More than a feeling?

Reliability and robustness of high-level music classifiers

C. Mostert

Delft University of Technology

TU Delft

More than a feeling?

Reliability and robustness of high-level music classifiers

by

C. Mostert

to obtain the degree of Master of Science
in Computer Science
Track: Data Science & Technology
at the Delft University of Technology,
to be defended publicly on Thursday August 20, 2020 at 10:30 AM.

Student number: 4473353
Project duration: November, 2019 – August, 2020
Thesis committee: Prof. dr. A. Hanjalic, TU Delft, chair
Dr. C.C.S. Liem MMus, TU Delft, supervisor
Dr. A. Panichella, TU Delft

TU Delft

Preface

"Just 'cause you feel it
Doesn't mean it's there"

Radiohead – There There, on *Hail to the Thief* (2003)

Before you lies my thesis which takes a critical look at high-level music classifiers. Do they actually perform as well as we believe? And can we trust their output to be a solid foundation for future research? Results using metrics based on label stability, label agreement and distributional differences show unexpected patterns in classifier outputs indicating that these outputs should not be taken as absolute truth and do not form a solid foundation for further research. The improvement of these high-level music classifiers is a multidisciplinary effort for which better evaluation methods are required. To this end, several approaches for more comprehensive classifier testing are presented, based on best practices in psychology and software testing. These approaches are not constrained to the field of Music Information Retrieval and can be applied to evaluate classifiers in other domains as well. This research was carried out from November 2019 to July 2020 and written to obtain a master's degree in Computer Science, specialization Data Science & Technology at the Delft University of Technology. The thesis committee consists of Prof. dr. A. Hanjalic (TU Delft, chair), Dr. C.C.S. Liem MMus (TU Delft, supervisor) and Dr. A. Panichella (TU Delft).

I've always had an interest in psychology and within the Multimedia Group I was able to combine this interest with that of Computer Science. In the first couple of weeks of the process, it was with great enthusiasm that I started exploring my initial idea: how much can we find out about any individual by their listening history? I figured state of mind would influence music choices, and that by observing these choices I would be able to infer this original state of mind or even predict where it would go in the future. However, the more I started exploring this idea the more questions came to mind: would the music mood labels — which were automatically generated by classifiers — actually be informative enough? And could they even be correct? What about different interpretations of the same song? These questions are what kickstarted the analysis of which the results are presented here.

The process of trying to answer these questions often felt like going down a rabbit hole: sometimes it seemed like trying to answer one question only spawned several more, and just when I thought I found an explanation for all the weirdness I was observing, along would come an observation that did not quite fit the rest. And meanwhile, due to the COVID-19 pandemic, the world around us was growing more uncertain by the day as well.

That is why I would like to express my heartfelt thanks to my supervisor, Cynthia, for making time in her busy schedule to help me navigate this rabbit hole and for showing genuine interest in and enthusiasm for the results I found along the way. To my friends and family for their support and their perseverance in listening to my ramblings. To my girlfriend, Iris, for supporting me and making me feel like I was never truly alone in writing this thesis and to anyone else who has helped me along the way, thank you.

And thank *you*, the reader, for taking the time to read at least a single page of my thesis. I genuinely appreciate your interest, and hope you enjoy reading the rest.

C. Mostert
Delft, July 2020

Contents

1 Introduction

Machine learning is becoming an increasingly prominent field in the last decade. While the field has a timeline that goes much further back — with theoretic statistical foundations like Bayes' theorem stemming from the early nineteenth century [84] — the last decade has seen an increase in popularity and feasibility of machine learning approaches. This increase in popularity and feasibility can mostly be attributed to the rise of Deep Learning methods, which have proven to be very versatile, being able to process many different forms of data, and in large quantities [88].

The earliest machine learning systems often relied on *deductive* learning, which is an approach that uses prior, provably correct knowledge to construct a model or algorithm [147]. This approach has the benefit of generating understandable models, since the logic used to construct such a model can be explicitly checked. However, in cases where the underlying logic is very hard to put into clearly defined rules, building a model using this deductive approach becomes more difficult. Take, for example, the task of detecting if there is a bird in a given picture. The rule based approach quickly becomes very complicated, since you would need to manually define the relation between certain pixels being colored and the presence of a bird. In addition, the bird could be present anywhere in the picture. How would you then deal with the size, position and rotation of the bird given a fixed set of rules? While incorporating some form of fuzzy logic in the rules allows a model to deal with a certain degree of uncertainty [156], the amount of uncertainty might simply be too large to create a straightforward model. In other cases, the task knowledge might simply not be available at all: a doctor might utilize a basic set of rules to predict if a patient has cancer, but often the only way to know this for sure is to have a biopsy [6]. Finally, these deductive approaches are *static*, if you decide that in addition to birds, you also want to be able to detect other animals, then for each animal you want to add the model needs to be changed, requiring a large amount of work.

Inductive learning approaches overcome these limitations by inducing these decision rules based on the available data. These approaches learn knowledge based on statistical regularities in the data [144]. Since these models are based on statistical properties of the data, the model can only be as good as the data it is supplied with, requiring a large amount of samples to get a good estimate. If only a small amount of data is available, then the inductive algorithms will perform rather poorly [111]. These inductive approaches have recently become more feasible, with massive amounts of data now being available for processing, forecast to reach 175 ZB (= 175 trillion GB) in 2025 [122] and increasing computing power able to process these large amounts of data owing to Moore's law [100]. This has enabled machine learning algorithms to be viable in many different applications, ranging from expression recognition in pictures using deep convolutional neural networks [21] to using a Support Vector Machine (SVM) to predict bankruptcy [134].

1.1. Problem statement

The problem with many of these inductive learning approaches is that they are Black box models — the accuracy can be tested on unseen data for which the true labels are known, but the underlying model is often not explainable. Some Neural Networks can have around 2.5 million tuned parameters

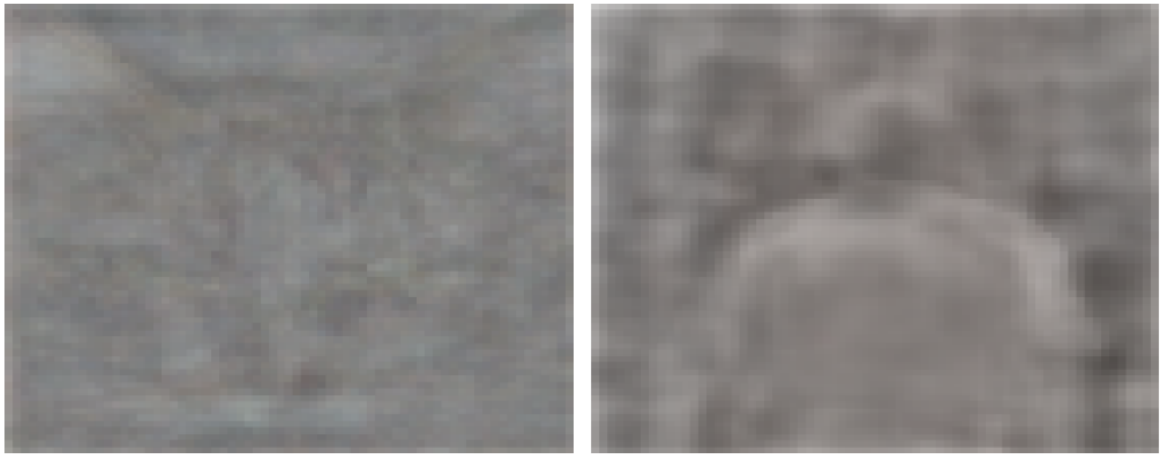


Figure 1.1: Visualization of trained neuron weights that detects cats (left) and human bodies (right) [87]

.

if 19 layers are used [124] making manual inspection and understanding of these underlying weights an infeasible task. Some Deep Learning approaches make it possible to visualize weights to get a rough idea of what the weights in a neuron model (see Figure 1.1), but such visualizations are often only intuitive for visual data and lose their interpretability for more abstract data. Furthermore, some argue that these visualizations for Deep Neural Networks are insufficient and that the focus should shift to making more interpretable Machine Learning models [125].

This inability to verify if a trained model is actually doing what you expect it to — often with the only available check being to run the model on some validation data for which you have the labels and calculating the accuracy — can be especially problematic when designing machine learning algorithms that aim to classify and detect abstract concepts. When these abstract concepts are ill-defined or subjective, there is a chance that the labels in the training data are inconsistent or do not reflect the properties of the data you wish to capture. In psychological literature this is often called *construct validity* [33]: the ability for the machine learning algorithm to properly capture the desired, abstract concept from the data. If there is a lack of construct validity, then the algorithm might be picking up on unintended statistical regularities in the data, instead of the desired construct we wish to capture.

In addition, there are a lot of other factors that need to be taken intro account depending on the type of data the machine learning algorithm operates on. In the case of music analysis, digital audio can be represented and encoded in many different ways (using different encoding schemes, file formats etc.) and audio codecs are developed based on human perception instead of signal consistency. A machine learning approach working with this kind of data needs to be able to see these different representations of the entitity (i.e. the recording) as the same, regardless of differences in the underlying digital signal. Furthermore, when working with any kind of multimedia content, human perception and interpretation of this data will always play an important role, since we almost always want to give more meaning to the data than it simply being a combination of 1s and 0s.

Given these challenges — where a machine learning approach needs to both interpret the data consistently and correctly — it might be optimistic to assume that an inductive Machine Learning approach, having been shown only a small amount of training examples, would be able to learn all of these complex concepts and apply them in a way that results in accurate results. However, companies like The Echo Nest (which later became Spotify) confidently provide a valence value for a given track[1] with the only guarantee being

> ”It’s no easy feat to have a computer listen to a song in three seconds and determine its emotional valence, but we’ve figured out how to do it.”[2]

The main question that *has not* been asked seems to be: did we actually figure out how to do it? Or

[1] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

[2] https://web.archive.org/web/20170422195736/http://blog.echonest.com/post/66097438564/plotting-musics-emotional-valence-1950-2013

does it *seem* like we figured it out? Given the inductive nature of many Machine Learning methods popular today, interpretability of the models is low, and the only reported test metric for many methods are either the classification accuracy [102][154] or R^2 value [155][38] on a small subset of data left out from training or using cross-validation [135]. This does not guarantee that the trained classifiers work well on real-world data not available in the training or validation sets, and since the same recording can be represented very differently digitally (see Section 2.3), it is feasible that when deployed 'in the wild' these classifiers might underperform when faced with the multitude of codecs and bitrates available. Furthermore, since inductive machine learning approaches generally do not result in interpretable models, it is very difficult to verify if the models did not pick up on some spurious correlation in the training data to produce the labels.

Sturm [137] compares classifiers which rely on such confounds in the data with Hans, the clever horse. Hans seemed to be capable of complex tasks such as arithmetic, answering by tapping his hoove the correct amount of times. However, by performing experiments in controlled environments, Dr. Oskar Pfungst was able to show that Hans did not have the ability to do arithmetic, and instead focused on the body language of the person asking the question by starting to tap when he would see the person tilt his head and stopping when the person would tilt his head back again [113]. It might be the case that many of the current machine learning classifiers function like Hans, where they seem to be able to perform complex tasks while they might actually rely on confounding factors in the training data.

1.1.1. High-level music classification

One such application of applying machine learning algorithms to ill-defined or subjective concepts is high-level music classification. This task requires a machine learning algorithm to label certain high-level musical properties like 'Is this song acoustic?' or in the case of music sentiment analysis automatically classifying the emotion that is present or that someone has towards a piece of music [97]. Examples of classified moods include sad moods, aggressive moods or even party moods [85]. This process involves extracting features from the music and then training a machine learning algorithm to map these features to some value in a chosen emotion model. Music sentiment analysis can help us better understand the emotional reaction to music, aiding the choice of music in for example music therapy to make treatment more efficient and predict how the client will respond [65] or tell us more about musical preferences linked to certain moods, which has applications in automatic mood-based music recommendation. This task requires combined knowledge from both psychology and computer science.

But as discussed, the currently popular inductive machine learning models hardly provide any interpretable results. With only the basic validation metrics, most machine learning classifiers seem to perform just fine. Since there seems to be no cause for concern regarding the accuracy of many of these classifiers, the data from a high-level music mood classifier might be used to draw conclusions about for example seasonal patterns of affective preference in music [109], the prediction of hit songs [157] or making claims about the direction pop music is going [71] resulting in large amounts of media coverage (See Figure 1.2).

All this uncertainty around these high-level music classifiers might be cause for concern. With no extensive ways of validating the trained models, classification accuracies on validation sets are often the only available metric. And if the classifiers perform worse then we would hope, then doing more research on the resulting data might lead us to draw incorrect conclusions. Given all these uncertainties about the performance of many of these high-level classifiers and the lack of literature discussing these uncertainties, it is important to take a step back and take a critical look at the existing classifiers: are they reliable? Can we measure the performance of such classifiers *without* relying on 'ground truth' labels in a validation set? Would these classifiers serve as a solid foundation for future research?

1.2. Research Questions

Given the problem statement as introduced in this chapter — namely the lack of trust we can place in the performance of high-level music classifiers and the lack of proper validation methods for such classifiers besides classification accuracy on a validation set with defined labels — this thesis aims to answer the following research questions:

- **RQ1**: How can multiple representations of the same input be leveraged to quantify the perfor-

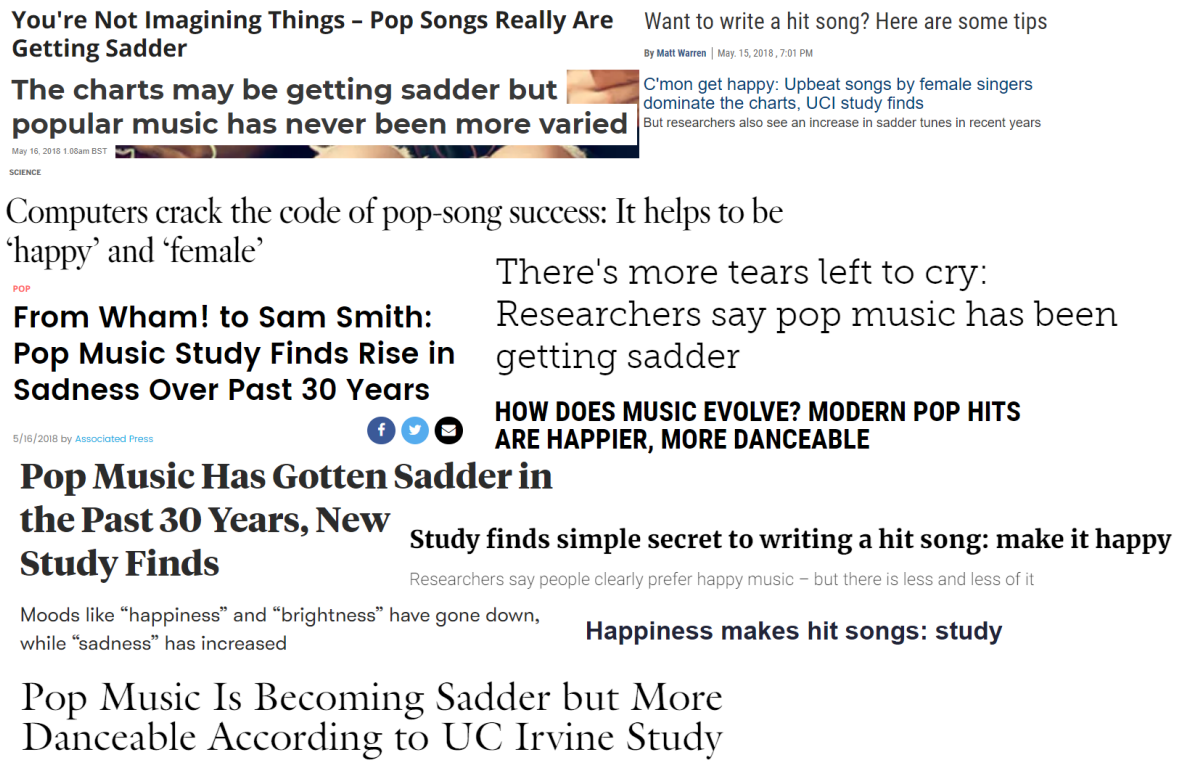


Figure 1.2: Media coverage based on one published paper [71] that used high-level music classifier data to draw a conclusion.

mance of a classifier in terms of stability?

  - **RQ1.1**: Using stability metrics, how does the representation of the audio (e.g. bitrate used, codec used) influence high-level music classifiers?
  - **RQ1.2**: Using stability metrics, how does software versioning (e.g. software version used to calculate features) influence high-level music classifiers?

- **RQ2**: How can outputs from multiple implementations of classification tasks be leveraged to quantify the performance of a classifier in terms of agreement?
  - **RQ2.1**: Using agreement metrics, how does the representation of the audio influence high-level music classifiers?
  - **RQ2.2**: Using agreement metrics, how does software versioning influence high-level music classifiers?
- **RQ3**: How can evaluation methods for such classifiers be improved using techniques from other disciplines like psychology and software testing?

The answer to **RQ1** will result in a metric that will give a general idea of how robust a classifier is to noise, or small differences in the representation of the input data. If it is observed that a classifier is unstable (i.e. when presented with many different representations of the same input, they produce many different labels), then this might indicate poor performance of the classifier. Such a metric can be applied to classifiers in all domains, as long as it is based solely on input-output pairs.

For the specific case of high-level music classifiers, subquestions **RQ1.1** and **RQ1.2** can be explored. These question will give a general idea on which properties of the audio influence the stability of the classifier, which can provide interesting information for both improving current high-level music classifiers (if a classifier, for example, always performs worse on a certain codec then further research into how we can improve the classifier on this codec can be conducted) or about in which settings the classifiers can be used reliably (if a classifier, for example, is only stable on lossless audio, then we can make sure that if we use this classifier as is in some future research that we use lossless audio).

Since actual ground truth labels are not available for unseen data, and even data in the training set might have ambiguous labels (as will be explored in Chapter 2), relying on the available labels in training data for quantifying classifier performance can be troublesome. However, by looking at outputs of different implementations of the same classification task we can leverage this redundancy to quantify the performance of the classifier. The metric resulting from **RQ2** will give us more insight into the construct validity of classifiers by studying how the outputs of different implementations correspond with each other. Since different implementations of classification tasks might have a different design or be trained with different data, chances are relatively high that when both classifiers agree with each other for the majority of the inputs we present to them that they have modeled the desired construct correctly. If the agreement between the different implementations is low, then chances are relatively high that both classifiers modeled some other property of the data instead of the desired one. While this does not give any guarantees about the correctness, — if agreement is high between two different implementations of a classification task, they might *both* be wrong — very low agreement scores would further indicate poor classifier performance. Again, for the specific case of high-level music classifiers, **RQ2.1** and **RQ2.2** are explored to find out how audio representation affects this metric.

While **RQ1** and **RQ2** present the first steps towards a new approach to studying the performance of machine learning classifiers based on input-output pairs, **RQ3** provides a framework for future research to further solidify these methods. The answers to this research question allow us to gain more insight into the different factors and how they influence the performance of classifiers based on literature from the field of software testing, and show how these validation methods can be applied to classifiers in many different domains.

1.3. Approach and Thesis outline

The main narrative structure of this thesis follows a U-shape as presented in Figure 1.3. Chapters 1 and 2 serve as the higher-level theoretical background with Chapter 1 giving a general introduction to the problem statement while Chapter 2 will provide background on general machine learning and classification, the specific field of music information retrieval and on the technical details of digital audio representation to highlight the many challenges that *all* machine learning approaches have to deal with before exploring the additional challenges to working with multimedia data and audio signals. The final section of Chapter 2 will provide an example of a high-level music classification task, that of Music emotion recognition, and will further explore the challenges that arise when dealing with music data. These first two chapters highlight the problem of classifier validation using 'ground truth' labels to quantify performance and motivate the need for validation methods that do not depend on such ground truth labels and validation sets.

After the explanation of these challenges regarding machine learning and the more specific application to audio signals, Chapters 3 and 4 present metrics to quantify classifier performance that do not depend on labeled validation data and are instead based on stability (answering **RQ1**) and agreement (answering **RQ2**), applying them to existing high-level music classification datasets.

Chapter 5 provides an even lower-level analysis based on the underlying label probability distributions produced by classifiers, motivated by the results presented in Chapters 3 and 4. Then, using the answers to **RQ1** and **RQ2** obtained from this analysis, Chapter 6 provides a higher-level analysis of the performance of these classifiers by trying to control for as many of the effects that have been observed in the previous chapters.

Then, Chapter 7 will answer **RQ3** by giving general recommendations on how future research might be conducted given all the observed results and the answers to **RQ1** and **RQ2** by presenting a framework with which classifiers can be tested that is based on a second literature study of software testing techniques. Finally, Chapter 8 will place these results in a broader context by analyzing what the results presented in this thesis mean for the field of machine learning as a whole, what can be learned from them and where the field should go from here, concluding the thesis.

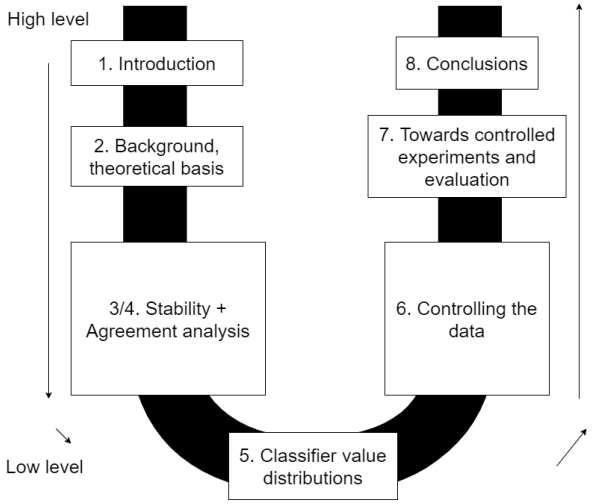


Figure 1.3: Structure of the thesis: starting at the high-level, then exploring the more low-level details specific to the music classifiers in the AcousticBrainz dataset before placing the results of this analysis in a broader context by drawing conclusions applicable to the broader field of machine learning.

2

Background

This chapter aims to give a general overview of the challenges in implementing machine learning classifiers. The analysis presented in this section will first focus on the challenges encountered in the more general task of machine learning. Then, challenges specific to the field of Music Information Retrieval (MIR) are presented and finally the (multi-disciplinary) challenges of the specific task of music emotion recognition are highlighted.

The main goal of this chapter is to illustrate the many challenges that need to be addressed in MIR, which highlight the importance of awareness, as well as the need for proper validation methods for classifiers in the field.

2.1. Machine learning and classification

Before diving deeper into the challenges that are specific to the field of Music Information Retrieval (MIR) and the specific task of high-level music classification, it is worth taking a closer look at the challenges that are inherent to all forms of machine learning. This section will describe three approaches to machine learning (deductive, inductive and transductive) and discuss their advantages and disadvantages in terms of performance, interpretability and data need. Then, the bias-variance tradeoff in machine learning is analyzed and the different validation methods and their problems are discussed.

2.1.1. Learning approaches

Machine learning is a form of *inference*: we wish to draw some conclusion based on the gathered evidence, or dataset. When making a logical argument, there are different ways of reasoning resulting in different approaches. Three approaches, which include examples of machine learning algorithms that employ these approaches will be further discussed in this section.

Deductive learning Deductive machine learning approaches follow the principle of deductive reasoning. It uses rules or hyptheses to deduce new knowledge. Using the classical syllogism structure [138], say we have the following rules or premises:

1. All cats are clever.
2. Lily is a cat

By combining these rules, we can now deduce that Lily is clever. If the knowledge used to formulate these rules is provably correct, then the deduced conclusion will also be provably correct. Early machine learning systems like expert systems aim to emulate this decision making ability automatically by encoding many such rules with the help of a human expert to automatically deduce new knowledge [73]. Such systems are essentially large decision trees, with each node applying one of the rules to reach the final label.

The important thing to note with this approach is that these rules, and thus the decision tree or whichever other model is used to encode these rules, are built directly using provably correct knowledge [147], and this model is then applied to new, unseen data to deduce the desired knowledge.

Inductive learning Inductive machine learning approaches employ inductive reasoning. Contrary to deductive reasoning, which is certain to result in a correct conclusion as long as all the premises used to reach this conclusion are correct, inductive reasoning allows for the possibility that the conclusion is false, even if the premises are all true [63]. Thus, inductive reasoning can result in weak arguments if the probability of the conclusion being false is relatively high.

In machine learning, this form of inductive learning uses examples (or training data in the dataset) to build a model that generalizes for unseen data. This model is not provably correct since the inductive reasoning results in a certain degree of uncertainty about the correctness. The main assumption can be described as follows:

> "Lacking any further information, our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning (...)" [98]

Examples of machine learning algorithms that employ the inductive approach are data driven approaches like Support Vector Machines [31], or Deep Learning algorithms [88] which are very popular right now.

For inductive learning, no explicit expert knowledge about the data is needed to build the machine learning model, since it is induced from the training examples. The downside is that there is a certain degree of uncertainty due to the inductive nature of the process: there will never be a 100% guarantee that the model will be correct.

Transductive learning A third approach is that of transductive learning. A transductive learning approach does not use inductive reasoning to first build a model from which the value of interest is then deduced, but instead directly calculates the value of interest from the examples [51]. One such example of a learning algorithm that uses transductive inference is the k-nearest neighbors algorithm [9], which classifies a point of interest by looking at the k examples that are most similar to this point, classifying it as the majority class in these examples. Since this transductive approach does not learn a generalized model, the data is used directly every time a new prediction is required.

