POLITECNICO DI MILANO
Master of Science in Computer Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

AUDIO FEATURES COMPENSATION BASED ON CODING BITRATE

Polo Regionale di Como

Supervisor: Prof. Augusto Sarti
Co-supervisor: Dr. Massimiliano Zanoni

Tesi di Laurea di:
Maria Stella Tavella, ID 842105

Anno Accademico 2016-2017

POLITECNICO DI MILANO
Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

AUDIO FEATURES COMPENSATION BASED ON CODING BITRATE

Polo Regionale di Como

Relatore: Prof. Augusto Sarti
Correlatore: Dott. Massimiliano Zanoni

Tesi di Laurea di:
Maria Stella Tavella, Matricola 842105

Anno Accademico 2016-2017

Dedicated to
my parents Bianca and Giuseppe,
my family and friends,
Teresa,
Davide.

Abstract

During the past years, due to the advent of digital media, the way in which we were used to buy, collect and discover music has drastically changed. Personal music collections have reached enormous sizes thanks to the increase of digital storage capability and the advent of musical streaming services has enlarged the online availability of music. In this scenario arises the need for methods capable of organizing, browsing and classifying such huge music collections.

Music Information Retrieval (MIR) is a research field that deals with the retrieval of useful informations from music. The most used informations in the MIR field are the one that are extracted directly from the audio file and we refer to them as being *features* or *audio descriptors*. One of the main problems inside the scientific community of MIR is the difficulty for researchers in finding uniform audio collections (i.e. audio collections composed of songs that are encoded with the same encoding parameters). The problem arises when non-uniform collections are used in MIR final applications, such as classification or clustering techniques. This fact leads to having a non-uniform set of audio descriptors, since features values appear to be influenced by lossy audio compression and in particular by the bit rate value used during the encoding process.

This thesis work proposes various methods that are capable of unifying such non-uniform music collections by compensating features values extracted from songs encoded at some bit rate value as if they were extracted

from song encoded with an higher bit rate value. We will show that it is needed to compensate feature values in the case of a real case MIR scenario such as *Music Emotion Recognition (MER)*, which makes use of various classification and clustering techniques, and that their performances are very much influenced by features extracted from non-uniform music data sets.

Sommario

Nel corso degli ultimi anni, con l'avvento dei mezzi di comunicazione digitali, il modo in cui si era abituati ad acquistare, collezionare e scoprire la musica è drasticamente cambiato. Le collezioni musicali personali hanno raggiunto dimensioni enormi grazie anche all'incremento della capacità di memoria sui dispositivi digitali e all'avvento di servizi che offrono musica in streaming, aumentando la disponibilità di musica online. In questo scenario insorge la necessità di metodi capaci di organizzare e classificare queste vaste quantità di musica.

Music Information Retrieval (MIR) è l'ambito di ricerca che si occupa di recuperare informazioni utili dal contenuto musicale. Il tipo di informazioni più utilizzati in ambito MIR sono quelle che vengono estratte direttamente dal file audio, meglio conosciute nell'ambito come *feature* o *descrittori audio*. Uno dei problemi principali all'interno della comunità scientifica del MIR è la difficoltà per i ricercatori nel reperire collezioni audio uniformi (collezioni audio composte da file audio codificati utilizzando gli stessi parametri in fase di compressione). Il problema sorge quando vengono utilizzate collezioni audio non uniformi per applicazioni MIR, come tecniche di classificazione o di clustering. Questo porta ad avere un set non uniforme di descrittori audio, dal momento che i valori delle feature risultano influenzati dalla compressione con perdite (lossy) e, in particolare, dal valore di bit rate utilizzato durante il processo di compressione.

Questo lavoro di tesi propone diversi metodi capaci di uniformare collezioni

musicali non uniformi attraverso la compensazione dei valori delle feature estratti da canzoni codificate utilizzando un certo valore di bit rate come se fossero estratte da canzoni codificate utilizzando un valore di bit rate più elevato. Mostreremo che è necessario compensare le feature nel caso di applicazioni MIR quali *Music Emotion Recognition (MER)*, che richiede l'utilizzo di varie tecniche di classificazione e di clustering, e che le prestazioni di questi sono influenzate nel caso di feature estratte da un data set musicale non uniforme.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In the last decade we assisted to a fast mutation in the way we avail ourselves of digital multimedia contents. The worldwide spread of fast internet connection together with the increasing of storage capability and the diffusion of digital audio formats has demolished several music distribution and collection limitations. A consequence is the enlarged availability of online multimedia contents. Moreover, with the advent of easy to use tools, devices and software for audio, images or video production, the amount of user generated contents has never been so massive with respect to the past. The production and, therefore, the fruition of music, witnessed to the same evolution. As a consequence of that, we assisted to the birth of several musical streaming platforms like Spotify, Apple Music and SoundCloud among the others, which now offer a large musical on-demand catalogue. Those new services have changed the way in which we collect, deliver, buy, discover and access to music.

In this scenario we need solutions that are smart, fast and scalable in order to browse, classify and recommend large quantities of music. Some of the already existing methods are based on context-based informations, they focus on the extraction and retrieval of contextual features and, therefore, they rely on humans to manually tag the musical excerpts, by creating metadata-

based systems. Adopting the context-based solution, however, turns out not to be optimal when dealing with very huge quantities of musical excerpts. Some other music retrieval methods are based on content-based techniques, that aim at extracting musical and perceptual properties directly from musical excerpts. These informations are called *features* or *audio descriptors* and are largely used in the Music Information Retrieval field. Recently, hybrid methods are being adopted, that combine both context and content-based techniques [7].

Content-based methods for the organization and retrieval of musical informations combine audio feature extraction [9] together with machine learning techniques [32] and have been shown to be more suitable and fleet for the purpose. Those machine learning methods serve to build systems for music classification and recommendation (e.g., based on mood [19, 34], genre [25, 33]); music transcription [2, 20]; forensic analyses (e.g., plagiarism [1] and bootleg detection [4]); blind environmental inference [28], artist Identification, Instrument Recognition [9]; and others.

All the aforementioned methodologies are part of the Music Information Retrieval (MIR) field, which is a branch of Multimedia Information Retrieval (MMIR) that aims at extracting informations from multimedia data sources, facilitating the management and classification of large collections of files, such as audio, image and video. MIR operates by the use of features, which are abstract representations of the digital audio file, computed directly from the raw audio signal. Audio feature analysis is a key step for a large variety of information retrieval applications.

The possibility of storing large collections of digital media would have been not so popular nowadays without the use of some sort of compression of the data. MP3 compression (also known as *MPEG-2 Audio Layer III*) is often applied to audio files and the majority of tracks now available on the internet is often not in a lossless format. However, frequently, musical collections can be made up of songs encoded at different bit rates or with different types of encoders and settings. We refer to this kind of collections as being

inhomogeneous [3]. This is the case of YouTube, Vimeo or SoundCloud, where users are allowed to upload music in any type of formats. Some other examples are Myspace, that hosts audio content encoded at 96kbps MP3, and iTunes, which provides content encoded as 128kbps AAC. Thus, audio encoding formats are not uniform across different collections and more often there are cases in which they are not homogeneous even in the same audio collection [44].

Nowadays, one of the crucial issues relating to MIR applications is the difficulty in collecting big amounts of audio excerpts. Due to the lack of online availability of large datasets, researchers often rely on collecting songs from various sources, mainly in different compressed format, with the result of having songs encoded with various encoders or at multiple bit rates [6]. It has been observed that the quality and performances of the final applications of MIR, such as classification or clustering, are influenced by the quality (i.e. encoding settings) of the audio files under analysis [27]. For that reason we need ways that make feature values extracted from the audio collection under analysis the more uniform as possible by using features compensation techniques.

Lossy compression is an irreversible process that aims at reducing the weight and bandwidth of some multimedia data by producing a copy of the original data containing less information with respect to the original one. That is to say that the same feature (information) extracted from the same audio file coded at different bit rates or, more generally, with different audio codecs, might not assume the same value. In other words we can assert that feature values are bit rate dependent. The same feature, when comparing songs at equal bit rate value, can be very significant and discriminative than when comparing the same feature in the case of songs encoded at different bit rate values. Since feature values change depending on the compression bit rate and thus are less informative in case of a non-homogeneous dataset, they negatively affect classification or clustering performances. We will prove this fact in this thesis by performing classification and clustering techniques and comparing the results obtained by using data sets encoded at

mixed or non-mixed bit rates values and using two different types of feature compensation techniques.

In this thesis we start by analysing the relationship and dependence between coding bit rate and audio feature values, we show that feature values are, indeed, bit rate dependent. We will show that it is possible (and in some cases worth) to compensate feature values extracted from audio files encoded at some given bit rate, as if they where extracted from the same audio file encoded with a different bit rate value (e.g. high), and that compensated features approach the ideal target value. The proposed method compensates the degradation of feature values due to lossy audio compression. Feature compensation enables the generation of homogeneus train and test data sets that can be used to perform classification tasks. We propose several supervised methods based on trained models. The models have been validated on the CAL500 database, a collection of 500 audio files ranging over a broad variety of musical genres, largely used for music information retrieval's final applications. In order to prove the effectiveness and accuracy of the feature compensation method, we apply it to a real MIR application, such as Music Emotion Recognition (MER).

1.2 Thesis Outline

Chapter 2 illustrates the current state-of-the-art methodologies. In chapter 3 we provide the theoretical background that comprehend the definition of the audio features used and the description of some of the most popular machine learning methods and algorithms. Chapter 4 is devoted to the description of the methodology setup and of the implementation of the various compensation methodologies. Experimental tests and results are shown in chapter 5. Chapter 6 concludes the work and suggests possible future improvements of the method.

Chapter 2

State of the Art

This chapter provides the reader with a review of the main works that are present in the literature that comprise the use of audio descriptors in Music Information Retrieval applications. We will discuss about the use of audio features in the MIR field, as well as about the effects that audio coding has on audio descriptors and their usage in performing supervised machine learning tasks related to music retrieval. Another section is devoted to the analysis of audio features among song collections encoded with different encoders/bit rates and their robustness in the case of audio classification and others music information retrieval applications and tasks, such as music emotion recognition, genre classification, artists identification, chord recognition and onset detection tasks. Some other works presented treat the problem of feature normalization against degradation due to audio compression. Then there will be presented an overview on general audio classification and we will devote a section introducing the problem of *Music Emotion Recognition, MER*, a method performed in this thesis with the intent of evaluating the effectiveness of our feature compensation method and test its efficiency in a real case MIR scenario.

2.1 The use of audio descriptors for MIR applications

In large audio collections it is very common to store audio files in a compressed format. This is due to the fact that there exists a large number of systems with which we can access to music and therefore the retrieval of audio files coming from different sources leads to the storage of non uniform collections, especially in the case of a personal audio collection and in case of collections employed for research purposes. However, also encoding parameters usually vary from collection to collection or even within the same collection. There exist a large variety of different parameters when it comes to audio encoding, such as the codecs, bit rates, etc. Having this non homogeneus audio assortment can be a problem when performing MIR tasks, since information retrieval's applications widely rely on the use of audio features, that are audio descriptors extracted directly from audio files. However, those audio descriptors turn out to be strongly dependent on audio compression, in particular they depend on the choice of the compression parameters.

For the purpose of this thesis we will mainly focus on the fact that audio features are bit rate dependent, thus final MIR applications are negatively influenced by this fact. In other words, feature values extracted from a song encoded at different bitrates will vary depending on the bit rate value chosen during the encoding process. We will see that the case of having an homogeneous audio collection, i.e. a data set of songs encoded with the same bit rate value, will hield at least coherent results in a number of MIR applications, while working with an heterogeneous audio collection, i.e. a data set of songs econded at mixed bit rate values, will result in poorer accuracy with respect to the former case.

One of the fundamental problems in the Music Information Retrieval field is Classification. Classification is a supervised machine learning technique and in the case of MMIR it is adopted when it is needed to *label* or *automatically categorize* large multimedia collections. In the field of Music

Information Retrieval (MIR), this problem refers to labeling each song based on, for instance, genre, artists, mood etc. This is a key problem that has already multiple applications and it is very useful for the purpose of music organization and retrieval.

Audio Content Analysis (ACA) refers to the extraction of informations from audio signals. Nowadays, lots of techniques are employed for that purpose and it is becoming fast and easy to collect intelligence on formal, perceptual, musical, and technical aspects of audio files. It is the case of key and tempo analysis, useful for automatic music transcription of musical scores, or the analysis of artists' performances in order to attempt modeling the human emotional affection in listening to music.

The most important step for the classification of audio files is audio feature extraction of numerical descriptors that serve to capture useful informations about the musical content. Those informations, in the case of classification, should be very representative with respect to the music. In fact, extracted features could provide, for instance, informations on the dynamic of a musical excerpt (usually energy features provide this type of clue), on Timbre (spectral features), Harmony and Rythm. Over the years the MIR community has developed fast methods and a large variety of tools for automatic feature extraction. The audio descriptors extracted from the audio files are organized as feature vectors in order to provide a representation of the examples to be classified. Given that, the goal of classification is to find a mapping from the feature space to the output labels in order to minimize the prediction error. Many audio features have been proposed for audio classification and there exist different taxonomies for their categorization. Depending on the classifier, feature representation for a song can be carried on in different ways, therefore features vectors can be organized as: a single feature vector per song, as a similarity measure calculated for each group of songs, another approach can be to calculate the mean or median value of each feature vector and so on [9]. In order to improve those types of classifiers it is always adviced to use the *most informative features*, i.e. the ones that account for, singularly or grouped, the largest amount of in-

formation on the musical content.

In Table 2.1 we show the most common audio features used for the task of Music Emotion Recognition [19] grouped by type of musical information, in order to provide the reader an idea on the importance of the feature extraction step in the MIR field.

TYPE	***FEATURES***
Dynamics	RMS energy
Timbre	MFCCs, spectral shape, spectral contrast
Harmony	Roughness, harmonic change, key clarity, majorness
Register	Chromagram, chroma centroid and deviation
Rhytm	Rhythm strength, regularity, tempo, beat histograms
Articulation	Event density, attack slope, attack time

Table 2.1: Most used audio features for Mood Classification.

2.2 The effect of lossy audio encoding on the robustness of audio features

In [27] is introduced the Audio Degradation Toolbox (ADT), a tool for the controlled degradation of audio signals, with the purpose of analysing the robustness of various audio processing algorithms against different combinations of audio degradations, including MP3 compression. This works shows that the performances of different music informatics algorithms strongly depends on a combination of method and degradation applied. Perceptual audio coders may affect descriptors since they introduce distortions and perturbations to the original audio signal, usually by reducing high-frequency content, by the introduction of blurring effects and smoothing of the spectral envelope.

In [40] MP3 encoding is used to study the robustness of MFCCs and Chroma features, which are among the most popular audio descriptors used in MIR research, since they capture significant informations on timbre and

tone. Results show that controlling encodings or analysis parameters does not increase robustness significantly when the sampling rate is 22050 Hz. Although the combination of different codecs and bit rates leads to more robust descriptors.

[31] also treats the influence of MP3 coding on MFCCs, showing that they are very robust in case of collections of songs encoded at high bit rates values, while they exhibits some distortions on audio collections encoded with low bit rate values.

In [39] is treated the case of audio compression on chord recognition tasks, comparing chroma features extracted among music files encoded at different bit rates with the one extracted from original PCM audio files. Results indicate that the compression ratio influences the sound quality of compressed songs, but in this case low bit rate compression does not strongly affect the chord recognition accuracy.

[44] evaluates the effect of audio encodings on the onset detection tasks. The results show that significant changes in onset detection accuracy only occur at bit rates lower than 32kbps.

In [12] results show that if the music collection is heterogeneous, i.e. audio files are encoded using mixed bit rate values, then MIR results are significantly different with respect to results obtained by using music collections that consist of raw audio files. In this study it is proposed a method for normalizing MFCCs which is effective in reducing the differences between MIR results from mixed music collections.

Despite studies have been conducted on feature normalization techniques, the solution for feature compensation has not been treated by the MIR community. In this thesis we provide the implementation of various feature compensation methods based on machine learning approaches. Results show that it is possible to compensate feature values in order to improve MIR's final application tasks.

2.3 The effect of lossy audio encoding on MIR classification tasks

The purpose of [6] is to evaluate the impact of audio encoding bit rate on the performance of audio-based music information retrieval tasks. Specifically, it treats the case of how the accuracy of an automatic genre classification task is effected by various audio encodings, using the Mel-frequency cepstral coeffcients (MFCCs) as audio descriptors. The work proposed in [6] shows that classification performances decrease as the encoding bit rate of the audio files under analysis decreases. This result indicates that audio encoding parameters have a singificant impact in this specific context, since low audio quality resutls in poorer classification performance and heterogeneous mixtures of audio encodings significantly impact classification accuracy.

In [40] the robustness against audio quality and compression of MFCCs and Chroma features is evaluated. This study shows that the amount of degradation may depend on the musical genre of the songs under analysis, thus affecting genre classification tasks. One of the main side effects of applying perceptual encorders is infact the introduction of filters that reduce the high-frequency spectral content of the audio file. On genres that make use of high-frequency sounds (e.g. cymbals and electronic tones) audio compression should impact more than in genres not including them (e.g. country, blues and classical). However, loss of performance does not include audio files encoded at high bit rate values, such as MP3 encoded at 128kbps, for which there is no evidence of decrease in classification accuracy.

[6] and [40] treat the case of genre classification and its performances against lossy audio compression of the data set.

In the following sections we will instead study the effects of having a data set of audio files compressed at various bit rate values in another popular MIR application, such as Music Emotion Recognition, by performing both classification and clustering techniques, in order to prove the effectiveness of the feature compensation method proposed.

2.4 Music Emotion Recognition (MER)

The problem of Music Emotion Recognition (Mood Classification), i.e. finding a relationship between music and human emotions in order to automatically tag music, has been studied for decades expecially in the field of psychology. Since antiquity the role of music in society has been seen not only as entertainment but also its social and psychological effects have been considered. Thus, the main goal of MER systems is to automatically organize, label and retrieve music according to emotions. This type of analysis is becoming more and more popular nowadays, since always more often, users find very helpful browsing music according to emotions.

Before the introduction of Music Emotion Recognition (MER), the retrieval and classification of music has been carried on using standard informations such as: name of the composer, work title, year of release etc. Those informations still remain important and widely used for a number of information retrieval applications. However, we still need to identify deeper and important cues that are representatives for humans and their relationship with music and emotions. The problem of Music Emotion Recogniton has been proposed for the first time in the MIR community back in 2007, during the annual Music Information Research Evaluation eXchange (MIREX) [15].

In this chapter we give an overview on the studies conducted for the representation of musical emotions in the field of psychology. Those models of emotions are currently employed for MIR classification systems. Classification algorithms and audio descriptors used for Music Emotion Recognition are very similar to the one used for genre classification. Despite the need for a training data set of mood annotations generated by humans through perceptual experiments, contextual informations may result incomplete or be entirely missing, due to the rapid growth of digital music libraries (i.e. in case of brand new music). Also, manual annotation is a very time and resources consuming activity. For this reason, content-based methodologies for the representation of music need to be introduced. Human emotions can be influenced by a number of musical attributes such as tempo, har-

mony, timbre, loudness, harmonicity and so on. These musical properties are acoustical features and they have been employed very successfully for mood classification. Audio features result to be very discriminative in case of both supervised (classification) and unsupervised (clustering) tasks, where the need of very representative feature vectors is crucial in order to learn the best model for class separability.

2.4.1 Representation of emotions

Music-IR systems tend to use either categorical descriptions or parametric models of emotion for classification or recognition. Those representations have been inferred by psychology researches. In those experiments, candidates were asked to listen to music and then to rate or tag predefined adjective that represents at best perceived emotions during the listeining experience. Emotions are grouped into four categories: *angry, fearful, surprised, happy, and sad* [26]. Despite cross-cultural studies suggest that there may be universal psychophysical and emotional cues that transcend language and acculturation [19], there exist several problems in recognizing moods from music. In fact, one of the main difficulties in recognizing musical mood is the ambiguity of human emotions. In other words, it is clear that different individuals perceive and feel emotions induced by music in different ways. Also, their individual way of expressing them using adjectives is biased by a large number of variables. In fact, the human ability to associate music with an emotional meaning depends on several factors, such as: the form and structure of music, the attitude and previous experience of the listener, their training, knowledge and psychological condition [14].