These three inference methods are related to each other as demonstrated in Figure 2.1. Note that for most inductive machine learning algorithms, the whole process involves induction (building a generalized model from the training examples) as well as deduction (using the rules established in this model to deduce the value of interest). Expert systems do not use induction on training examples to build a generalized model, instead drawing upon the specialized knowledge of an expert to directly build the model that can be used for deducing new knowledge.

2.1.2. Performance vs. interpretability

Historically, expert systems were one of the first successful forms of Artificial Intelligence [127]. This makes sense, since by directly building the model using verified logic, the inductive inference step is skipped. As long as the model is built using verifiably correct knowledge, the deductions made using this model will also be correct (due to the nature of deductive reasoning). When these expert systems were popular — they were created in the 1970s but became popular in the 1980s [89] — storage capacity was scarce and computing power was low compared to now. In contrast, inductive inference relies heavily on the quality of the training samples. As an example, say we wish to minimize the standard error of the mean of our collected data to improve our inductive reasoning. Since the formula for the standard error of the mean is:

$$\sigma_{\bar{x}} = \frac{s}{\sqrt{n}}$$

with s the sample standard deviation and n the amount of observations,

we can either collect better data (lower s) or collect much more of it (increase $\sqrt{n}$). Thus, to get good quality results using inductive approaches it is crucial to have a large amount of data, which was simply not available at the time.

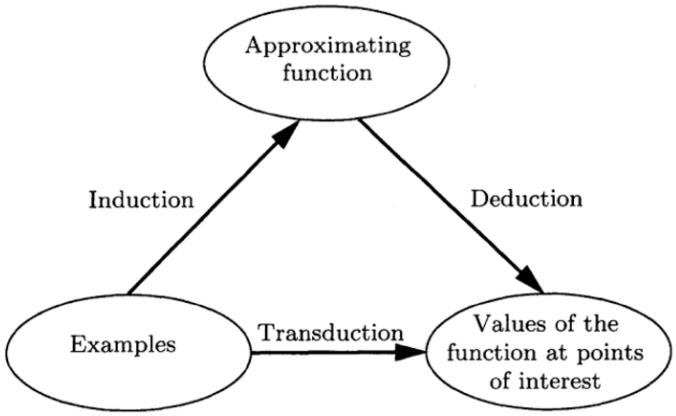


Figure 2.1: The relationship between the three described learning approaches. [145]

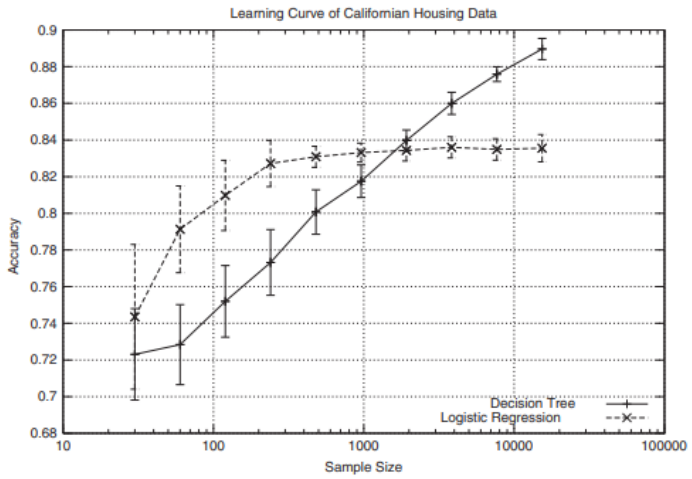


Figure 2.2: Effect of an increasing sample size on the accuracy of relatively simple classification algorithms. [111]

Why then has machine learning now adopted the inductive learning approach as one of the most popular approaches? Due to the growing amount of data available (with the amount of data forecast to reach 175 ZB in 2025 [122]) classical classification algorithms that might have historically performed rather poorly can now perform quite well: see for example Figure 2.2 where the relatively simple decision tree algorithm can perform much better when the sample size increases substantially. In addition to the increase in data, due to the increase in computing power (owing to Moore's law [100]) more complex induction algorithms like Deep Neural Networks allow for even better classification performance [88]. In addition, the expert knowledge required for directly building a rule-based system often is not available (take for example the bird detection example in Chapter 1), further motivating the need for inductive approaches that try to learn a generalized model for deduction from example data.

However, as conventional wisdom states: "There is no such thing as a free lunch". By investing effort into developing more complicated inference methods the resulting models are less transparent to manual inspection. The problem with these black box models is that is very difficult to truly understand what the model has learned and which aspects of the data are used to reach the conclusions:

> "The problem is that the knowledge gets baked into the network, rather than into us." [25]

Many of these unexplainable, black-box models seem to work just fine, with many of the current state

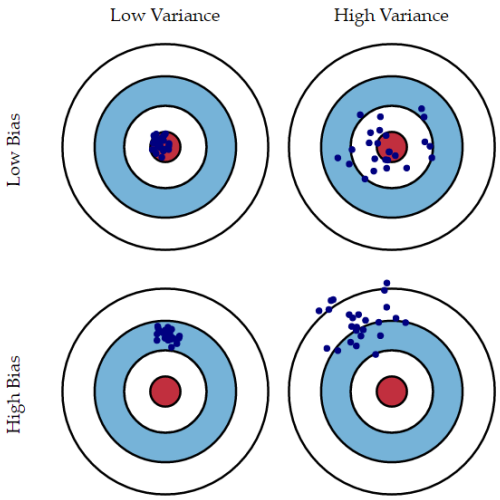


Figure 2.3: Bias vs. variance for machine learning classifiers illustrated. [74]

of the art models being unexplainable but high performing[1], however due to the non-transparency of these models we do not understand *how* they make their decisions and it becomes a very difficult task to explain these models after they have been trained [125].

2.1.3. Bias–variance tradeoff

Another general challenge in machine learning is the challenge of balancing the bias and variance of the trained model. Generally speaking, say we have a machine learning classifier that tries to classify input data as belonging to either class A, B, C or D for which we have $45, 40, 10$ and 5 examples, respectively. Since we use inductive reasoning to build a model from these examples, we want to fit a model to these data points that will generalize well to unseen data. Training might go as follows:

1. Our initial algorithm inherits a large *bias* from this data, since it sees that guessing either label A or B is correct for most of the data and consequently does not learn enough from the data to be able to detect classes C or D. It *underfits* the data by simply looking at the frequency instead of the features we are interested in.

2. To prevent this, we might increase the complexity of the inferred model to decrease the bias. However, this increasingly complex model might also start considering random noise in the data as important to the classification, thus increasing the *variance* by overfitting on the data.

Visually, this difference between classifiers with high and low biases and variances is often illustrated as in Figure 2.3: a classifier with a high bias is likely to be wrong (off-target) when applied to unseen data and a classifier with high variance is likely to include a lot of class-noise (i.e. errors in classification) when applied to unseen data. It is worth noting that there is some discussion about if this trade-off is universal or only applies to certain machine learning algorithms [103], but the main conclusion remains the same: to train a classifier that works as we would hope we need to make sure that both the bias and the variance of the classifier are as low as possible.

Bias in data While the complexity of the machine learning algorithm might play a large role in the bias of a classifier, as described above, such bias can also arise from the data the algorithm was trained on. Due to the fact that most current machine learning approaches rely on induction — hypothesising that the seen examples generalize to all cases — representative samples need to be fed to the algorithm to prevent data-driven bias.

Possible biases might include an imbalanced dataset: in the case of a music classifier this might be the overrepresentation of certain genres in the training dataset. But bias might also arise due to human

[1] A majority of the state of the art approaches uses a deep learning approach or other approach which is not interpretable: `https://paperswithcode.com/sota`

factors: when crowdworkers are tasked with labeling data they might be influenced by psychological effects like the bandwagon effect [15] where workers might follow behaviour of a group instead of trusting their own decisions or ambiguity effects where certain options become less attractive if missing information is perceived [45].

2.1.4. Validation

As described in Section 2.1.3, classifier performance can depend on more than just the accuracy of the machine learning algorithm. Other factors like the training data are just as important. However, due to the non-transparency of many of these high-performing machine learning classifiers it becomes very difficult to verify how well they actually work.

There are many ways of validating a machine learning classifier, the simplest of which reports the classification accuracy on some test dataset (which was left out at training time) as simply:

$$accuracy = \frac{correct\ classifications}{total\ predictions}$$

While this metric is simple to understand, it might give a false sense of 'safety': a high accuracy on a test set only indicates that the classifier works well *on that specific set*, with no guarantees on how well the classifier will work with 'in-the-wild' data. If the data used to train a classifier and the data used to calculate the accuracy both include some bias, then the accuracy reported will be high, but the resulting classifier will have inherited this bias from the training data. If then this bias is not present in 'in-the-wild' data, the classifier might not perform as well as the accuracy reported in the lab might lead us to believe.

While some validation methods like k-fold cross-validation aim to counteract this dependency on the validation data used by calculating the accuracy on multiple different 'folds' of the data, this metric still suffers from accuracy problems due to high variability [13] and requires ground-truth values necessitating the collection and labeling of data which might include human and data biases.

Measuring how well any given classifier is performing on real-world data can thus be challenging, given that most validation methods require some sort of ground-truth answers. Other approaches that do not require ground-truth to be available might be gathering user feedback to assess and retrain the learned model [136] or creating interpretable machine learning architectures which allow for more insight in how the algorithm makes the decisions [153]. Such interpretable machine learning has only recently started to become popular, leaving many open questions about how this evaluation can be carried out [153].

2.2. Music information retrieval

The primary goal of MIR is to facilitate access to the large amounts of music that are available digitally in a similar way to Information Retrieval (IR) systems, which focus solely on text data. Concisely, researchers in the field of MIR wish to make it possible for users to search for music not only using text-based queries (like the title of the song) but also by using musically framed queries [41]. Examples of such musically framed queries might include the user singing a snippet of a song which they do not know the title of to find it, or searching for music that is congruent with their current mood.

Music data can be challenging to work with, there are many different representations, music is experienced differently by users depending on factors like their mood and preferences and the field of MIR requires multidisciplinary knowledge. These domain features of the MIR field will be further elaborated on in the following sections.

2.2.1. Representation

Music can be represented in formats with varying levels of structure. The most unstructured way of storing music is by storing a digitally encoded version of the analog sound wave. Due to the unstructured nature of this data, it needs to be transformed into some structure to be able to query it. Different ways of giving structure to this data include: an embedding layer to get a representation of the audio [32], calculating low level features of the audio to represent it as a vector of numerical data, representing it as time-stamped events like MIDI encoding [5] or as musical notation. Excluding the approaches that require machine learning to calculate features or embedding layers, the basic representations of music are visualized in Figure 2.4.

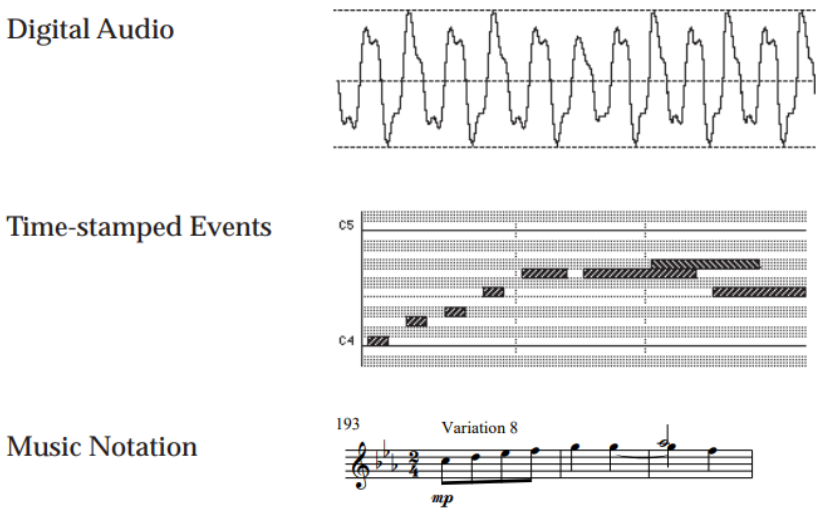


Figure 2.4: Three basic representations of music data. From top to bottom more structure is added to the respresentation. [22]

.

In addition to the different ways that the same piece of music can be stored digitally, a piece of music can also be represented in different ways before it is digitized. Live versions, covers, remasters and rerecordings are all different representations of the same song, resulting in different audio signals and representations.

There are many different representations for music, and this adds a challenge to developing MIR applications. Choosing a representation that includes all the relevant information about the music for the task at hand, while keeping storage and computation costs feasible (MIDI files are much smaller than digital audio, but probably do not include tiny nuances expressed by the musician in an audio recording) might be difficult. In addition, the concept of a piece of music having many different representations like cover versions adds additional challenge to MIR compared to IR — in IR a word written down by a different author will be the exact same word, in MIR the same recording performed by a different performer can result in an entirely different audio signal.

2.2.2. Experience

Another unique feature of music data compared to for example image data of an object, is the artistic expressiveness of the data. Working with music data requires modeling of the abstract properties of music that make us feel and enjoy it, with no objective measurement. Take for example the task of classifying the emotion present in music (described in greater detail in section 2.4). This task requires interpretation of not just the audio data of the music itself, but also of the experience and the perception of the data. Instead of training on an objective task (i.e. is this a picture of a bird) it involves modeling and predicting how people perceive and react to music (see Figure 2.5).

Due to the nature of music data — it is a form of artistic expression, which is perceived differently by people in different moods [57][140], with different personalities [123], from different cultures [10] or even due to neurological differences [62] — this human factor always plays a role and should be taken into account.

2.2.3. Multidisciplinarity

Finally, partly due to this human factor present in the interpretation of music data, the field of MIR requires multi-disciplinary knowledge. Given the many different modalities of music data (e.g. raw audio which would require some signal processing background, lyrics which would require some background in Natural Language Processing), the effects that music can have on people and how they choose what music to listen to (requiring knowledge from psychology, cognitive science etc.) and the more 'classical' computer science problems like handling large datasets and designing efficient algorithms to turn this data into something useful (e.g. using a machine learning classifier to output some label from

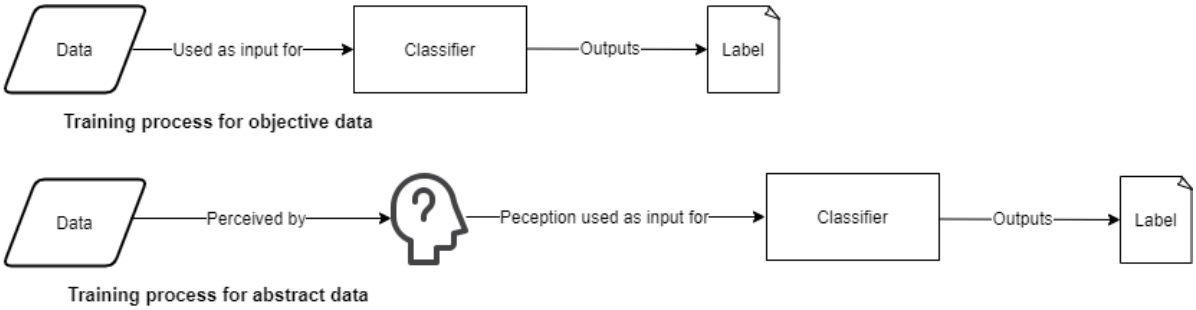


Figure 2.5: Classifier pipeline difference for objective and abstract data.

the audio data). Futrelle and Downie [49] present a summary of these different disciplines, however it is reasonable to assume that this list is not exhaustive given the many different contexts and environments that music might be applied to.

This marriage of all these different disciplines is what makes the field of MIR exciting, but also what poses a challenge. Successful MIR research should not focus on just a single discipline: a computer scientist might be able to perfect a given machine learning pipeline, but without critical inspection of the representation or human interpretation of the input data such a pipeline will probably underperform. The abundance of different fields and techniques (and their dependencies) that are often required for MIR tasks, of which a simplified overview is given in Figure 2.6, illustrate the importance of having such diversified knowledge.

The challenge, then, is to have a clear overview of both the disciplines that are involved in any specific MIR task and to take the specialized knowledge from those fields into account when conducting research in this field either by broadening individual knowledge or by cooperating with different disciplines.

2.3. Digital audio representation

Our perception of music relies on how our ears interpret analog sound wave information. Computers however are not able to handle 'true' analog data since the data needs to be stored using binary bits of information. Thus, to be able to use musical audio data as input for a machine learning algorithm, the analog audio needs to be represented digitally. This digital audio coding chain can be seen visually in Figure 2.7. Note that the process of audio encoding and decoding is based on human perception of the resulting audio signal: the goal is to maximize the perceived quality at the end of the pipeline, while also minimizing the amount of information needed to represent the audio signal [18].

The simplest encoding method, used for storing music on CDs, is pulse code modulation (PCM) [18]. PCM samples the analogue audio in a regular interval (sample rate) and then the amplitute of this sample is quantized to the nearest value using N bits (bit depth). This process can be seen visually in Figure 2.8. For digital audio the term bitrate is often used instead of sample rate and bit depth, and since bitrate is defined as the amount of bits that are processed per unit of time [60] this results in:

$$bitrate = samplerate \times bitdepth$$

CDs for example are encoded using a sampling rate of 44100Hz and a bit-depth of 16 bits per channel, with 2 channels used for stereo audio [72]. In theory, increasing either the bit-depth or the sample rate will result in a more accurate representation of the analog audio. However, even with the standard CD parameters, one second of stereo audio already uses $44100 \times 16 \times 2 = 1.4112$ Mbits of data.

Audio compression In an effort to reduce storage requirements for digital music, this digital audio representation can be compressed, often with the goal of being able to store more music and reduce transfer times of music over the internet [94]. Many different algorithms for coding and decoding data in an efficient way exist. These systems are often called *codecs*, short for coder-decoder[2]. Different codecs can take different approaches for compressing the audio, and they can be categorized as:

[2] https://www.merriam-webster.com/dictionary/codec

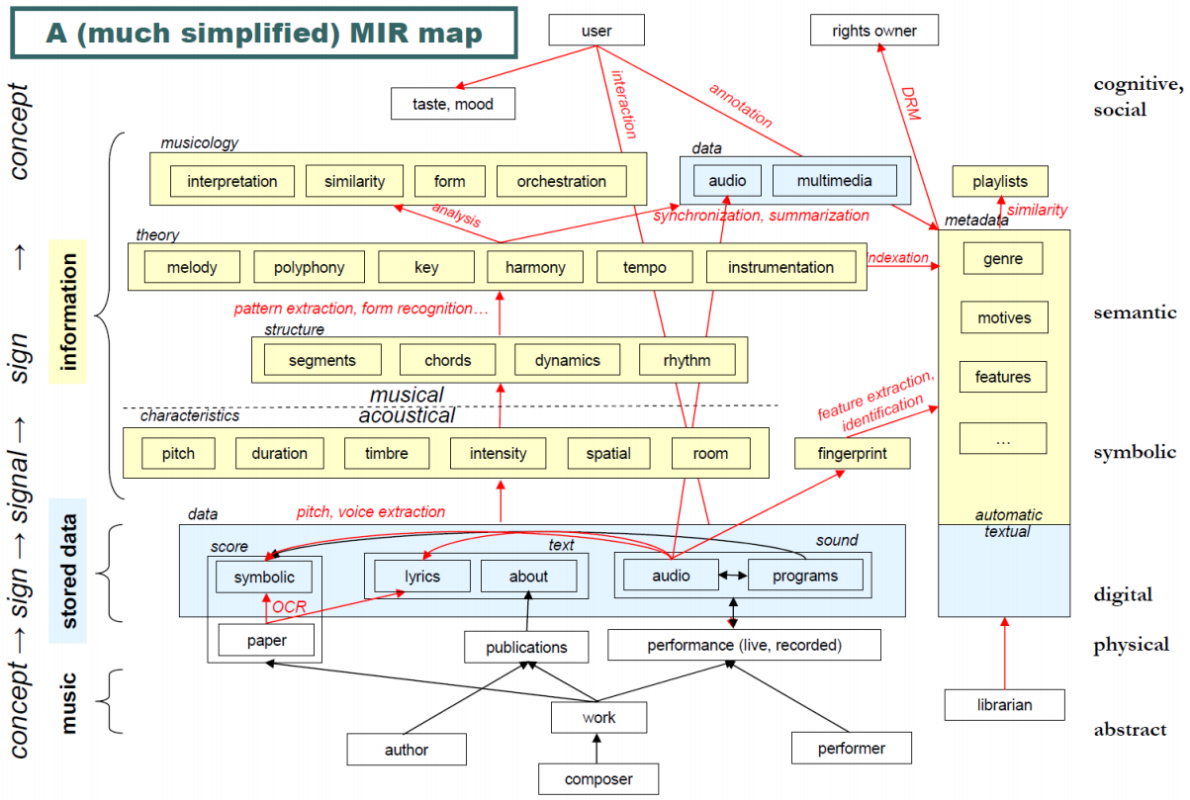


Figure 2.6: A simplified map of concept knowledge required for MIR tasks and their dependencies. [47]

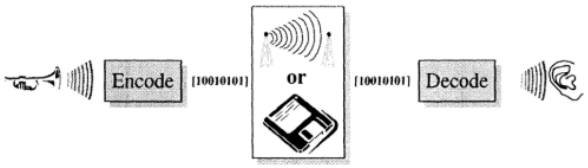


Figure 2.7: The digital audio coding chain. [18]

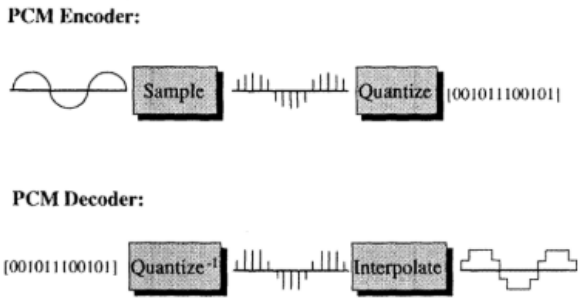


Figure 2.8: The Pulse Code Modulation process. [18]

- *Lossless* codecs, like FLAC[3], which compress the PCM data from a CD source without loss of quality
- *Lossy* codecs, like MP3[4] or Vorbis[5] which alter the signal in a way that does affect the resulting quality, but compresses the signal more than lossless codecs

Lossy codecs like MP3 use a principle called *perceptual coding* [76] to remove certain parts of the audio information that are, for example, beyond the range of human hearing before further compressing the signal. Because these approaches are based on human perception, the algorithms are designed and evaluated using listening tests [34]. Due to the subjective nature of these tests, the encoded signals might be very different. Note that the algorithms are designed based on maximizing the quality based on *human perception*, thus it is conceivable that different encoding approaches have a different effect on the interpretation for a machine learning algorithm.

Codec differences To illustrate that similar sounding music fragments can have very different encoded audio signals based on the codec and encoding parameters used, several short music fragments encoded using different codecs and bitrates[6] were visualized as a spectrogram using Spek[7] in Figure 2.9. When looking at this figure it seems conceivable that some high-level classifiers might pick up on these differences in the audio signal.

2.4. Music sentiment analysis

An example of such a task in the MIR domain is that of music sentiment analysis. This task lies on the intersection of multiple disciplines, with both a psychological and data science angle as described in section 2.2.3. Thus, it is very important to have a good overview of the various psychological factors that play a role in designing such a classifier. This section aims to analyze this specific task as an example that illustrates the many different challenges in dealing with such a multidisciplinary task. First, a formal description of the music sentiment analysis task will be given. Then, several different applications will be explored. After the task is introduced, the psychological aspects at play will be discussed to illustrate the multi-disciplinary knowledge required for executing the task successfully. The factors discussed in the subsequent sections are: the challenges of modeling sad music, the influence of extramusical factors like memories, interpretation issues regarding the labels used for both training and further analysis, and the different ways in which these factors can be handled in designing a classifier.

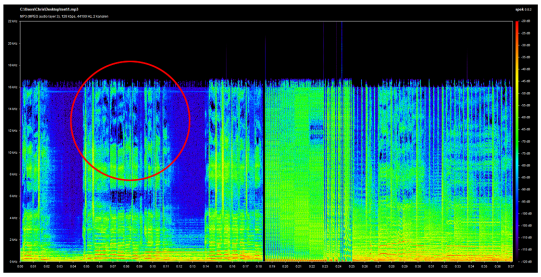
2.4.1. Definition

Music sentiment analysis is a domain specific version of the broader definition of sentiment analysis coined by Picard [114], who defined it as a field of science related to sentiment, derived from sentiment or exerting influence on sentiment. In the general case, sentiment refers to a feeling, attitude, evaluation or emotion that is associated with an opinion [23], however in the music sentiment analysis literature the terms emotion and sentiment are often used interchangeably. This results in two different tasks that both fall under the umbrella of music sentiment analysis:

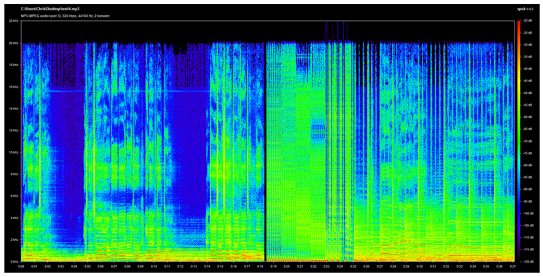
- The task of automatically classifying the emotion that is communicated by the music, i.e. the emotion *perceived* by the listener in the music.
- The task of predicting which emotion is *induced* by the music in the listener. This definition better corresponds to the notion of sentiment, since the opinion of the listener about the piece now also plays a role.

Both tasks have a common machine learning pipeline, which can formally be described as follows: given an audio signal, a, as input, output an emotion label l according to some psychological emotion

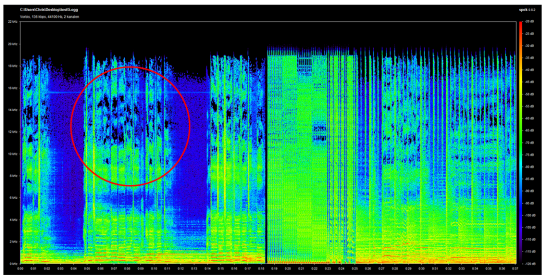
[3] https://www.loc.gov/preservation/digital/formats/fdd/fdd000198.shtml
[4] https://www.loc.gov/preservation/digital/formats/fdd/fdd000012.shtml
[5] https://www.loc.gov/preservation/digital/formats/fdd/fdd000117.shtml
[6] http://nigelcoldwell.co.uk/audio/
[7] http://spek.cc/



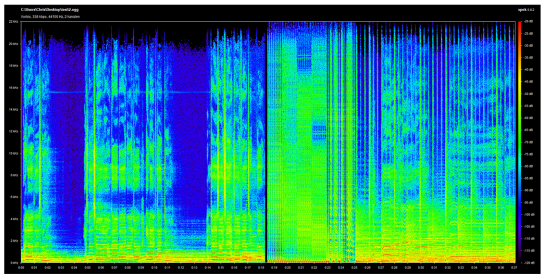
MP3, 128kbps CBR



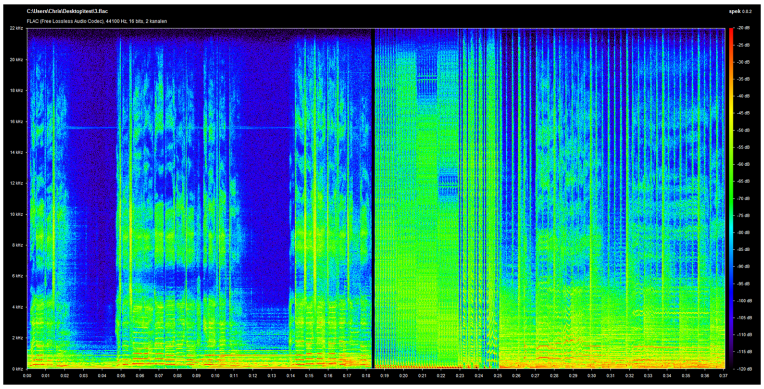
MP3 320kbps CBR



OGG Vorbis ~128kbps VBR



OGG Vorbis ~320kbps VBR



FLAC lossless

Figure 2.9: Spectrograms for five different encodings of the same CD quality music fragment. The x-axis is time, the y-axis is frequency and the color is amplitude (with hotter colors being a higher amplitude). The red circle illustrates one of the differences between codec signals, while the bitrate is roughly the same for both signals, the OGG Vorbis encoded fragment has more information missing in the red circle and includes more of the higher frequency data.

model that best describes either a or the emotion induced by a. Optionally, additional contextual data (C) might be used to better predict the perceived or felt emotion from the audio data.

$$a(+C) \xrightarrow{emotion\_classifier} l$$

Often times, the audio signal is not directly mapped to the emotion label, but instead some form of feature extraction is applied to the audio signal to get low-level features (LL) like timbral texture [142] or loudness [56]. These extracted low-level features can then serve as the input for the classifier that outputs the emotion label:

$$a \xrightarrow{feature\_extraction} LL(+C) \xrightarrow{emotion\_classifier} l$$

Converting the original audio signal to a collection of low-level features that describe the audio is also useful for legal reasons. Due to copyright laws, directly sharing music data is not legal [79], however sharing the low level features is permitted. This allows projects like AcousticBrainz [118] to gather a large collection of low level features which can then be used to train the emotion classifier without having to run a feature extractor.

The analyses in the subsequent chapters will mostly focus on the high-level classifiers like the emotion classifiers as described here. Since these high level features are subject to psychological interpretation, a study of their performance can yield interesting insights for high-level feature classification using music data. Low level features should in theory be more stable. Projects like Essentia [16] already provide implementations of them and they have already been analysed [143].

2.4.2. Use cases

Music sentiment analysis methods can potentially have many different applications, and depending on the context in which they are used it might be very important that the classifiers operate accurately and under the right assumptions. This list of use cases is not meant to be comprehensive, but illustrates that music sentiment analysis can have many different approaches, ranging from commercial applications to healthcare. Depending on the use case, it can be very important to know how much trust can be put in the results from the classifiers.

Music recommendation One possible, low risk application of music sentiment analysis is for the task of music recommendation. By inferring the current mood of the listener — either from the perspective of analysing what music they choose to listen to, or from the perspective of how they probably feel after listening to certain pieces of music — it becomes possible to recommend music that is congruent with their current mood, increasing the chance of the user liking the recommended piece. Other approaches might explicitly ask the user which mood they wish to be in, and guide them towards that mood by selecting appropriate pieces of music [61]. These applications do not necessarily have to be limited to a single listener. If the communicated mood in pieces of music can be accurately estimated this might also help in certain creative processes like selecting an appropriate soundtrack for a trailer based on the mood of the scenes or automatically generating accompaniment that communicates a certain mood for a given melody [27]. Depending on the application, the severity of making an error is relatively low: if a piece of recommended music is disliked a user can simply skip it, and if a selected soundtrack does not fit, a different one can be chosen.

Opinion prediction An application closer to the definition of sentiment as described in section 2.4.1 might be to facilitate opinion mining on pieces of music. It might be desirable to predict the emotional response or opinion of a listener to a piece of music before it is officially released. If such a system would work accurately, it would become possible to tweak the music to get the desired response or positive reaction to your work. In essence, it could provide 'pre-emptive feedback' on if the song communicates the emotions that you wish it to. These applications are a bit less fault tolerant than the recommendation approaches, given that a wrong prediction might cause the final work to be received less favorably.

Music therapy Music therapy is defined as:

> "The use of music in clinical, educational and social situations to treat clients or patients with medical, educational, social or psychological needs." [148]

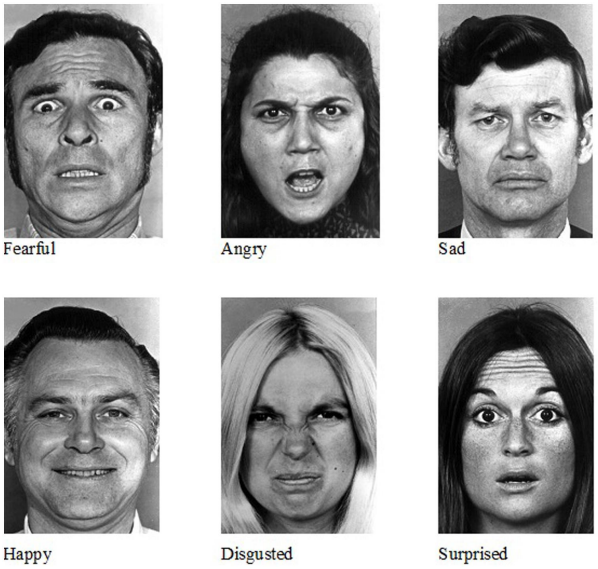


Figure 2.10: Examples of the facial features corresponding to six basic emotions in Ekman's emotion model [86].

which can include listening to music to influence the mood and behavior of a client [148]. If a classifier can accurately predict which emotions a piece of music might induce in a listener, then this could help with selecting appropriate music for therapy purposes. It can be beneficial for the effectiveness of the therapy to take user preferences into account [65] and by being able to predict the emotional response of the client the selection of music to use can become easier. However, because therapy directly deals with the mental health of a client, errors made by the classifier can have ethical ramifications.

2.4.3. Emotion models

To facilitate computational emotion recognition, complex human emotion needs to be transformed into some simpler representation using an emotion model. Some of the models might be better a better fit for the task of music sentiment analysis than others. This section will give a brief overview of the different kinds of emotion models and discuss their correctness for the sentiment analysis task. A distinction can be made between *discrete*, *continuous* and *hybrid* models.

Discrete models This approach to emotion modeling, originally proposed by Darwin, treats emotions as separate discrete entities that were evolved traits universal to the human species [36]. This same ideology is adopted in in the work of Ekman, who introduced and showed the notion of six basic discrete emotions, resulting in a classification of one of these six emotions based on facial features [43]. Ekman later added more emotions to this basic emotion model, arguing that these basic emotions like fear, happiness and sadness are primitive emotions that are 'hardwired' into our brain through evolution [44]. Examples of the facial features corresponding to the different basic emotions according to Ekman's model can be seen in Figure 2.10. This model defines very utilitarian emotions which facilitate fast and automatic reactions to increase the chances of survival, more complex emotions like boredom or annoyance are not directly modeled in this basic emotion model.

Dimensional models To be able to model more complex emotions, dimensional models allow for continuous values that fall between discrete emotion labels. Perhaps the most well known model is the two-dimensional circumplex model of affect by Russell [126]. This model consists of two dimensions: one for arousal (or degree of excitement) and one for valence (from negative to postive). According to this model, almost all emotions can then be mapped to (x, y) coordinates in this two-dimensional space. See Figure 2.11 for an illustration of the original circumplex model by Rusell. Because of the continuous nature of this model, other complex emotions like boredom can be modeled as a combination of the arousal and valence dimensions, falling somewhere in the bottom left quadrant with negative valence and low arousal values.

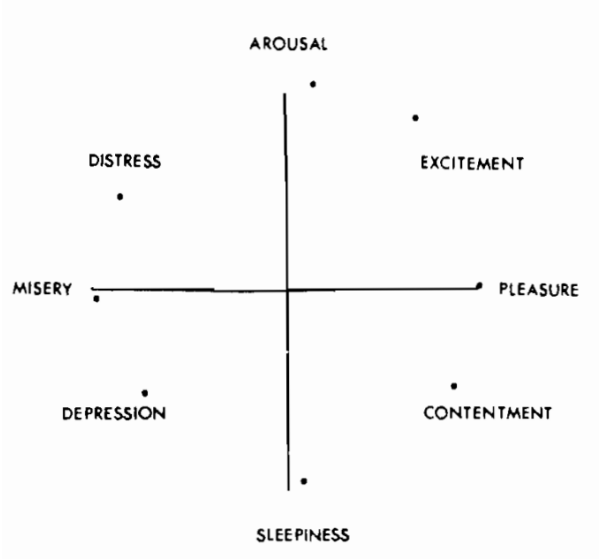


Figure 2.11: Circumplex model of affect by Russell, the X-axis represents valence (from left to right, negative to positive and the Y-axis represents arousal (from bottom to top, low to high). Eight affect terms have been mapped in this space. As an example, 'excitement' is high arousal and positive valence. [126]

.

It can be more difficult to classify and name specific emotions using dimensional models. In theory, all possible emotions are present in the dimensional space, but we do not have an infinite amount of labels. Thus, results from a dimensional model might be more difficult to interpret than the discrete models. If we would be able to combine the properties of the dimensional model with the finite amount this limitation in interpretability might be overcome.

Hybrid models Hybrid emotion models such as the wheel of emotion by Plutchik [116] combine properties of the discrete and dimensional models. In his model, Plutchik [116] defines eight basic emotions. Each complex emotion can then be modeled as a combination of multiple basic emotions, with the intensity being modeled using a dimensional scale. This effectively combines the discrete labels from the discrete emotion models with the continuous values in dimensional models. The resulting model is then able to capture more complex emotions as combinations of the basic emotions, with a varying degree of intensity. The resulting emotion space is three-dimensional, see Figure 2.12 for a visualization.

Some of the described models seem to be a better fit for the task of music sentiment analysis then others, however it can be argued that all models have their limitations. Of all described models, the discrete emotion model allows for the least amount of possible labels, when applying this model for describing emotion in music the hypothesis of having consistent, but relatively biased results seems likely. It could be theorized that when trying to describe a the emotion in a song, someone would pick the emotion label that is closest to what they feel is correct, and with not that many choices the chances of agreement between different descriptions would be relatively high. However, when tasked to label the emotion present in music using a discrete model, results were not significantly more consistent then when using other emotion models [42]. This could be due to the fact that emotion in music is more complex than the basic emotions, resulting in the absence of an appropriate label for many songs. If a song is some combination of happy and sad, then participants are forced to choose between one of the two viable options, resulting in low inter-rater agreement. In addition, it is questionable if models constructed upon the basic emotions are appropriate for music sentiment analysis, given that emotion might be perceived differently in an aesthetic context [131][129][80], and the basic emotion models were built for use in a more utilitarian context.

The dimensional models allow for an 'infinite' number of labels, which in theory would mean that every song can be modeled using these models. The amount of possible values does not affect the performance of this model [42], however this model suffers from the same problems as the basic emotion model as it is built with the assumption of 'real world' emotions, not taking into account the aesthetic

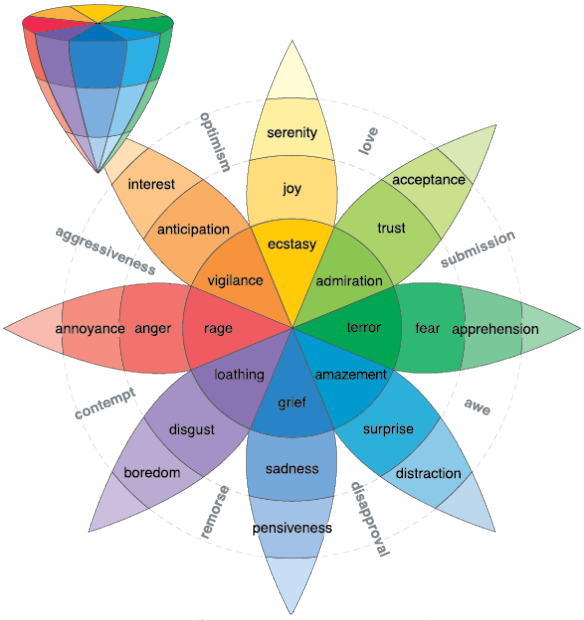


Figure 2.12: Visualization of Plutchik's hybrid model of emotion [116]. Each color represents a basic emotion, darker colors are more intense, lighter colors are less intense variants of the basic emotion. The white regions allow for combinations of basic emotions. Illustration taken from Plutchik [117].

context in which they are applied.

Domain specific models, like GEMS [158] were built specifically with the task of music sentiment analysis in mind, designed to overcome the problem of applying an emotion model in a context it was not designed for [158]. GEMS is shown to outperform the basic emotion as well as the dimensional models for the task of music sentiment analysis, suggesting that domain specific emotion models are the optimal choice. However, GEMS has the drawback of having discrete labels, and thus it might not be able to model all musically induced emotion, especially when the emotion in the music is inherently ambiguous, as will be explored in the following sections.

2.4.4. Modeling of sad music

Differences in the perception of emotion on an individual level make constructing an emotion model for music sentiment analysis even more difficult. The neural systems responsible for the perception of emotion vary from person to person [62] and these individual differences can greatly influence the emotions that music can induce, especially when dealing with negative emotion in music [52]. Of particular interest for this thesis is the modeling of sad emotion in music.

The emotional reaction to sad music can be hard to model because the definition of sad music is not well defined: is music in a minor key sad? Or is music sad when the lyrics are negatively valenced? How then do you label an upbeat happy sounding song with dark lyrics[8]?. In addition, while music with a sad mood is perceived as sad, and thus would have negative valence in a dimensional model, this music generally does not induce negative or unpleasant emotion in the listener [146] or has a limited negative impact [80]. However, some listeners do genuinely feel sadness while listening to music, which might be explained by individual differences in prolactin concentrations [70], differences in association [131], personality [146] or gender [57]. These individual differences in emotion induced by sad music make it difficult to correctly model emotion for all cases.

Kawakami et al. [80] make an important distinction between *perceived* and *felt* emotion for music sentiment analysis, which can facilitate the modeling of emotion for music with a sad mood. Generally, the perceived emotion (i.e. which emotion is *communicated* by the music) can be modeled using a dimensional model [42][158] but due to individual differences in the felt emotion (i.e. which emotion does this music make me *feel*) and the paradox of music with negative valence inducing feelings of

[8]For example: Foster the People - Pumped up Kicks. An upbeat, happy sounding song with lyrics about gun violence `https://genius.com/Foster-the-people-pumped-up-kicks-lyrics`

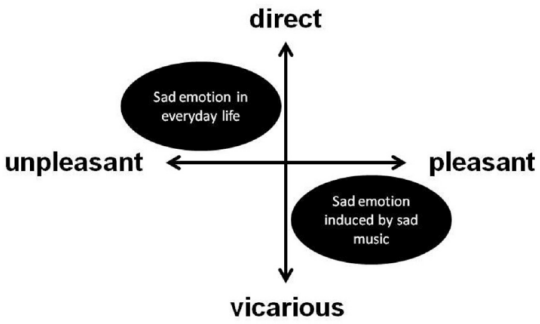


Figure 2.13: Pleasant-unpleasant, direct-vicarious model. [80]

positive valence these dimensional models are insufficient for modeling felt emotion. They suggest a new model of musical emotion which incorporates a ”direct” to ”vicarious” dimension to deal with this paradox (see figure 2.13), however this model is not extensive and instead highlights the problems that arise when using standard psychology models in the application of music sentiment analysis.

2.4.5. Musical memory

Extramusical factors like autobiographical memories [30] can also influence the emotion induced by music on an individual basis, making the modeling of felt emotions even more difficult. As Bower [19] showed, emotion and memory are linked, with most people being able to better remember personal experiences congruent with their current mood. This same principle applies to music, with people remembering more details about songs from their past which invoke some emotion [132]. It is important to note that the direction of causation is unclear (i.e. does the mood invoke the memory or does the memory invoke the mood?). If memories of events from the past that are linked to the music are able to invoke a certain mood in the listener, predicting this in an emotion model becomes impossible without knowing all the memories of the individual or taking this uncertainty into account. As an example, take a generally happy sounding song. While predicting that this song will induce pleasant feelings for everyone seems like a safe bet, for some this song might remind them of a sad or traumatic event from the past like losing a dear friend, which would probably induce negative emotions.

However, this extreme example might not be realistic, since the influence of memories on the felt emotion from music is mostly positively valenced: Janata et al. [75] showed that for music, positively valenced experiences are recalled more readily than negative ones, with music often reminding people of periods, friends or significant others. Reported feelings induced by music linked to memories mostly included happiness, youthfulness and nostalgia. Using this knowledge, the uncertainty of felt emotion due to individual memories might be taken into account by adjusting the results from a used model to be slightly more positively valenced than reported.

2.4.6. Interpretation issues

These ambiguities regarding the definition and modeling of sad music as well as the personal differences also influence a key step in the training of classifiers: data collection and labeling. Any inductive learning approach can only be as good as the data it is provided with, since inductive learning is reliant on the data it is trained on to discover statistical regularities that define the labels that are to be classified. For any machine learning problem, ground truth labels need to be supplied so that the learning algorithm is trained on representative samples for the given labels. While this is generally doable for objective tasks, like detecting objects in a photograph (data can be collected and labeled by for example crowdsourcing, and the labels are generally unambiguous), this becomes much more difficult for the music sentiment analysis task. Ground truth labels need to be assigned to training examples by humans, and if these labels are ambiguous or subject to personal differences then the training examples are probably inconsistent. This inconsistency in the subjective judgement of which labels correspond to which training examples can result in *class noise* in the training data. Even relatively small amounts of class noise can have a large negative influence on classifier accuracy, and the choice in classifier can be very important for robustness to this noise (see Figure 2.14).

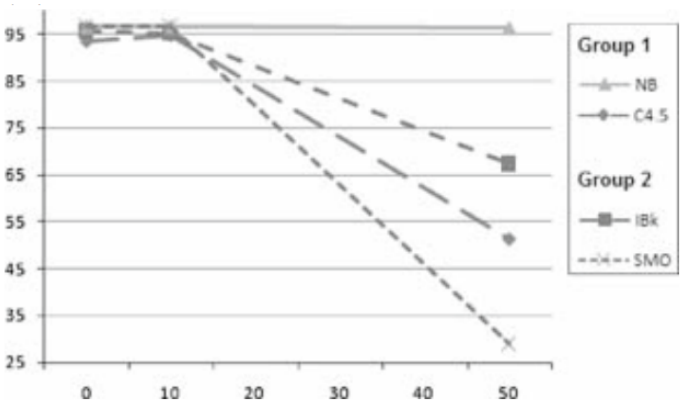


Figure 2.14: The effect of class noise (x-axis) on classifier accuracy (y-axis) using a balanced label distribution. NB = Naïve Bayes probabilistic classifier [78], C4.5 = C4.5 decision tree [119], IBK = IBk instance-based algorithm [8], SMO = SMO Support Vector Machine [115]. [104]

Because the available data for training music sentiment classifiers is sure to contain noise, it is important to prevent the performance impact of the noisy data on the classifier, often this is done either by avoiding overfitting so that the classifier does not try to fit the noise [29][120] or by filtering out noisy training examples from the data entirely [77][20]. However, due to the difficulty of labeling the training data, approaches are often only trained on a few hundred to a thousand training examples [154][85], making removing examples not a very viable option. It is very important then, that a classifier is able to deal with noisy data resulting from the described psychological factors.

In addition, it is important to use the labels output by such a classifier in a psychologically sound way. For example, if we wish to predict the mood of a listener based on mood labels in the listening history, we might be tempted to think that if a user recently listened to more sad music than usual, then they are probably feeling more sad than usual. This might not be the case, however. Some literature suggests that people in a sad mood do not necessarily listen to a larger quantity of sad music and instead opt to listen to a smaller quantity of happy music because overtly happy music feels inappropriate for their current mood [140]. Other literature suggests that people mediate their own mood using entertainment like music and movies, suggesting that people in a sad mood might actually seek out uplifting content to try and repair their mood [159]. Due to these varying motivations for seeking out certain songs, it might be difficult to correctly interpret the generated labels.

Other factors like social, environmental, cognitive and biological factors also influence this same listening behavior [64] making it difficult to attribute it solely to the mood of the listener. Finally, many listeners might not even be actively selecting music they feel like listening to, instead opting to use the shuffle feature or putting on a popular playlist, with variety in music choice and shuffling being an important factor in listening strategies [12]. However, it can also be argued that this is not a problem — listening has generally shifted away from purposefully selecting albums to listen to with most people now listening to music using streaming services — making it important to focus on the interaction of both self and not self chosen music [149]. All of these factors influence the interpretation of the resulting labels, and if they are not taken into account then the interpretation might be flawed.

2.4.7. Application

With the various psychological concepts that might influence the correctness of music sentiment analysis options having been discussed in the previous sections, it is valuable to explore how music sentiment analysis applications might handle these concepts in their models.

Emotion types Due to the factors described in Sections 2.4.4 and 2.4.5 it is very important to clarify which emotion type, perceived or felt emotion, a music sentiment analysis approach is trying to model. If the goal of the classifier is to model felt emotion, then the psychological factors described in the sections above should be taken into account in the model, or at the very least be acknowledged to make sure that the resulting classifier will not be used to draw conclusions that might be incorrect from a psychological point of view. Many of the approaches found in the literature focus on modeling

the perceived emotion in music, this might be because perceived emotion prediction seems to be a more objective, and thus easier task. However, while factors like the emotion source might have less of an impact for the perceived emotion prediction task, individual differences play a role in both how emotion is perceived and felt [62], and thus these psychological factors can not be ignored, even when modeling perceived emotion. If music sentiment analysis papers would explicitly mention the goal being the modeling of perceived or felt emotion, it would be easier to take these psychological limitations into account.

Emotion models The dimensional model seems to be among the most popular emotion models to use for the task of music sentiment analysis. This is in line with the assumption that dimensional models can sufficiently model perceived emotion in music [158]. However, this model is not domain specific and might not be the best pick for the aesthetic context in which it is used. While in theory GEMS seems like a logical pick, many applications opt to use the dimensional model [90][155][38] or some form of discrete classes [102][154][152]. While every model has positive and negative aspects, it can be argued that the dimensional and discrete emotion models are insufficient for modeling the complexities of felt emotion in music. For this task, other emotion models might be necessary, like the Emotion State Transition Model (ESTM) used in Han et al. [61] which is able to model emotion transitions as well as include context information like location, occupation and current mood to more accurately model the correct emotional response. This model is certainly a step in the right direction for the automatic classification of felt emotion, however some contextual information like the current mood of a listener or their occupation might not be readily available, making this a more accurate but difficult to implement approach.

Sad music modeling Handling the difficulty of modeling sad moods in music seems to be a particular challenge. While this factor seems to impact the modeling of felt emotion the most, differences in the perception of emotion also exist [62] making this factor relevant for the modeling of perceived emotion as well. Han et al. [61] handle this by explicitly asking which mood a listener wishes to attain from listening to specific music to give more insights in their emotional goal. Other approaches implement a degree of uncertainty in the predictions by utilizing fuzzy classification to try to account for the possible differences in perception of emotion between users [154].

Emotion source Handling extramusical factors like linkage between autobiographical memories and certain moods in a model seems almost impossible without having full information about the user of the system running the predictions. Approaches like the one by Han et al. [61] might capture some of this information by gathering as much contextual information about a listener as possible, like education, hobbies and musical background. This model is able to leverage these extramusical factors in the music suggestion system, but this data needs to be available.