Categorical representation of emotions

Categorical approaches for the representation of emotions comprehend the finding and the organization of a set of adjectives/tags/labels that are emotional descriptors, based on their relevance and connection with music. One of the first studies concerning the aforementioned approach is the one conducted by Henver and published in 1936, in which the candidates of the experiment were asked to chose from a list of 66 adjectives arranged in 14

groups [14] as shown in Figure 2.1. In a more recent study, Zenter et al. inferred a set of 801 "general" emotional terms into a subset of 146 terms specific for music mood rating [26]. Mood adjectives used in the MIREX Audio Mood Classification task [15] have been categorized songs into five mood clusters, shown in Table 2.2

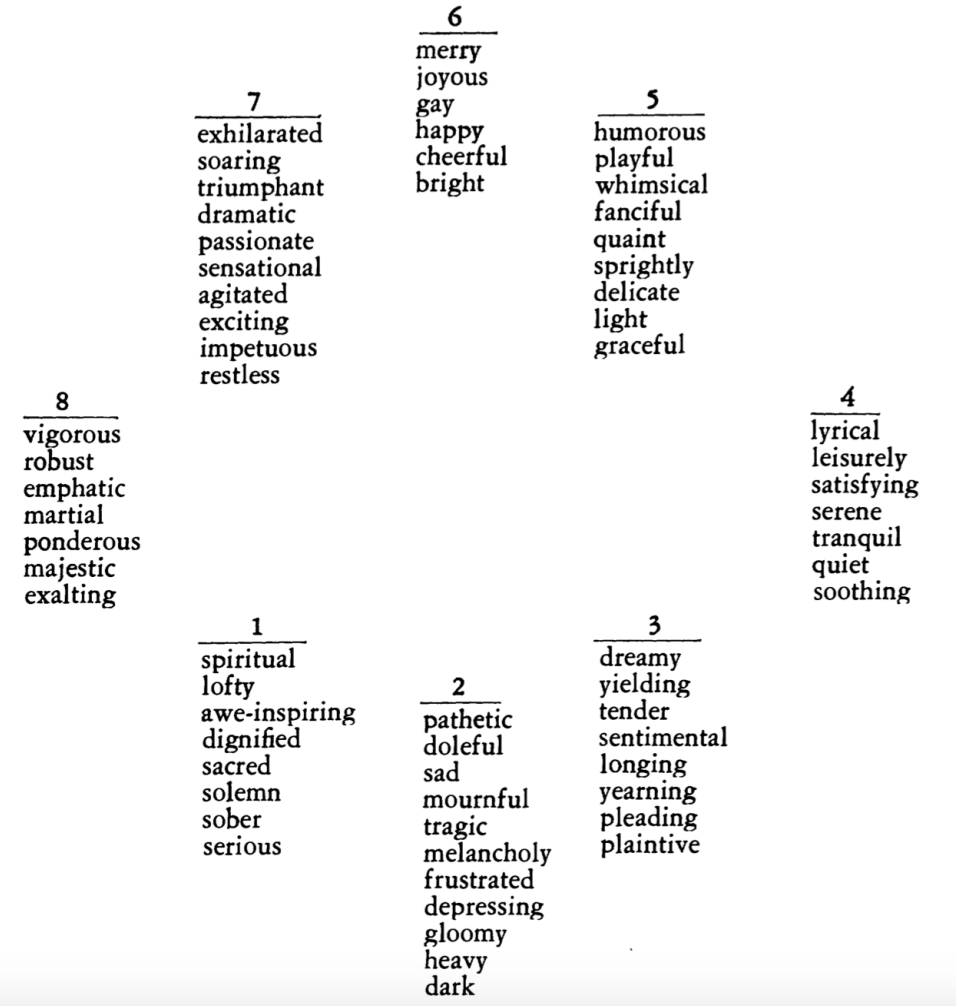


Figure 2.1: Hevner Adjective Clusters.

Clusters	***Mood Adjectives***
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 2.2: Mood adjectives used in the MIREX Audio Mood Classification task.

Scalar/dimensional representation

Other researches, such as the one conducted by Russell [30], suggests that mood can be scaled and measured by a continuum of descriptors or simple multidimensional metrics. In this scenario, sets of mood descriptors are organized into low-dimensional models, one of that models is the *Valence-Arousal* (V-A) space (see Figure 2.2), in which emotions are organized on a plane along independent axes of arousal (intensity) and valence (an appraisal of polarity), ranging from positive-to-negative.

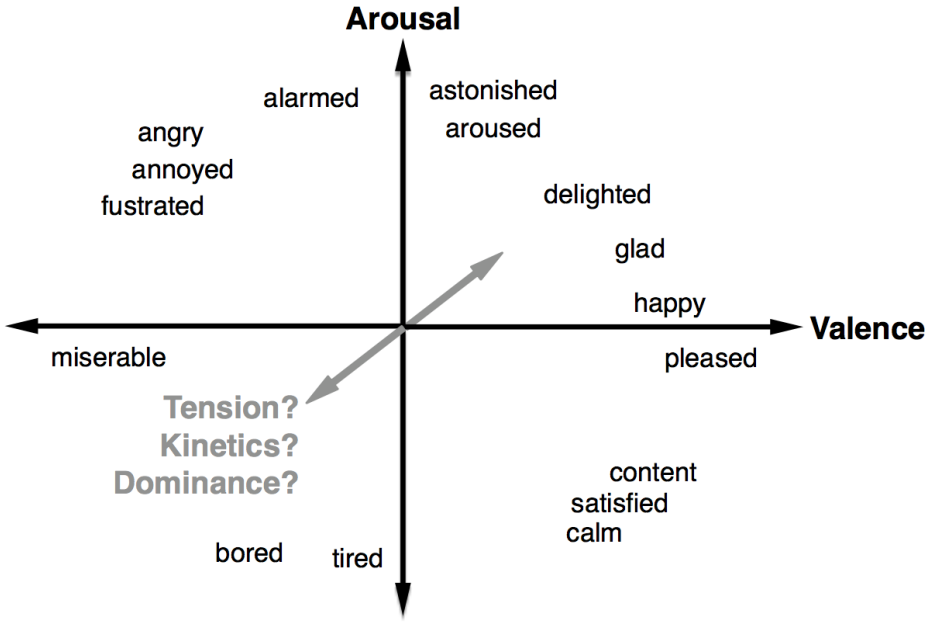


Figure 2.2: Russell's Valence-Arousal space.

Chapter 3

Theoretical Background

In this chapter we provide the theoretical background our methods are based on. First we provide a general overview on audio coding techniques, then we present a detailed description of the audio features used for this thesis work. In the last section an overview of some of the most used machine learning algorithms and methodologies is presented.

3.1 Overview on Audio Coding

Audio coding is a special technique employed mostly in order to provide a more efficient transmission of data, by reducing the bandwidth and storage of audio files. Audio codecs can be divided into two categories:

- *Lossless.* The first theory about data compression was formulated by Claude E. Shannon in 1949. He proved that there exist a limit in how much a data can be compressed without losing any information. Therefore, the main property of lossless audio codecs is that they can be de-coded to their uncompressed and original form.

- *Lossy.* A lossy audio coding format reduces the bit resolution of the compressed data, i.e. the number of bits of information per sample. Original data can not be retrieved back because some of the information is lost during compression.

Lossy encorders take advantage from psychoacoustical studies, reasearches aiming at understanding human ear and brain interactions with respect

to musical stimuli. A *perceptual audio codec* is a codec that takes advantages of human ear characteristics and psychoacoustical models in order to cut useless informations (less human audible sounds/frequencies, not perceived by the human ear) out of the audio file by reducing audio fidelity. In particular, these techniques exploit a perceptual phenomenon such as the frequency-dependent abdoslute *threshold of earing*, depicted in Figure 3.1, which is a curve describing the amount of energy needed by a pure tone to be detected by a listener in a noiseless environment [43].

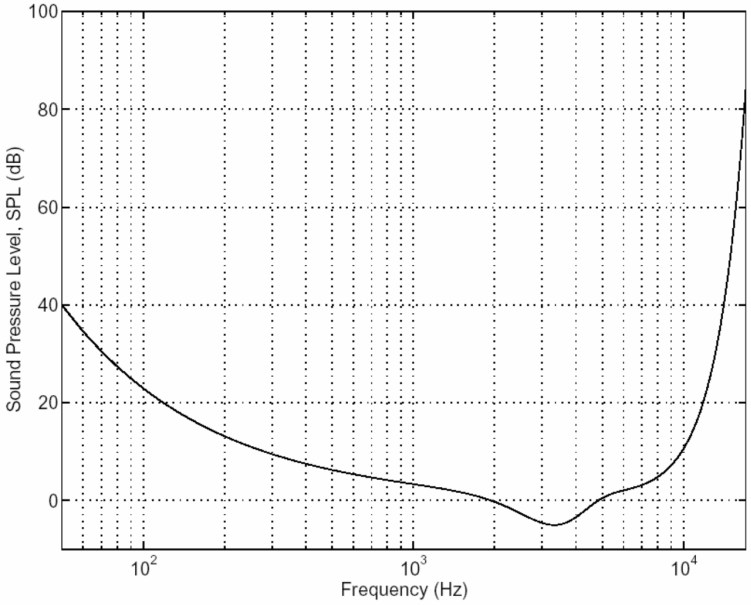


Figure 3.1: The absolute threshold of hearing.

Another exploited phenomenon is *masking*, which is the effect that occurs when the perception of a sound is affected (*masked*) by the presence of another, usually louder, sound. The masking effect can occur either in the time domain or in the frequency domain. Both the masking phenomenon and the absolute threshold of earing are exploited by audio compression in order to discriminate between useful and useless informations, i.e. the one inaudible by the human ear, by the use of low pass filters and the removal and smoothness of certain spectral peaks of the audio signal. Thus, as a consequence, the analysis of lossy-compressed audio signals, rather than the analysis of lossless versions of the same signals, may lead to different

results. One of the main encoding parameters is the **bit rate**. It is an user option that has to be set before starting with the encoding process. The bit rate represents the amount of information (data) that can be stored for every second of uncompressed audio. The choice of bit rate allows the user to modify the quality of the encoding and thus of the resulting audio file. The Layer III standard defines bitrates from 8 kbit/s up to 320 kbit/s, default is usually 128 kbit/s. Higher bit rate values implies more precision in measuring the samples and therefore much more audio resolution.
MP3 encoders also allow the possibility to specify between two different types of bit rates [43] :

- *Constant Bitrate* (CBR): usually a default setting. It indicated that every part of an audio file is encoded using the same amount of bits.
- *Variable Bitrate* (VBR): allows the bit rate to vary depending on the dynamics of the audio signal. Frames within a song can be encoded using different bit rate values.

There exist different types of lossy audio codecs:

- *MP3*: The MPEG-1 Audio Layer 3 was released in 1993 becoming the most popular audio format used for music files. It is supported by the majority of platform and devices.
- *AAC*: Advanced Audio Coding, developed in 1997 as the successor of the MP3 but it never prevailed over it. The AAC algorithm for music compression is more complex than the one used by MP3 and it can achieve better results compared to the same encoding parameters used for MP3.
- *OGG (Vorbis)*: it is a free and oper source software for lossy audio encoding and it is able to produce a smaller file size with an equivalent audio quality compared to the others formats.
- *WMA*: Windows Media Audio is the proprietary format created by Microsoft. In terms of performances and quality it is similar to AAC and OGG (thus better than the MP3), but the fact that it is proprietary makes it usually not well supported by various devices and platforms.

For the purpose of this thesis we will only use the MP3 (MPEG-1 Audio Layer 3) as audio codec, since it is the most used for audio compression.

3.2 Audio Features

The main purpose of Music Information Retrieval (MIR) community is to study and investigate on the automatical understanding, organization and retrieval of useful informations from music. All the elements involved in music can be collected and described as going from low level, which is related to the sound signal and audio content, to higher levels of abstractions, related to the human perception of sounds and music. These elements are generally referred as features or descriptors.
Audio features can be grouped and described as follows:

- **Low-Level Features (LLF)**: are the important building blocks for audio systems, they are extracted directly from the audio signal using signal processing techniques and they are able to characterize each type of sound.
- **Mid-Level Features (MLF)**: unlike low-level features, they capture intrinsic properties of music that humans preceive. They add further levels of semantics by referring to structural components of music such as *Melody*, *Harmony* and *Rhythm*.
- **High-Level Features (HLF)**: represent an high degree of semantic abstraction, which make them easily understandable by humans. They are the result of the composition of LLF and MLF. Example of HLF are: mood, genre, social tags.

For the purpose of this thesis we used a mixture of low-level and mid-level features. Table 3.1 shows the audio features used, grouped by type.

In the next section we will provide an overview of the features used in this work. We refer to [42] and [17] for a more extensive explanation.

TYPE	*FEATURES*
Energy	RMS Energy, Intensity, Intensity Ratio, Loudness.
Spectral	Spectral Centroid, Spectral Skewness, Spectral Kurtosis, Spectral Inharmonicity, Spectral Slope, Spectral std deviation, Irregularity k, Irregularity j, Tristimulus, Spectral Variance, Spectral Flux, Spectral Contrast Valleys, Spectral Contrast Peaks, Spectral Contrast Mean, Chromagram, MFCCs, Crest, Odd-Even Ratio, Spectral Rolloff, Sharpness, Spectral Smoothness, Spread, Flatness.
Temporal	Zero Crossing Rate.
Waveform	Average Deviation, Kurtosis, Variance, Skewness.

Table 3.1: Summary of all the used features, grouped by type.

3.2.1 Energy Features

Energy features are very useful for Music Information Retrieval studies. Measuring the energy distribution and evolution of a sound can characterize the listener's perceptual experience or even the genre of the musical excerpt under analysis.

Root-Mean-Square (RMS)

It is a measure of the total energy of a signal. It is defined as:

$$F_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N-1} x(n)^2}. \tag{3.1}$$

Where $x(n)$ represents the audio signal and N is the total number of samples. Figure 3.2 shows the RMS evaluated on 30 seconds of a musical excerpt. The feature was extracted using the Python library *Librosa* [1].

[1] https://librosa.github.io/librosa/

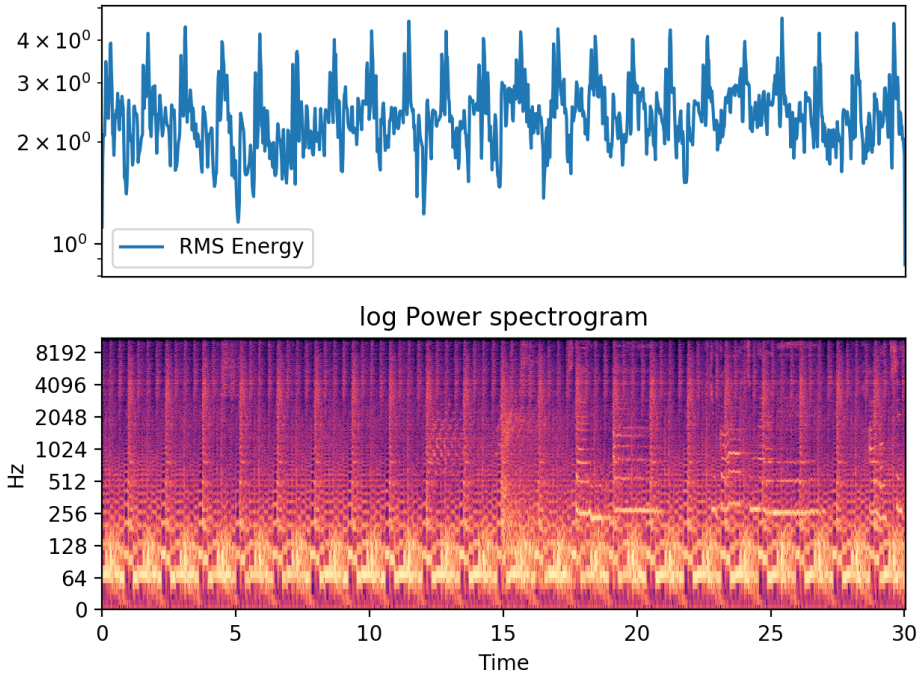


Figure 3.2: Root-Mean-Square (RMS) Energy of the song "Army of Me" by Bjork.

Intensity and Intensity Ratio

The signal is divided into k sub-bands, ranging from L_p to H_p. For each sub-band the intensity ration is defined as the ratio of that sub-band's intensity to the overall intensity I,

$$I_{ratio} = \frac{1}{I} \sum_{k=L_i}^{H_i} S(k), \tag{3.2}$$

where $S(k)$ is the amplitude of the k-th bin of the frequency spectrum and the Intensity I is computed as the summation of all the components:

$$I = \sum_{k=0}^{F_s/2} S(k). \tag{3.3}$$

Loudness

Loudness is the hearing sensation referred to the intensity. It is associated to the sound intensity of the stimulus, and it depends on the spectral content

of the sound signal. While sound intensity can be measured, the sensation of loudness can only be determined by the listener by comparing two audible stimuli and deciding whether they are equal or not in loudness. These experiments are useful for psychoacoustical purposes and studies.

3.2.2 Spectral Features

Short Time Fourier Transform (STFT)

The Short Time Fourier Transform (STFT) is a special Fourier transform used to determine the spectral content of local sections of a signal that changes over time. It is used for the computation of spectral features. The STFT is evaluated by dividing the signal into segments and then computing the Fourier transform of each segment separately. Its mathematical formulation is:

$$X_m(\omega) = \sum_{n=-\infty}^{\infty} x(n)\omega(n - mR)e^{j\omega n}, \tag{3.4}$$

where:

- $x(n)$ is the input signal at time n
- $w(n)$ is a length M window function (e.g. Hamming)
- $X_m(\omega)$ is the Discrete Time Fourier Transform (DTFT) of the windowed data centered at time mR
- R is the hop size between successive DTFTs in samples.

In figure 3.3 we show the STFT extracted from 30 seconds of the song *"Brazil"* by Django Reinhardt.

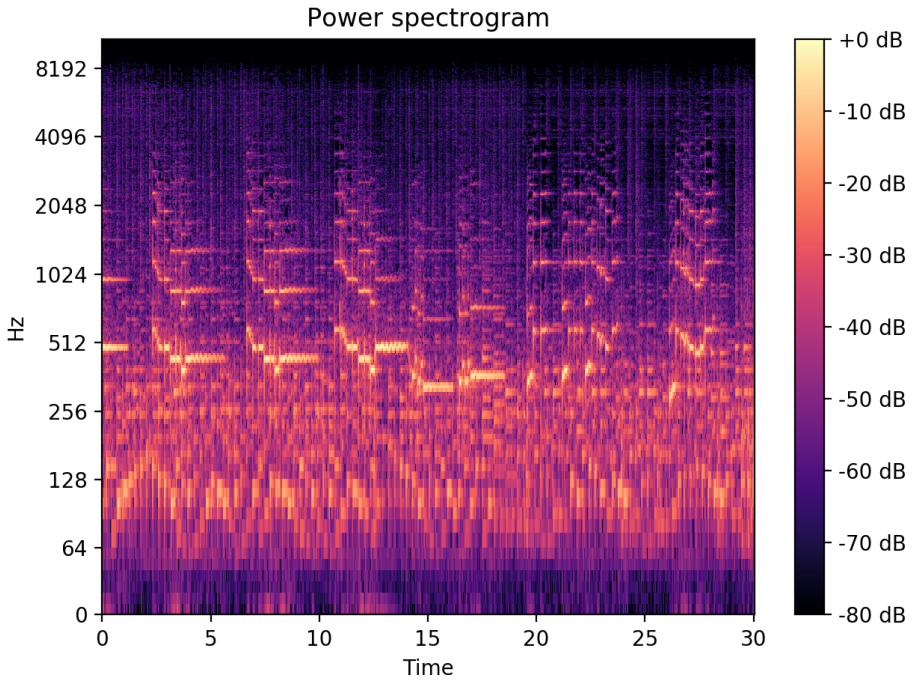


Figure 3.3: STFT of the song "Brazil" by Django Reinhardt.

Spectral Centroid

The Spectral Centroid determines the centroid (center of gravity) of the magnitude spectrum of the STFT. It is a measure of the amount of high or low frequencies components present in the spectrum of a sound, it is related to the *brightness* of the sound.

$$F_{SC} = \frac{\sum_{k=1}^{K} f(k) S_l(k)}{\sum_{k=1}^{K} S_l(k)}. \tag{3.5}$$

$S_l(k)$ is the Magnitude Spectrum, $f(k)$ is the frequency and K is the total number of frequency bins.

Figure 3.4 shows the Spectral Centroid extracted from 30 seconds of the song *"Army of Me"* by Bjork.

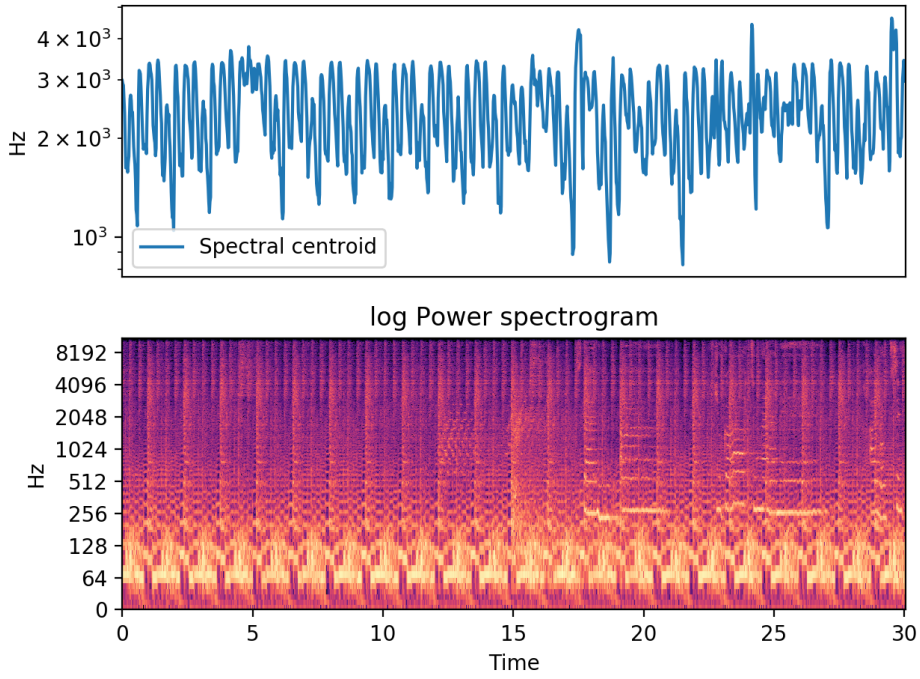


Figure 3.4: Spectral Centroid of the song "Army of Me" by Bjork.

Spectral Spread

The Spectral Spread is a measure that identify the frequency range of a sound around the spectral centroid. It is defined as the normalized second centered moment of the spectrum, i.e. the spectral variance.

$$F_{SS} = \sqrt{\frac{\sum_{k=1}^{K}(f(k) - F_{SC})^2}{\sum_{k=1}^{K} S_l(k)}} \tag{3.6}$$

The spectral spread accounts for the sensation of timbral fullness or richness of a sound.