2.4.8. Key challenges

This section aimed to highlight the many (inter-disciplinary) challenges that should be addressed when constructing a machine learning classifier in the MIR field, further highlighting the need for evaluation methods independent of ground-truth labels. The following factors have been identified and discussed:

- **Use cases**. The fault tolerance of the classifier depends on the use case where the classifier will be deployed. While the discussed psychological factors apply to all applications, extra caution should be taken when applying them in a high risk domain such as music therapy.
- **Choice of emotion model**. The emotion model chosen for representing the emotion in music can have a large impact on the performance and correctness of the classifier. None of the presented models are perfect, and thus the choice should depend on which factors are most important for the application in which the classifier will be used. The correct choice of model requires knowledge from the field of psychology. Incorrect choices for emotion models will make representation of the emotion more difficult.
- **Sad music modeling**. The 'sad' mood of music deserves careful consideration. As described, modeling it can be difficult. When designing a classifier, especially when dealing with the sad

mood, making a distinction between *perceived* and *felt* emotion can be very useful to make sure that the assumptions made in designing the classifier are correct.

- **Musical memory**. Musical memory is a factor that can influence perceived and felt emotions on an individual level, and it is very difficult to incorporate this in a model without modeling some uncertainty. When designing a classifier, the possible influence of this factor should be taken into account.
- **Interpretation issues**. Due to ambiguities in the labeling of music data using labels from emotion models, class noise will be present in the data. It is important to make sure that the trained classifier is relatively robust to noise, or the actual performance might decrease substantially.

  Labels output by a classifier should also be interpreted carefully, taking into account that factors other than mood influence the listening behavior of users, and that listening strategies might differ.

2.5. Conclusions

While some ways of dealing with some of the difficult factors that might play a role in any MIR system have been discussed in Section 2.4.7, factors like interpretation issues are barely discussed in the literature. The example task presented in Section 2.4 demonstrates that there can be many factors that ultimately influence the performance of any trained music classifier, and it is reasonable to assume that any other task in the field of MIR will contain similar (but perhaps different) aspects that might influence the performance of the trained classifier. Given all these extra-musical factors it might be the case that some MIR classifiers fail to actually learn the abstract concept we wish it to learn and pick up on statistical confounds in the data instead. This calls for more elaborate evaluation of the *construct validity* of the classifiers without relying on a labeled test set, which will be discussed in Chapter 4. In addition to these extra-musical factors, digital audio representation can also vary according to the codec or bitrate used to encode the audio as shown in Section 2.3, highlighting the importance of proper analyses of aspects like classifier stability, which will be discussed in Chapter 3. Such analyses will give more insight into the effect of these potentially troublesome factors on the stability and performance of high-level feature classifiers like the sentiment analysis classifiers described in this section or classifiers in general, and will allow us to better understand MIR and general machine learning pipelines. Furthermore, it will provide a good basis towards better evaluation methods.

3

Classifier stability

The literature study in Chapter 2 demonstrated the need for evaluation methods for machine learning classifiers that do not depend on any ground truth labels. This chapter will explore **RQ1**: How can multiple representations of the same input be leveraged to quantify the performance of a classifier in terms of stability? by defining several metrics based on the variance of label probabilities when a classifier is presented with multiple representations of the same input and applying it to a large corpus of music data.

The reasoning for quantifying part of the performance of a classifier based on this variance in output probabilities is as follows: Any classifier should give back the same answer (or nearly the same answer) if the representation of the data is slightly different. In the case of high-level music classifiers, different representations of the same input might refer to a recording of a song, released on a specific album, but digitized in different ways. For example, three different representations of the song `Bohemian Rhapsody` by `Queen` from the album `A Night at the Opera` might include a CD-quality FLAC file, an MP3-file encoded at 320kbps and a Vorbis file encoded at a variable bitrate. These three digital files are different representations of the exact same piece of music from the same source, having the same high-level characteristics. A small change in audio quality does not change the mood of the song, for example.

If the classifier works as intended then these small changes in the digital representation of the input should not have an influence on the output labels. It is desirable for a classifier to be *stable* in the labels that they output across different representations of the same input, indicating that the classifier is able to handle these representational differences in the input. Intuitively, since many of the high-level music classification tasks such as detecting the mood of a song are based on the human perception and interpretation of the music, classifiers should ignore these small variations in audio quality to calculate these answers and rather use broader concepts and features available in the audio that humans use to assess the mood of a piece of music: features like the melody, key or instrumentation used.

However, as Section 2.3 demonstrated, the digital audio signal can change quite a lot when different encoding settings are used. If the high level classifiers are susceptible to these changes in the input signal, then the output of the classifiers must in some way take these almost irrelevant features (codecs and bitrates are based on listening tests, and many people find it difficult to hear quality differences [34]) into account in calculating the high-level features. This would hint at the possibility that, either due to how the classifiers are designed or how they are trained, the classifiers overfit on irrelevant data. If this would be the case, then it is questionable if the classifiers actually model the concept we are interested in by utilizing the correct musical knowledge, or if the system is a 'horse system':

> "A system appearing capable of a remarkable human feat, e.g., music genre recognition, but actually working by using irrelevant characteristics (confounds)." [137]

3.1. Data Availablity

To be able to draw any meaningful conclusions about the stability of currently available high-level music classifiers, we need a dataset comprising multiple representations of the same recording, with resulting high-level classifier labels and label probabilities calculated for all entries in the dataset. One way of

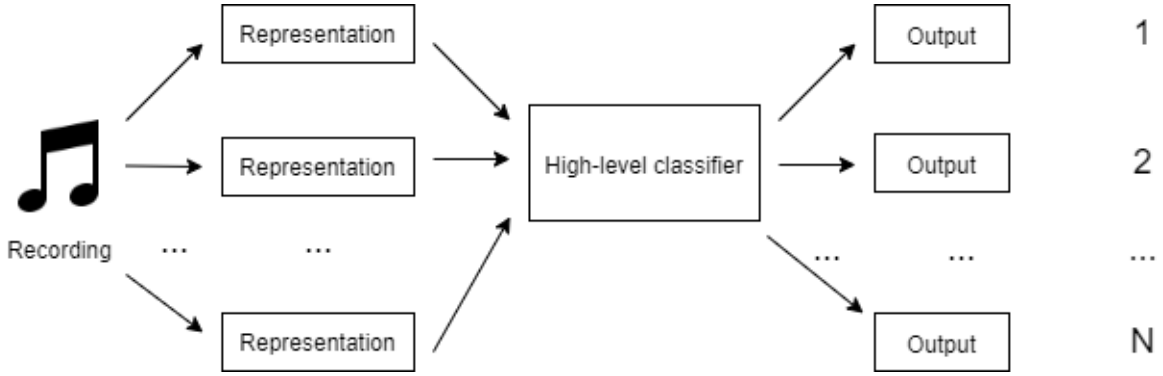


Figure 3.1: Data structure of the AcousticBrainz dataset. For each recording multiple representations are uploaded by users (with different codec, bitrate etc.). These different representations of the same recording are then all ran through the same classifier and the outputs of the classifier are stored. The full dataset contains multiple high-level classifiers and multiple recordings.

creating such a dataset would be to collect a large amount of CD-quality, commercial recordings (commercial because classifiers, like the ones from Spotify, are probably trained on in-house commercial music) and then alter the representation by encoding the source file with different codecs and different bitrates or adding other small adjustments that change the audio signal without the recording sounding totally different. While this approach has the advantage of being able to fully control which factors are changed across representations, creating such a dataset is outside of the scope of this thesis due to time constraints (having to generate the data, manually having to label data with the 'correct' labels through expert judgment...) and legal issues of acquiring a large corpus of commercial music [79] and making that data available for open science.

3.1.1. AcousticBrainz

The alternative is using an existing dataset for the stability analysis. Luckily, the AcousticBrainz project [118] exists. The goal of AcousticBrainz is to "*crowd source acoustic information for all music in the world and to make it available to the public*"[1] and it includes output from different high-level classifiers. The platform relies on users to send in data, and offers an easy to use GUI that can be ran over the personal music collection of a user. This music (in whatever format the user has it stored as) is first transformed into a representation of the audio by calculation low-level features using Essentia [16]. These low-level features are legal to share, and are sent to the AcousticBrainz servers to be stored there. Finally, using these low-level features AcousticBrainz runs a list of high-level feature classifiers on the low-level features extracted from the audio and stores these results. A full list of the high-level classifiers that are ran on the AcousticBrainz data can be found at the AcousticBrainz website[2]. A summary of the classification accuracies as reported by AcousticBrainz can be found in Table 3.1. Note that according to the classification accuracies most of these classifiers perform quite well with most having a classification accuracy of over 80%. The data included in the AcousticBrainz dataset is described visually in Figure 3.1, showing that this dataset is suitable for stability analysis since we have multiple representations of the same input and each of these inputs has a corresponding output produced by the classifier we wish to test. We can use these multiple outputs to quantify the classifier performance in terms of stability.

The AcousticBrainz dataset is a good match for our analysis due to the 'real-world' nature of the data. Most high-level classifiers are trained on a rather limited selected training dataset (see Table 3.1), whereas the AcousticBrainz dataset includes data on millions of recordings[3] and due to the way the data is collected — every user simply uses their personal music collection — a large sample of non-synthetic differences in representation of the audio is included. In addition, for every submission the metadata fields — like the bitrate or codec of the song, and even the software version used to calculate the underlying audio features — are also saved, allowing us to analyze the effect of these fields on the stability of the classifiers. When looking at the accuracies reported in Table 3.1 the classifiers seem

[1] https://acousticbrainz.org/
[2] https://acousticbrainz.org/datasets/accuracy
[3] https://acousticbrainz.org/statistics-graph

Classifier	Source / Dataset	Dataset size	Reported accuracy
danceability	In-house MTG collection	306	92.41%
gender	In-house MTG collection	3311	87.21%
genre_dortmund	Music Audio Benchmark Data Set [66]	1820	60.25%
genre_electronic	In-house MTG collection	250	91.70%
genre_rosamerica	In-house MTG collection created by a musicologist [58]	400	87.56%
genre_tzanetakis	GTZAN Genre Collection [142]	1000	75.53%
ismir04_rhythm	ISMIR2004 Rhythm Classification Dataset [24]	683	73.21%
mood_acoustic	In-house MTG collection [85]	321	92.98%
mood_aggressive	In-house MTG collection [85]	280	97.50%
mood_electronic	In-house MTG collection [85]	332	86.38%
mood_happy	In-house MTG collection [85]	302	83.27%
mood_party	In-house MTG collection [85]	349	88.38%
mood_relaxed	In-house MTG collection [85]	446	93.20%
mood_sad	In-house MTG collection [85]	230	87.83%
moods_mirex	MIREX Audio Mood Classification Dataset [69]	269	57.09%
timbre	In-house MTG collection	3000	94.32%
tonal_atonal	In-house MTG collection	345	97.67%
voice_instrumental	In-house MTG collection	1000	93.80%

Table 3.1: List of high-level classifiers available in the AcousticBrainz dataset and their reported accuracies.

to be working very well. If other metrics show that these classifiers are probably not working as well as these accuracies lead us to believe, then this can give some interesting insights into 'perceived accuracy' on testing data vs. 'real accuracy' on real-world data.

3.2. Processing the AcousticBrainz data

AcousticBrainz provides a data dump from the 30th of January, 2015[4] containing the outputs of the high-level classifiers described in section 3.1. The data consists of 1.806.725 JSON files with filenames following the format: `[recording MBID]-[submission no.]`. MBIDs are described by MusicBrainz as follows:

> "In a nutshell, an MBID is a 36 character Universally Unique Identifier that is permanently assigned to each entity in the database, i.e. artists, release groups, releases, recordings, works, labels, areas, places and URLs." [5]

These JSON files were parsed using a Python script to extract the fields we are interested in analyzing: all of the outputs from the classifiers listed in Table 3.1 as well as additional metadata as described in Table 3.2 were stored in a Pandas[6] dataframe for easy data manipulation. Not all included JSON files had entries for the high-level classifier fields. Submission for which the high-level features were not calculated were filtered out. Of the 1,806,725 submissions, 1,805,912 submissions had associated high-level feature data. A small subset of the resulting data can be seen in Figure 3.2.

Because in this chapte we are interested quantifying the performance of classifiers based on the stability of the labels over different representations of the same recording, recordings for which there was only one representation were filtered out. After filtering this resulted in high-level classifier data for 941,018 submissions across 299,097 different recordings, with an average of around 3.1 submissions per recording.

[4] https://acousticbrainz.org/download
[5] https://musicbrainz.org/doc/MusicBrainz_Identifier
[6] https://pandas.pydata.org/
[7] https://essentia.upf.edu/streaming_extractor_music.html
[8] https://essentia.upf.edu/gaia/index.html
[9] https://github.com/MTG/essentia

	MBID	ID	(danceability, danceable)	(danceability, not_danceable)	(gender, female)	(gender, male)	(genre_dortmund, alternative)	...
1	000009a8-34f1-4c58-a8de-1d99809cd626	0	3.000001e-14	1.000000	0.622127	0.377873	4.812406e-02	...
		1	2.751654e-01	0.724835	0.588847	0.411153	2.766604e-01	...
2	00000baf-9215-483a-8900-93756eaf1cfc	0	9.999093e-01	0.000091	0.500000	0.500000	1.232104e-01	...
		1	9.999142e-01	0.000086	0.500000	0.500000	1.222987e-01	...
		2	3.000001e-14	1.000000	0.622127	0.377873	1.199233e-03	...
	...	...	...	...	...	...	...	...
k-3	ffffcf12-d32e-4b8e-91bc-139de844528d	0	7.875147e-01	0.212485	0.978986	0.021014	1.446512e-01	...
k-2	ffffdf6a-cbbe-40e4-a900-53f55723fa14	0	2.258485e-08	1.000000	0.471415	0.528585	3.723815e-01	...
k-1	ffffefa3-45ae-4f72-9177-523b316a6534	0	3.936251e-02	0.960638	0.189658	0.810342	4.516886e-12	...
		1	3.936261e-02	0.960637	0.189648	0.810352	4.516782e-12	...
k	fffff998-2ebd-47d0-9476-c5cbfdc1cd92	0	3.540457e-01	0.645954	0.604256	0.395744	1.486034e-02	...

1805912 rows × 84 columns

Figure 3.2: Small subset of the dataframe used for the analysis. Every row is indexed as MBID, submission ID (entries with the same MBID but a different submission ID are submissions by different users), every column is indexed as (classifier, label), entries are the probability of the submission for a recording having a specific label, according to that classifier.

The colored boxes demonstrate the indexing used for the metrics in this Chapter, with the red box corresponding to $(MBID_1, danceability, danceable)$ and the blue box to $(MBID_2, danceability, not\_danceable)_0$.

Metadata field	Description
analysis_sample_rate	Sample rate used for analysis
bit_rate	Bitrate of the source audio file
codec	Codec used to encode source audio file
downmix	Type of downmixing used
equal_loudness	Boolean indicating if the volume of the audio file was normalized
length	Length of the source audio file in seconds
lossless	Boolean indicating if the source audio file was encoded using a lossless codec
replay_gain	Replay gain (normalization) value
essentia_high	Version of Essentia used for high-level feature extraction
extractor_high	Music extractor[7] version used for high-level feature extraction
gaia_high	Gaia[8] version used
essentia_low	Version of Essentia used for low-level feature extraction
essentia_git_sha_low	GitHub commit SHA of Essentia version[9] used for low-level feature extraction
essentia_build_sha_low	Build SHA of Essentia version used for low-level feature extraction
extractor_low	Music extractor version used for low-level feature extraction

Table 3.2: List of metadata fields that were included in the analyzed DataFrame. All fields are present in the AcousticBrainz dataset. See also `https://acousticbrainz.org/data#sample-data`.

3.3. Stability Metrics

To quantify the stability of the classifiers over either label probabilities or the labels themselves we define several variance metrics. By defining such a metric we can quantify the stability of a single classifier (since we expect the classifier to be stable across representations, lower is better) without relying on ground-truth data and compare different classifier stabilities (given classifiers x and y we might infer that x is more stable than y).

As an example, take the Recording "Let It Be" by The Beatles from the AcousticBrainz data. According to the first submission on AcousticBrainz[10] this song is instrumental, sung by a female vocalist and not sad, while the second submission[11] claims that this song is non-instrumental, is sung by a male vocalist and is sad. Many more of these labels seems to flip depending on the submission. This instability of many of the labels raises concerns about the classifiers. Does this also happen on a large scale? Which classifiers are more stable? We define the stability metrics to quantify this classifier stability. This section will briefly describe these stability metrics used for this analysis and motivate their applicability to the problem at hand.

3.3.1. Variance

First, we look at the case where we wish to quantify the stability of the results given by a classifier over one input. For the AcousticBrainz data this means that we have one recording, identified by an MBID, which has n different submissions, with $n > 1$. We then want to classify how stable the probability values of a classifier c are for a certain label l over the different submissions. As an example, we might have the classifier $c = danceability$, for which $l \in \{danceable, not\_danceable\}$. If we identify a set of probability values by $(MBID, c, l)$ (see Figure 3.2) this could result in a probability vector $(MBID_1, danceability, danceable) = [0.9, 0.8, 0.4]$, as an example. Now we can use the statistical concept of variance, defined as the average of the squared deviations from the mean. If indexing from 1, the mean is defined as:

$$\overline{X} = \frac{\Sigma_{i=1}^{n} X_i}{n}$$

And then the variance is defined as:

$$var(X) = \frac{\Sigma_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$$

In the example, using the variance as described here as a metric for stability, the stability for the recording with $MBID = MBID_1$ would be:

$$X = (MBID_1, danceability, danceable) = [0.9, 0.8, 0.4]$$

$$\overline{X} = \frac{0.9 + 0.8 + 0.4}{3} = \frac{2.1}{3} = 0.7$$

$$var(X) = \frac{(0.9 - 0.7)^2 + (0.8 - 0.7)^2 + (0.4 - 0.7)^2}{2} = 0.07$$

Note that Bessel's correction term of $n-1$ is used in the denominator to give an unbiased estimation of the sample variance for the population. Also note that for classifiers with two possible labels, e.g. $danceability$ with $l \in \{danceable, not\_danceable\}$ the probability of the labels add up to one, causing the opposite label to have a probability of $1 - p$. Due to this, the variance over both labels will be the same. For classifiers with more than two labels the probabilities will still add to 1, but variances over labels can be different.

3.3.2. Pooled variance

Using the variance metric as described above allows us to quantify the stability of the classifier over multiple representation or submissions for one recording specified by an MBID. However, we we wish to combine the calculated variances in a way which reflects the stability of the classifier over the entire set of recordings. To combine the calculated variances over the different recordings, we make the following assumptions:

[10] https://acousticbrainz.org/0cdc9b5b-b16b-4ff1-9f16-5b4ba76f1c17?n=0
[11] https://acousticbrainz.org/0cdc9b5b-b16b-4ff1-9f16-5b4ba76f1c17?n=1

- We have different populations (one population per MBID-classifier-label pair, identified as $(MBID_i, c, l)$ of probability values, each with a different mean value depending on the song used for calculating these probabilities.
- The variance of each population is the same, and is caused by the behaviour of the classifier.
- The variance estimation is more accurate if more samples are used to calculate it.

Simply taking the mean value of the calculated variance values would be one possibility, however this would give an equal amount of weight to variances which were calculated using a small amount of submissions and variances calculating using a larger amount of submissions. Since we assume that the variances which are calculated using a larger amount of samples are more accurate for estimating the variance in output labels caused by the classifier, we can pool these variances together by taking the weighted average, giving more weight to variances calculated over larger sample sizes.

Assume we wish to calculate the pooled variance for classifier c and label l. Each recording-classifier-label pair is a population indexed by $i = 1...k$, so for a given classifier and label the populations are $[(MBID_1, c, l), (MBID_2, c, l), ..., (MBID_K, c, l)]$.

Then, the pooled variance is defined as:

$$\overline{var}(c, l) = \frac{\Sigma_{i=1}^{k}(n_i \times var((MBID_i, c, l)))}{\Sigma_{i=1}^{k} n_i}$$

where n_i is the sample size of population i

3.3.3. Normalized Entropy

The variance metric as described in Sections 3.3.1 and 3.3.2 only works on vectors of continuous (e.g. probability) values. However, since the classifiers output probabilities that correspond to discrete output labels, we might also want to look at the stability of the discrete labels. Defining a metric to quantify stability over discrete labels allows us to:

- Analyze the bias of a classifier by checking if output labels are somewhat evenly distributed over the different submissions and recordings
- Analyze the stability of a classifier on the discrete labels for the different submissions of one recording

To make this possible, the probability data is first transformed into discrete labels using the following rule. Given classifier c which has the set of labels L_c corresponding to c, MBID i, submission ID j and the probability values are indexed as $(MBID_i, c, l)_j$, we transform these probabilities for a given MBID and submission ID to a discrete label by selecting the label which has the highest corresponding probability value by looping over all possible values $l \in L_c$ and keeping the one for which $(MBID_i, c, l)_j$ was highest. Using the same example as in 3.3.1 where we had the probability values of label $danceable$ of $[0.9, 0.8, 0.4]$, applying this rule would give us $[danceable, danceable, not\_danceable]$.

After the data is transformed into discrete data, we can calculate the information entropy $H(MBID_i, c)$ over these labels [133]:

$$H(MBID_i, c) = -\Sigma_{l \in L_c} P(MBID_i, c, l) \log_2 P(MBID_i, c, l)$$

Where $P((MBID_i, c, l))$ is the probability of label l in classifier c, following the observed empirical distribution of discrete labels within the population corresponding to $MBID_i$ and L_c is the set of possible labels for classifier c. Using the same example, the entropy value is calculated as:

$$(MBID_1, danceability) = [danceable, danceable, not\_danceable]$$

$$P(MBID_1, danceability, danceable) = \frac{2}{3}$$

$$P(MBID_1, danceability, not\_danceable) = \frac{1}{3}$$

$$H(MBID_1, danceability) = -(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}) \approx 0.92$$

Because not all of the classifiers have the same amount of labels, we wish to normalize this calculated entropy by dividing it by the maximum possible value, resulting in a metric that takes on values in the range $[0, 1]$. Since it holds that $H(X) \leq \log_2 |(L_X)|$ [50, Property 2.6] where L_X is the set of possible labels for input X then we define the normalized entropy $\hat{H}(MBID_i, c)$ as follows:

$$\begin{aligned}\hat{H}(MBID_i, c) &= -\Sigma_{l \in L_c} \frac{P(MBID_i, c, l) \log_2 P(MBID_i, c, l)}{\log_2 |(L_c)|} \\ &= -\Sigma_{l \in L_c} P((MBID_i, c, l)) log_{|L_c|} P((MBID_i, c, l))\end{aligned}$$

Thus, we can calculate the normalized entropy metric for a given classifier by calculating the entropy over the sampled probability values of the discrete labels and setting the base of the log equal to the amount of different labels that classifier can output.

This metric can model the stability or the bias as follows. If the normalized entropy is 1 (the maximum value), then the distribution of labels is uniform. For stability of a classifier this is the worst case, for the bias of the classifier this is the best case since we want a classifier to be relatively 'sure' of their answer for a specific recording, but across recordings it should not give back the same answer every time assuming that the data has a uniform label distribution for the ground truth. Thus, for stability the normalized entropy metric can be interpreted in a similar way as the variance, a lower value means that the classifier outputs more stable labels. For quantifying the bias of a classifier, higher values are better.

We can use this metric in the two different ways mentioned at the start of this section: calculating the normalized entropy over *all* submissions and *all* MBIDs (the entire column as defined in Figure 3.2, which will be denoted as $\hat{H}(c)_{all}$) will allow us to quantify the bias of the classifier, and calculating the normalized entropy over the different submissions of *one* MBID (denoted as $\hat{H}(MBID_i, c)$ contains only the labels for classifier c on MBID i) will allow us to quantify the stability of the classifier on a single recording according to the discrete labels.

3.3.4. Pooled normalized entropy

Much like the metric described in Section 3.3.2, we can also pool the normalized entropy. This will again allow for the usage of not just one recording, but all recordings present in the dataset for estimating the stability of a classifier. The approach is similar to that of calculating the pooled variance, however the entropy is calculated over *all* labels instead of over *one* of the possible labels.

Assume we wish to calculate the pooled normalized entropy for classifier c. Since each MBID is a population indexed by $i = 1...k$ with sample sample size n_i the pooled normalized entropy is defined as:

$$\overline{\hat{H}}(c) = \frac{\Sigma_{i=1}^{k}(n_i \times \hat{H}(MBID_i, c))}{\Sigma_{i=1}^{k} n_i}$$

where $(MBID_i, c)$ is the vector of discrete labels corresponding to classifier c and MBID index i.

3.4. Analysis

With the dataset described in Section 3.1 and the metrics described in Section 3.3 we have all the tools we need to analyze how stable the various high level classifiers in the AcousticBrainz data are. This section will quantify the performance of the classifiers in terms of stability using both the probability and discrete label variance metrics showing how such a metric can be valuable for studying the performance of classifiers. In addition, this section will explore **RQ1.1**: Using stability metrics, how does the representation of the audio (e.g. bitrate used, codec used) influence high-level music classifiers? and **RQ1.2**: Using stability metrics, how does software versioning (e.g. software version used to calculate features) influence high-level music classifiers?

3.4.1. Comparing classifiers using stability metrics

The most basic analysis using these metrics is to check if the classifiers are stable or unstable in the most general sense. For this the entire AcousticBrainz dataset filtered on recordings with at least two submissions can be used (as described in Section 3.2). Both the variance over the label probabilities

as well as the variance of the discrete labels will be explored. Keep in mind that, if the classifiers correctly model the concept that we expect them to and calculate their outputs based on relevant data, we would expect the classifiers to be stable across different representations (submissions) of the same recording. Thus, if this would be the case then the *variance of label probabilities and discrete labels should be zero or very close to zero*.

Stability of label probabilities Stability of label probabilities was calculated for all high-level classifiers described in Section 3.1 using the variance and pooled variance metrics described in Sections 3.3.1 and 3.3.2 respectively. The results of running the pooled variance calculations are presented in Table 3.3.

Table 3.3: Pooled variance results for the various high-level classifiers per label. A lower value for $\overline{var}(c,l)$ indicates that the classifier outputs more stable label probabilities. The 95% confidence interval is given by $[\overline{var}(c,l)] \pm$ [95% CI term]

Classifier	Label	$\overline{var}(c,l)$	95% CI term
voice_instrumental	instrumental	0.073534	2.338615e-04
	voice	0.073534	2.338615e-04
danceability	not_danceable	0.066811	2.310532e-04
	danceable	0.066811	2.310532e-04
timbre	bright	0.058701	2.068124e-04
	dark	0.058701	2.068124e-04
tonal_atonal	atonal	0.056830	2.004491e-04
	tonal	0.056830	2.004491e-04
mood_electronic	not_electronic	0.049802	1.670732e-04
	electronic	0.049802	1.670732e-04
mood_party	party	0.046484	1.646424e-04
	not_party	0.046484	1.646424e-04
mood_aggressive	not_aggressive	0.045529	1.987949e-04
	aggressive	0.045529	1.987949e-04
mood_relaxed	not_relaxed	0.037327	1.310168e-04
	relaxed	0.037327	1.310168e-04
mood_acoustic	not_acoustic	0.028181	1.418287e-04
	acoustic	0.028181	1.418287e-04
ismir04_rhythm	Rumba-American	0.026770	1.466862e-04
mood_happy	happy	0.024462	9.786015e-05
	not_happy	0.024462	9.786015e-05
gender	male	0.023747	7.823300e-05
	female	0.023747	7.823300e-05
genre_dortmund	electronic	0.020293	1.093386e-04
moods_mirex	Cluster5	0.018311	6.523851e-05
ismir04_rhythm	VienneseWaltz	0.017491	9.410717e-05
	ChaChaCha	0.016644	9.712531e-05
	Tango	0.015979	8.963356e-05
genre_tzanetakis	jaz	0.012092	8.665969e-05
genre_rosamerica	hip	0.009747	6.383294e-05
mood_sad	not_sad	0.009161	4.655576e-05
	sad	0.009161	4.655576e-05
genre_rosamerica	roc	0.007562	4.756549e-05
genre_electronic	ambient	0.007066	3.671622e-05
genre_rosamerica	pop	0.006909	3.817121e-05
	rhy	0.006791	4.373959e-05
	cla	0.006490	5.998254e-05
	jaz	0.005267	4.525560e-05
genre_electronic	trance	0.004749	2.783810e-05

Continued on next page

Table 3.3: Pooled variance results for the various high-level classifiers per label. A lower value for $\overline{var}(c,l)$ indicates that the classifier outputs more stable label probabilities. The 95% confidence interval is given by $[\overline{var}(c,l)] \pm$ [95% CI term]

Classifier	Label	$\overline{var}(c,l)$	95% CI term
moods_mirex	Cluster3	0.004434	2.160711e-05
	Cluster2	0.004325	2.359543e-05
genre_electronic	house	0.004142	3.193548e-05
genre_rosamerica	dan	0.003365	3.601890e-05
genre_dortmund	folkcountry	0.002837	2.894233e-05
moods_mirex	Cluster4	0.002795	1.217487e-05
genre_tzanetakis	roc	0.002406	3.838028e-05
ismir04_rhythm	Jive	0.002105	2.321592e-05
	Waltz	0.002034	2.112069e-05
genre_electronic	techno	0.001898	1.926141e-05
ismir04_rhythm	Samba	0.001808	8.378892e-06
genre_tzanetakis	hip	0.001785	3.014589e-05
genre_dortmund	rock	0.001768	2.117224e-05
	alternative	0.001726	1.266411e-05
	blues	0.001565	2.070428e-05
genre_tzanetakis	cla	0.001387	3.485617e-05
ismir04_rhythm	Rumba-International	0.000989	1.466324e-05
moods_mirex	Cluster1	0.000965	7.281328e-06
ismir04_rhythm	Quickstep	0.000797	1.182080e-05
genre_dortmund	jazz	0.000760	1.551551e-05
genre_electronic	dnb	0.000497	8.578323e-06
genre_tzanetakis	blu	0.000480	1.131812e-05
	pop	0.000411	1.282443e-05
genre_rosamerica	spe	0.000399	7.951778e-06
genre_tzanetakis	cou	0.000353	4.955996e-06
genre_dortmund	raphiphop	0.000295	1.196482e-05
ismir04_rhythm	Rumba-Misc	0.000277	8.268435e-06
genre_tzanetakis	dis	0.000272	5.035058e-06
	reg	0.000236	4.457647e-06
	met	0.000154	7.204082e-06
genre_dortmund	pop	0.000042	5.592759e-07
	funksoulrnb	0.000011	3.613095e-07

The results presented in Table 3.3 suggest that classifiers like `voice_instrumental` and `danceability` score much worse using this stability metric than classifiers like `genre_dortmund` which had a much lower pooled variance score. While these results seem to suggest that such classifiers are very stable, this might be due to a classifier being very biased. If a classifier gives back almost the same label probabilities every time, then the pooled variance metric will consequently be very low. To distinguish between biased and unbiased classifiers, the normalized entropy as described in Section 3.3.3 was calculated over all submissions across all recordings for each classifier. In addition, we take the average of the pooled variance for all different labels for one classifier to summarize the results. For most of the classifiers that give binary values, this value is the same for both labels and the average will also be equal to these values. This mean of the pooled variances over the labels of one classifier will be denoted as $\overline{\overline{var}}(c)$. The mean pooled variance in combination with the normalized entropy gives us an overall overview of how stable and biased the classifiers are. These results are presented in Table 3.4. By plotting this overview we can visually see which classifiers perform well based on the measured stability and bias metrics, with the best performing classifiers located in the bottom-right of the figure (see Figure 3.3).

Note that by looking at the pooled variance metric combined with the normalized entropy metric, classifiers that appeared to perform well in Table 3.3, like `genre_tzanetakis` and `genre_dortmund` seem to score well on the pooled variance metric due to them being relatively biased — a low value for $\hat{H}(c)_{all}$ means that the label entropy over all submissions was relatively low, in other words the

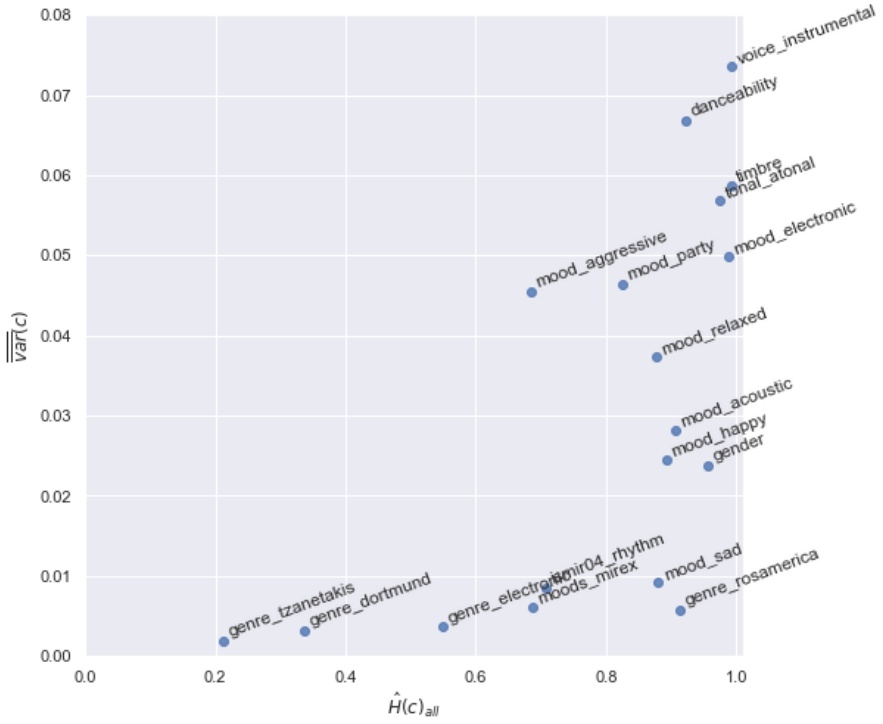


Figure 3.3: Visual representation of the relation between the bias of the classifier and the stability based on the mean pooled variance as reported in Table 3.4. Higher values of $\hat{H}(c)_{all}$ indicate a lower bias and are better. Lower values of $\overline{\overline{var}}(c)$ indicate a higher stability and are better. Thus, classifiers closer to the bottom right of the figure perform best according to these metrics.

classifier outputs the same label for most of the submissions. Classifiers in the bottom right of Figure 3.3 like `genre_rosamerica` or `mood_sad` score well on the pooled variance metric *without* showing a large bias.

Stability of discrete labels The same stability analysis can also be ran on the discrete labels instead of the probability values for every label if we first transform these label probabilities according to the method as described in Section 3.3.3. Then, the normalized entropy metric as described in Section 3.3.3 will give an indication of the stability of the classifier labels.

The reasoning for running an analysis on both the stability of the label probabilities and the discrete labels can be explained using an example: say we have a classifier c that outputs label probabilities for a binary classification, so labels a and b. Depending on the value of these label distributions, the different metrics will give back different results.

Say the label probabilities of c for label a are $[0.1, 0.11, 0.09]$, then by analyzing the variance of these labels we see that the probabilities vary slightly, however since none of the probabilities take on a value higher than 0.5 due to the low variance, the label output by the classifier will still be $[b, b, b]$ and the normalized entropy represents this lack of label variance by giving back a value of 0. If however the label probabilities are closer to the critical value of 0.5, say the probabilities of c for label a are $[0.499, 0.501, 0.502]$ then the corresponding labels will be $[b, a, a]$ and the normalized entropy will give back a relatively high amount of variance, while the label probabilities vary only a very small amount.

There is also the case in which the label variances are simply very high, say the probabilities of c for label a are $[0.1, 0.6, 0.9]$, in this case both the label variance and the normalized entropy metric will report a high variance. Thus, both metrics are will capture large amounts of variance, but the normalized entropy is able to incorporate the cases where the label flips due to a low amount of variance. The tradeoff is, however, that a low amount of variance further away from the critical value will *not* be captured by the normalized entropy.

Calculating the Pooled normalized entropy metric on the various high-level classifiers, and comparing it with the normalized entropy ran across all submissions allows us to quantify both the variance

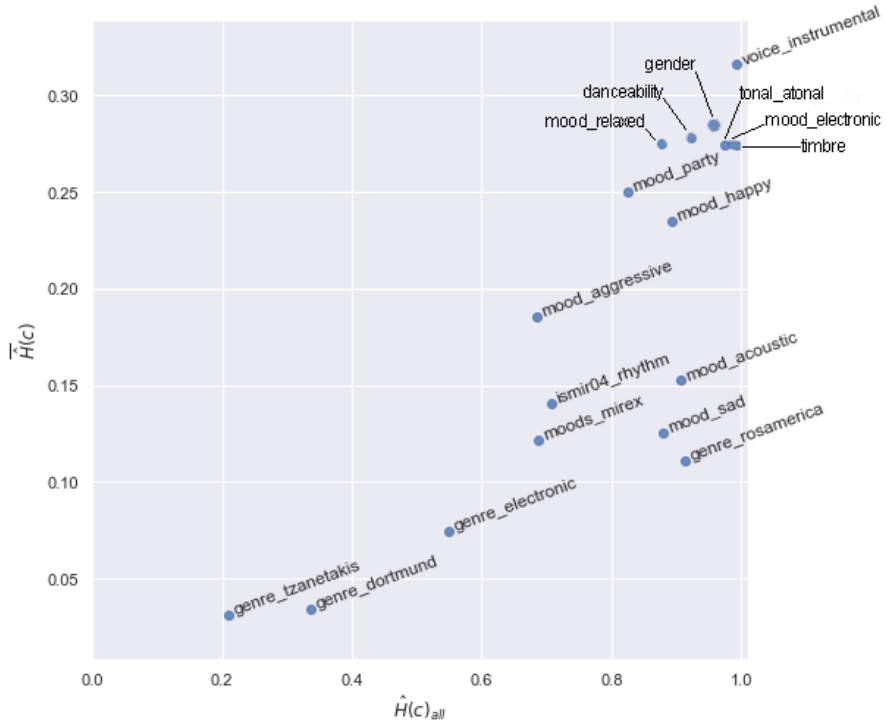


Figure 3.4: Visual representation of the relation between the bias of the classifier and the stability based on the mean normalized entropy as reported in Table 3.4. Higher values of $\hat{H}(c)_{all}$ indicate a lower bias and are better. Lower values of $\overline{\hat{H}}(c)$ indicate a higher stability and are better. Thus, classifiers closer to the bottom right of the figure perform best according to these metrics.

Classifier	$\hat{H}(c)_{all}$	$\overline{\overline{var}}(c)$	$\overline{\hat{H}}(c)$
genre_tzanetakis	0.210992	0.001957	0.030846
genre_dortmund	0.337265	0.003255	0.034370
genre_electronic	0.550274	0.003670	0.074624
mood_aggressive	0.684244	0.045529	0.185294
moods_mirex	0.686701	0.006166	0.121858
ismir04_rhythm	0.707772	0.008489	0.140486
mood_party	0.825267	0.046484	0.250273
mood_relaxed	0.877459	0.037327	0.275083
mood_sad	0.880068	0.009161	0.125632
mood_happy	0.892659	0.024462	0.234724
mood_acoustic	0.905857	0.028181	0.152564
genre_rosamerica	0.914024	0.005816	0.110834
danceability	0.922382	0.066811	0.278202
gender	0.956663	0.023747	0.285081
tonal_atonal	0.975303	0.056830	0.274210
mood_electronic	0.987134	0.049802	0.275585
timbre	0.992234	0.058701	0.274619
voice_instrumental	0.993270	0.073534	0.316232

Table 3.4: Summary of classifier biasedness ($\hat{H}(c)_{all}$) and stability based on the label probabilities ($\overline{\overline{var}}(c)$) and discrete labels ($\overline{\hat{H}}(c)$). A higher value for $\hat{H}(c)_{all}$ means that the classifier is less biased, a higher value for either $\overline{\overline{var}}(c)$ or $\overline{\hat{H}}(c)$ means that the classifier is more unstable.

and the bias. The results are presented in Table 3.4 and visually in Figure 3.4. Note that these stability using the mean pooled variance and the pooled normalized entropy largely correlate with each other, with classifiers scoring high on one metric often scoring high on the other as well. This indicates that a lot of the label probabilities produced by these classifiers are probably *not* around the critical point of 0.5 where the label flips even for small variances in label probabilities. One exception to this seems to be the `gender` classifier which scored relatively well using the mean pooled variance (with 7 classifiers scoring better, some of which were much more biased than the `gender` classifier) but scored relatively poorly using the pooled normalized entropy metric with only `voice_instrumental` scoring worse. This might indicate that for the `gender` classifier many of the label probabilities were around 0.5 meaning that the classifier on average was quite unsure about the correct label.

3.4.2. Effect of bitrate and codec on stability

From the analysis in the previous section it appears that many of the classifiers are not stable accross the many different representations of the same recordings. The following sections will explore subquestions **RQ1.1** and **RQ1.2** by analyzing the effect of some of the properties that these representations have, like the bitrate and codec used to encode the audio using the metrics defined in this chapter. The goal is to see if these representational differences lead to a part of the instability in the labels output by the high-level classifiers. Although the dataset includes an `analysis_sample_rate` field, all audio files were upscaled to a sample rate of 44100 before the low-level feature extractor was ran, thus all fields for this are the same giving us no information to compare.

As described in section 2.3, the bitrate and the codec used to encode an analog audio signal to a digital audio signal can result in very different audio signals. However, these audio signals should all sound roughly the same. High-level classifiers which classify the mood of a recording should output the same labels on different representations of the audio if the classifiers correctly look at the relevant musical concepts to calculate the label. If classifier stabiblities seem different for the different representations of the same recordings using the stability metric, then this would further hint at the fact that the classifiers look at the 'wrong' parts of the data to draw their conclusions. Thus, if the classifiers were to model mood using only the relevant musical properties of a recording, we would expect no differences in label variance for recordings with different bitrates or codecs.

Bitrate analysis To study the effect of the bitrate on the stability of the label probabilities produced by the classifiers, the original dataset as described in Section 3.1 is partitioned into parts using the metadata associated with every submission. Some bitrates were only present in a handful of submissions in the dataset, so the groups were split using the following steps:

- Filter out all bitrates for which there are less than 10,000 submissions in the dataset that use this bitrate.
- Using the dataset with only these common bitrates, filter out recordings for which there are no submissions that share a bitrate. (i.e. a recording with two submissions with different bitrates is filtered out, since this gives us no information about the stability)

After filtering, the distribution of submissions of various bitrates that remain is presented in Table 3.5. It is interesting to note that most submissions that remain are either lossless or have relatively 'low' bitrates of around 128000 and 192000.

By calculating the Pooled variance over these different slices of the data, the stability of the classifiers over the different bitrates can be calculated, see Figure 3.5 for the plot. Note that while this analysis does hint that classifiers working with higher bitrates are more stable — with classifiers generally showing a lower pooled variance score for the higher bitrates and lossless clearly having the lowest variance for all classifiers — it does *not* conclusively give any statistical guarantee that this difference in variance is caused only by the bitrate. Due to the nature of the dataset, with most recordings only having around 3 submissions in total, the dataset does not contain any submissions for which at least two classifier outputs are available for *all* bitrates. This means that, while this analysis hints that bitrate plays a role in classifier stability, the underlying data for each of the bitrates was different, and fully controlled experiments would be needed for statistically conclusive results which are unfortunately not possible using the AcousticBrainz data. Still, the results seem to suggest that bitrate plays a role in the stability of these classifiers as captured by the metrics defined in this Chapter.

Bitrate	Count
lossless	255740
192000	45830
128000	35085
320000	32517
160000	8221
256000	4153
Total	381546

Table 3.5: Distribution of submission bitrates after filtering out uncommon bitrates and recordings for which the submissions did not share a bitrate.

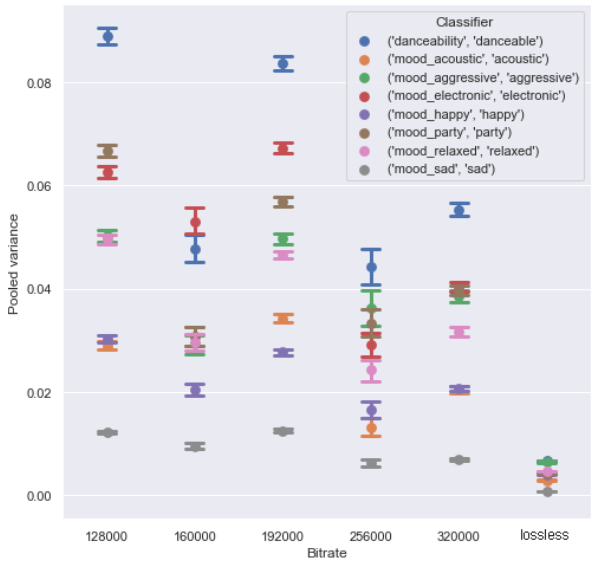


Figure 3.5: The effect of bitrate on the stability of various high-level mood classifiers, measured by pooled variance over the label probabilities.

Bitrate	Codec			
	flac	mpc7	mp3	vorbis
0	255317	230	0	0
128000	0	0	34944	122
160000	0	0	8074	145
192000	0	0	44768	1062
256000	0	0	3560	589
320000	0	0	25093	7405

Table 3.6: Codec distribution accross bitrates after filtering out uncommon bitrates and codecs, and filtering out all submissions that do not share a bitrate and codec with another submission for that recording. Bitrate '0' corresponds to lossless encoding.

Codec analysis It is also interesting to see if the codec used to encode the recording has an effect on the stability of the high-level classifiers. For bitrate it might be expected that lower bitrates, which correspond to a 'worse' audio quality, might add more noise to the audio signal and consequently worsen the stability of the classifier. However, while different codecs encode the analog audio signal in different ways (see Section 2.3), we might assume that different codecs with the same bitrate have roughly the same audio quality. Thus, we would expect that there are no differences in the label probabilities output by the high-level classifiers if we compare the stability between different codecs while making sure that the bitrate used for encoding the audio is exactly the same.

To test this, we first filter out the uncommon bitrates much like in the bitrate analysis. Then, on the dataset with only the common bitrates we filter out the uncommon codecs. Finally, using the data resulting from these two filtering steps, we filter out submissions for recordings that do not share a codec *and* a bitrate with another submission for that recording. Due to the many filtering steps, we end up filtering out a lot of data resulting in the unbalanced class distributions presented in Table 3.6.

Because the differences in class sizes after filtered is very large, running the pooled variance metric directly on this data will not result in a fair comparison: if a class has *much* more submissions, it also has *much* more 'chance' to be unstable due to there simply being more submissions to represent the variance. Thus, to make the comparison between codecs fair, first the variances for the full data as represented in Table 3.6 are calculated and stored, duplicating the variance values according to the sample size. Then, the variances corresponding to the largest group are downsampled without replacement to the same size as the smallest group and finally the variance is pooled by taking the average of both groups.

As an example, say we have group a with 5 submissions and group b with two submissions. Group a needs to be downsampled to the size of group b and contains two recordings, with the first recording having two submissions and the second one having three submissions. The variance is calculated over these recordings, say the variance over the first recording is 0.1 and over the second recording it is 0.2. Due to the first recording having two submissions and the second recording having three submissions this would be stored as $vars = [0.1, 0.1, 0.2, 0.2, 0.2]$. Note that if we would take the mean of $vars$ we would calculate the weighted average, or pooled variance, as described in Section 3.3.2 (since that would equal $\frac{(2\times0.1)+(3\times0.2)}{5}$). However, before pooling the variance, we downsample $vars$ so that it is of equal size to group b which has two submissions, now we might have $vars\_downsampled = [0.1, 0.2]$. Finally, we take the mean of $vars\_downsampled$ to pool the variance. Since variances calculated on recordings with more submissions have more entries in $vars$, the probability that those variances are sampled and included in $vars\_downsampled$ is larger than for recordings with less submissions — thus still giving more weight to recordings with more submissions — but the submissions are downsampled to give two groups of equal size and allow for a fairer comparison. Again, like with the bitrate analysis presented above, this does not control for the underlying data since the pooled variance metric requires at least two submissions *per bitrate-codec combination* and finding pairs of the same recording with the two encoding pairs requires a recording to have at least four submissions with the correct codec and bitrate, which unfortunately almost never occurs in the AcousticBrainz data.

Still, a comparison for lossless codecs can be seen in Figure 3.6 and a comparison between the mp3 and vorbis codec for lossy bitrates can be seen in Figure 3.7. Results seem to hint that the codec used to encode the audio also plays a role in terms of classifier stability, with the more common flac codec seemingly resulting in better stability than mpc7 and vorbis seemingly resulting in more stable

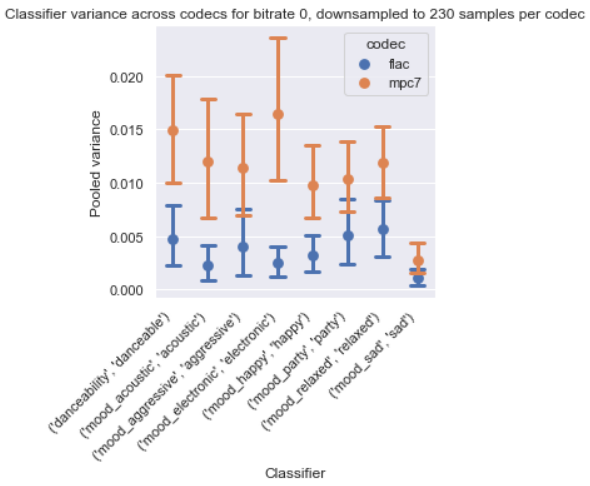


Figure 3.6: Comparison of classifier stability for two different lossless codecs. The flac data was downsampled to 230 samples to balance the classes.

results than mp3 for most bitrates. Note however that the observed differences are much smaller when compared to the differences in the bitrate analysis above, suggesting that bitrate has a larger effect than codec. And again, more controlled experiments would be needed to statistically prove this. Nonetheless, these results seem to indicate that the codec used to encode audio input for a classifier also influences the stability of subsequent high-level classifiers .