Spectral Skewness

The Spectral Skewness provides an estimation of the symmetry property of the spectral distribution in a frame. It can be obtained through the following mathematical expression:

$$F_{SSK} = \frac{\sum_{k=1}^{K}(S_l(k) - F_{SC})^3}{KF_{SS}^3}. \tag{3.7}$$

Depending on its value the presence or absence of asymmetric concentration is identified. A value equal to zero indicates a symmetric distribution of spectral energy around the spectral centroid.

Spectral Kurtosis

The spectrum can be characterized in terms of its peakiness. This property can be expressed by means of the spectral kurtosis.

$$F_{SK} = \frac{\sum_{k=1}^{K}(S_l(k) - F_{SC})^4}{KF_{SS}^4}. \tag{3.8}$$

In particular, the kurtosis describes to what extent the spectral shape resembles or differs from the shape of a Gaussian bell curve. For values below zero the spectral energy tends towards a uniform distribution. Such behavior is typically observed for wide-band sounds. Values larger than zero characterize a peaked spectral shape concentrated around the spectral centroid. Such a spectral shape is typically obtained for narrow-band sounds.

Spectral Inharmonicity

Partial tones can be characterized by their degree of harmonicity, which expresses the amount of consonance or dissonance perceived in a music excerpt.
Harmonic tones are composed of a fundamental tone and overtones whose frequencies are integer multiples of the fundamental freqyency f_0. Given a set of partials, a measure of inharmonicity can be obtained by computing the energy-weighted absolute deviation of the estimated partial tone frequencies $f_{\hat{\mu}}$ and the idealized harmonic frequencies $\hat{\mu} f_0$.

$$F_{SH} = \frac{2}{f_0} \frac{\sum_{k=1}^{K} |f_k - k f_0| \, (S_l(k))^2}{\sum_{k=1}^{K} S_l(k)^2}. \tag{3.9}$$

Which ranges from 0 (purely harmonic) to 1 (inharmonic).

3.2.3 Irregularity K

Defined by [Krimphoff et al. 1994] [22] and it is the sum of the considered amplitude minus the mean of the previous, next and considered amplitude

$$F_{IR_K} = \sum_{k=2}^{N-1} \left| S_l(k) - \frac{S_l(k-1) + S_l(k) + S_l k + 1}{3} \right|. \tag{3.10}$$

3.2.4 Irregularity J

The Irregularity defined by Jensen [18] is computed as the square of the amplitude differences between adjacent partials.

$$F_{IR_J} = \frac{\sum_{k=1}^{K} (S_l(k) + S_l(k+1))^2}{\sum_{k=1}^{K} S_l(k)^2}. \tag{3.11}$$

Tristimulus

The relative energy of partial tones can be quantified by three parameters which measure the energy ratio of the partials. For the first partial we have:

$$F_{T_1} = \frac{S_l(k=1)^2}{\sum_{k=1}^{K} S_l(k)^2}, \tag{3.12}$$

for the second, third, and fourth partial

$$F_{T_2} = \frac{\sum_{k \in 2,3,4}^{K} S_l(k)^2}{\sum_{k=1}^{K} S_l(k)^2}, \tag{3.13}$$

and the remaining partials

$$F_{T_3} = \frac{\sum_{k=5}^{K} S_l(k)^2}{\sum_{k=1}^{K} S_l(k)^2} \tag{3.14}$$

respectively, where $S_l(k)$ represents the amplitude of the k-th partial tone in the l-th frame, with $k \in 1, 2, .., K$.

Spectral Flatness

It is a measure of the peakness of a signal, defined as the ratio between the geometric mean and the arithmetic mean of the magnitude spectrum.

$$F_{SF} = \frac{\sqrt[K]{\prod_{k=0}^{K-1} S_l(k)}}{\sum_{k=1}^{K} S_l(k)}. \tag{3.15}$$

A higher spectral flatness value indicates a more uniform spectral distribution, whereas a lower value implies a peaked and sparse spectrum.

Spectral Rolloff

It is defined as the frequency below which the 85% of the total energy is contained

$$\sum_{k=0}^{K_{roll}} S_l(k) = 0.85 \sum_{k=0}^{K} S_l(k). \tag{3.16}$$

The lower the value of K_{roll}, the more spectral energy is concentrated in low-frequency regions. Spectral rolloff estimates the amount of high frequency in the signal.

Figure 3.5 shows the exatracted Spectral Rolloff on 30 seconds of the song *"He War"* by Cat Power.

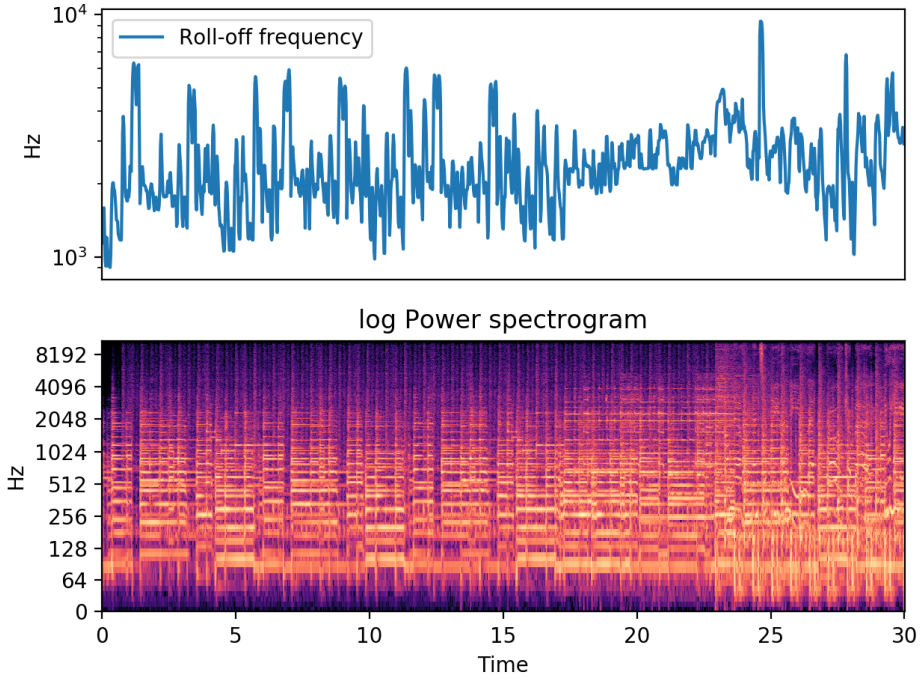


Figure 3.5: Spectral Rolloff of the song "He War" by Cat Power.

Spectral Flux

It measures the amount of spectral change between consecutive signal frames. The Spectral Flux is defined as the squared differences in frequency distribution of two successive time frames, measures the rate of local change in the spectrum

$$F_{SF} = \frac{1}{K}\sum_{k=1}^{K}[log(|(S_l(k)| + \delta) - log(|S_{l+1}(k)| + \delta)]^2, \tag{3.17}$$

where δ is a parameter used in order to avoid overflow during the calculus.

Sharpness

Sharpness is used as a Timbral descriptor and it is a measure of how much the spectrum of a sound is in the high end, and can be computed as a weighted sum of the specific loudness level in various bands.

Mel Frequency Spectral Coefficients (MFCCs)

The Mel Frequency Spectral Coefficients are a set of features that model the spectral envelope and the human auditory response. They are spectral low-level features based on the Mel-Frequency scale, a model that considers the human auditory system's perception of frequencies and provide a psychoacoustical representation of the spectral content. MFCCs are widely used in the field of MIR expecially for the discrimination between speech and music or for genre classification.
MFCCs are obtained from the Discrete Cosine Transform (DCT) of a power spectrum on a nonlinear Mel-Frequency scale (computed by a mel-filter bank). The mathematical formulation is:

$$c_i = \sum_{k=1}^{K_c} \left\{ log(E_k) cos[i(k - \frac{1}{2})\frac{\pi}{K_c}] \right\} with 1 \leq i \leq N_c, \tag{3.18}$$

where c_i is the i-th component, E_k is the spectral energy measured in the critical band of the i-th mel-filter, N_c is the number of filters and K_c is is the amount of cepstral coefficients ci extracted from each frame.
In Figure 3.6 we show the plot of the MFCCs for the song *"Strawberry Fields Forever"* by The Beatles.

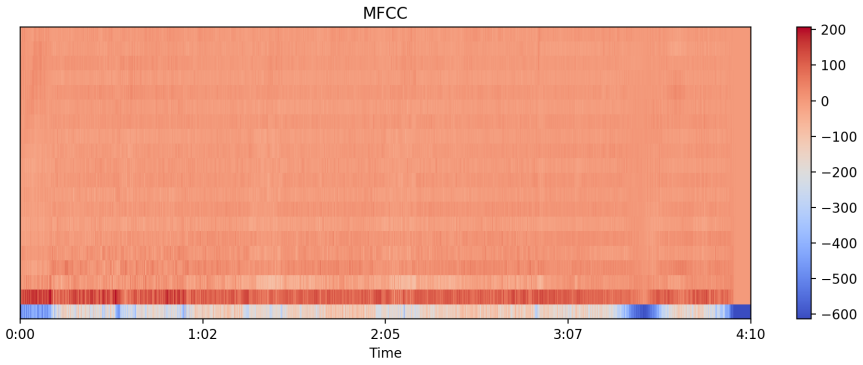


Figure 3.6: MFCCs of the song "Strawberry Fields Forever" by The Beatles.

MFCCs are derived as follows:

- Pre-processing in time domain.
- Take the Fourier transform of the signal.
- Convert the powers of the spectrum obtained in the mel scale, using triangular overlapping windows.
- Take the logs of the powers at each of the mel frequency and compute the DCT.
- Finally retrieve the MFCCs, that are the amplitudes of the resulting spectrum.

The Mel scale is a perceptual scale of pitches. A popular formula for the convertion of f hertz into m mels is:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.19}$$

Odd-Even Harm

They represent the even and odd harmonics of the spectrum, they are derived as follows:

$$F_{EH} = \sqrt{\frac{\sum_{k=1}^{K}(S_l(2k)^2}{\sum_{k=1}^{K}(S_l(k))^2}}, \tag{3.20}$$

$$F_{OH} = \sqrt{\frac{\sum_{k=1}^{K+1}(S_l(2k-1)^2}{\sum_{k=1}^{K} S_l(k)^2}}. \tag{3.21}$$

Chromagram

The chromagram is a special time-frequency representation of the musical signal, the spectral energy of the signal is mapped onto spectral bins that correspond to the twelve semitones of the chromatic scale.
Figure 3.7 shows an example of Chromagram extraction performed with *Librosa*.

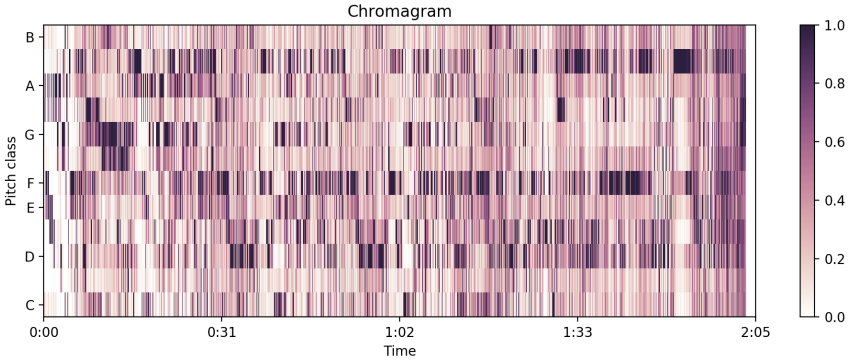


Figure 3.7: Chromagram of the song "Strawberry Fields Forever" by The Beatles.

Crest

Is is related to the noisness/harmonicity of the related signal and to the flatness of the spectrum:

$$F_{Crest} = \frac{maxS(k)}{\frac{1}{K}\sum_k S(k)}. \tag{3.22}$$

Spectral Contrast (mean, peaks, valleys)

The Spectral Contrast is a measure of the relative distribution of the harmonic and inharmonic components. It is computed by segmenting the track into overlapping frames and computing the spectrum, then the signal is filtered with an octave-scale filter that divides it into seven sub-bands. Let consider p-th sub-bands $(a_1^p, (a_2^p, ..., (a_K^p)$, peaks and valleys are defined as:

$$peak = log\frac{1}{\alpha K}\sum_{k=1}^{\alpha K} a_k^p, \tag{3.23}$$

$$valley = log\frac{1}{\alpha K}\sum_{k=1}^{\alpha K} a_{K-k+1}^p, \tag{3.24}$$

respectively, with α being a regulatization parameter.
Finally the Spectral Contrast is defined as:

$$contrast = peak - valley \tag{3.25}$$

Figure 3.8 shows an example of Spectral Contrast extracted from an audio excerpt.

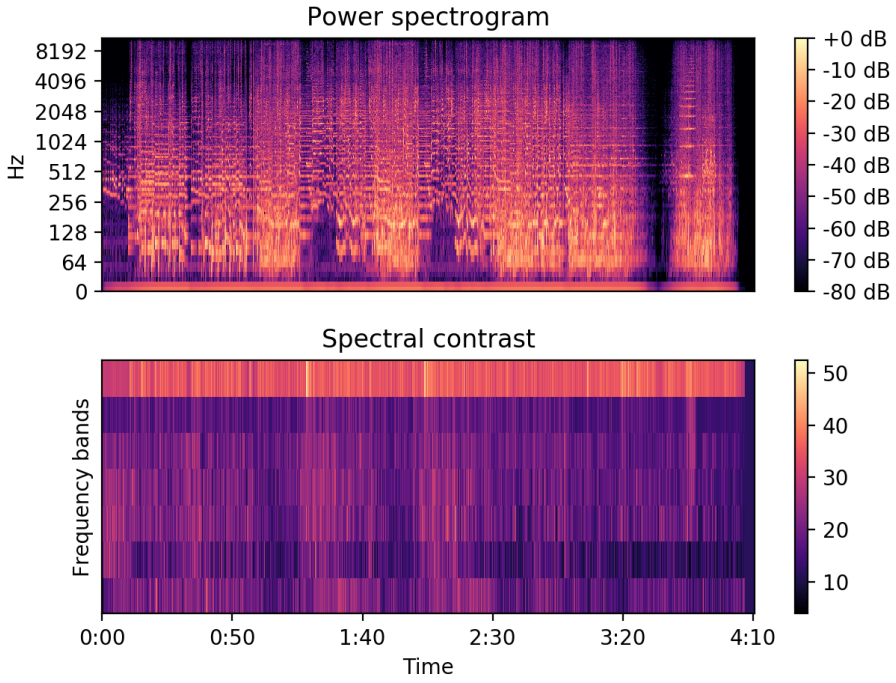


Figure 3.8: Spectral Contrast of the song "Strawberry Fields Forever" by The Beatles.

Spectral Slope

Accounts for the rate of the spectral components that decrease towards higher frequencies. The Spectral Slope is obtained by computing the linear regression of the spectral amplitudes:

$$F_{SL} = \frac{1}{\sum_k S(k)} \frac{K \sum_K f(k) \cdot S(k) - \sum_K f(k) \cdot \sum_K S(k)}{K \sum_K f(k)^2 - (\sum_K f(k))^2}, \tag{3.26}$$

where $f(k)$ is the k-th spectral component relative to the spectral amplitudes $S(k)$.

Spectral Smoothness

It is the difference between adjacent spectral components:

$$F_{SSMO} = \sum_{K} \left| 20logS(k) - \frac{20logS(k-1) + 20logS(k) + 20logS(k+1)}{3} \right|. \tag{3.27}$$

Spectral Variance

It is related to the variance of the spectral amplitudes:

$$F_{SVAR} = \frac{1}{K} \sum_{k} (S(k) - \bar{S})^2. \tag{3.28}$$

Spectral Standard Deviation

The standard deviation of the magnitude spectrum is obtained by taking the square root of the Spectral Variance.

3.2.5 Temporal Features

Zero-Crossing Rate

It is a rough measure of the noisiness of a signal. It is the rate of sign-changes of a signal, i.e. the rate at which the signal changes its value from positive to negative or back. In the field of music retrieval it is useful to classify percussive sounds.

$$F_{ZCR} = \frac{1}{2} \sum_{n=1}^{N-1} |sgn(x(n)) - sgn(x(n)| \frac{F_s}{N}. \tag{3.29}$$

F_s is the sample rate and N the number of samples of the signal $x(n)$.

3.2.6 Waveform Features

Average Deviation

It is the mean of the absolute deviation of each sample of the signal from the samples mean:

$$avgdev = \frac{\sum_{n=1}^{N} |x(n) - \bar{x}|}{N}, \tag{3.30}$$

where N represents the number of samples in the frame, x_n is the n-th sample and $\bar{x}$ is the mean value of the samples.

Kurtosis

The definition is equal to the Spectral Kurtosis but in this case it is applied to the samples of each frame.

Variance

The definition is equal to the Spectral Variance but in this case it is applied to the samples of each frame.

Skewness

The definition is equal to the Spectral Skewness but in this case it is applied to the samples of each frame.

3.3 Machine Learning Methods

Machine learning methods are widely used in the field of statistics, data mining and artificial intelligence and their main purpose is learning from data. The typical scenario is the one in which we have some quantitative measurements (or categorical) that we wish to predict using a set of *features*. We have a *training set* data which is used to build prediction models. We use these models (or *learners*) to predict the outcome of new unseen objects using a *test set*. A good model is a model that is able to predict accurately such an outcome. This type of approach is called *supervised learning* since we use a *training set* to guide the learning process, opposed to the *unsupervised learning*, in which we are given only the features without any measurements of the outcome. Types of supervised learning are classification and regression problems, an example of unsupervised learning is clustering.
In Figure 3.9 and 3.10 we show the block diagrams for a generic supervised and unsupervised machine learning method respectively.

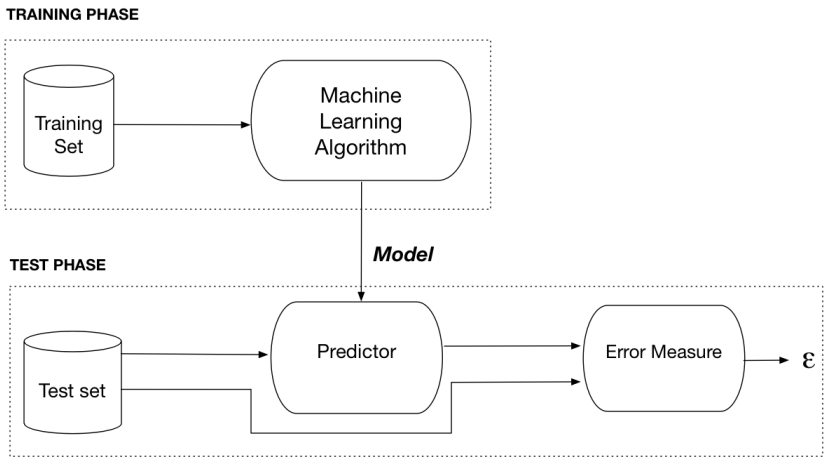


Figure 3.9: Block diagram of training and testing phase for a generic supervised machine learning problem.

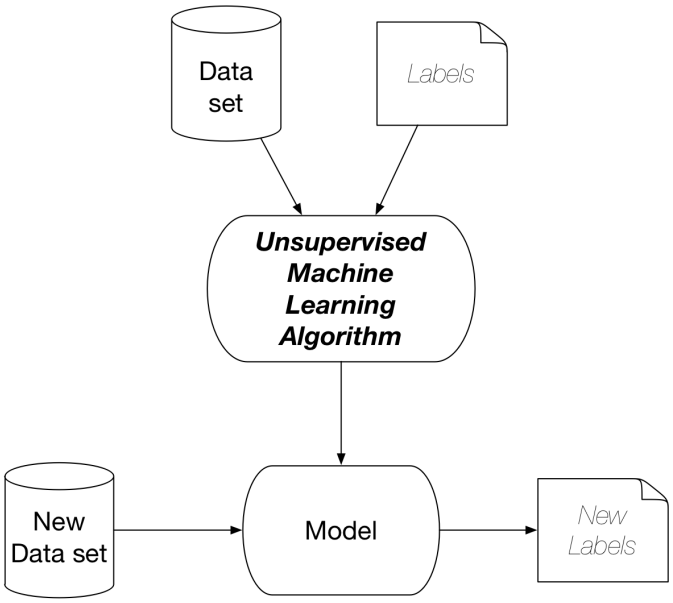


Figure 3.10: Block diagram of a generic unsupervised machine learning problem.

3.3.1 Classification

Classification is the problem of categorizing new observations on the basis of models computed using a training set of data containing observations whose categories are known. An example of classification could be the genre classification in which we try to classify between, for instance, *pop* and *rock* music based on some properties extracted from the audio files under analysis. This type of classification belongs to the category of the binary problems, where there are only two possible classes. An example of multi-class problem is the one in which we consider more than two genres (classes) as possible outputs to the system.

Support Vector Machine

SVMs are among the most used techniques for classification. Given a set of training examples and known associated categories, an SVM training algorithm builds a model for assigning new observations to classes. The searching for the right model consists in constructing the optimal hyperplane so that the examples beloging to different classes are divided by a gap

that is the widest as possible. New observation examples are then mapped into the feature space and, depending on their position with respect to the hyperplane, are classified accordingly.
In case of linear SVM, the hyperplane is defined as:

$$g(x) = w^T x + \omega_0, \tag{3.31}$$

where w is the direction of the hyperplane and ω_0 is its position.
The training phase of an SVM consists in finding the hyperplane for which

$$\begin{aligned} w^T x + \omega_0 &> 1, \quad \forall x \in \omega_1 \\ w^T x + \omega_0 &< -1, \quad \forall x \in \omega_1 \end{aligned} \tag{3.32}$$

Figure 3.11 shows a graphical example in the case of linear and non linear SVM.

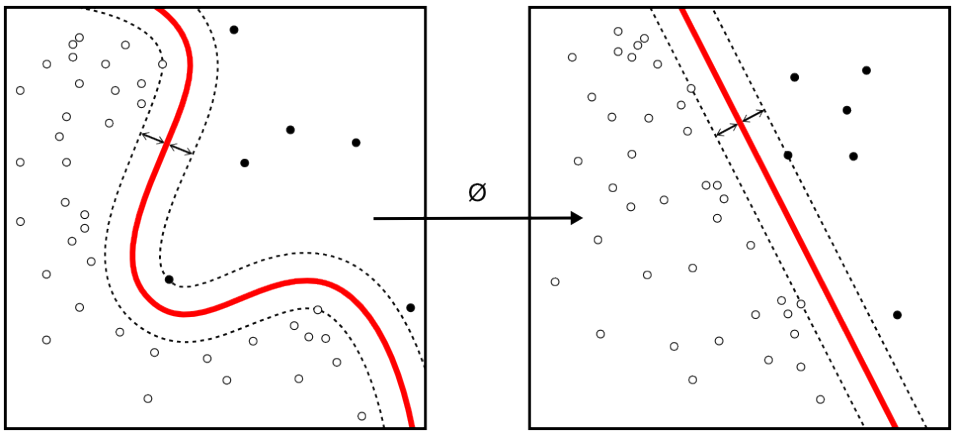


Figure 3.11: Example of linear (right-hand side) and non-linear (left-hand side) SVM.

3.3.2 Regression Models

Given a training set composed by N pairs of inputs and outputs:

$$(x_i, y_i), i \in \{1, ..., N\} \tag{3.33}$$

where x_i is a $1 \times P$ feature vector and y_i is the real value to predict, a regressor $r(\cdot)$ is a function that minimizes the error ε between the expected and the predicted values. A typical measure for the prediction error is the mean squared error (MSE).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.34}$$

Regression analysis is an example of supervised learning and it is used for prediction and forecasting.
A regressor is estimated during two steps:

- *training phase*, where a training set is used to estimate the regression function.
- *testing phase*, where a test set is used to estimate the regression preformances.

Linear Regression

Consider the case in which we have an input vector $X^T = (X_1, X_2, ..., X_p)$ and want to predict a real-valued output Y. The linear regression model takes the form:

$$Y = f(X) = \beta_0 + \sum_{j=1}^{p} X_j\beta_j, \tag{3.35}$$

where the β_j's are unknown coefficients and X_j is the input.
A linear model assumes that the regression function is linear or that the linear model is a reasonable approximation. The parameters β are estimated using a set of training data $(x_1, y_1)...(x_N, y_N)$ and the most popular method for the estimation of such parameters is the *least squares* method, in which the coefficients $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ are chosen in order that the following

cost function is minimized:

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{N} x_{ij}\beta_j)^2 \tag{3.36}$$

Polynomial Regression

Polynomial Regression is considered to be a special case of a linear regression problem. It is a type of regression analysis in which the relationship between the independent variable X and the dependent variable Y is modelled as an n-th degree polynomial

$$Y = f(X) = \beta_0 + \sum_{j=1,n=1}^{p} X_j^n \beta_j. \tag{3.37}$$

Ridge Regression

In ridge regression the coefficients are shrinked by modifying equation 3.35 by adding a new penalty term that controls the amount of shrinkage in order to avoid building a model with high variance. The ridge coefficients minimize the penalized residual sum of squares

$$\hat{\beta}^{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{N} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{3.38}$$

Here $\lambda \geq 0$ is the parameter that controls the amount of shrinkage: the larger λ, the greater the amount of shrinkage.
Ridge regression is employed in cases in which there are many correlated variables in a linear regression model and the coefficients can become poorly determined and the model exhibits high variance. For that reason, it may happen that a large positive coefficient on one variable can be canceled by a similar large and negative coefficient. By imposing the size constraint this problem is minimized.

Support Vector Regression (SVR)

Support Vectors can be applied also to regression problems, in this case we refer to them as Support Vector Regression (SVR) problems. SVR has been

sucessfully applied in the field of forecasting and time series prediction. This method offers many degrees of freedom during the modelling process, since it admits the selection of kernel functions together with their parameters. The standard Support Vector Regression (SVR) algorithm uses the ϵ-insensitive loss function as proposed in [41]. This type of function allows a tolerance to errors not greater than ϵ.

Let consider a number of training data points $\{(x_1, y_1), ..., (x_l, y_l)\}$ from which we want to construct the regression model. The SVR algorithm applies a transformation to the original data points from the initial Input Space Φ to an higher-dimensional Feature Space F in which we construct a linear model that corresponds to a non-linear model into the original space Φ.

$$\Phi : R^n \rightarrow F, w \in F \tag{3.39}$$

$$f(x) = \langle w, \Phi(x) \rangle + b \tag{3.40}$$

The function f that we are trying to find in order to fit to the data must have a deviation less than ϵ and have to be as flat as possible. That means we seek for a small weight vector w. One way to do this is to minimize the quadratic norm of the vector w.

minimize

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) \tag{3.41}$$

subject to:

$$\begin{aligned} y_i - \langle w, \Phi(x_i) \rangle - b &\leq \epsilon + \xi_i \\ \langle w, \Phi(x_i) \rangle + b - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, 2, ..., l \end{aligned} \tag{3.42}$$

In 3.41 parameter C accounts for the trade-off between generalization capacity and accuracy in the trainin data, while parameter ϵ defines the tolerance to errors.
In Figure 3.12 the situation is depicted graphically. We can see that the points outside the shaded region contribute to the cost and the deviations are linearly penalized.

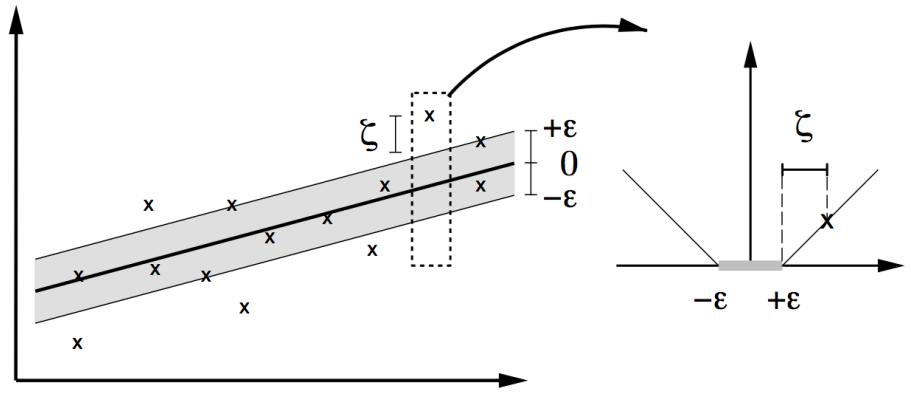


Figure 3.12: Graphical interpretation of a linear SV machine.

We present in the following equations a more convenient representation of the problem stated above.

$$w = \sum_{i=1}^{l} (\alpha - \alpha_i^*) \Phi(x_i) \tag{3.43}$$

$$f(x) = \sum_{i=1}^{l} (\alpha - \alpha_i^*) K(x_i, x) - b \tag{3.44}$$

In 3.44 α_i and α_i^* are the dual variables, and the expression $K(x_i, x)$ represents the inner product between $\Phi(x_i)$ and $\Phi(x)$, which is known as to be the kernel function. From this function we obtain a solution for the original regression problem.
In Table 3.2 the most used kernel functions for SVR are presented, together with their mathematical expressions.

Where σ denotes the kernel width.

Type of kernel	Definition
Polynomial	$k(x, x^{'}) = (x^T x^{'} + c)^d$
Gaussian	$k(x, x^{'}) = exp(-\frac{\|x-x\|^2}{2\sigma^2})$

Table 3.2: Caption for the table.

3.3.3 Clustering

Clustering is the most important non supervised learning problem. Given a set of unlabeled data, clustering algorithms try to find a structure within data in order to group the one that are similar to each other. If two or more objects are similar it means that they belong to the same class, conversely, if they are not similar, it means they belong to different classes.

K-Means Clustering

One of the simplest unsupervised algorithm is the K-Means Clustering. Let consider a set of clusters, the idea of K-Means is to find k centroids, one for each cluster. Centroids need to be placed away from each other, then we take each point in the data set and associate it to the nearest centroid. When this process finishes the centroids are recomputed and the data points are re-assigned to their closed centroid. This process is iterated until no more changes can be done, since centroids can not modify their position anymore. Figure 3.13 shows an example of a K-Means clustering algorithm in 6 steps.

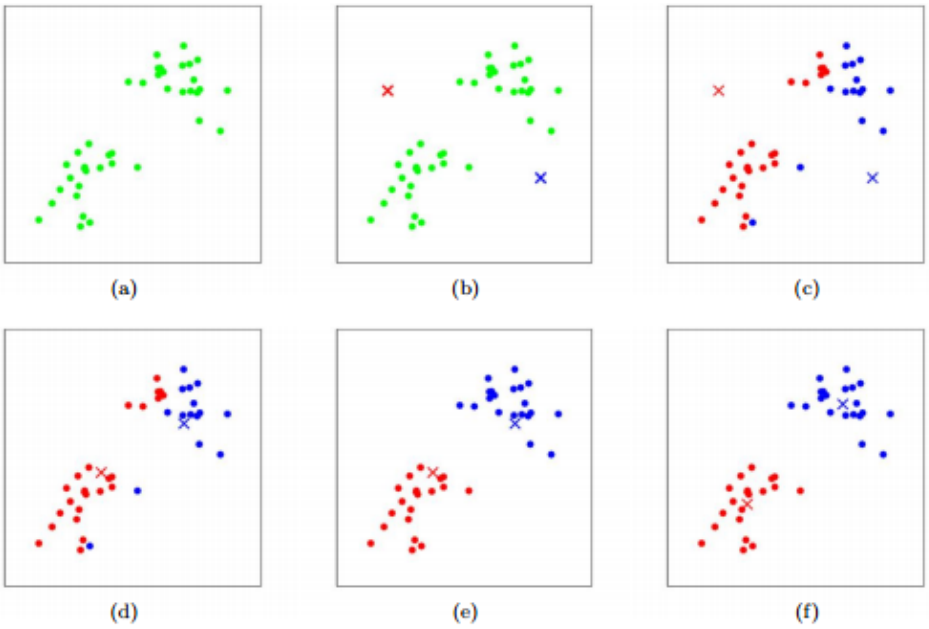


Figure 3.13: Example of a K-Means clustering algorithm.

Chapter 4

Method Overview

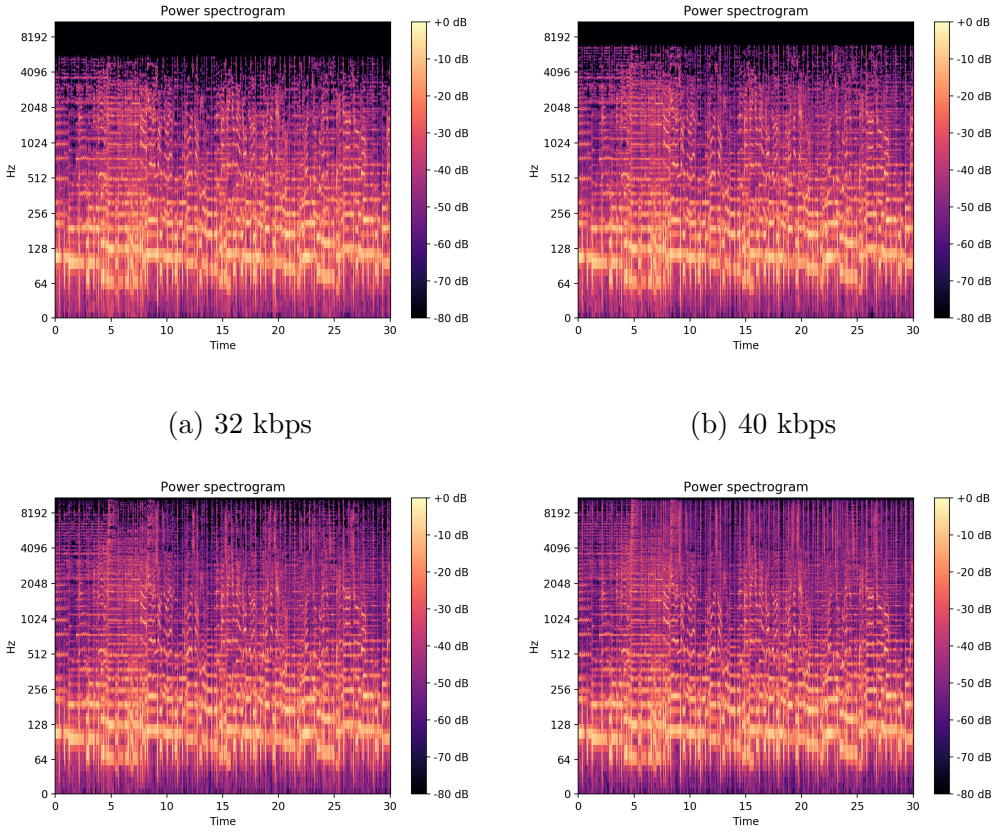
In this chapter we examine the effects caused by lossy audio compression on audio features value. In the last section we provide a detailed description of all the algorithms used for audio features compensation and the task of Music Emotion Recognition is presented.

4.1 The effect of lossy compression on audio features

In Chapter 3 we presented an overview on data compression. We discussed, earlier in Chapter 2, about the key role and importance of feature extraction for Music Information Retreival's applications. The goal of this thesis is the compensation of audio features extracted from lossy-encoded audio files. The audio compression algorithm used is the MP3 (also known as MPEG-1 or MPEG-2 Audio Layer III). MP3 has become very a popular coding format for digital audio. It is a type of lossy compression that works by reducing the accuracy of certain portion of a continuous sound, by exploiting perceptual coding and psychoacoustical models.

The feature compensation method is employed in order to improve the accuracy and performances of final music information retrieval tasks, such as genre or mood classification and clustering. We focused on MP3 lossy compression and, in particular, on the impact that the encoding at various

bit rate values have on the extracted audio features. We show the effect of lossy audio compression in figures 4.1 and 4.2, that represent the STFT extracted from the same 30 seconds musical excerpt encoded at different bit rate values. The STFT is a very important feature because it is the starting point for the computation of a great number of other Spectral features.



(a) 32 kbps

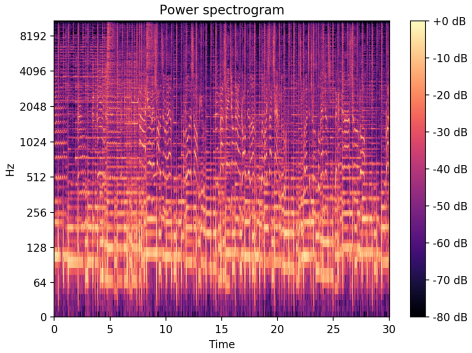
(b) 40 kbps

(c) 64 kbps

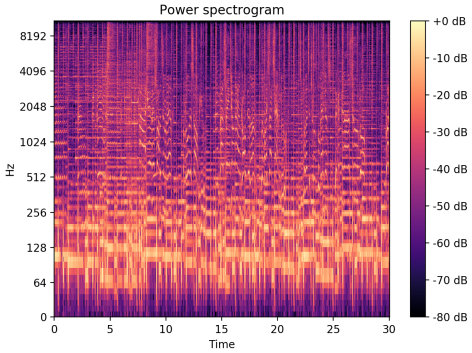
(d) 96 kbps

Figure 4.1: STFT extracted from 30 seconds of the musical excerpt "Song for Bob Dylan" by David Bowie encoded at various bit rate values

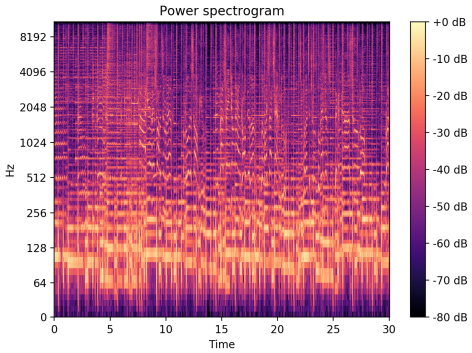
The effect that coding bit rate produces on the Short-Time-Fourier-Transform (STFT) is widely visible. In fact, when the coding bit rate is very low (Figure 4.1 (a)), the STFT amount of spectral content results very poor in the high frequency range.



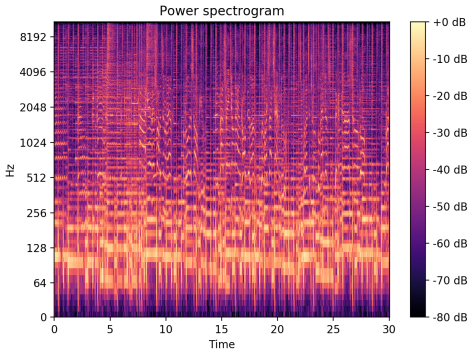
(c) 112 kbps



(d) 160 kbps



(c) 224 kbps



(d) 320 kbps

Figure 4.2: STFT extracted from 30 seconds of the musical excerpt "Song for Bob Dylan" by David Bowie encoded at various bit rate values

The effect of lossy compression is visible as it aims at approximating or discarding useless informations that are considered not audible by the human ear by exploiting psichoacoustical models of sound perception. As a consequence of that, it seems clear that the same feature extracted from the same audio excerpt encoded at different bit rate values will not assumes the

same value. We say that feature values are bit rate dependent.

In Figure 4.3 and 4.4 we see the feature-bit rate dependence more clearly by showing the values of Spectral Centroid and Sharpness features respectively, extracted from 3 songs (song 1, 2 and 5 from CAL500) encoded at different bit rate values. As shown in the figures, feature values are bit rate dependent. Due to lossy audio compression the same feature, extracted from a song encoded at different bit rate values, assumes always diverse values. This feature-bit rate dependence is highlighted in the figures as an interpolated spline of the various feature values.

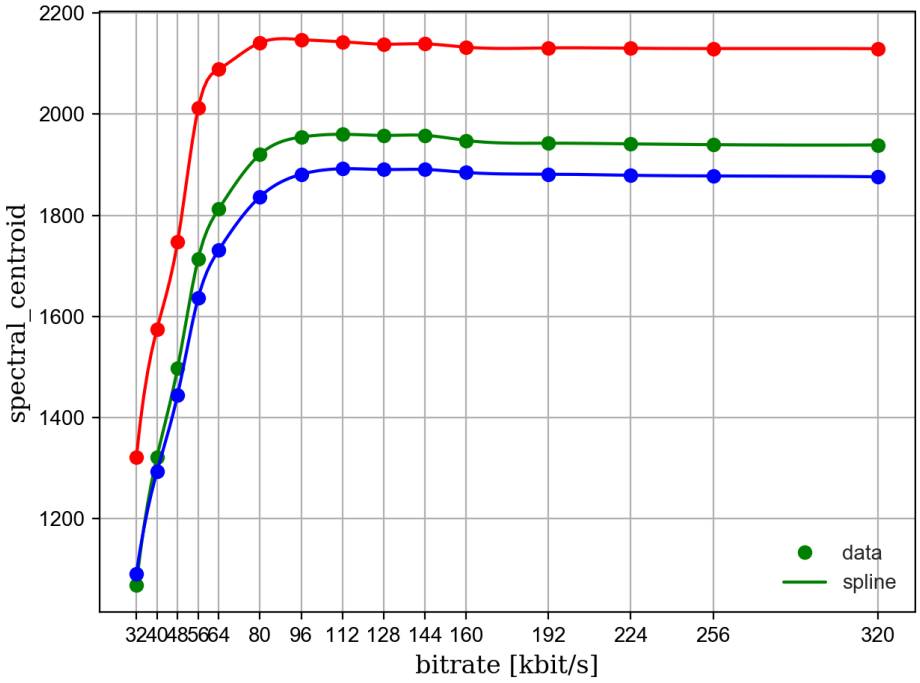


Figure 4.3: Behavior of feature Spectral Centroid extracted from 30 seconds excerpts of songs 1 (green), 2 (red), and 5 (blue) of CAL500 dataset coded at different bitrates. Dots represent measured values, lines are spline interpolations.

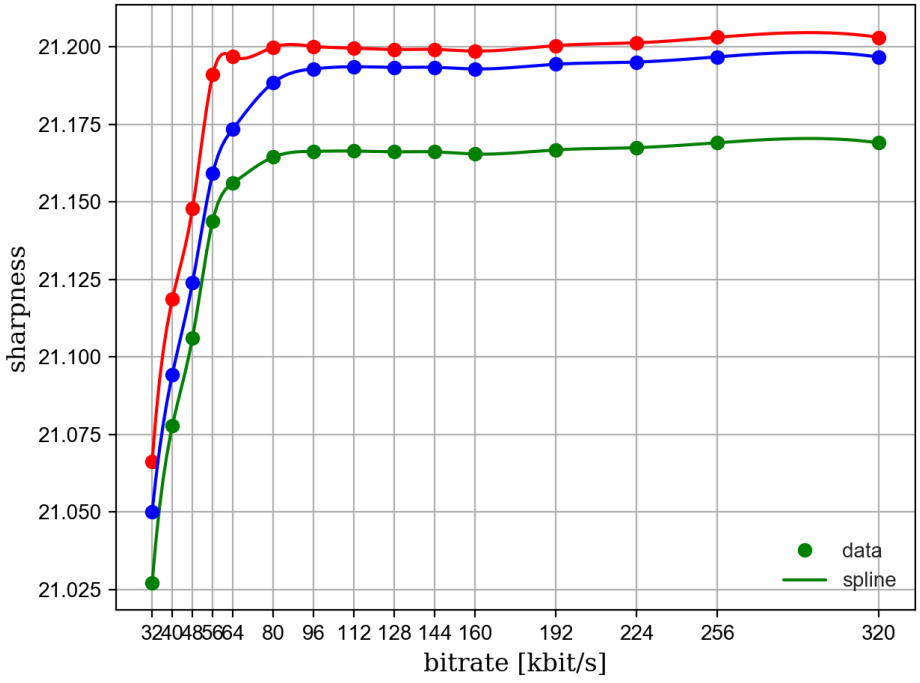


Figure 4.4: Behavior of feature Sharpness extracted from 30 seconds excerpts of songs 1 (green), 2 (red), and 5 (blue) of CAL500 dataset coded at different bitrates. Dots represent measured values, lines are spline interpolations.