3.4.3. Effect of other metadata on stability

Finally, **RQ1.2** remains: Using stability metrics, how does software versioning (e.g. software version used to calculate features) influence high-level music classifiers? Some of the labels might be calculated using a different version of the high-level or low-level classifiers. While there were several different version fields present in the metadata (see Table 3.2), only the `essentia_low` field took on different values in the dataset. All 1,805,912 submission were run with `essentia_high = 2.1-beta1`, `extractor_high = music 1.0`, `gaia_high = 2.4-dev` and `extractor_low = music 1.0`. The `essentia_low` field has 663,866 submissions with version `2.1-beta1` and 1,142,046 submissions with version `2.1-beta2`.

Running the pooled variance calculations on both subsets, one for each low level essentia version, can give us the following insights:

- The low-level essentia version implementation might have changed a lot, resulting in significantly different low-level feature representations of the recordings. Since the high-level features are calculated using these low-level features, the label stability might be effected by the version. If this would be the case, then by running the pooled variance calculation on the subsets where the low-level essentia version is the same might result in significantly lower label probability variance than when running the calculation on the complete set as in Section 3.4.1.
- The high-level features should in theory not train on 'irrelevant' low-level features. If this is the case, and the high-level classifiers are robust enough against small changes in the low-level features, then there should be no observable difference in stability between the two slices of the dataset with different low-level Essentia versions. If we do observe a difference then either the low-level features in one version are incorrect, or the high-level feature classifiers are sensitive to changes in the low-level features.

The resulting comparison can be seen in Figure 3.8. Note that even though the `2.1-beta1` version is less represented in the dataset with only 663,866 submissions, the label probabilities using this version are significantly higher than the label probabilities calculated on the low-level features output by version `2.1-beta2`. Again, while no statistical guarantees can be given due to the change in the underlying data, these results suggest that the software version of the low-level extractor used to generate features can have a rather large impact on the stability of a classifier trained on these

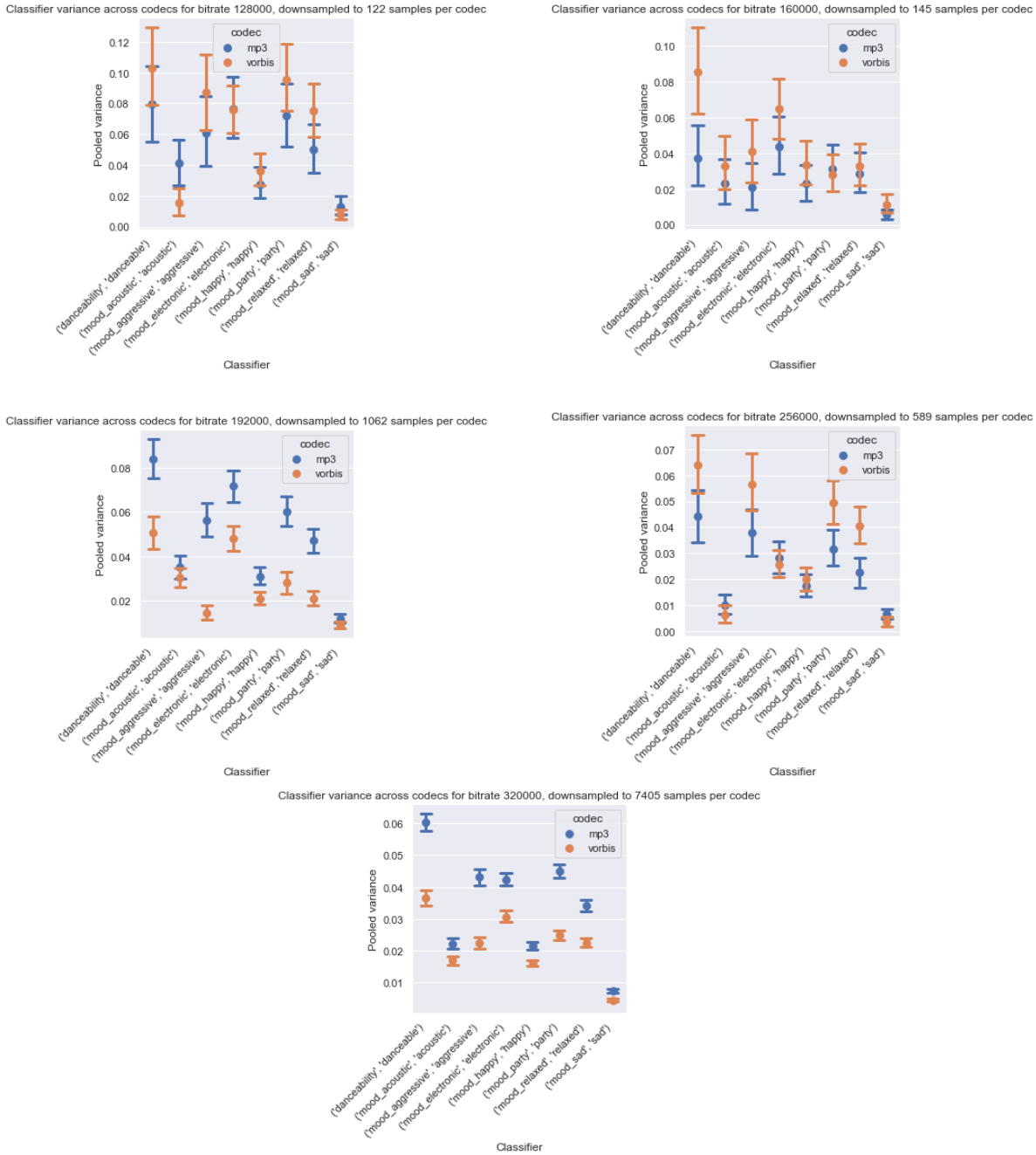


Figure 3.7: Comparison of classifier stability for the most common codecs mp3 and vorbis. Refer to the plot titles for bitrate and downsampling parameters.

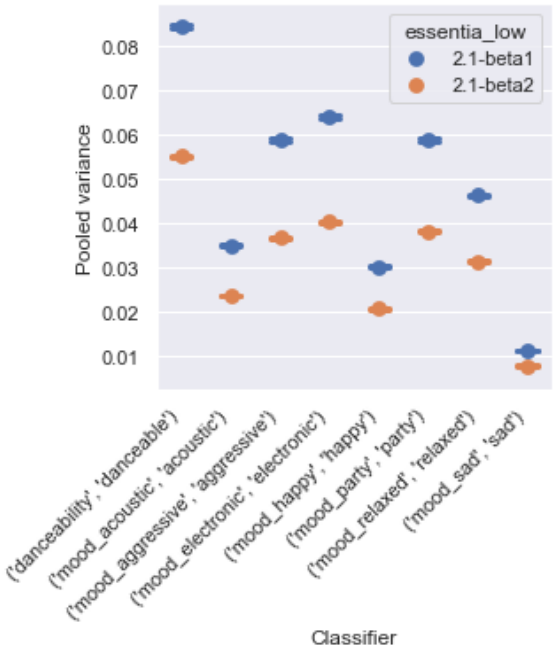


Figure 3.8: Classifier variance across Essentia low level versions.

features — with differences in pooled variance values being substantially larger than those observed in the codec analysis — indicating that further research into the effect of such software versions on the performance of trained classifiers can be very valuable.

3.5. Discussion

The results presented in this section give a broad overview of how many of the MIR classifiers trained in lab conditions perform in the wild. Ideally, we would have seen results indicating that most classifiers had a low bias and low variance. Unfortunately, many of the classifiers showed quite a bit of instability in both label probabilities and the discrete labels themselves when presented with in-the-wild data.

On the bias-variance tradeoff. Many of the classifiers like genre_tzanetakis and genre_dortmund which scored relatively well on the stability metric (having low values for $\overline{\overline{var}}(c)$ and $\overline{\hat{H}}(c)$) seemed to achieve this relative stability by being quite biased, showing very low scores for $\hat{H}(c)$ in comparison to the other classifiers. Interesting to note is that some classifiers which have to deal with relatively abstract concepts like mood_sad or mood_happy scored considerably better than more concretely defined concepts like voice_instrumental, which might seem counter-intuitive.

When this relationship between classifier bias and variance is plotted visually (as in Figures 3.3 and 3.4, especially in the latter) it seems that even though some argue that the bias variance tradeoff should not necessarily be a tradeoff [103], almost all classifiers seem to have to sacrifice some stability in order to be more unbiased.

Uncertain classifiers. When comparing the values of $\overline{\overline{var}}(c)$ and $\overline{\hat{H}}(c)$, it seems that some classifiers like the gender classifier that on average score well on this mean pooled variance do not necessarily have to score well when quantified with the pooled normalized entropy metric. This tells us that some classifiers, like the gender classifier, often make predictions with relatively low confidences of around ≈ 0.5, causing the discrete label to flip while the variance in the confidence can remain relatively low. Labels from such classifiers should be interpreted extra carefully.

The effect of representation on classifier stability. The results presented in Section 3.4.2 and 3.4.3 further show that these classifiers behave unexpectedly on in-the-wild data, providing evidence that it is likely that properties of the input signal such as the codec and the bitrate used to encode it

play a role in the stability of the results obtained from high-level music classifiers. While it is difficult to pinpoint the exact effect of these different audio properties — the nature of the dataset does not allow for studying the effects of just one property, since users organically submitted different recordings with different audio properties, resulting in different subsets of data when filtering on a specific metadata field — the most important conclusion presented by this data is that audio encoding definitely seems to play an important role for classifier stability, which is in line with earlier work that showed that such properties also effect low-level audio features [143].

Furthermore, while the nature of the dataset does not allow for a strong, statistical claim, it seems that lossless codecs perform significantly better than lossy ones — with higher bitrates resulting in lower pooled variance values (Figure 3.5) and lossless codecs resulting in significantly lower pooled variances (≈ 0.010, Figure 3.6) compared to lossy codecs (ranging from 0.010 to 0.120, Figure 3.7). And even when the bitrate is the same, there seem to be differences in stability when using different codecs (Figure 3.7).

Conclusion. Answering **RQ1** in short, stability metrics like those presented in this Chapter are useful for quantifying the performance of classifiers in the wild. The resulting analysis seems to suggest that many classifiers in the high-level music classification domain seem to be perform sub-par, with classifiers being either quite biased or relatively unstable given slightly different representations of the same input data. While this certainly does not mean the resulting labels obtained from these classifiers are useless, it does indicate that we should carefully consider how we use these labels in future research. Considering them as absolute truth might be dangerous and lead to incorrect scientific decisions. In addition, since the metrics are concerned only with input-output pairs, they can be applied to measure the performance of classifiers in other domains as well.

As for **RQ1.1** and **RQ1.2**, it seems that both bitrate and codec as well as software versioning might play a role in the stability of high-level music classifiers. However, it is conceivable that more factors like for example the platform a classifier is ran on or the CPU architecture[12] might also influence the stability of these classifiers in the wild, opening the door for more elaborate testing setups for future research which will further be discussed in Chapter 7.

[12]See for example this issue on the Essentia GitHub page: `https://github.com/MTG/essentia/issues/179`

4

Classifier Agreement

While Chapter 3 showed that classifier performance can be quantified using the sensitivity to differences in input representation like codec and bitrate, and that the studied classifiers are sensitive to small variations in the low-level features produced by feature extraction algorithms earlier in the machine learning pipeline, classifier performance can also be low when the classifiers do not correctly classify the construct they are supposed to model. For instance, if we train a mood classifier to label two different moods, and the training dataset for the one mood on average has a higher BPM, then the classifier might simply resort to classifying based on BPM instead of mood if no additional constraints are applied. The results from Chapter 3 suggest that the classifiers in the AcousticBrainz dataset probably are sensitive to more than just the relevant musical concepts in the data, but to further analyze if such classifiers are able to model the complex concepts that we try to train them on we can use the notion of *classifier agreement* and explore **RQ2**: How can outputs from multiple implementations of classification tasks be leveraged to quantify the performance of a classifier in terms of agreement?

This concept of looking at the agreement between multiple implementations of the same classification task is inspired by the more general process of how we conduct science: it is often difficult to assess if outcomes we are observing (like the accuracy rating of a music classifier in lab circumstances) or inferences and conclusions we draw from them are truly correct. Here, we are interested in gaining some insights into the *validity* of the classifiers, and a parallel can be drawn between this applied case and the more general domain of psychological testing. In psychological testing we often wish to measure some abstract psychological construct that is not directly observable, but is often measured with surveys by asking many different questions which all hopefully capture the same concept. More specifically, we aim to test test for the *construct validity* of these classifiers to see if they provide a correct

> ”measure of some attribute or quality which is not ’operationally defined’” [33]

In science in general, we often relate measurements to outcomes of previous research which were shown to be valid [33], and if these measurement correspond with the previously accepted theory and data then we reasonably assume that the new measurements are correct as well. For the data available here we do not have any previous classifier results which we can assume to be valid (since Chapter 3 showed us that this is probably not the case), but we can do something similar: we can look at different measurements (outputs produced by different classifiers) and see if they relate to each other in a way that makes sense from a psychological point of view. If this is not the case, then this gives us evidence for the *lack* of construct validity in these classifiers.

The idea is relatively straightforward: say we have two classifiers, A and B that both supposedly model the same musical concept. Because both classifiers are designed and trained by different people and on different data, the trained weights and overall architectures probably differ significantly. If we assume that both classifiers have correctly identified the musical concepts required to model the desired property of the music, then when we run the trained classifiers A and B on a new set of data, unseen by both classifiers, both should give back similar label probabilities. If in most cases A and B give back different label probabilities for recordings in this unseen validation dataset, then either:

- Our assumption that *both* classifiers have correctly identified and modeled the musical concepts required is false, *or*
- *One* of the classifiers has correctly identified and modeled the musical concepts, while the other classifier overfit on irrelevant data present in the representation of the recordings, *or*
- *Both* classifiers overfit on irrelevant data present in the representation of the recordings, but in a different way

Consequently, if the correlation between label probabilities given by A and B is high then this tells us either:

- Both classifiers correctly identified and modeled the musical concepts required, *or*
- Both classifiers overfit on irrelevant data present in the representation of the recordings, but *in the exact same way*.

We assume that the second conclusion here is more unlikely, given that the classifiers are designed by different groups of people, using different datasets and training procedures. Thus, if we observe high correlations between the label probabilities produced by A and B then it is *more probable* that the classifiers correctly model the musical concept. If we observe relatively low correlations, then it is probable that either one or both of the classifiers incorrectly model the musical concept.

To analyze this classifier agreement, two different approaches will be applied. The first will measure the **intra-dataset agreement** between different high-level classifiers. For this, the AcousticBrainz dataset can be used, since many different high-level classifiers are run on the same data and there are classifiers in the dataset that should in theory model the same concept like label `not_electronic` by `mood_electronic` and label `acoustic` by `mood_acoustic` (since if a recording is highly likely to be acoustic, and electronic music is not acoustic, then the recording should also be not electronic with a high probability). The second part will measure the **inter-dataset agreement**, which will incorporate a second dataset and set of high-level classifiers, sourced from Spotify.

As an example, the first submission of ”Let It Be” by the Beatles on AcousticBrainz reports a probability of 0.904 of the recording not being acoustic, or 0.096 of being an acoustic track. Spotify, when queried with the same song reports back a value for acousticness of 0.631. While AcousticBrainz was very certain that this song was *not* acoustic, the best guess provided by Spotify is that the song probably *is* acoustic, meaning the two classifiers do not agree with each other. Does this also happen on a larger scale? What does this mean for the classifiers? These are the questions we wish to explore in this chapter.

4.1. Data acquisition

For the intra-dataset agreement, the same AcousticBrainz dataset as described in Section 3.1 can be used. We can use the entire dataset, since we do not need to filter out recordings which only have one submission: every submission is used as input for all the high-level classifiers so every submission can be used for the analysis. For the inter-dataset agreement testing this data is matched with Spotify data.

4.1.1. Spotify data

For the inter-dataset agreement we need at least one additional source of high-level music classification outputs that overlap with those present in the AcousticBrainz data. Spotify is a good match, having an API[1] that provides high-level classifier output for any given input song as long as it is available on Spotify. Spotify has a very large catalog of music with over 50 million recordings [4], which results in a very high chance that if a recording is present in the AcousticBrainz dataset, it will also be present on the Spotify platform and provides a way to fetch results of several different high-level features for a track given a spotify ID using their API [3]. The list of available high-level features relevant to the analysis of high-level music classifiers in the Spotify API is presented in Table 4.1.

The Spotify data differs from the AcousticBrainz data in that we have no way of knowing which internal representation of the music was used as input for the classifier, while AcousticBrainz includes

[1] `https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/`

Field	Type	Description
acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	float	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	float	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
instrumentalness	float	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Table 4.1: List of relevant high-level features that can be fetched using the Spotify API. [3]

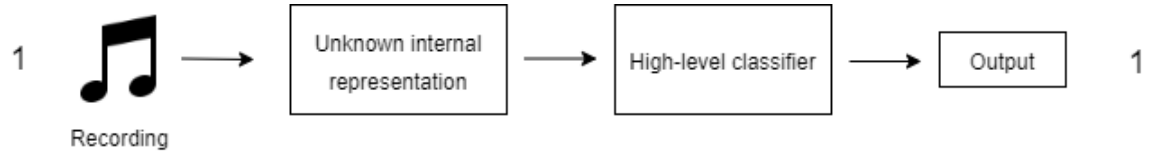


Figure 4.1: Structure of the Spotify dataset. Each recording that is queried through the Spotify API has some unknown internal representation that was used as input for the classifier, and only the classifier output is provided. Every recording only results in 1 output.

this information for every submission in the dataset. Furthermore, for every song only one output is provided instead of the N outputs provided by AcousticBrainz. Visually the structure of the Spotify dataset is shown in Figure 4.1.

4.1.2. Gathering and processing the data

The data for AcousticBrainz has already been gathered, so the data as described in Section 3.1 can simply be loaded in again for the intra-dataset analysis part. For the inter-dataset analysis, high-level feature data needs to be fetched from the Spotify servers using their API. The API must be queried using a Spotify ID, and unfortunately the AcousticBrainz dataset uses MBIDs to identify recordings. Thus, before the data can be fetched from Spotify, the MBIDs present in the AcousticBrainz dataset need to be mapped to Spotify IDs.

Constructing the mapping The Echo Nest used to run a service called the Rosetta Stone, which allowed for the translation of music IDs from platform to platform, including from MBID to Spotify ID [1]. Unfortunately, as of the 31st of May, 2016, The Echo Nest API was shut down[2] and while the Spotify API that replaced it does offer high-level features, the Rosetta Stone functionality was no longer available. Luckily, thanks to the Million Song Dataset [14] results of querying the Rosetta Stone for a million songs were saved[3]. Using these results requires several steps:

- First, a mapping between MBID and MSD ID (used by the Million Song Dataset) provided by AcousticBrainz[4] is used to get an MSD *track* ID (starting with `TR`) for some of the recordings present in the AcousticBrainz dataset. This mapping includes 250,000 MSD IDs, resulting in 370,000 matches with the AcousticBrainz dataset (since a MSD ID may map to more than one MBID). After filtering out matches between AcousticBrainz and MSD for which no high-level features were available in the AcousticBrainz dataset, 239,201 matches are left.
- Next, using these 239,201 matches between MBID and MSD track ID, we map the MSD track IDs to MSD song IDs using an SQL database file containing most metadata about each track[5] provided by MSD[6]. This results in a three way mapping between MBID, MSD track ID and MSD song ID for all 239,201 matches.
- Unfortunately, the metadata provided by MSD contains some matching errors [2], resulting in some MSD tracks IDS being matched with incorrect MSD song IDs. MSD provides a list of MSD track ID - MSD song ID mismatches[7], which were used to filter out matches in our mapping that can not be trusted. Of the 239,201 matches, 225,706 matches remain.
- Now that the MSD song IDs are available for 225,706 entries in the AcousticBrainz dataset, the saved results from the Rosetta Stone[8] can be used to look up the corresponding Spotify IDs.
- Since the saved results from Rosetta sometimes include empty responses (perhaps an API or connection failure occured when constructing the archive) some IDs can not be mapped to Spotify

[2] `https://web.archive.org/web/20160330095059/http://developer.echonest.com/`
[3] `https://labs.acousticbrainz.org/million-song-dataset-echonest-archive/`
[4] `https://labs.acousticbrainz.org/million-song-dataset-mapping/`
[5] `http://millionsongdataset.com/sites/default/files/AdditionalFiles/track_metadata.db`
[6] `http://millionsongdataset.com/pages/getting-dataset/`
[7] `http://millionsongdataset.com/sites/default/files/tasteprofile/sid_mismatches.txt`
[8] `https://labs.acousticbrainz.org/million-song-dataset-echonest-archive/`

	title	artist	msd	msd_songid	mbid	spotify
0	No One Could Ever	Hudson Mohawke	TRMMMRX128F93187D9	SOGTUKN12AB017F4F1	7f9264e2-be8e-4e8e-a58e-04026d096a85	41RpZW2lxAdnqDd2nMBzLQ
1	(Looking for) The Heart of Saturday Night	Shawn Colvin	TRMMMUT128F42646E8	SOBARPM12A8C133DFF	d3e506fc-574a-4b0a-8c5d-9618c1644164	5SM86TB7dU5n9Y23wLgcBY
2	Ethos of Coercion	Dying Fetus	TRMMMQY128F92F0EA3	SOKOVRQ12A8C142811	ae7be227-5112-4d21-adbb-662e4c23a90d	0ghgsfOnoXJT7jsS63U8et
3	Nervous	Nicolette	TRMMMPN128F426610E	SOGFWVT12A8C137C64	f4fddf93-9e1d-4fcc-843a-e4f11ae69a28	6mxDT6y9Sdp2802sieuwOA
...	...	...	...	...	...	...

143957 rows × 6 columns

Figure 4.2: Sample of the final mapping constructed for analyzing the inter-dataset agreement between Acousticbrainz (using MBID as an identifier) and Spotify.

IDs. After filtering out these entries, 143,957 matches between MBID and Spotify remain. This mapping is the mapping that will be used for the inter-dataset agreement testing. A small sample of the full mapping is presented in Figure 4.2

Note that this mapping relies on the accuracy of several different datasets, and might still contain some matching errors after filtering out the *known* errors. This might result in slightly noisy data, but we assume that the relatively large sample size diminishes the impact of this on the resulting analyses.

4.2. Metric

Since we wish to analyze the agreement between label probabilities (which are floats) and see if these label probabilities between the different classifiers 'agree' with each other, we need some metric that can quantify the relationship between the label probabilities output by both classifiers. For this the Pearson Correlation [48] can be used:

$$r_{x,y} = \frac{\Sigma_{i=1}^{n}(x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \overline{x})^2 \Sigma_{i=1}^{n}(y_i - \overline{y})^2}}$$

Where x and y are vectors of label probabilities given by two different classifiers, and $\overline{x}$ is the mean of vector x. Intuitively, if the two classifiers agree with each other, then when a label probability in vector x is high, the corresponding label probability vector y will also be high. Thus, when using the Pearson correlation and by making sure that the label probability vectors are chosen in a way such that a higher value for the one label should also mean a higher label value for the other, a higher Pearson correlation implies a higher level of agreement between the two classifiers, increasing our trust in the construct validity of the classifier under study.

All correlation calculations in this analysis use the Pearson Correlation implementation of SciPy[9], which report a two-sided p-value using a beta distribution on the interval $[-1, 1]$.

Using this metric, we would expect high levels of correlation between the chosen classifier label probabilities, given that the chosen label probability vectors should result in similar probabilities.

4.3. Agreement Analysis

For each of the following analyses, first the theoretical agreement (i.e. which labels should give back roughly the same probabilities) between different classifier labels will be analysed, before the correlation results calculated using the metric are presented.

4.3.1. Intra-dataset agreement

This section will describe the results of the agreement analysis on the AcousticBrainz dataset. Given the classifiers and labels present in the AcousticBrainz dataset (for a full overview, refer to Table 3.3), the labels presented in Table 4.2 should model the exact same concept and thus should have a very high correlation coefficient.

However, the resulting correlations as presented in Table 4.3 are worryingly low, with some of the labels even having a slight negative correlation. Given that the classifiers under study here should

[9] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

model the *exact* same concept, and the data should not be noisy at all — the analysis is ran on the AcousticBrainz data, and thus we can be sure that the input for both classifiers was exactly the same in all cases — we would expect the correlations to be much higher. Results like these highlight that we should not put much trust in the construct validity of the analysed classifiers.

Classifier, label A	Classifier, label B
genre_dortmund, blues	genre_tzanetakis, blu
genre_dortmund, jazz	genre_tzanetakis, jaz
genre_dortmund, pop	genre_tzanetakis, pop
genre_dortmund, rock	genre_tzanetakis, roc
genre_dortmund, jazz	genre_rosamerica, jaz
genre_dortmund, pop	genre_rosamerica, pop
genre_dortmund, rock	genre_rosamerica, roc
genre_rosamerica, cla	genre_tzanetakis, cla
genre_rosamerica, hip	genre_tzanetakis, hip
genre_rosamerica, jaz	genre_tzanetakis, jaz
genre_rosamerica, pop	genre_tzanetakis, pop
genre_rosamerica, roc	genre_tzanetakis, roc

Table 4.2: Genre labels that model the exact same concept. These labels should have a very high correlation in the correlation analysis.