4.2 Data visualization/distribution of the training set

The following section is devoted to the data visualization and distribution of the training set. This type of analysis is presented in order to prove some of the choices that have been made on models, algorithms and machine learning methods.

In Figures 4.5 and 4.6, the scatter matrices for features Spectral Centroid and MFCC 17 are being shown respectively. Each sub matrix is the plot of each pair of bitrates, on the x-axis there is the starting bitrate b_o, on the y-axis we have the target bit rate b_t. We noticed that the distribution of the data is very much linear, expecially where bitrates gets closer to the highest bit rate value of $320kbps$. If the distribution is linear it means that it would be sufficient to use linear regression methods in order to approach the trend

of the training dataset. We will show this trend in Section 5.

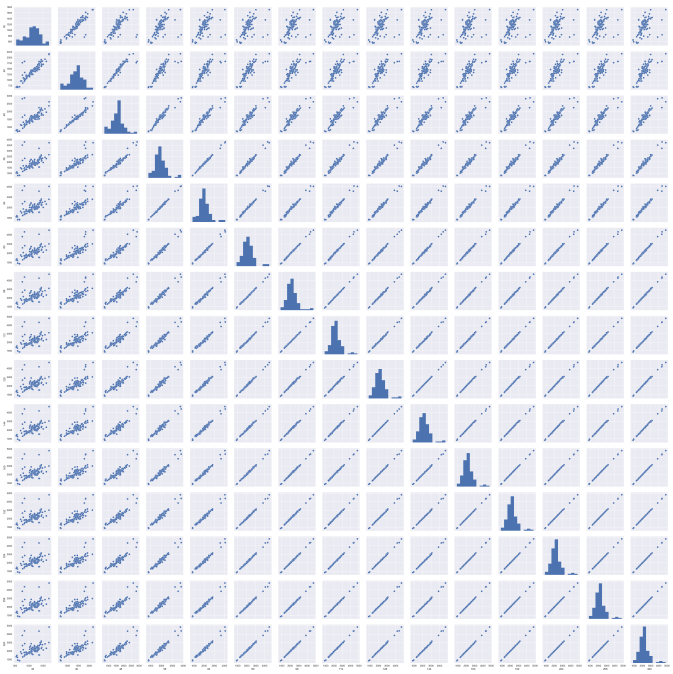


Figure 4.5: Scatter matrix for the feature Spectral Centroid.

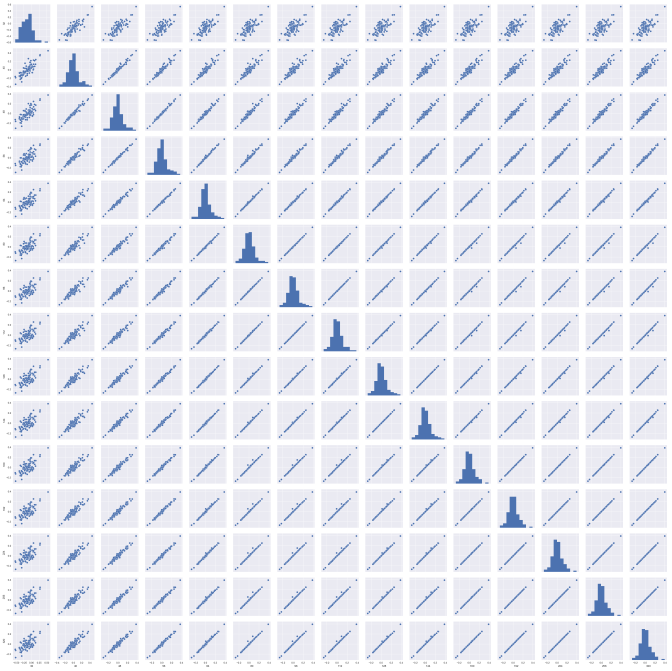


Figure 4.6: Scatter matrix for feature MFCC 17.

4.3 System overview

The feature compensation algorithm aims to predict a feature value obtained from a song we have compressed at an original bitrate b_0, as if it was compressed at a target bitrate $b_t \neq b_0$. However, it easy to realize that the case $b_t < b_0$ is trivial, since it is sufficient to re-compress the song at target bitrate b_t before feature extraction. Conversely, if $b_t > b_0$, recompressing the audio file at bitrate b_t is useless, since lossy audio compression is an irreversible process. The audio file at b_0 contains less informations compared to the one encoded at b_t and in this case the re-compression results not to be feasible and another strategy must be adopted. In this section we outline the proposed strategy to cope with the feature compensation problem in the challenging scenario of $b_t > b_0$. In particular, the bit rate values used are:

$$bitrates_{(kbps)} = [32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160, 192, 224, 256, 320]. \tag{4.1}$$

In Figure 4.7 we show the general schema of the feature compensation process proposed in this thesis work.

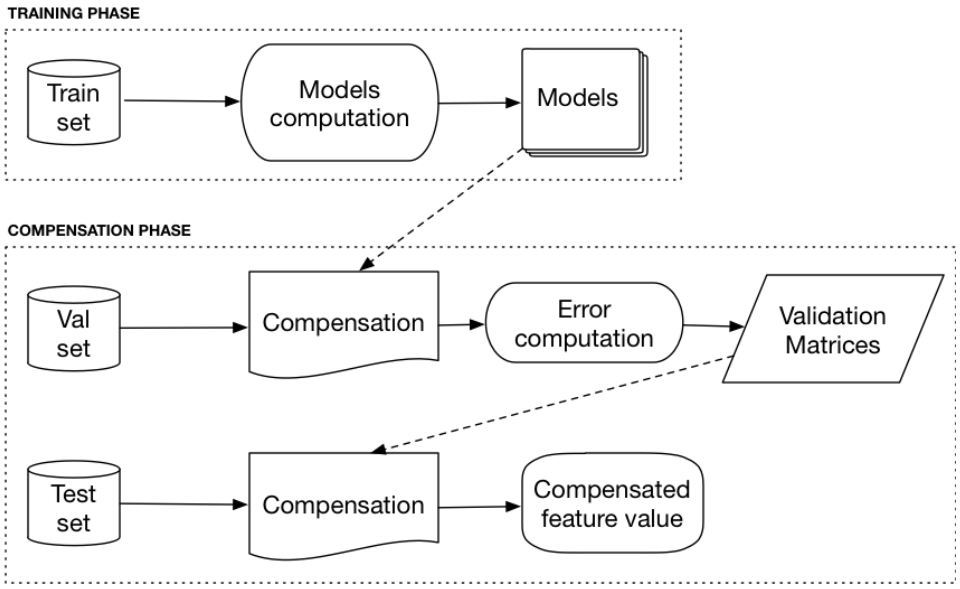


Figure 4.7: Block diagram of the feature compensation process.

The feature compensation process is composed by two main phases:

- **Training phase**: during this phase models are computed using a training data set.
- **Compensation phase**: the computed models are used for the feature compensation. This phase is divided in two more phases:
  - *Validation phase*: a validation set is used and the feature compensation is performed using the computed models. An error measure is computed in order to keep trace of the amount of compensation ability for every bit rate pairs (b_0, b_t) and for every feature f in order to further compensate only features that really take advantage from the compensation.
  - *Test phase*: a test set is employed in this phase and for every tuple (f, b_0, b_t) a check is done on the compensation ability by looking at the error values stored during the validation phase. If the considered value is above a certain threshold, the compensation can be performed.

We employed two main techniques for the training phase, that are:

- **Rule-based audio features compensation**: for each feature a set of models is computed. This set of models is the set of all the compensation curves fitted on features-bitrates values. Each song in the data set is encoded a certain number of times using different bit rate values. Features values are extracted from the same song at different bit rate values. Those feature values are fitted using a spline interpolating curve. A curve is computed for each song in the training set. The set of curves computed for a specific feature is the trained model (Fig. 4.3 and 4.4). During the compensation phase, for each tuple (f, b_0, b_t), the new feature value at b_0 is projected into the plane containing all the curves that correspond to the feature f. The closest curve to the new feature value at b_0 is selected and the compensated feature at b_t becomes the feature value at b_t that belongs to the selected curve.
  This method can be improved by using machine learning techniques

in order to account for a more accurate possible dependence between bit rates.

- **Machine learning based audio features compensation**: in this method, unlike the rule-based in which we build one model for each feature, we compute a model for each feature f and for each bitrate pair (b_0, b_t) in order to consider possible dependencies between the pair (b_0, b_t). Validation and test phase are the same as described in the rule-based approach.

4.4 Rule-based audio features compensation

The *rule-based audio features compensation algorithm* operates by first computing a set of *feature-bitrate models* for each feature, using a *training set*. Those models are the spline interpolations of the feature values considered at different bitrates values. For each feature f we create a model, this model is composed by all the curves fitted on the data of the training set and each song in the training set has its own curve.
Figure 4.8 shows the model of feature *Zero Crossing Rate* composed by songs from 1 to 20 of the CAL500 song dataset.

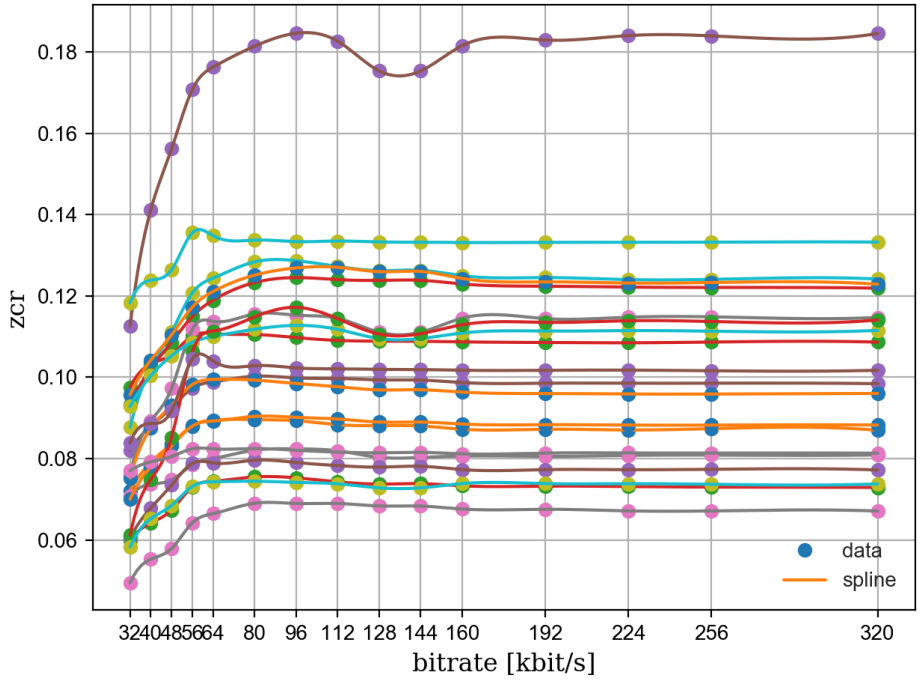


Figure 4.8: Model of feature Zero Crossing Rate composed by songs from 1 to 20 of the CAL500 song dataset.

The compensation algorithm works by considering all the possible combination of bit rate pairs (b_0, b_t) for each feature.
The algorithm takes the feature value of the songs at bit rate b_0 and computes the distances between this value and the various models (interpolated splines). The closest model is selected and the compensated feature assumes the correspondent value at bit rate b_t that lies on the selected spline.
Once the models have been computed for each feature, a different set (*validation set*) is used to validate them by applying the feature compensation algorithm and by computing an error measure. The error is computed in order to measure the compensation ability of the method. The error measure is defined as:

$$|\text{feature error after compensation}| < |\text{feature error without compensation}|\,, \tag{4.2}$$

where the *feature error after compensation* represents the error between the

compensated feature value and the actual feature value at b_t, and the *the feature error without compensation* is the error between the actual feature value at b_t and the feature value at b_0.
Using the errors computed in the *validation phase* we finally test the compensation method on a different set, the *test set*. Using the errors computed in the *validation phase*, we apply a rule on whether the compensation can or can not take place when tested on the *test set*.
All the algorithm's steps are detailed in the following sections.

4.4.1 Feature-bitrate models computation

The first step of the algorithm consists in building a set of feature-bitrate models for every feature.
Let consider:

$$X_{s,f}(b) \quad with \quad b \in B, f \in F, \tag{4.3}$$

where:

- X is the average value of the extracted feature vector.
- f is the considered feature (i.e. Spectral Centroid, Zero-Crossing Rate etc..).
- s is the compressed audio segment from which the feature has been extracted.
- b is the bitrate value.
- B is the set of all possible bitrates (Equation 4.1).
- F is the set of all the extracted features (Table 3.1).

For simplicity we will now focus on describing the process of building a model for a specific fixed feature $\bar{f}$.
Let consider S_{tr}, that is the set of *training songs* and $\bar{s} \in S_{tr}$, which is an audio excerpt. We compute $X_{\bar{s},\bar{f}}(b)$ for the finite set of bitrates $b \in B$.
The feature bitrate model is estimated as the natural spline interpolation of the points $X_{\bar{s},\bar{f}}(b)$ as shown in Figure 4.9.

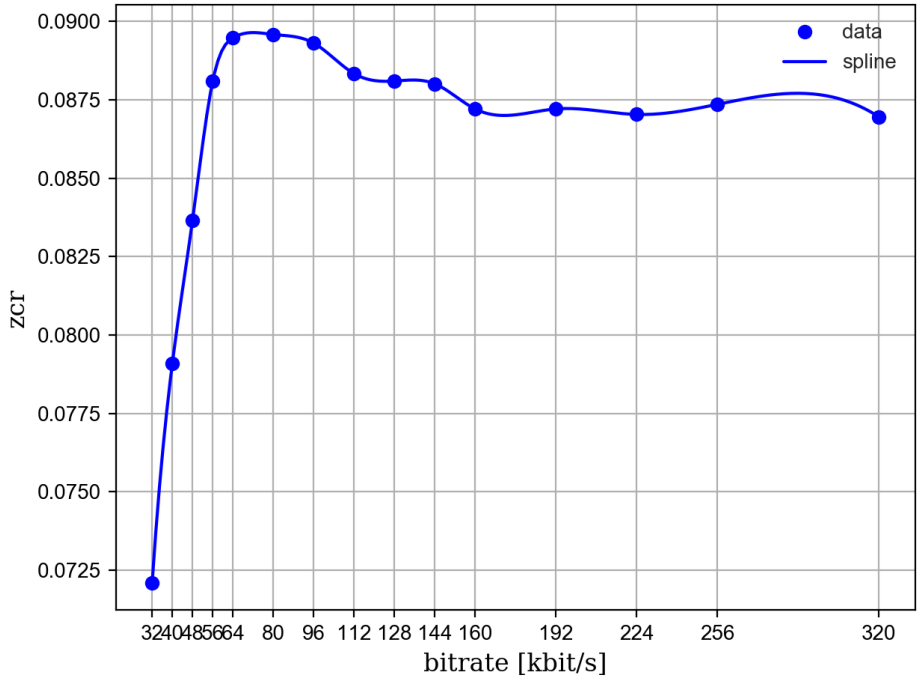


Figure 4.9: Spline Interpolation for feature Zero Crossing Rate and song 1 of the CAL500 song dataset.

The spline interpolation can be formalized as:

$$\begin{aligned} \hat{X}^i_{\bar{s},\bar{f}}(b) &= a^i_{\bar{s}} \cdot b \\ &= a^i_{\bar{s},3} b^3 + a^i_{\bar{s},2} b^2 + a^i_{\bar{s},0}, \\ & b \in [b_i, b_{i+1}], \end{aligned} \tag{4.4}$$

where $\mathbf{a}^i_{\bar{s}} = [a^i_{\bar{s},3}, ..., a^i_{\bar{s},0}]$ is the vector of coefficients describing the polynomial model fitted, $\mathbf{b} = [b^3, b^2, b, 1]$ is the vector of bitrates in which the polynomial is evaluated and $[b_i, b_i + 1]$ determines the bit rate interval (b_0, b_t) At the end of the models computation we store S_{tr} models for each feature.

4.4.2 Feature compensation

After the computation of the models, we perform a validation step in order to determine which triplets (f, b_0, b_t) acqually takes advantage from the

compensation. This step is useful because for some features and for some bit rate pairs, it may happen that non-compensated features are closer to target values more than the compensated ones. This may happen, for instance, in cases in which b_0 and b_t are very close bit rate values. In this case, the compensation results useless. We compare compensated and target features using a validation set S_{val}.

Let consider the case in which we need to compensate feature $\bar{f}$ extracted from excerpt $\hat{s} \in S_{val}$ from bit rate b_0 to bit rate b_t. For doing that we first compute the feature value $X_{\hat{s},\bar{f}}(b_0)$ at the given bit rate, then we select the model (computed in the previous step) that best fits the song under analysis. We select the best model by searcing for the model $a_i^s, s \in S_{tr}$ associated to the polynomial curve whose value at b_0 is the closest to the feature value $X_{\hat{s},\bar{f}}(b_0)$, extracted from the song under analysis. Thus, we compute:

$$\begin{aligned} \hat{s}^* &= \underset{s \in S_{tr}}{argmin}(\left|\hat{X}^i_{s,\bar{f}}(b_0) - \hat{X}^i_{\hat{s},\bar{f}}(b_0)\right|) \\ &= \underset{s \in S_{tr}}{argmin}(\left|a^i_s \cdot b_0 - \hat{X}^i_{\hat{s},\bar{f}}(b_0)\right|), \end{aligned} \tag{4.5}$$

where $b_0 = [b_0^3, b_0^2, b_0, 1]$ and $\hat{s}^*$ is the estimate of the song whose feature-bitrate curve can be used ad a model for $\hat{s}$.
We then compute the feature value at b_t using the estimated model:

$$\hat{X}^i_{\hat{s},\bar{f}}(b_t) = \hat{X}^j_{\hat{s}^*,\bar{f}}(b_t) = a^j_{\hat{s}^*} \cdot b_t, \tag{4.6}$$

where $b_t = [b_t^3, b_t^2, b_t, 1]$.
We then repeat the feature compensation in S_{val} for each tuple (s, f, b_0, b_t) and then we check the condition:

$$\left|\hat{X}_{s,f}(b_t) - X_{s,f}(b_t)\right| < \left|X_{s,f}(b_0) - X_{s,f}(b_t)\right|, \tag{4.7}$$

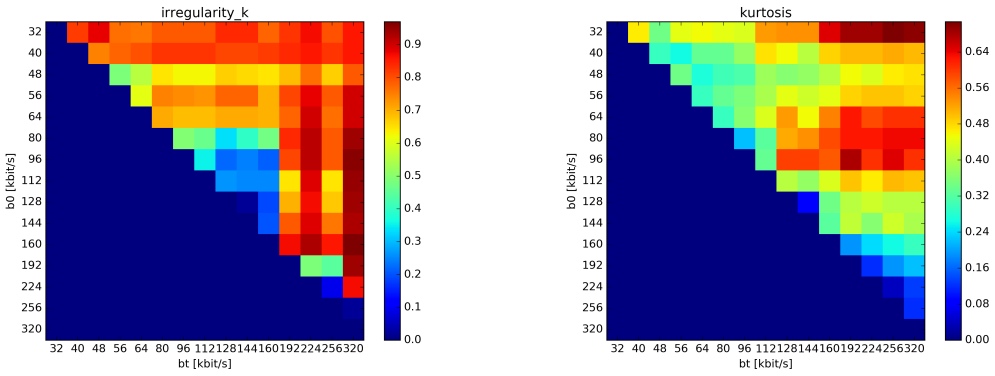
which measure if the feature error after compensation (left-hand side) is less than the error without compensation (right-hand side). Whe then fill a matrix

$$M(s, f, b_0, b_t) = \begin{cases} 1 & \text{if (4.7) is verified,} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

The fraction of songs whose feature f is worth being compensated from bitrate b_0 to b_t is then given by

$$\bar{M}(f, b_0, b_t) = \frac{1}{|S_{val}|} \sum_{s \in S_{val}} M(s, f, b_0, b_t). \tag{4.9}$$

Figures 4.10 (a) and (b), represent validation matrices $\bar{M}$ for feature *Irregularity J* and *Kurtosis* respectively. Each cell of this matrix contains the value of the overall compensation ability for the considered feature and bit rate pair. Values can range from 0 to 1.
Note that the lower triangular part of the matrices (i.e. the case $b_t \leq b_0$) has not been considered since the relevant scenario is that of $b_t > b_0$



(a) (b)

Figure 4.10: Validation matrix for feature Irregularity k and Kurtosis.

After the validation step, the feature compensation algorithm is applied on the test set S_{test}.
Given a feature f, extracted from a song $\in S_{test}$ at bit rate b_0, and given a target bit rate b_t, we compensate the feature by applying (4.6) only if the *worthy condition* is verified

$$\bar{M}(f, b_0, b_t) > th, \tag{4.10}$$

where *th* is a *threshold* equal to 0.5 in our experiments.
Equantion (4.10) indicates if a tuple (f, b_0, b_t) is worth to be compensated, based on the analysis on the validation set.
Figure 4.11 shows an example of the *rule-based feature compensation* algorithm in the case of $b_0 = 96$, $b_t = 320$.

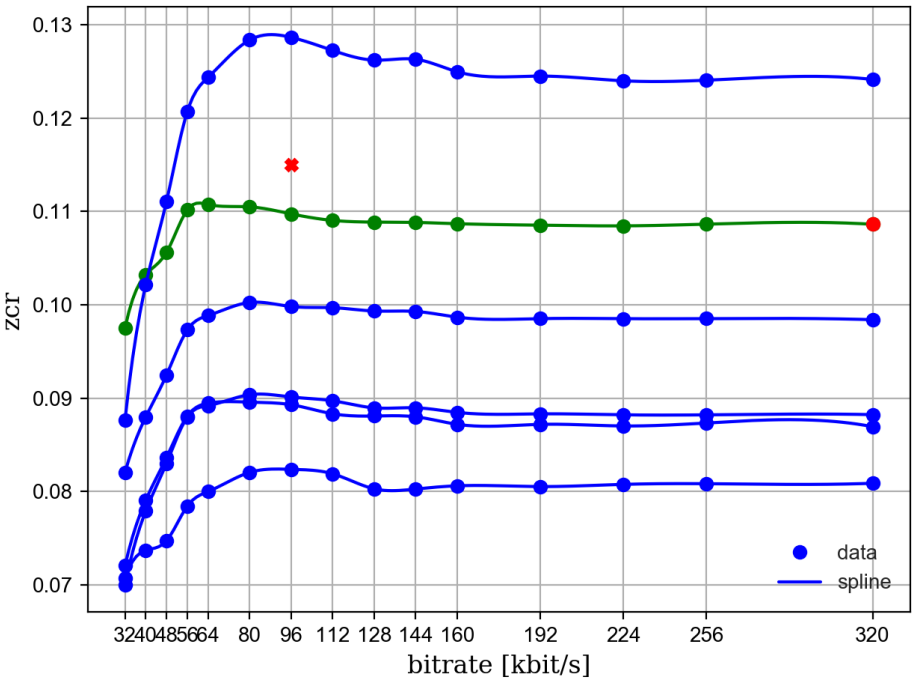


Figure 4.11: Rule-based feature compensation example.

In Figure 4.11, Splines are the different models, the red cross represents the feature value at bit rate b_0 of the song we are trying to compensate. The closest model is the one in green and the compensated feature value at bit rate b_t is shown as a red circle.
The rule-based audio feature compensation method compensates feature values based on a simple rule on the choice of the best model, which is to select the closest model to the feature values at b_0 and assign as compensated feature the feature value at b_t belonging to the chosen model. Another

problem is that models are too poorly constructed since there is too less models with respect to the possible bit rate combination pairs ($b_0.b_t$), for that reason we try building new models using other methods, that are machine learning-based and treat the feature compensation problem as a regression problem.

4.5 Machine Learning based audio feature compensation

The use of machine learning based approaches is needed in order to construct more reliable bit rate models. In particular, unlike the rule-based audio feature compensation method, in this approaches, we construct a model for each tuple $(f, b_0, b_t) \in S_{train}$ and in the compensation step we make predictions using the computed model for each tuple $(f, b_0, b_t) \in S_{test}$.
Also in this case we compute *validation matrices* during the *validation phase* in order to compensate feature values only when it is worthed. We tested the following machine learning methods for feature compensation:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Support Vector Regression (SVR): using Linear, Non-Linear (Radial Basis Function) and Polynomial kernels.

The steps for the machine learning approach are similar to the ones shown in the previous sections:

- *Models Computation*: The train set S_{tr} is employed for the *learning* step in order to find a model for each feature f, for each bit rate pair (b_0, b_t). The best model has been chosen using a grid search method for parameters selection.
- *Validation*: Using the validation set S_{val}, the models computed in the previous step have been tested for each feature f, for each song $s_{val} \in S_{val}$ and for each bit rate pair (b_0, b_t). The amount of correctly compensated features have been calculated by using (4.7) as error measure.
- *Test*: Computed models have been used to perform *prediction* on the test set, S_{test}. Before proceeding with the compensation, a check on the validation matrices is done in order to compensate only the tuples

(f, b_0, b_t) that really take advantage of the compensation, i.e. the ones that follow the worthy condition (4.10).

Features are compensated differently depending on the method adopted, thus, using only one method results in poor compensation ability for a certain number of bit rate pairs and features. In order to overcome this problem we propose the *Fusion method* which is a method that merges all the proposed models and finds, using the validation matrices, the best model in order to perform the feature compensation for each tuple (f, b_0, b_t).

4.6 Fusion Method

The Fusion method is a mixed technique for audio features compensation. With this method we aim at employing only one of the model presented in the previous sections for each tuple (f, b_0, b_t), by choosing among all the avaliable ones.
The Fusion method does not have a *model computation* or *validation* phase, since it selects among the models already computed with the *rule-based* and *machine learning based* audio features compensation methods and using the corresponding validation matrices.
Figure 4.12 shows the block diagram of the Fusion method.

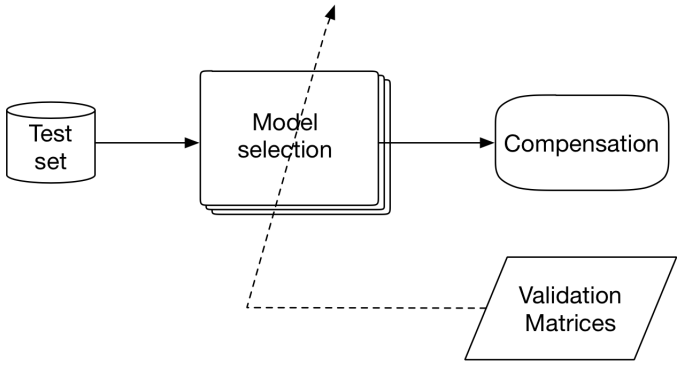


Figure 4.12: Block diagram of the Fusion method.

For each tuple (f, b_0, b_t) we select the best model $m \in M$, where M is the set of all possible models. The best model m is defined as the one that scored the highest compensation ability during the validation phase of the *rule-based* and *machine learning based* audio features compensation approaches. In order words, for each tuple (f, b_0, b_t) we first perform a check on all the validation matrices of all the models corresponding to the feature f, reading the value stored in the cell (b_0, b_t). All the values are stored so that we end up having M values as M are all the possible models. We select the maximum value and compensate the tuple (f, b_0, b_t) with the corresponding model if and only if this value is above a certain threshold (0.5 in our experiments).

4.7 Music Emotion Recognition (MER)

The effectiveness of the feature compensation method has been tested in an automatic Music Emotion Recognition (MER) application. Music Emotion Recognition is among the most popular MIR applications, it is about classification and tagging of music according to human emotions. MER is based on the analisys of audio features extracted from the audio files under analysis, thus, testing its performances in case of a *non-homogeneus* data sets, i.e. song collections made of songs encoded at different bit rate values and testing its performances in the case of a compensated data set, can provide an idea of the effectiveness of the feature compensation in the MIR field.
We confronted two main problems:

- *Binary problem*: by selecting two pairs of affective terms from the CAL500 annotations (*happy-sad, arousing-calming*) as described in [34].

- *Multi-class problem*: by selecting four classes corresponding to the excerpts annotated by the following four affective terms: *Emotion-Happy, Emotion-Angry, Emotion-Calming and Emotion-Emotional.*

Two set of features have been used: *MFCCs*, which are very used for the purpose of MER, and features selected from the whole features set (Table 3.1) through a ReliefF [29] technique, an algorithm that performs feature selection (i.e. it selects a subset of features that are very discriminative for that specific application, among all the available features).

Both *Supervised* (i.e. using a set of labelled examples for training the classifier) and *Unsupervised* (i.e. using unlabelled examples and thus skipping the training phase) techniques have been tested.

Supervised MER

In Figure 4.13 the block diagram for a generic supervised MER problem is depicted.

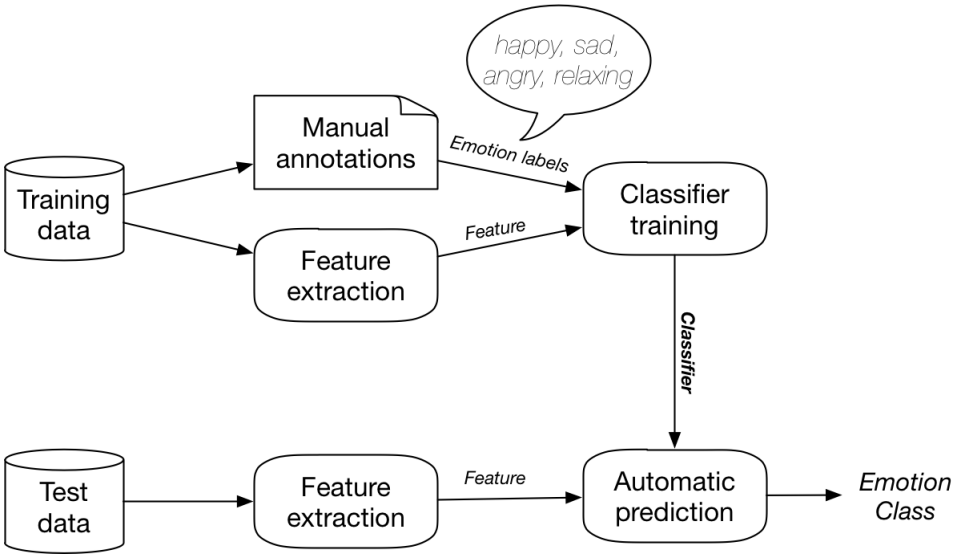


Figure 4.13: Music Emotion Recognition (MER) block diagram.

In general, a supervised MER task is composed by the following steps:

- *Training phase*: The training data set is composed of annotations and audio descriptor extracted from the audio file. In this phase, a classifier is trained using audio features vectors and annotations (emotions) as labels/classes.

- *Test phase*: The classifier is then fed with unseen examples (i.e. feature vectors) that are automatically labelled.

The machine learning techniques used for this classification task are: *Support Vector Machine (SVM)* and *K-Nearest Neighbors (KNN)*, which represent some of the most used techniques for automatic annotation [25]. The data set was divided into training and test set. The test set was selected to be composed of about the 25% of the whole dataset. A 3-fold cross validation has been used in order to retrieve the optimal parameters space

for each model.

The model was trained using the feature vectors containing the feature values (of MFCCs and features selected by the ReliefF algorithm) at bit rate b_t (X_{target}) of all the songs in the training set, and tested on:

- X_{target}: the feature vectors containing the feature values at bit rate b_t.
- X_{source}: the feature vectors containing the feature values at bit rate b_0.
- X_{compM}: the feature vectors containing the compensated feature values using the worthy condition (4.10)
- X_{comp}: the feature vectors containing the compensated feature values without using the worthy condition.

Unsupervised MER

The two aforementioned problems, binary and multi-class, have been also tested in an unsupervised machine learning scenario.
The algorithm used for this problem is the K-Means. We tested the ability of the clustering algorithm to generate separate clusters. A description of the K-Means algorithm steps can be found in Chapter 3.

Chapter 5

Experimental Results

In this section we present the experimental setup together with the test results of the various feature compensation methods described in Chapter 4 and provide an analysis of the effect of the feature compensation in real applications by performing Music Emotion Recogniton classification and clustering, whose results are shown in section 5.4.

5.1 Experimental Setup

In this section we provide a description of the data set employed for the purpose of feature compensation and Music Emotion Recognition. Moreover we provide a description of the programming language, libraries and tools used.

5.1.1 Dataset description and manipulation

The Computer Audition Lab 500-Song (CAL500) data set was first proposed in [36] and it is composed of 500 popular music songs, ranging from a large variery of musical genres. The original format of all songs in the dataset is FLAC (lossless format).
For each song in the data set, we first performed the extraction of 30 seconds for each excerpt. Then every song has been encoded at multiple bitrate values, using the MP3 (MPEG-1 or MPEG-2 Audio Layer III) as audio codec.

The *training set* was composed of 200 songs, the *validation set* was composed of 100 songs and the remaining 200 form the *test set*.
The bitrate values used can be found in equation 4.1.

We then performed the extraction of all the audio features already listed in Chapter 3, Section 3.1, Table 3.1, and retained only the *mean value* of each feature vector for each excerpt.

5.1.2 Programming language, libraries and tools

- *Programming language*: Python[1]
- *Audio encoding library*: FFmpy[2], a Python package that allows the usage of FFmpeg[3] on Python.
- *Audio analysis library*: Librosa[4].
- *Feature exctraction tools*: Vamp Plugins[5] have been used in the features extraction step. In particular we used: Libxtract[6], Essentia[7], BBC Vamp Plugin[8], Queen Mary Vamp Plugins[9].
- *Machine learning library*: Scikit-Learn[10], an open source Python tool for data analysis and data mining.

[1] https://www.python.org/
[2] https://pypi.python.org/pypi/ffmpy
[3] https://ffmpeg.org/
[4] http://librosa.github.io/librosa/
[5] http://vamp-plugins.org/
[6] http://libxtract.sourceforge.net/
[7] http://essentia.upf.edu/documentation/
[8] https://github.com/bbc/bbc-vamp-plugins
[9] http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html
[10] http://scikit-learn.org/

5.2 The feature compensation method: an overview

In the previous chapters we showed the effects that lossy audio compression has on features values. In particular, we showed that the feature value of a song encoded at different bit rate values assumes different values depending on the compression bit rate used. Since audio features are crucial in a great number of Music Information Retrieval application, and given the fact that the majority of audio collections are composed of songs encoded at different bit rate values or with different encoders or encoding paramenters (we identified these types of collections as being *inhomogeneous* collections), solutions to the problem of having variable feature values depending on encoding parameters must be adopted. A solution to this problem is the one proposed in this thesis and it is about compensating the feature values. We recall the various steps for the feature compensation method:

- *Models Computation*: This is the *learning* phase, in which, for each tuple (f, b_0, b_t), a model is computed.
- *Compensation*: The compensation phase is composed by two main steps:
  - *Validation*: During this phase models are tested using a validation set S_{val} and the validation matrices are computed.
  - *Test*: In this phase the actual compensation takes place by considering one feature f and all its possible bit rate values combination at a time for every song in the test set. Therefore, for each tuple (f, b_0, b_t), first we perform a check on the validation matrices, in order to only compensate the tuples that really take advantage from the compensation.

Table 5.1 shows the list of features and their corresponding index in the bar plots depicted in the following sections.

index	*feature*	*index*	*feature*
0	Spec. Centroid	21	Non zero count
1	Spec. Skewness	22	Odd-Even Ratio
2	Spec. Kurtosis	23	RMS Amplitude
3	Spec. Inharmonicity	24	Skewness
4	Spec. Slope	25	Crest
5	Spec. Std. Dev.	26	Spec. Variance
6	Irr. k	27	Std. Dev.
7	Irr. j	28	Sum
8	Spread	29	Variance
9	Flatness	30	Spec. Flux
10	Rolloff	31	Energy
11	Sharpness	32	RMS Delta
12	Smoothness	33	Intensity
13	ZCR	34:40	Intensity Ratio
14:16	Tristimulus	41:47	Spec. Cont. Valley
17	Avg Deviation	48:54	Spec. Cont. Peak
18	Kurtosis	55:61	Spec. Cont. Mean
19	Loudness	62:73	Chroma Features
20	Mean	74:93	MFCC

Table 5.1: Features index legend.

5.2.1 Feature compensation test results

Results of the compensation are shown as bar plots. On the x-axis there are the features indexes, values of bars on the y-axis represent the compensation ability for each feature, which represents the mean error evaluated on the whole *test set*.

Rule-based audio feature compensation method

In Fig. 5.1 we present the results of the test phase for the Rule-based audio feature compensation method.
It can be noticed that in the majority of the cases the compensation ability exceeds the 60% and that only in few cases (feature $14 - 15, 22, 31$), the compensation is never applied resulting in a 0% of compensation ability for those specific features.

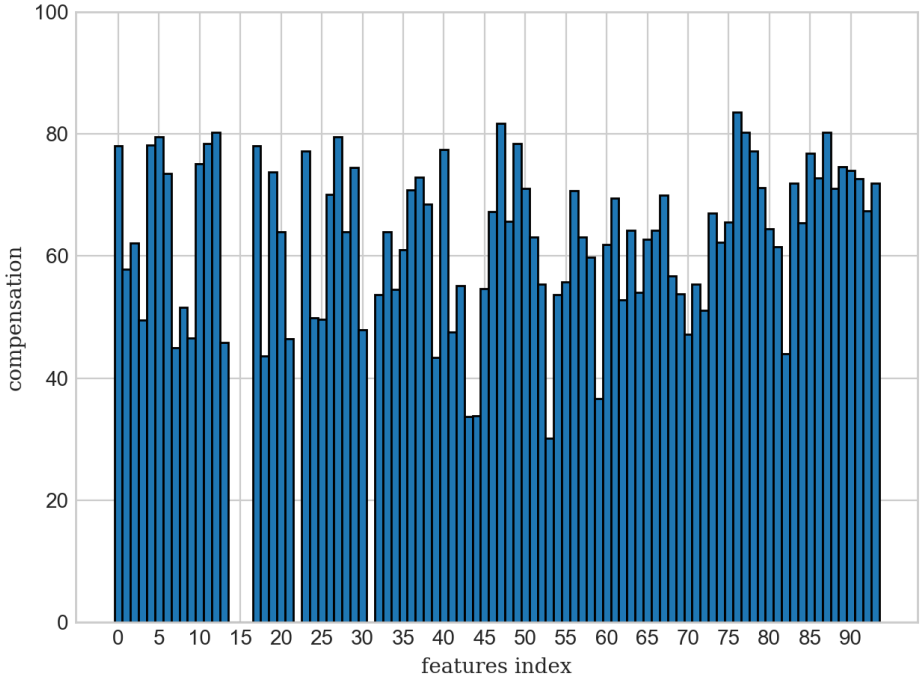


Figure 5.1: Rule-based audio feature compensation method test results.

Linear Regression

Fig. 5.2 shows the test results of the test phase for the Linear Regression method.

We notice that the majority of features exceed the 60% of compensation ability and that there is only one case in which the compensation could not be applied (feature 9).

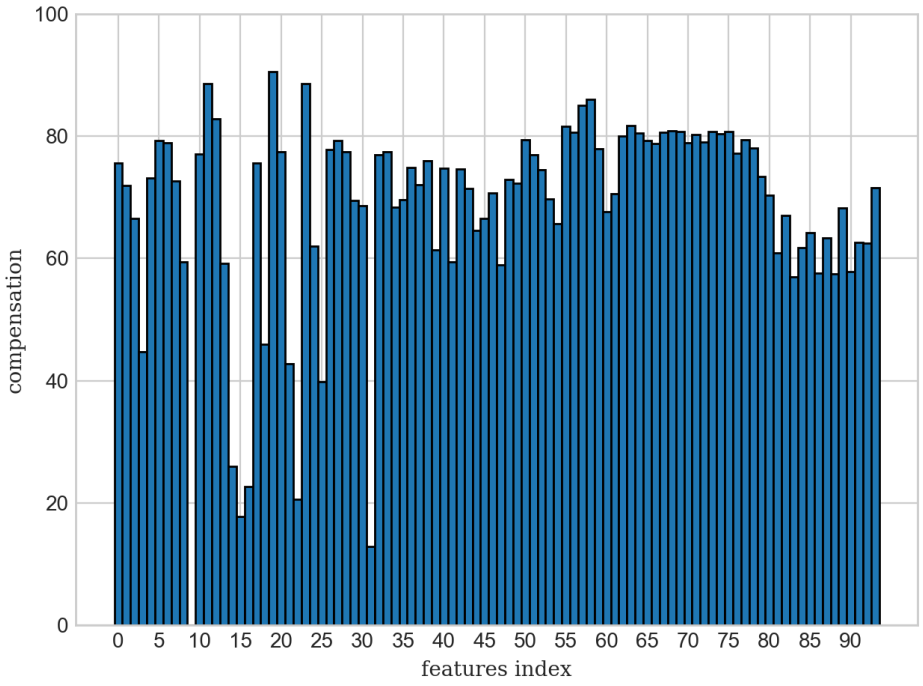


Figure 5.2: Linear Regression test results.

Polynomial Regression

In Fig. 5.3 and 5.4 the results of the test phase are plotted for the Polynomial Regression method (degree 2 and 3 respectively).
Here results for both methods appear to be pretty much similar and although there are a lot of cases in which the compensation has never been applied, some features reach the 100% of compensation (features $66, 72$). Meaning that in all cases the compensated value approached the real feature value at b_t more than the one at b_0, i.e. the compensation has been worthed in all the cases.

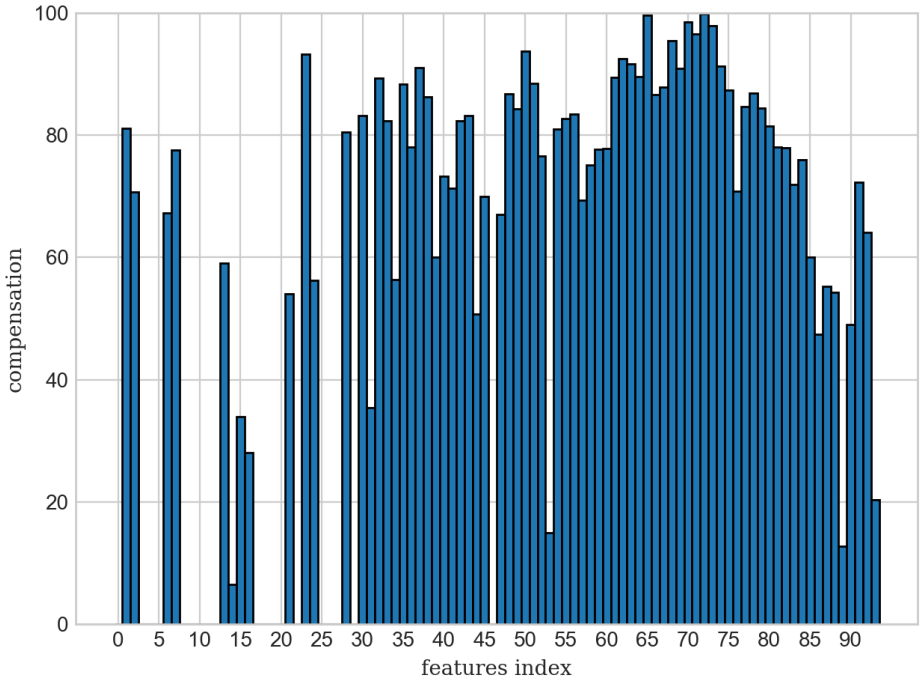


Figure 5.3: Polynomial Regression, degree 2, test results.