Classifier, label A	Classifier, label B	Pearson r	p
genre_rosamerica, cla	genre_tzanetakis, cla	0.287475	<.001
genre_dortmund, rock	genre_rosamerica, roc	0.237809	<.001
genre_dortmund, jazz	genre_rosamerica, jaz	0.217246	<.001
genre_dortmund, pop	genre_rosamerica, pop	0.113847	<.001
genre_dortmund, jazz	genre_tzanetakis, jaz	0.0809686	<.001
genre_rosamerica, pop	genre_tzanetakis, pop	0.0561556	<.001
genre_rosamerica, hip	genre_tzanetakis, hip	0.0481102	<.001
genre_rosamerica, jaz	genre_tzanetakis, jaz	0.0206385	<.001
genre_dortmund, blues	genre_tzanetakis, blu	0.00790441	<.001
genre_dortmund, pop	genre_tzanetakis, pop	-0.0500181	<.001
genre_dortmund, rock	genre_tzanetakis, roc	-0.059596	<.001
genre_rosamerica, roc	genre_tzanetakis, roc	-0.074783	<.001

Table 4.3: Pearson correlations between the classifiers described in Table 4.2.

The label pairs presented in Table 4.4 have a relation that is not exactly one-to-one, but we would still expect them to agree with each other to a moderately high degree based on the reasoning presented in the Table. Interestingly enough, the correlation results as presented in Table 4.5 for these 'looser' relations are higher than those for the exact relations in Table 4.3 with some classifier pairs like `mood_aggressive` - `mood_relaxed` even having quite high correlation values.

While these more exact concepts like genre seemed to perform better than the more abstract concepts like mood in the stability analysis presented in Chapter 3, using the agreement metric they score poorly, further suggesting that such classifiers might only display stable behaviour due to them failing to capture the correct underlying construct or by simply outputting the majority class most of the time as was also observed in Chapter 3 by the normalized entropy metric.

Furthermore, it is interesting to see that using this agreement metric, the `mood_happy` and `mood_sad` classifiers seem to perform the worst while they performed relatively well in the stability analysis in Chapter 3. This might be due to the fact that mood as a construct is relatively hard to grasp compared to constructs like `danceability` or `acousticness`, as was also shown in the literature study in Section 2.4.

Classifier, label A	Classifier, label B	Reasoning
danceability, danceable	mood_party, party	Should be moderately high, assuming that dancing is an activity common at parties
danceability, danceable	genre_rosamerica, dan	Should be high, assuming that music with genre 'dance' is danceable
mood_acoustic, acoustic	mood_electronic, not_electronic	Should be high, since acoustic music is non-electronic
mood_aggressive, aggressive	mood_relaxed, not_relaxed	Should be high, since aggression is a non-relaxed mood
mood_electronic, electronic	genre_dortmund, electronic	Should be moderately high, assuming that music with an electronic 'mood' is also of genre electronic
mood_happy, happy	mood_sad, not_sad	Should be high, since happy music should not be sad
mood_happy, happy	mood_party, party	Should be moderately high, assuming that people party to happy music

Table 4.4: Mood based labels that should have a relatively high correlation. While these labels do not model the exact same concepts, they should still be somewhat correlated given the reasoning.

Classifier, label A	Classifier, label B	Pearson r	p
mood_aggressive, aggressive	mood_relaxed, not_relaxed	0.588771	<.001
mood_acoustic, acoustic	mood_electronic, not_electronic	0.579642	<.001
danceability, danceable	mood_party, party	0.527954	<.001
mood_electronic, electronic	genre_dortmund, electronic	0.481804	<.001
danceability, danceable	genre_rosamerica, dan	0.33418	<.001
mood_happy, happy	mood_party, party	0.200259	<.001
mood_happy, happy	mood_sad, not_sad	0.131705	<.001

Table 4.5: Pearson correlations between the classifiers described in Table 4.4.

However, while the agreement scores are much higher than those between the genre classifiers, it can be argued that they are still relatively low. Thus, based on this metric employed on the AcousticBrainz data it seems like caution is in order when assuming that high-level music classifiers are able to correctly capture the desired constructs.

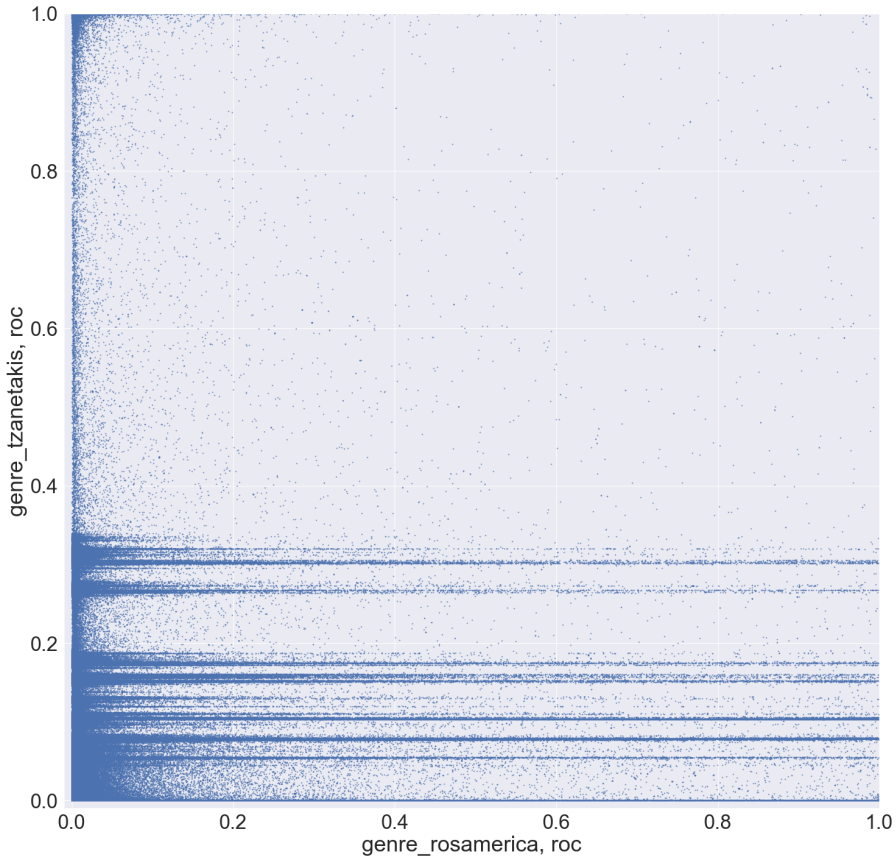
Correlation plots Some of the relatively low Pearson correlations for the genre classifiers and the mood classifiers were further investigated by plotting the label probabilities in a two dimensional plane. This visualization allows us to see the data distribution of the two labels to see why they do not correlate highly. The two plots for the classifier combinations that resulted in the lowest correlation can be seen in Figure 4.3.

Note that several anomalies can be seen, with large parts of the distribution being empty and some label probabilities containing most of the data points for the genre classifiers, and missing values in the distribution of the mood classifiers. All other genre classifiers showcased the same 'banding' behaviour that can be seen, with many inputs resulting in specific probability values. The pattern of missing probability values around 0.5, and higher densities for other values was also present with all other mood classifiers.

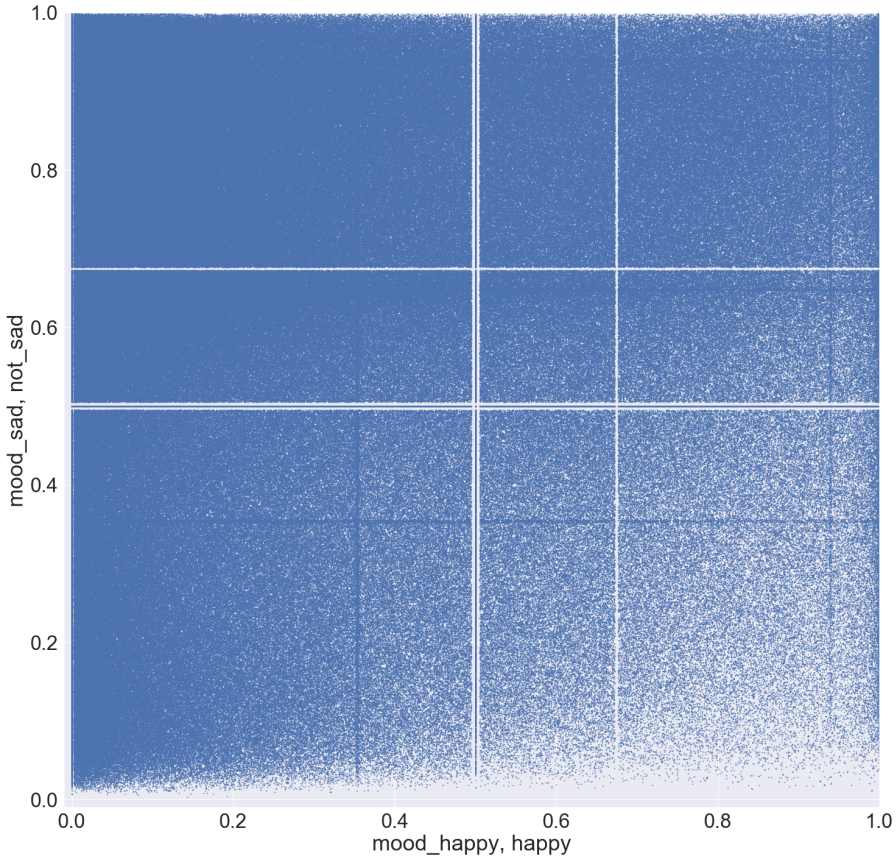
These anomalies in the distribution can not be easily explained using either the stability or agreement approaches discussed so far, and prompted the in-depth analysis of the label probability distributions of the classifiers presented in Chapter 5.

4.3.2. Inter-dataset correlation

For the Inter-dataset correlation analysis high-level feature data was downloaded using the Spotify API and the mapping produced in Section 4.1.2. Looking at the high-level features present in the AcousticBrainz dataset (Table 3.3) and in the Spotify data (Table 4.1), results in the correlation hypotheses



(a) Value distribution behaviour for two genre classifiers in the AcousticBrainz dataset.



(b) Value distribution behaviour for two mood classifiers in the AcousticBrainz dataset.

Figure 4.3: Two correlation plots for the genre and mood classifiers with the lowest Pearson correlation values. Note that the behaviour was similar for other genre and mood classifiers.

Spotify feature	AcousticBrainz feature
acousticness	mood_acoustic, acoustic
danceability	danceability, danceable
energy	mood_relaxed, not_relaxed
instrumentalness	voice_instrumental, instrumental
valence	mood_happy, happy
valence	mood_sad, not_sad

Table 4.6: Correlation hypotheses for features present in the Spotify data and in the AcousticBrainz data. Each row represents a pair of classifiers that should be correlated to a relatively high degree.

as presented in Table 4.6. Again, most of these concepts are largely the same, and if both the AcousticBrainz and Spotify classifiers properly captured the underlying construct then we would expect high agreement scores.

Since in the Spotify data there is no notion of multiple submissions per recording, we have the AcousticBrainz dataset which has at least one, but possibly multiple submissions for one recording (which are relatively unstable as shown in Chapter 3) which need to be compared to the Spotify data for which only one 'submission' is available per recording. This results in the following two ways of calculating the inter-dataset correlation between these datasets:

- Duplicate the Spotify label probability for every AcousticBrainz submission for a recording. As an example: if AcousticBrainz has three submissions for a recording resulting in label probabilities $[0.2, 0.3, 0.4]$ and Spotify has a label probability of 0.3 for this recording the correlation would be calculated as $r_{x,y}$ with $x = [0.2, 0.3, 0.4]$ and $y = [0.3, 0.3, 0.3]$. This will be named `corr_1`.

- Take the average of the label probability in the AcousticBrainz dataset so that each recording has only one label probability and *then* calculate the correlation between the datasets. Using the same example we would then calculate $r_{x,y}$ with $x = \frac{0.4+0.3+0.2}{3} = [0.3]$ and $y = [0.3]$. This will be named `corr_2`

If not specified further, then `corr_1` is used when reporting the Pearson coefficient for the inter-dataset correlation analysis. Table 4.7 shows the results of the correlation analysis on this data using the two approaches described above. Note that the two correlation approaches result in similar correlation scores.

Spotify feature	AcousticBrainz feature	Pearson r corr_1	corr_2
acousticness	mood_acoustic, acoustic	0.649977	0.670740
danceability	danceability, danceable	0.258386	0.277237
energy	mood_relaxed, not_relaxed	0.454956	0.472379
instrumentalness	voice_instrumental, instrumental	0.282487	0.319092
valence	mood_happy, happy	0.236280	0.263427
valence	mood_sad, not_sad	0.153313	0.145066

Table 4.7: Pearson correlations for high-level classifiers between Spotify and AcousticBrainz using the two different approaches.

Also interesting to note is that these results again show that the agreement scores are lowest for the `mood_happy` and `mood_sad` classifiers, further solidifying the idea that such classifiers might not properly capture the underlying mood constructs of the input data. When compared to the intra-dataset agreement scores as presented in Table 4.5, most of the agreement scores are similar, with the biggest difference being that the danceability here scored relatively low while in the inter-dataset agreement analysis danceability was one of the highest scoring classifiers. This might indicate that the Spotify implementation of danceability is different which might be due to either implementation or interpretation differences.

4.4. Effect of audio representation on classifier agreement

Since Sections 3.4.2 and 3.4.3 showed that the audio bitrate and codec might affect the variance in label probabilities produced by the high-level classifiers, it is worth investigating how much of an effect such different audio representations have on the agreement metric — it might be the case that representational differences only influence the stability of a classifier without impacting construct validity. For this analysis, the correlations under study are the inter-dataset ones between AcousticBrainz and Spotify. The data has been preprocessed much in the same way as in Section 3.4.2, filtering out uncommon bitrates. Since this analysis is concerned with correlation, and not variance, filtering out recordings for which there was only one submission is not necessary.

4.4.1. Effect of bitrate and codec

Bitrate effects To study the effect of the bitrate on the correlation between the Spotify and the AcousticBrainz classifiers, the only preprocessing that is applied on the AcousticBrainz data is filtering out submissions that have an uncommon bitrate in the metadata and then selecting the submissions for which Spotify data is available using the mapping constructed in 4.1.2. For this, much like in Section 3.4.2, all submissions with a bitrate that occurs less than 10,000 times in the entire dataset are filtered out. See Table 4.8 for the distribution across bitrates used for this analysis.

Bitrate	Count
0	71696
192000	35177
320000	24465
128000	20700
160000	8463
256000	5536

Table 4.8: Amount of submissions that remain for each bitrate in the AcousticBrainz dataset after filtering out the uncommon bitrates and selecting only those for which Spotify data is available. A bitrate of 0 indicates lossless.

Note that for the Spotify data, no information is available about which audio representation was used for calculating the high-level features. Thus, the analysis looks at the different *subsets* of the AcousticBrainz dataset with a certain bitrate and calculates the correlation against the *entire* Spotify dataset. In this way we can gain some basic insight into the effect of the bitrate of the input data on the ability of the classifier to correctly capture the desired construct from this input data. See Figure 4.4 for the results of running these correlation calculations.

Again, just like with the analysis in Section 3.4.2 the underlying data for each of the subsets with a specific bitrate is different since for most recordings submissions have only been made using one or two different bitrates. This means that while this analysis suggests that higher bitrate input data results in higher agreement scores indicating that such classifiers are able to perform better on higher quality input data, we can not be sure if the underlying data also influenced this metric. More controlled experiments will be needed to exactly measure the effect of bitrate on such agreement scores.

Codec effects To study the effect of the codec used to encode the data and the correlation between the AcousticBrainz and Spotify data we can now take a more controlled approach than in Section 3.4.2 since the nature of our data and metric now allows for it. Since we wish to study the effect of the codec used to encode the input data on the agreement between the classifiers in AcousticBrainz and Spotify we prepare the data as follows:

1. Say we wish to study the difference in agreement between the AcousticBrainz and Spotify classifiers for input data encoded using codec C_1 and data encoded using codec C_2, both using bitrate B.

2. Since we require Spotify data to be available for analysing this agreement, we first use the mapping constructed in Section 4.1.2 to filter out recordings in the AcousticBrainz data for which we do not have any Spotify data.

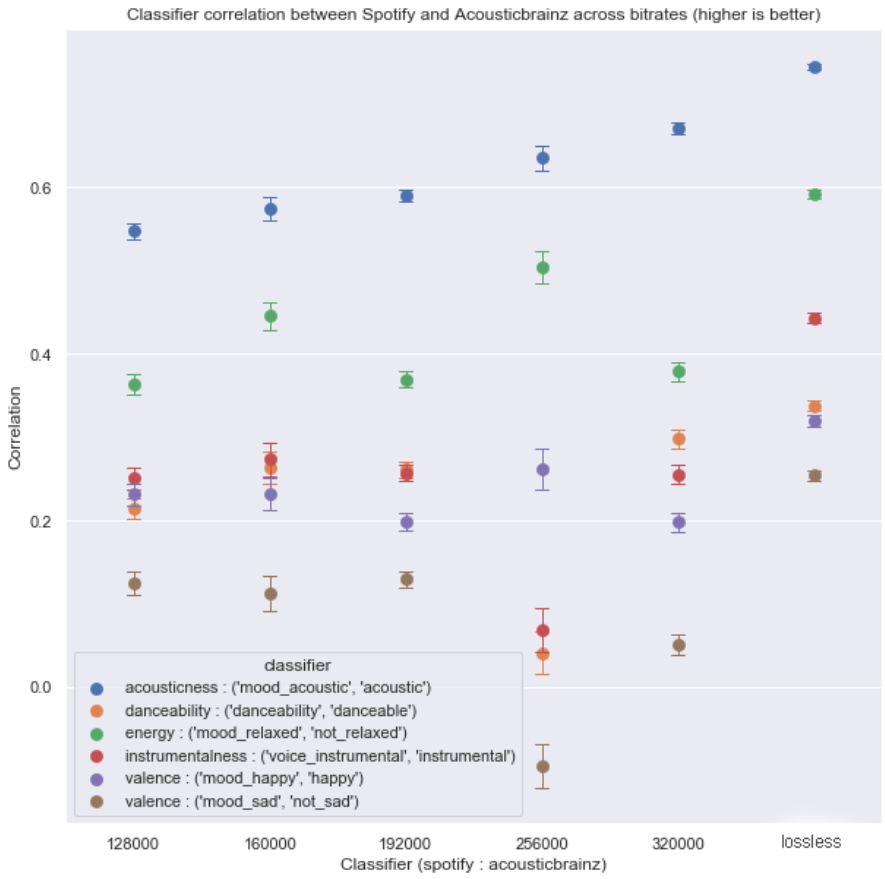


Figure 4.4: Label probability correlations between Spotify and AcousticBrainz classifiers for subsets of AcousticBrainz data with different bitrates. Bars indicate the 95% confidence intervals for the correlation.

3. Then, this data is filtered so that only AcousticBrainz submissions using the specified bitrate B remain.

4. On the resulting data, all pairss of submissions are found for which the MBID (the recording) is the same and one of the submissions was encoded using codec C_1 and the other using C_2. Save the submission encoded using C_1 in a vector A and the submission encoded using C_2 in vector B.

Now, given a bitrate and a pair of codecs we have vectors A and B of length n of corresponding pairs encoded using the two different codecs, for which we know for sure that the underlying data (the recording) of each pair we compare is exactly the same. Note that while finding such pairs would have been desirable in the analysis in 3.4.2, due to the stability metric requiring at least two submissions to be calculated not enough of those pairs existed in the data.

Say the corresponding vector of label probabilities obtained from the Spotify data is given by S. We want to test if there is any significant difference in the agreement between vector A and S and B and S (since then the codec has a significant effect on the agreement). To do this we calculate the Absolute Error (AE) for each of the elements in both vectors by subtracting the label probability as given by Spotify from the label probability as given by AcousticBrainz and taking the absolute value.

Then, we can use the paired samples t-test (implementation by SciPy[10] to see if there is a significant difference in the Mean Average Error (MAE) between the two vectors (which correspond to the exact same input data, only encoded using a different codec). For this test, the null hypothesis is that both samples have identical average values and if the p-value is sufficiently small then we can reject this null hypothesis of equal averages for these two samples. The resulting analyses are presented in Table 4.9.

The data showed a significant effect for codec for some of the classifiers using bitrate 192000 and all classifiers using bitrate 320000. Note that bitrate 320000 had the largest sample size ($n = 1203$). Interestingly enough for the cases where the bitrate was 192000 and the data showed a significant difference in MAE this was mostly in favour of mp3 — with mp3 having lower values for MAE than vorbis. For bitrate 320000 this was the inverse, where vorbis consistently had lower values of MAE than mp3. This might indicate that the mp3 codec performs better — at least in terms of resulting classifier agreement — at lower bitrates while vorbis performs better at higher bitrates. For the bitrates 160000 and 256000 almost no significant differences were observed, but this might also be due to the relatively small sample size. Again, it would be beneficial to run further controlled experiments. However, these first results do seem to indicate that the codec, and thus the representation of the input data, can affect the performance of classifiers that use this data.

4.4.2. Effect of other metadata

Using the same approach as in the codec comparison presented above, but instead searching for pairs where the low-level feature extractor differs, we can check if the low-level feature extractor version used affects the agreement between AcousticBrainz and Spotify in a more controlled way than in Chapter 3, again due to the agreement metric not requiring multiple submissions of the same input resulting in more data being available. The resulting paired t-test analysis is presented in Table 4.10. The results agree with those using the stability metric in a less controlled fashion as presented in Section 3.4.3: classifier results ran over features generated by version 2.1-beta2 seem to be better, resulting in higher agreement with the Spotify data. A further study into the source code of these two version could be very beneficial in finding out why the performance of classifiers calculated with different versions seem to change significantly.

4.5. Discussion

This Chapter provides a second metric to quantify classifier performance without relying on ground truth data. Instead, it relies on different implementations of the same classification task, and the application on the AcousticBrainz data presented in this chapter provides further evidence suggesting that high-level music classifiers might not perform as well on unseen, in-the-wild data. This section will discuss the construct validity of the classifiers under study, the effects of other metadata fields on the classifier agreement and the distributional anomalies observed in the correlation plots.

[10] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

	MAE		t	p
	mp3	vorbis		
(acousticness, (mood_acoustic, acoustic))	0.215574	0.192192	0.23	.820
(danceability, (danceability, danceable))	0.439985	0.326822	3.11	.006**
(energy, (mood_relaxed, not_relaxed))	0.286043	0.286476	-0.007	.994
(instrumentalness, (voice_instrumental, instrum...	0.600029	0.474101	0.987	.337
(valence, (mood_happy, happy))	0.202633	0.356557	-1.910	.072
(valence, (mood_sad, not_sad))	0.377937	0.388755	-0.171	.866

(a) Bitrate: 160000, n=19

	MAE		t	p
	mp3	vorbis		
(acousticness, (mood_acoustic, acoustic))	0.172869	0.193616	-2.14	.033*
(danceability, (danceability, danceable))	0.459768	0.526277	-4.71	<.001***
(energy, (mood_relaxed, not_relaxed))	0.411277	0.463551	-3.62	<.001***
(instrumentalness, (voice_instrumental, instrum...	0.578092	0.587861	-0.44	.659
(valence, (mood_happy, happy))	0.369526	0.398368	-2.42	.016*
(valence, (mood_sad, not_sad))	0.278248	0.256190	2.28	.023*

(b) Bitrate: 192000, n=299

	MAE		t	p
	mp3	vorbis		
(acousticness, (mood_acoustic, acoustic))	0.180786	0.177941	0.13	.897
(danceability, (danceability, danceable))	0.326517	0.331549	-0.35	.730
(energy, (mood_relaxed, not_relaxed))	0.238797	0.240557	-0.12	.906
(instrumentalness, (voice_instrumental, instrum...	0.346797	0.313191	1.10	.277
(valence, (mood_happy, happy))	0.245510	0.235400	0.55	.583
(valence, (mood_sad, not_sad))	0.372572	0.376357	-0.20	.840

(c) Bitrate: 256000, n=74

	MAE		t	p
	mp3	vorbis		
(acousticness, (mood_acoustic, acoustic))	0.161619	0.125306	6.33	<.001***
(danceability, (danceability, danceable))	0.439657	0.306358	19.98	<.001***
(energy, (mood_relaxed, not_relaxed))	0.424091	0.270790	18.94	<.001***
(instrumentalness, (voice_instrumental, instrum...	0.551969	0.294039	21.17	<.001***
(valence, (mood_happy, happy))	0.307287	0.283616	3.82	<.001***
(valence, (mood_sad, not_sad))	0.353860	0.320952	6.20	<.001***

(d) Bitrate: 320000, n=1203

Table 4.9: Results of paired t-test analysis comparing the Mean Average Error of AcousticBrainz data encoded using the specified bitrate and codec and the corresponding Spotify data. Lower values for MAE indicate higher agreement between the AcousticBrainz and Spotify data.
*: $p \leq 0.05$
**: $p \leq 0.01$
***: $p \leq 0.001$

	MAE		t	p
	2.1-beta1	2.1-beta2		
(acousticness, (mood_acoustic, acoustic))	0.174532	0.148726	54.23	<.001***
(danceability, (danceability, danceable))	0.430322	0.311341	242.42	<.001***
(energy, (mood_relaxed, not_relaxed))	0.424959	0.259122	267.79	<.001***
(instrumentalness, (voice_instrumental, instrum...	0.603558	0.302066	333.66	<.001***
(valence, (mood_happy, happy))	0.303454	0.265384	81.74	<.001***
(valence, (mood_sad, not_sad))	0.353758	0.310109	123.90	<.001***

Table 4.10: Result of paired t-test analysis comparing the Mean Average Error of AcousticBrainz data ran on features generated by the specified Essentia version and the corresponding Spotify data. Lower values for MAE indicate higher agreement between the AcousticBrainz and Spotify data.
*: $p \leq 0.05$
**: $p \leq 0.01$
***: $p \leq 0.001$

Construct validity The relatively low correlation scores presented in this chapter might indicate that the classifiers under study failed to correctly capture the underlying construct they wish to model. This is in line with the results presented in Chapter 3 that already seemed to indicate that these classifiers utilize data that is irrelevant to the desired construct to produce the labels, with many of the classifiers sensitive to small changes in the audio signal that do not change the higher level constructs of the audio.