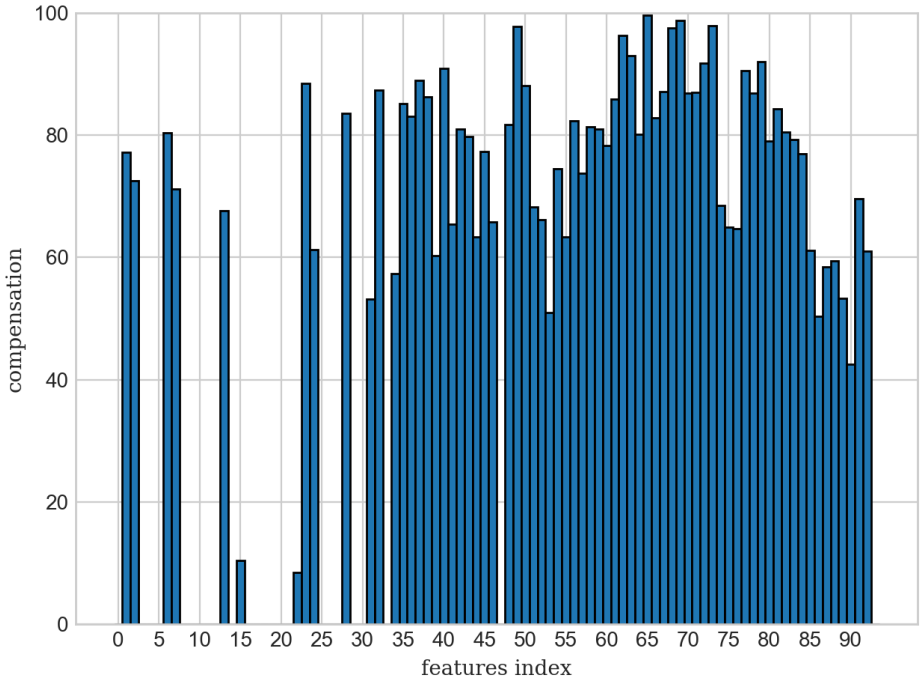


Figure 5.4: Polynomial Regression, degree 3, test results.

Ridge Regression

In Fig. 5.5 we show the plot of the results of the test phase for the Ridge Regression method. It can be noticed that some features try to approach the 100 % (features 34-38) more than others meaning that for some features models computed using the Ridge Regression result to be more accurate with respect to others.

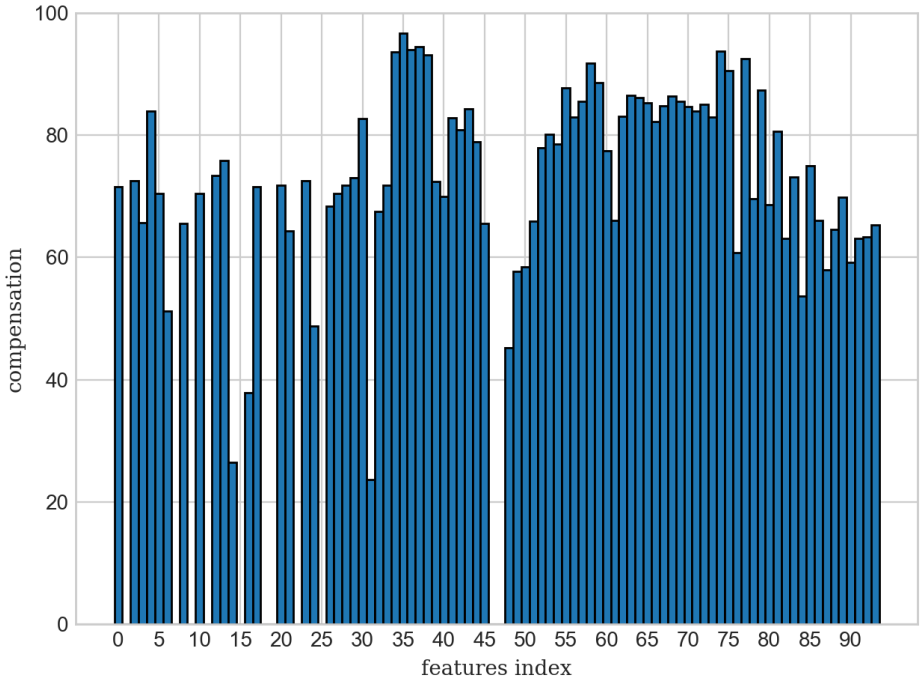


Figure 5.5: Ridge Regression test results.

Support Vector Regression, Linear Kernel

Fig. 5.6 shows the results of the test phase are plotted for the SVR method with linear kernel. This method works better than the other SVR methods (SVR with polynomial and non-linear kernel). Since the distribution of data appears linear it is rightful to expect better results using a linear kernel.

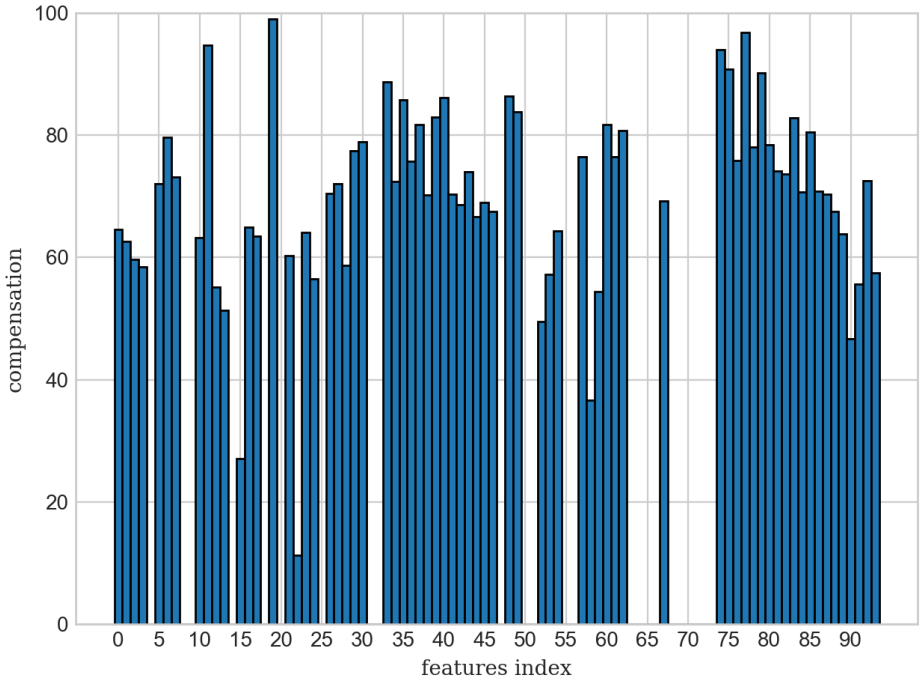


Figure 5.6: SVR Linear test results.

Support Vector Regression, Non-linear Kernel

In Fig. 5.7 we show the results of the test phase for the SVR method with non-linear kernel.

The plot shows that results using a non-linear kernel broadly approach the one obtained with a linear kernel. It can be noticed by looking at the central and right-hand portions of the figure and by make a comparison with the same portions of the graph in Fig. 5.6.

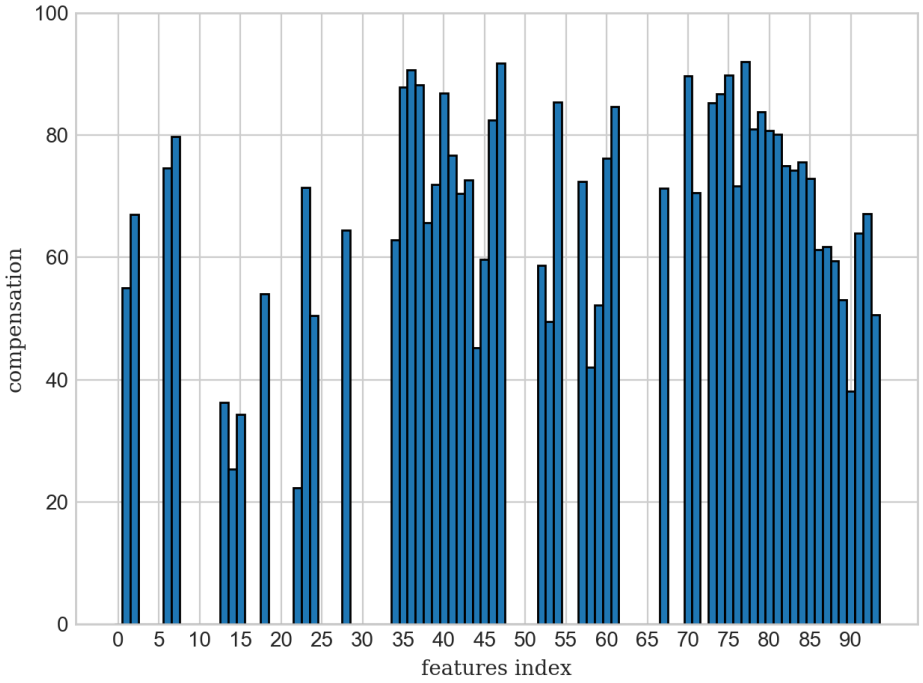


Figure 5.7: SVR RBF test results.

Support Vector Regression, Polynomial Kernel

In Fig. 5.8 we show the results of the test phase for the SVR method with polynomial kernel of degree 2.
We notice that, in this case, performances decrease significantly with respect to the previous methods.

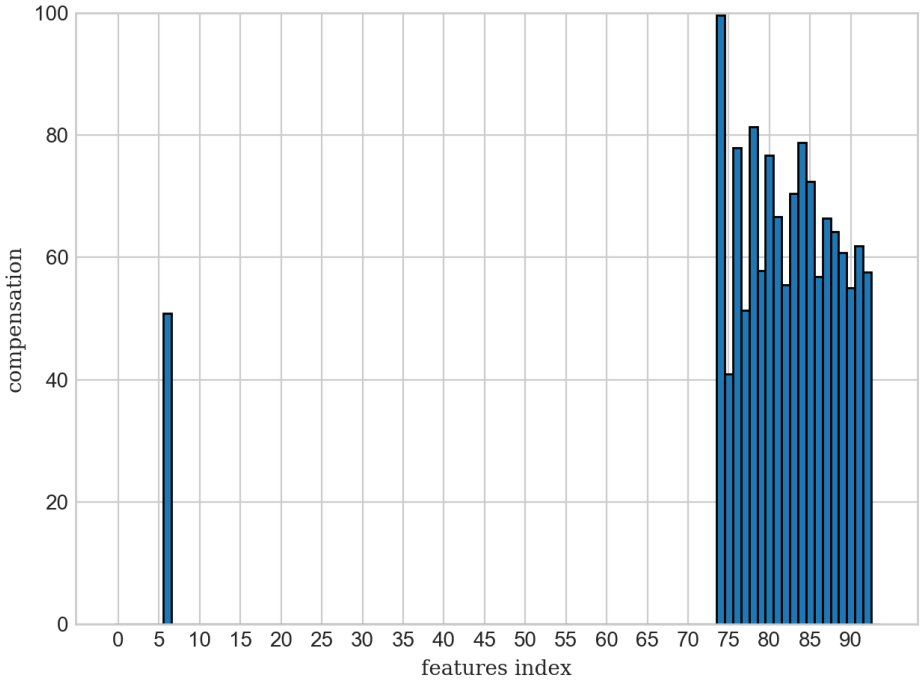


Figure 5.8: SVR Polynomial, degree 2, test results.

Fusion Method 1

Fig. 5.9 shows the results of the Fusion method performed using all the computed models from all the proposed methods.
Fusion method approaches the Rule-based and the Linear Regression method. There are only few cases in which features have a low compensation ability and 0 cases in which the compensation has never been applied.

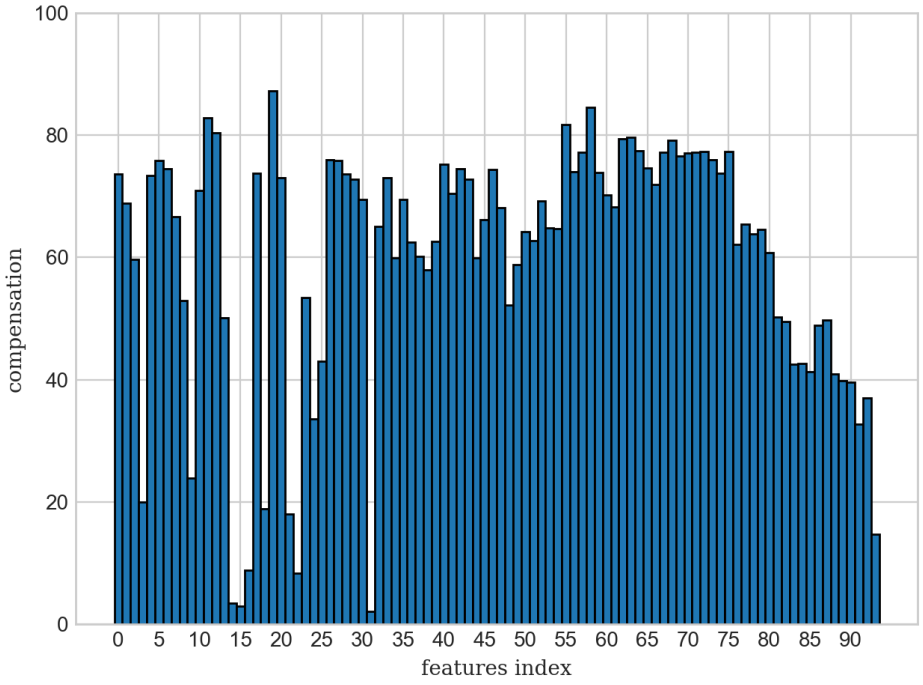


Figure 5.9: Fusion Method test results.

Figure 5.10 shows the differences between the Fusion method and the Linear Regression Method, since it results to be the one that provides the best performances. The upper part of the graph indicates the cases in which the Fusion method outperformed the Linear Regression method, while the lower part indicates the cases in which the Linear regression outperformed the Fusion method.

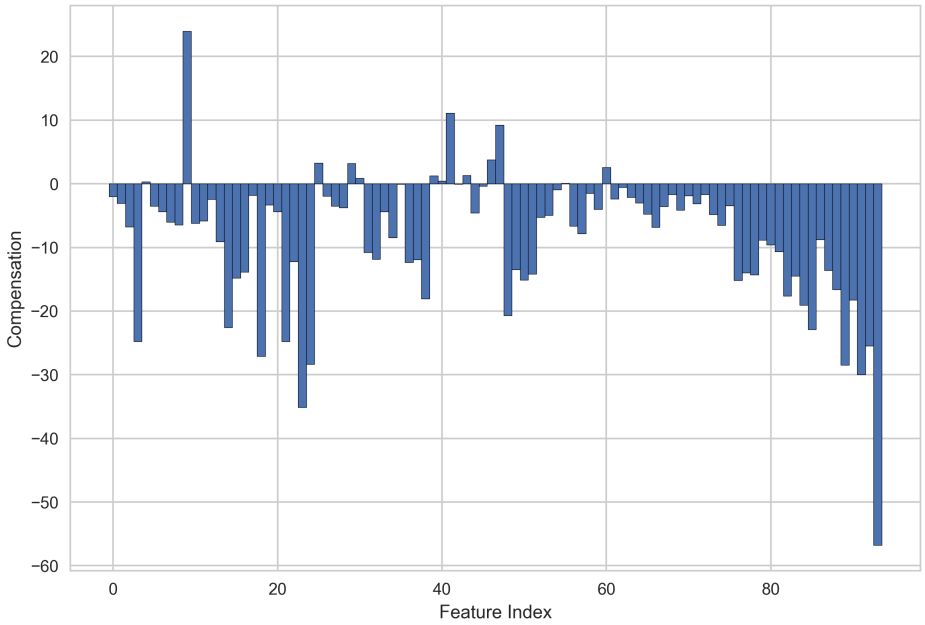


Figure 5.10: Differences between Fusion Method and Linear Regression method.

In order to provide a clearer idea of the compensation ability of each tested method we provide Table 5.2 that shows the overall percentage of compensated features per model. This measure accounts for the overall compensation ability of the model under analysis and it is evaluated by considering only the amount of compensated features that exceeded the 50% (see the bar plots) in the test phase. This measure is useful in finding out what are the best models that can be employed for the Fusion method. The overall percentage of compensated features per model is computed as:

$$model_{comp\%} = \frac{comp_m \cdot tot_f}{100}, \tag{5.1}$$

where $comp_m$ is a variable that is increased every time the compensation

value of a feature is greather than 50% and tot_f in this case is equal to 94 which is the number of features considered in this thesis. Table 5.2 is useful in order to better estimate and understand the overall performances of each method and to design a better Fusion method. Tested models are ranked according to their percentage of compensated features from the highest to the lowest.

	model name	*comp. %*
1	Linear Regression	78.96
2	Ridge Regression	73.32
3	Rule-based	67.68
4	**Fusion Method 1**	66.74
5	Polynomial Reg, degree 3	62.98
6	Polynomial Reg, degree 2	62.04
7	SVR Linear	61.1
8	SVR RBF	47.0
9	SVR Polynomial, degree 2	17.86

Table 5.2: Percentage of compensated features per model.

Fusion Method 2

Table 5.2 suggests that the highest percentage of compensated features belongs to the *linear regression* and *ridge regression* models, with the 78.96% and 73.32% respectively. By exploiting this fact, we performed another fusion test by considering only these two models, instead of considering all the models as in the case of the first Fusion test. Results are shown in Figure 5.11 and in some cases they perform better with respect to the Fusion test 1 (features 55-75).

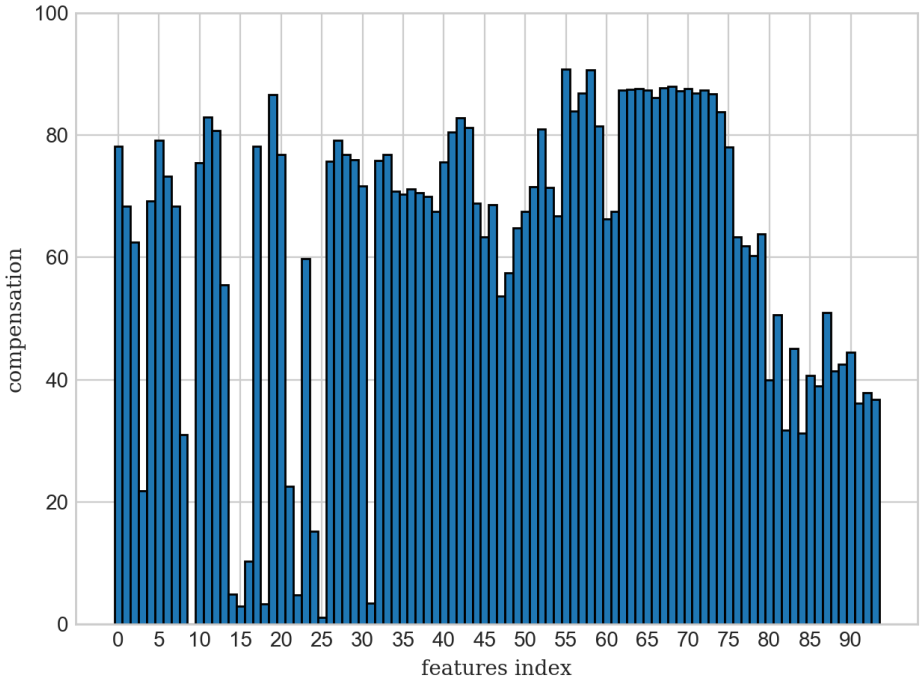


Figure 5.11: Fusion Method 2 test results.

Figure 5.12 shows the differences between the Fusion method 2 and the Linear Regression Method. The upper part of the graph indicates the cases in which the Fusion method 2 outperformed the Linear Regression method, while the lower part indicates the cases in which the Linear regression outperformed the Fusion method 2.

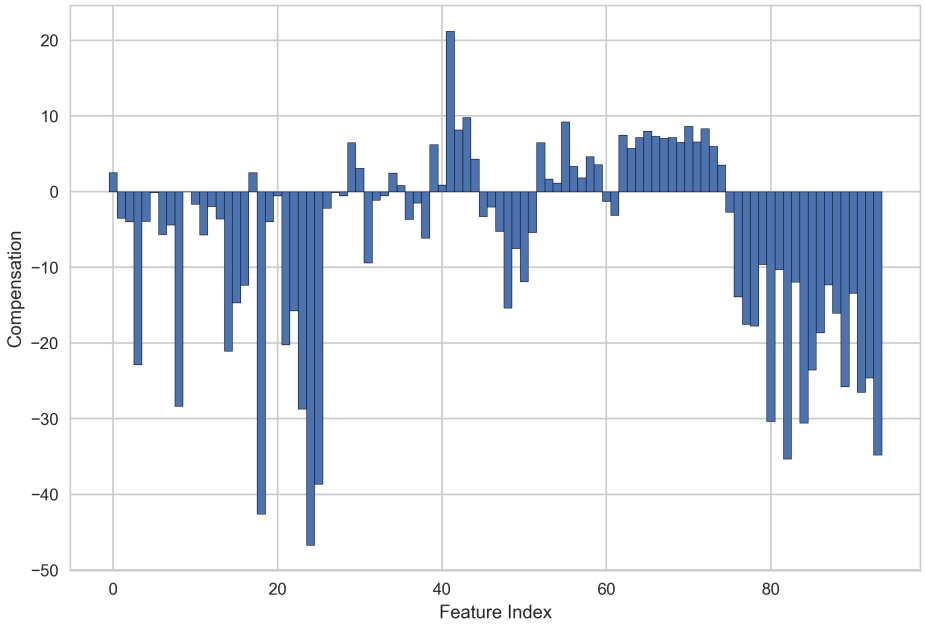


Figure 5.12: Differences between Fusion Method 2 and Linear Regression method.

Table 5.2 is updated by adding the percentage of compensated features belonging to the Fusion test 2. The updated results are shown in Table 5.3. Results obtained with Fusion method 1 and with Fusion method 2 are very close to each other. This means that it is possible, and maybe less computational demanding, to use the Fusion method by only considering the models that result in the highest overall compensation ability (in this case 2), instead of using all of them (i.e. 8).

	model name	*comp. %*
1	Linear Regression	78.96
2	Ridge Regression	73.32
3	Rule-based	67.68
4	**Fusion Method 1**	66.74
5	**Fusion Method 2**	65.8
6	Polynomial Reg, degree 3	62.98
7	Polynomial Reg, degree 2	62.04
8	SVR Linear	61.1
9	SVR RBF	47.0
10	SVR Polynomial, degree 2	17.86

Table 5.3: Percentage of compensated features per model (updated).

5.3 Analysis of the results

Table 5.2 proves that in almost all the cases, it is possible (and worth) to perform feature compensation, since in all cases but one, the overall compensation ability is higher than the 50%.
From the results it is also clear that, in almost all the cases and more than the 50% of the times, compensated features are very close to the original value of the feature. We discovered that the best models for performing the task of feature compensation are *Linear Regression* and *Ridge Regression* with over the 73 % of compensation ability.
Performing feature compensation in a mixed fashion (i.e. testing the Fusion method) results suitable because it outperforms half of the methods if they are employed singularly.
The percentage of compensated features in the case of the second fusion test (the one with Linear and Ridge regression models) performed is only 65.8%,

which is slightly less than the percentage of the Fusion method performed considering all the models (66.74%).

The fact that Linear regression, Ridge regresion and the the Rule-based approach, in this case, outperform both fusion methods could be due to the choice of the training dataset used in the model computation phase and to the validation datasets used for the validation phase. It may happen, for instance, that some models perform better in the validation phase for a certain number of bit rate paris but worse in the test phase. The datasets must therefore be chosen carefully in order to find the best possible combination between validation and test set.

In Figure 5.13 we show, for each feature, the best compensation among all the computed models, selected by considering the compensation percentage among all the models and selecting the one provided the maximum value of compensation.

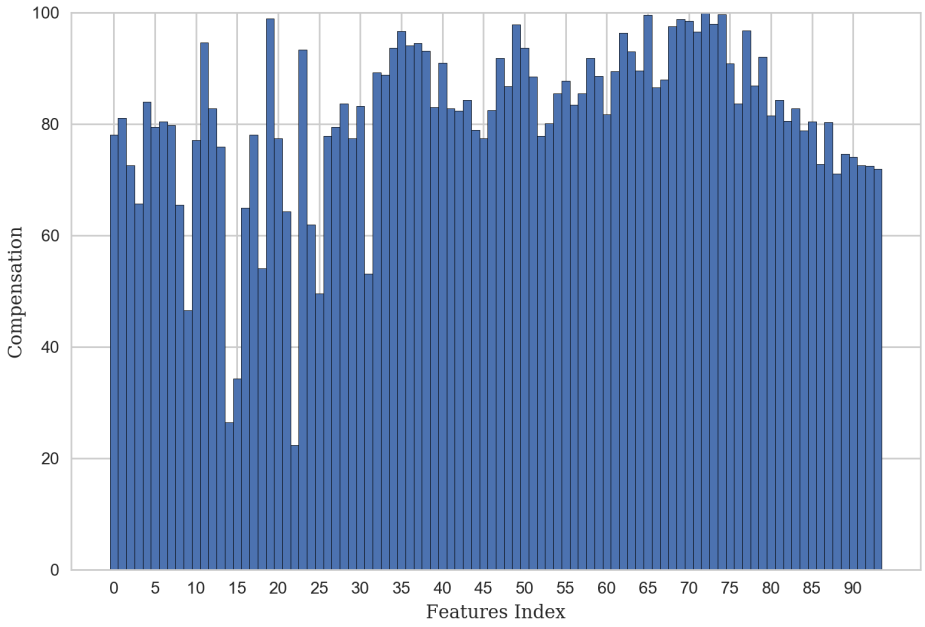


Figure 5.13: Result of the best compensation models.

5.4 Music Emotion Recognition (MER) results

In order to test the effect of the feature compensation methods in a real case Music Information Retrieval's scenario, we performed Music Emotion Recognition (MER) by considering both supervised (classification) and unsupervised (clustering) techniques.
We considered two main problems:

- *Binary problem*: We selected two pairs of affective terms from the CAL500 vocabulary *(happy-sad, arousing-calming)*.
- *Multi-Class problem*: We selected four classes *Emotion-Happy, Emotion-Angry, Emotion-Calming and Emotion-Emotional*.

The dataset used is the CAL500 composed by 500 songs. For the supervised (classification) method, the train test was chosen to be composed of the 75% of the total dataset, while the test set was chosen to be the 25%. The target bit rate was set to 320 kbps and the starting bit rate values have been chosen randomly between 32 and 256 kbps (see equation (4.1)). In order to test different sets of features we considered two cases: i) MFCCs only and ii) features selected from the whole features set by using a ReliefF [29] technique, which is an algorithm that performs feature selection by selecting a subset of features that result to be very discriminative for that specific application, among all the available features.
The supervised machine learning techniques used are Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).
The unsupervised machine learning technique used is K-Means Clustering.
Table 5.4 and 5.5 show the results of the MER classification and clustering respectively, where:

- X_{target}: is the feature value at bit rate b_t.
- X_{source}: is the feature value at bit rate b_0.
- X_{comp}: is the compensated feature value without using the worthy condition.
- X_{compM}: is the compensated feature value using the worthy condition.

Concerning the supervised scenario, the model was trained on the X_{target} dataset, whereas the test were performed separately considering $X_{target}, X_{source}, X_{comp}, X_{compM}$.
For the unsupervised scenario we exploited the *Calinski-Harabasz (CH)* [11] measure as quality index.
Since from the previous section one of the best models for features compesation resulted to be the Linear Regression model, X_{comp}, X_{compM} have been computed using the Linear regression models.
In Tab. 5.4 we show the resulting accuracy of the binary and multi-class classification and in Tab. 5.5 we show the clustering results in which the higher is the Calinski-Harabasz index, the better is the clustering.

			SVM (%)	KNN (%)
		X_{target}	85.42	85.42
	MFCC	X_{source}	77.08	77.08
		X_{comp}	85.42	85.42
happy		X_{compM}	85.42	85.42
sad		X_{target}	89.58	77.08
	ReliefF	X_{source}	89.58	62.05
		X_{comp}	89.58	83.33
		X_{compM}	89.58	75.0
		X_{target}	79.89	77.25
	MFCC	X_{source}	75.13	70.37
		X_{comp}	79.37	76.72
arousing		X_{compM}	79.37	77.25
calming		X_{target}	81.48	74.07
	ReliefF	X_{source}	66.67	55.03
		X_{comp}	66.67	73.54
		X_{compM}	66.67	71.43
		X_{target}	44.44	50.0
	MFCC	X_{source}	44.44	33.33
happy		X_{comp}	44.44	44.44
angry		X_{compM}	44.44	44.44
calming		X_{target}	44.44	50.0
emotional	*ReliefF*	X_{source}	44.44	38.89
		X_{comp}	44.44	44.44
		X_{compM}	44.44	44.44

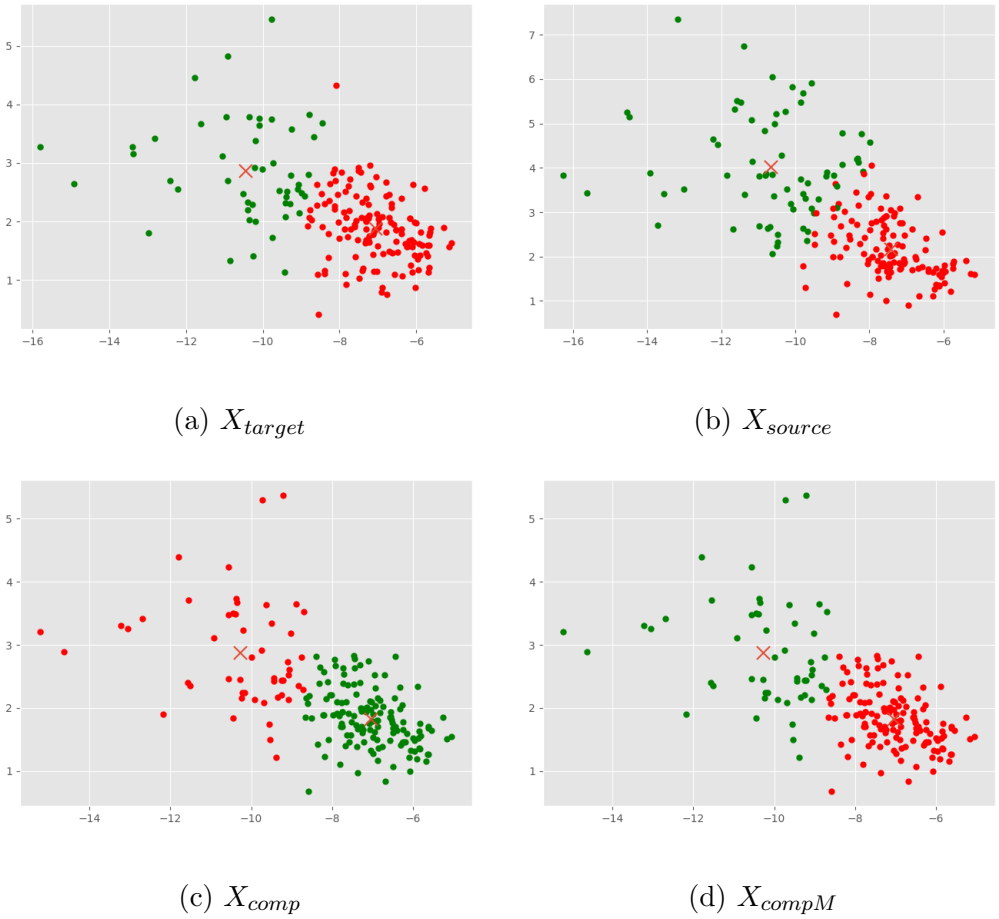
Table 5.4: Accuracy results of the binary and four mood classes using supervised machine learning methods.

			CH
	MFCC	X_{target}	23.86
		X_{source}	14.16
		X_{comp}	26.89
happy		X_{compM}	26.79
sad	*ReliefF*	X_{target}	10.10
		X_{source}	0.0
		X_{comp}	16.63
		X_{compM}	0.18
	MFCC	X_{target}	86.61
		X_{source}	61.62
		X_{comp}	91.57
arousing		X_{compM}	91.12
calming	*ReliefF*	X_{target}	34.26
		X_{source}	0.0
		X_{comp}	56.31
		X_{compM}	0.35
	MFCC	X_{target}	2.14
		X_{source}	2.0
happy		X_{comp}	2.07
angry		X_{compM}	2.07
calming	*ReliefF*	X_{target}	0.63
emotional		X_{source}	0.0
		X_{comp}	0.80
		X_{compM}	0.37

Table 5.5: Results of the binary and four mood classes using unsupervised machine learning methods.

5.4.1 Music Emotion Recognition (MER), analysis of the results

Both supervised and unsupervised investigations confirm the need to perform feature compensation in case of a non-homogeneous data set.
In fact, in almost all the cases, the results obtained using X_{target} outperform those obtained using X_{source}.
Results shows that in most of the cases, tests using the feature compensation (with and without the worthy condition) are very close to the case of X_{target}.
This fact means that compensated features (with and without worthy condition) approach the real value of the feature at b_t and that using X_{source}, i.e. the feature value at b_0, is not optimal when performing MIR applications.
In the specific case of MER, it appears clear that feature values play a key role in order to obtain acceptable results in the identification of the various classes and clusters.
In Fig.5.14 we show the *happy/sad* music emotion recognition and the composed clustering. We noticed that the shape of the clusters in the case of X_{target}, X_{comp} and X_{compM} are very similar, while in the case of X_{source} clusters are not well separated and some examples seem to overlap in the border region between the two clusters. This happens because the use of an hinomogeneous dataset (i.e. composed of songs encoded at different bit rate values) produces features vectors containing values that are not easily comparable to each other, thus resulting in being poorly discriminative when projected on a feature space.



(a) X_{target} (b) X_{source}

(c) X_{comp} (d) X_{compM}

Figure 5.14: Happy/Sad clustering plot.

Chapter 6

Conclusions and Future works

In this last Chapter we review the problems addressed in this thesis work as well as the resolutive methods proposed to solve them.
We then present future studies on the problem of feature compensation as well as developements of the proposed techniques.

6.1 Conclusions

In this thesis work we addressed the problem of compensating feature values extracted from songs encoded with mixed bit rate values. We showed the effect that audio coding causes on audio features and in particular on Music Information Retrieval applications, which results in poor classification accuracy and clustering ability. We therefore showed that it is possible to overcome the problem of having an hinomogeneous audio data set (i.e. a song collection composed of songs encoded at different bit rate values) by performing feature compensation. This allows the creation of an homogeneous dataset (i.e. a data set composed of features extracted from song encoded using the same bit rate value), that is more suitable for the purpose of MIR final applications. In particular, it is needful for obtaining choerent results and high accuracy when performing classification or clustering problems.

We presented various methods for compensating feature values extracted from songs encoded at different bit rate values. We proved, with various tests, that it is possible to compensate feature values and that the compensation results useful in the 78.96 % of the cases.

We discussed about the effect of audio coding on feature values and its consequent effect on MIR applications such as genre classification and others. We tested our feature compensation method on a real case scenario in the case of Music Emotion Recognition (MER) and proved that the case of having a not homogeneous dataset is an issue that badly affects the ability of classification and clustering methods in finding a proper subdivision of the feature space and in providing high accuracy results. In fact, in the binary classification *arousing-calming* and the case of the use of MFCCs features, the accuracy value of X_{target} for the KNN is 77.25 %, while X_{source} only scored 70.37 % and both X_{comp} and X_{compM} approach X_{target} with 76.72 % and 77.25 % respectively.
The case of multi-class classification confirms better performances in case of the use of an homogeneous dataset. In fact, in this case, the accuracy value for X_{target} is 50.0%, the one for X_{source} is only 33.33 % while both X_{comp} and X_{compM} try to approach X_{target} by scoring 44.44%.

The results obtained by applying clustering techniques confirms the ones obtained for the classification. In fact, for the four moods case (*happy, angry, calming, emotional*) the CH values for X_{target}, X_{source}, X_{comp} and X_{compM} are 2.14, 2.0, 2.07, 2.07, respectively, confirming the lowest score belonging to X_{source}.

6.2 Future Developements

Further investigations need to be done by considering a larger number of lossy encoders as well as encoding settings and parameters, since, for the purpose of this thesis, we only focused on MP3 compression at various bit rate values.
The use of only 500 songs from the CAL500 dataset may be not enough when

using machine learning methods. In fact, the fusion method resulted to be affected by the choice of the training and validation datasets. Therefore, a larger song data set need to be used in order to build models that are more accurate during the compensation.
Other future works will be devoted to the analysis and modeling of the *feature-bitrate behaviour* and to the developement of a feature compensation toolbox.

Bibliography

[1] *Audio Forensics Meets Music Information Retrieval - A Toolbox for Inspection of Music Plagiarism.* Zenodo, August 2012.

[2] A. Sarti B. Di Giorgi, M. Zanoni and S. Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony, 2013.

[3] T. Bertin-Mahieux and D.P.W. Ellis. Large-scale cover song recognition using the 2d fourier transform magnitude, 2012.

[4] P. Bestagini, M. Zanoni, L. Albonico, A. Paganini, A. Sarti, and S. Tubaro. Feature-based classification for audio bootlegs detection. In *2013 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 126–131, Nov 2013.

[5] Marina Bosi and Richard E. Goldberg. *Introduction to Digital Audio Coding and Standards.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[6] Michael Casey, Ben Fields, Kurt Jacobson, and Mark Sandler. The effects of lossy audio encoding on genre classification tasks. In *Audio Engineering Society Convention 124*, May 2008.

[7] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges, 2008.

[8] S. Chu, S. Narayanan, and C. C. J. Kuo. Environmental sound recognition with time;frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, Aug 2009.

[9] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, April 2011.

[10] Michael Haggblade, Yang Hong, and Kenny Kao. Music genre classification. 2011.

[11] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2):107–145, 2001.

[12] Shuhei Hamawaki, Shintaro Funasawa, Jiro Katto, Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. *Feature Analysis and Normalization Approach for Robust Content-Based Music Retrieval to Encoded Audio with Different Bit Rates*, pages 298–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[14] Kate Hevner. Experimental studies of the elements of expression in music. *American Journal of Psychol- ogy, 48:246–268*, 1936.

[15] X. Hu, S. J. Downie, C. Laurier, M. Bay, and A. Ehmann. The 2007 mirex audio mood classification task: Lessons learned. In *9th International Conference on Music Information Retrieval*, Philadelphia, USA, 14/09/2008 2008.

[16] Kurt Jacobson, Matthew Davies, and Mark Sandler. The effects of lossy audio encoding on onset detection tasks. In *Audio Engineering Society Convention 125*, Oct 2008.

[17] D. Jannach, C. Weihs, G. Rudolph, and I. Vatolkin. *Music Data Analysis: Foundations and Applications*. Chapman and Hall/CRC Computer Science and Data Analysis Series. Taylor & Francis, 2016.

[18] K. K. Jensen. Timbre models of musical sound: From the model of one sound to the model of one instrument. *University of Copenhagen*, 1999.

[19] Youngmoo E. Kim, Erik M. Schmidt, Raymond Migneco, On G. Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A. Speck, and Douglas Turnbull. Emotion recognition: a state of the art review. In *11th International Society for Music Information and Retrieval Conference*, 2010.

[20] A. Klapuri and T. Virtanen. Automatic music transcription, 2008.

[21] Anssi P. Klapuri. Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282, 2004.

[22] J. Krimphoff, S. McAdams, and S. Winsberg. Caractérisation du timbre des sons complexes. ii: Analyses acoustiques et quantification psychophysique. *Journal de Physique*, 4(C5):625–628, 1994.

[23] C. Laurier, M. Sordo, Joan Serrà, and Perfecto Herrera. Music mood representations from social tags. In *International Society for Music Information Retrieval (ISMIR) Conference*, pages 381–386, Kobe, Japan, 26/10/2009 2009.

[24] T. Li and M. Ogihara. Detecting Emotion in Music. Baltimore, USA, October 2003.

[25] Tao Li and G. Tzanetakis. Factors in automatic musical genre classification of audio signals. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 143–146, Oct 2003.

[26] D. Grandjean M. Zentner and K. R. Scherer. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion, vol. 8,4, pp. 494–521*, 2008.

[27] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In Alceu de Souza Britto Jr., Fabien Gouyon, and Simon Dixon, editors, *ISMIR*, pages 83–88, 2013.

[28] N. Peters, H. Lei, and G. Friedland. Room identification using acoustic features in a recording, June 12 2014. US Patent App. 14/097,369.

[29] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1):23–69, 2003.

[30] J. A. Russell. A circumspect model of affect. *Journal of Psychology and Social Psychology, vol. 39, no. 6, p. 1161*, 1980.

[31] S. Sigurdsson, K. B. Petersen, and T. Lehn-Schiøler. Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. In *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR)*, 2006.

[32] B. L. Sturm. Alexander lerch: An introduction to audio content analysis: Applications in signal processing and music informatics. *Computer Music Journal*, 37(4):90–91, Dec 2013.

[33] B.L. Sturm. A survey of evaluation in music genre recognition, 2012.

[34] Bob L. Sturm. Evaluating music emotion recognition: Lessons from music genre recognition? *International Conference on Multimedia and Expo*, pages 1–6, 2013.

[35] Bo Shao Tao Li, Mitsunori Ogihara and DingdingWango. Machine learning approaches for music information retrieval, theory and novel applications of machine learning, meng joo er and yi zhou (ed.), intech, doi: 10.5772/6687. 2009.

[36] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 439–446, New York, NY, USA, 2007. ACM.

[37] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 439–446, New York, NY, USA, 2007. ACM.

[38] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, pages 293–302, 2001.

[39] Aiko Uemura, Kazumasa Ishikura, and Jiro Katto. *Effects of Audio Compression on Chord Recognition*, pages 345–352. Springer International Publishing, Cham, 2014.

[40] J. Urbano, Dmitry Bogdanov, Perfecto Herrera, Emilia Gómez, and Xavier Serra. What is the effect of audio quality on the robustness of mfccs and chroma features? In *15th International Society for Music Information Retrieval Conference*, pages 573–578, Taipei, Taiwan, 27/10/2014 2014.

[41] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[42] Yi-Hsuan Yang and Homer H. Chen. *Music Emotion Recognition*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011.

[43] Y. You. *Audio Coding: Theory and Applications*. Springer US, 2010.

[44] Markos Zampoglou and Athanasios G. Malamos. Music information retrieval in compressed audio files: a survey. *New Review of Hypermedia and Multimedia*, 20(3):189–206, 2014.

[45] M. Zanoni, D. Ciminieri, A. Sarti, and S. Tubaro. Searching for dominant high-level features for music information retrieval. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2025–2029, Aug 2012.