What is interesting, and perhaps most worrisome, is that the correlations between the genre classifiers — which should model the exact same construct — were relatively low and in some cases these label probabilities even correlated negatively. This might indicate that the genre labels like 'classical' or 'rock' are too broad, giving the classifier insufficient information to learn to correctly classify this construct. It might, however, also be the case that the `genre_tzanetakis` classifier is underperforming, since all correlation scores < 0.10 included this classifier as one of the two classifiers used to calculate the metric. However, the highest correlation of ≈ 0.29 for the genre 'classical' also included the `genre_tzanetakis` classifier.

The high-level constructs which have some intuitive relation to each other, of which was hypothesised that these would produce relatively lower correlation scores, scored higher than the genre classifiers for all but one of the classifier pairs, with some classifier pairs scoring correlations above 0.5. Here, the correlation between `mood_happy, happy` and `mood_sad, not_sad` was the lowest, implying that music that is classified as happy can also be sad at the same time. While this contradicts Rusell's 2D circumplex model of affect [126] where happiness and sadness have opposite scores on the valence dimension, it is in line with the opinion of some musicologists who believe that music can evoke both emotions at the same time, however the concepts of happiness and sadness have showed negative correlation in the past [42].

Classifier correlations between Acousticbrainz and Spotify were also relatively low, with only `acousticness` correlating relatively highly (> 0.5) with the Acousticbrainz classifier `mood_acoustic, acoustic`. Again, the correlation relating to the `mood_happy, happy` and `mood_sad, sad` classifiers were lowest, this time when correlated with the `valence` feature in Spotify. This might indicate that either the mood is one of the more tricky constructs to model (which is in line with the analysis presented in Section 2.4) or that the valence model employed by Spotify does not translate well to the model in AcousticBrainz, where there is one binary classifier for sadness and one for happiness.

Metadata effects The results presented in this section also further support the main observation of Chapter 3: audio representation matters for high-level music classifiers. While Chapter 3 hinted at the possibility that relatively 'lower' quality music might result in more *instability* in the label probabilities produced by these classifiers, this Chapter showed that higher bitrates generally also seem to result in higher agreement scores between the classifiers in AcousticBrainz and Spotify. It might be the case that the agreement is higher simply because the labels used for calculating these scores are more stable. However, some of the classifiers that performed well on the stability metric — like the genre classifiers — performed poorly on the agreement metric, which makes this seem unlikely. It might be

the case that the classifiers are better able to capture the underlying high-level constructs on higher quality audio representations, resulting in the higher agreement score.

Data distributions The correlation plots presented in this chapter indicate that the underlying distribution of label probabilities for the high-level music classifiers is not uniform. Many label probabilities rarely occur at all, while others seem to be overrepresented. These distributional anomalies can not be easily explained and deserve a more in depth analysis which will be presented in Chapter 5 since it might be that these anomalies also affect the performance of these classifiers much like the stability and agreement.

Conclusion The agreement scores presented in this chapter further indicate that the high-level classifiers under study might be underperforming on in-the-wild data, raising doubts about the ability of the classifiers to capture the correct high-level constructs in the training phase in the lab. While the low agreement scores might be a result of the low stability observed in Chapter 3, other factors like data representation or distributional anomalies might also play a role.

5

Classifier value distributions

Chapters 3 and 4 showed that many of the high level classifiers are not very stable over slight variations of the same input data, and that many of the the classifiers have relatively low agreement scores, both internally and when compared with the Spotify data. While these results on their own already suggest that the chance of validity of such classifiers is limited, the agreement analysis also showed that some of the classifiers have strangely distributed label probabilities with some values occurring more often and others values not occurring at all (see Figure 4.3).

Since the underlying distribution of the data can have a large effect on for example the correlation scores — a classifier which always gives back the same value can not correlate with another classifier — these observed anomalies in the distribution of the label probabilities deserve a more thorough analysis. This chapter will take a closer look at the label probability distributions for several of the high-level classifiers present in the AcousticBrainz data, and investigate a number of possible causes for these anomalies using available metadata, genre data and a decision tree approach by employing a distributional distance metric. Finally, these distributions of the AcousticBrainz classifier probilities will be compared with the distribution of the Spotify labels.

5.1. AcousticBrainz label distributions

Since the main focus of this thesis is the stability and reliability of high-level music classifiers related to mood, and because plots will become less readable if distributions for all classifiers are plotted, this analysis will focus on the distribution of label probabilities produced by the following classifiers and labels (the first term of the tuple denotes the classifier, the second term denotes the label):

- (danceability, danceable)
- (mood_acoustic, acoustic)
- (mood_aggressive, aggressive)
- (mood_electronic, electronic)
- (mood_happy, happy)
- (mood_party, party)
- (mood_relaxed, relaxed)
- (mood_sad, sad)

Each of these combinations of classifier and label produces a vector of probability values in $[0, 1]$, with every value resulting from running the classifier on a submission. To visualize the distribution, these probability values are binned into N bins of equal size. If we index the bins from $0 \ldots N-1$ and denote the ith bin as B_i then:

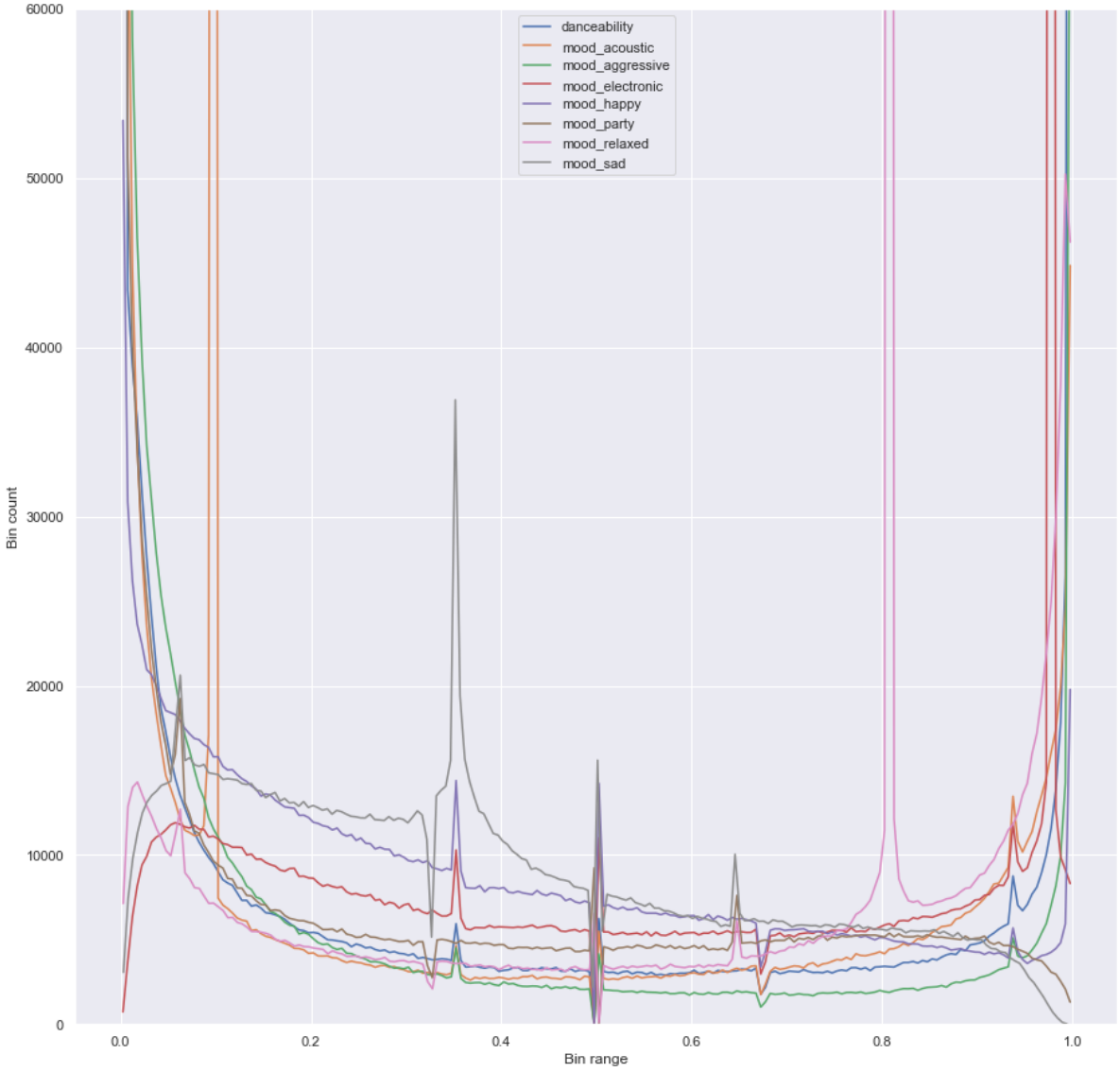


Figure 5.1: Distribution plot of label probability distributions of several high-level classifiers by means of binned counts using 200 bins. The x-axis denotes the lower bound of the range of a bin, the y-axis denotes the amount of probability values that fall in the corresponding bin.

$$range(B_i) = \begin{cases} [\frac{i}{N}, \frac{i+1}{N}], & \text{if } i = N-1 \\ [\frac{i}{N}, \frac{i+1}{N}), & \text{otherwise} \end{cases}$$

So if $N = 4$ then $range(B_0) = [0, 0.25)$ and $range(B_3) = [0.75, 1]$. Then, the amount of values in each bin is counted. Then, the binned counts can be plotted to visualize the distribution of the label probabilities produced by the classifiers. Figure 5.1 shows a plot of these binned counts for the listed classifiers.

When studying Figure 5.1 several strange anomalies in the distribution can be observed. The most obvious ones are the very large peaks of count for mood_acoustic at around 0.1, mood_relaxed at 0.8, mood_electronic at around 0.95 and mood_sad around 0.35. However, different subsets of classifiers also show smaller peaks in the distribution at the same probability values. These subsets are *not* always the same, for instance the classifiers mood_sad, mood_party and mood_relaxed have a relatively low count of values around 0.32, but the subsequent peak at 0.35 includes mood_sad, but *not* mood_party or mood_relaxed. These peaks in the label probability distribution might explain some of the relatively low correlations observed in Section 4 and the low stabilities reported in Section 3. Thus, it can be

Name	Classifier	Range	Full		Genre	
			#MBIDs	#submissions	#MBIDs	#submissions
acoust_spike	(mood_acoustic, `acoustic`)	$[0.09, 0.10]$	282,605	358,747	60,261	94,268
relaxed_spike	(mood_relaxed, `relaxed`)	$[0.805, 0.815]$	373,555	485,184	72,739	119,050
electronic_spike	(mood_electronic, `electronic`)	$[0.972, 0.982]$	315,626	401,151	64,944	101,915
sad_spike	(mood_sad, `sad`)	$[0.346, 0.362]$	57,697	75,688	8,854	14,242

Table 5.1: Data slices used for distributional comparisons. We indicate the classifier confidence range for which a submission was considered to be anomalous, and list the counts of unique MBID recordings and overall submissions, both for the full corpus and our genre-filtered corpus.

valuable to try and understand what might cause these peaks, since this might give us extra insight into why some of these classifiers seem to underperform on in-the-wild data. The following theories might explain these anomalies:

1. Due to some error in an older version of the software used to analyze the audio, many of the classifiers erroneously reported the same label probabilities for a large amount of submissions. This was fixed in a later version and the other probability values in the distribution were populated later.

2. The underlying dataset is highly skewed towards a certain music genre or acoustic quality of the music, which makes some of the peaks like the large peak in low acousticness probability a correct assessment. Even if this is true, the peak should probably not be as 'sharp', since if an assessment of 0.1 is very common for the data we would expect an assesment of 0.11 to be relatively equally common.

3. The classifiers are highly susceptible to codec or bitrate differences which cause the large peaks. For example, if a classifier is unable to accurately assess submissions with codec X and bitrate Y and instead returns the same value every time for these submissions, it is feasible that when this is a commonly used codec this would result in the large peaks. The rest of the distribution would be filled out by submissions using a different codec or bitrate.

For all analyses in this chapter we focus on the four largest observable peaks, subdividing the full AcousticBrainz data as described in Table 5.1. Note that these ranges correspond to the largest peaks as observed in Figure 5.1.

As a first approach to testing the theories described above, we take the following approach: say we are interested in investigating if the codec might be an important factor in causing the peak for a certain classifier in the distribution. If the full dataset is denoted as X, and the peak for the classifier we are interested in has values ranging from 0.5 to 0.55 then we can take subset X_s for which all submissions had a label probability for this classifier that falls within this range. Then the ratio of submissions that use a specific codec within this subset is calculated. The same ratio is calculated on the entire dataset. A large difference in ratio of a codec within the peak compared to within the entire dataset might suggest and effect of the codec for causing the large peak.

Table 5.2 shows percentual differences in the amount of lossless music present in the peaks. It shows that submissions that correspond to one of the peaks often included *less* lossless music compared to the entire distribution. Table 5.3 shows differences in codec percentages suggesting that submissions that fall in one of the peaks were often encoded with the mp3 codec, and less often with lossless codecs like flac or alac and Table 5.4 shows differences in version percentages when compared to the entire distribution showing that many of the submissions that correspond to one of these anomalous peaks were ran on features produced by an older version of Essentia. These results effectively take the first steps towards testing theories 1 and 3 by showing that there is a relatively large amount of submissions that fall within these anomalous peaks that were calculated using a specific version or codec. Interesting to note is that the `sad_spike` does not show such differences of codecs or versions used, having ratios that are comparable to the full distribution. Thus, some classifiers show this behaviour due to other underlying causes, indicating that different classifiers are sensitive to different aspects in the machine learning pipeline.

Since these exploratory results are relatively sensitive to sample size differences, Section 5.1.1 will present a more formal analysis. However, the results presented here hint at the possibility that version

Name	Percentage lossless
full	0.325822
acoust_spike	0.105501
relaxed_spike	0.162542
electronic_spike	0.117786
sad_spike	0.364800

Table 5.2: Percentage of lossless submissions in each of the described peaks. The dataslice with the name 'full' simply includes all submissions.

	mp3	flac	vorbis	aac	alac
full	0.598476	0.320427	0.039505	0.033874	0.005316
acoust_spike	0.844213	0.083374	0.028633	0.017084	0.022121
relaxed_spike	0.783913	0.144755	0.031341	0.018086	0.017748
electronic_spike	0.832754	0.097053	0.029009	0.016009	0.020728
sad_spike	0.570315	0.361550	0.032647	0.027785	0.003250

Table 5.3: Percentage of submissions using a specific codec for each of the described peaks. The dataslice with the name 'full' simply includes all submissions.

and codec might both have an effect that causes the anomalies observed in the distribution, giving some support for theories 1 and 3. The formal analysis will also have to check theory 2 to see if these peaks correspond to an overrepresentation of a certain genre.

5.1.1. Distribution comparison

For a more formal analysis of the three theories, the following approach is taken:

Say we want to know if a certain codec is represented more in the acoust_spike than it is in the 'non-anomalous' data. To get a good baseline of this non-anomalous data with which we can compare the anomalous data that is defined by any of the four named peaks, we slice the full dataset as follows:

1. Identify all submissions that were anomalous in *any* of the four described peaks. In set notation: $anomalous = \{acoust\_spike \cup relaxed\_spike \cup electronic\_spike \cup sad\_spike\}$
2. Then we select all submissions in the full dataset that were not a member of either of these four anomalous sets. In set notation, the baseline would be: $baseline = \{full - anomalous\}$

By constructing this baseline, we ensure that we have a consistent dataset to check the distributional differences in metadata fields with. This baseline set consists of 1,239,882 submissions for 855,266 unique MBID recordings.

Then, since we are interested in checking if there is a distributional difference in the metadata fields for anomalous and non-anomalous data, we count the appearances of the metadata values for each of the defined peaks. Say there are three codecs, x, y, z, and A contains 3 submissions using codec x, 2 submissions using codec y and 1 submission using codec z. We take these counts and make vector $counts(A) = [3, 2, 1]$. The same is done for subset B, resulting in the vector $counts(B)$. Finally, these count vectors are normalized so that they sum to one, so $counts(A) = [\frac{3}{6}, \frac{2}{6}, \frac{1}{6}]$, effectively giving us the sampled probability of observing the metadata. This transformation of submissions into

	2.1-beta2	2.1-beta1
full	0.632393	0.367607
acoust_spike	0.112207	0.887793
relaxed_spike	0.197968	0.802032
electronic_spike	0.125277	0.874723
sad_spike	0.637921	0.362079

Table 5.4: Percentage of submissions using a specific low-level Essentia version for each of the described peaks. The dataslice with the name 'full' simply includes all submissions.

	acoust_spike	relaxed_spike	electronic_spike	sad_spike
bit_rate	0.418759	0.322867	0.39447	0.166321
codec	0.339619	0.26285	0.322216	0.0569593
downmix	0.000893558	0.000526145	0.000697558	0
length	0.145746	0.150273	0.148331	0.31763
lossless	0.281495	0.21319	0.265893	0.0166514
essentia_low	0.609062	0.521436	0.594767	0.145326
essentia_git_sha_low	0.672409	0.584972	0.660398	0.23105
essentia_build_sha_low	0.70419	0.615245	0.692346	0.239792

Table 5.5: JSDs between counts over metadata categories for anomalous vs. baseline submissions. For metadata categories that are not listed, found JSDs were always 0, meaning no distributional differences were observed.

vectors of probabilities based on counts of metadata essentially allows us to view the anomalous and non-anomalous data as probability distributions over the metadata. Thus, if we wish to know if the distribution in the peak is very different from the distribution in baseline, a metric comparing two different distributions can be used.

Distribution distance metric To quantify the distributional differences, the Jensen-Shannon-Distance (JSD) metric [46] is used, which is defined as:

$$JSD(p,q) = \sqrt{\frac{D(p\|m) + D(q\|m)}{2}}$$

where m is the pointwise mean of p and q and D is the Kullback-Leibler (KL) divergence [82]. As an advantage over the KL divergence, the JSD is symmetric, and always has a finite value within the $[0, 1]$ range [92].

Comparing metadata For each metadata category in the full acousticbrainz dataset (for which the sample sizes are listed in the 'Full' column in Table 5.1) the JSDs are calculated between the frequency occurrence of these category values, counted over all submissions within the anomalous spike, vs. all submissions in the baseline. As some categories can assume many different values (e.g. replay_gain), the JSD is only calculated over values that occur at least 10 times in both of the count vectors (the count vector for the anomalous peak and for the baseline) to prevent overinflation of the distance metric for such categories. The resulting JSD values are presented in Table 5.5. These results can be interpreted as follows: the higher the JSD value for a given peak and metadata field, the larger the distributional difference this metadata field had between anomalous and non-anomalous data. Thus, the higher this value, the more 'skewed' the metadata distribution was in the anomalous data, hinting at the possibility that this metadata might play a larger role in the formation of these anomalous peaks.

Inspecting the JSD values in Table 5.5, it appears that the largest difference between the anomalous and non-anomalous data could generally be found in the low level essentia version used, followed by the bitrate and codec used. This is in line with the results presented in Chapters 3 and 4 which also showed that the version, codec and bitrate can have an effect on the stability of and agreement between several high-level classifiers. The JSDs for the sad_spike differ slightly from the three largest spikes, with length showing the largest difference, but this is followed by the version, bitrate and codec fields, again showing that not all classifiers respond in the same way when being presented with inputs of different representations.

These results provide evidence for theories 1 and 3, with the strongest evidence for theory 1, indicating that the anomalous peaks might have been caused by a software error in an older version of Essentia. Slightly weaker evidence is found for theory 3, with relatively high JSD-values for the bit_rate, codec and lossless fields accross all studied classifiers.

Comparing genre While the results presented above provide evidence for the theory that the low level extractor or different codecs and bitrates have some effect on the formation of the strange peaks in the distribution, theory 2 still should be examined. It is interesting that the sad_spike showed length as

	acoust_spike	relaxed_spike	electronic_spike	sad_spike
discogs	0.121217	0.0943394	0.112098	0.108356
last.fm	0.141714	0.116578	0.134499	0.144426
tagtraum	0.138084	0.114325	0.130844	0.135816

Table 5.6: JSDs between frequency counts over genre categories, for anomalous vs. baseline submissions.

the highest metadata effect. This might suggest that — in addition to the version and codec differences — the underlying data might also cause some of the peaks. It might be the case that the dataset is overpopulated with songs of a certain genre, and if those songs generally have a different length, then this we might have indirectly observed this skew of genre in the underlying data through the length data. Thus, it is worthwhile to do a similar JSD analysis on the genre distribution differences between the peaks and a baseline.

The main issue with this is that ground-truth data for the genre is not available in the AcousticBrainz dataset. While there are genre *classifier* outputs available, these are far from reliable as demonstrated in Sections 3 and 4. The AcousticBrainz Genre Dataset [17] provides ground truth genre information aggregated from Discogs[1], Last.fm[2] and Tagtraum[3]. Of these sources, TagTraum aggregates multiple genre sources and reaches 90.4% agreement [130], giving something resembling ground truth genre information data for our submissions. Processing this genre data for analysis consisted of the following steps:

1. The data [17] was downloaded[4], and the following tab-separated files were used as a source for the ground truth: 'acousticbrainz-mediaeval2017-discogs-train.tsv', 'acousticbrainz-mediaeval2017-lastfm-train.tsv', 'acousticbrainz-mediaeval2017-tagtraum-train.tsv'.

2. Since these three datasets include genre labels for different sets of MBIDs, the MBIDs of these three genre sources were intersected, resulting in a list of MBIDs for which genre labels from all three sources were available.

3. This list of MBIDs was then matched against those of the submissions in the AcousticBrainz Dataset. This results in a subset of the full AcousticBrainz dataset for which we now have three ground-truth genre labels.

4. From this genre intersected subset, the anomalous data was again selected, resulting in data slices with sizes reported in Table 5.1 (under the 'Genre' column). The baseline slice is again created by taking all submissions that are not in either of the four anomalous peaks, resulting in a baseline of 267,394 submissions for 128,687 MBID recordings.

5. The main genres in every peak and in the baseline were counted, subgenres were discarded, resulting in possible values of 'pop, rock, classical' etc.

6. These genre distribution vectors are normalized so that every row adds up to one in the same way as in the metadata analysis.

7. Finally, the JSDs of genre distributions in the peak and the baseline are calculated, again only comparing values that occur at least 10 times in both count vectors. The resulting JSDs are shown in Table 5.6.

The results in Table 5.6 show that the anomalous data do not have a large skew in genre distribution, with a relatively low JSD-values (≈ 0.10) compared to many of those reported in Table 5.5. However, a JSD-value of ≈ 0.10 does indicate that there might be *some* merit to theory 2, hinting that the underlying data might also play a small role in the formation of these peaks and perhaps also the resulting stability and agreement of these classifiers — especially for the sad_spike, where a JSD-value of ≈ 0.10 is relatively high compared to the JSD-values over the other metadata fields.

[1] https://www.discogs.com/
[2] https://www.last.fm/
[3] http://www.tagtraum.com/msd_genre_datasets.html
[4] https://drive.google.com/drive/folders/0B8wz5KkuLnI3elJLMjh2cS1ha2s

	acoust_spike	electronic_spike	relaxed_spike	sad_spike	total
essentia_build_sha_low	5	5	5	3	18
codec	4	4	4	2	14
length	3	3	3	5	14
essentia_git_sha_low	1	2	1	2	6
bit_rate	2	2	2	0	6
replay_gain	1	1	1	2	5
essentia_low	0	0	1	0	1
lossless	0	0	0	1	1

Table 5.7: Maximum classification tree scores for each of the metadata fields when following the ctree approach. A higher total score means that this metadata field on average appeared in higher tree positions, indicating that this metadata field gives us more information on data being anomalous.

5.1.2. Decision tree approach

As an alternative approach to ranking the importance of certain metadata fields in causing the anomalous peaks in the distribution, a Decision Tree approach is taken. A Decision Tree is a machine learning classifier that continuously splits the data to reach a classification label, resulting in an explainable tree structure [101]. Running this second analysis can increase our certainty in the theories much like how we scored the classifiers based on agreement. If these two different analyses result in similar conclusions, then we can be more sure that those conclusions are correct. This is done because, like with many of the analyses presented in this thesis, it can be hard to make statistical claims due to the nature of the data which often does not allow for controlled experiments. The machine learning task is formulated as follows:

1. Make a subset of the AcousticBrainz dataset based on submissions that fall within a given peak, just as in Table 5.1. All submissions that do not fall in one of the peaks are part of the baseline set.

2. For each peak we are investigating, label all submissions that are part of that peak as 'anomaly' and add the baseline submissions with tag 'non-anomaly'.

3. Train a decision tree (classification tree) on this labeled data, using the metadata fields as input and the label as output. The task of the classification tree is then to try and predict if a given submission will be anomalous, given only the metadata corresponding to this submission. Since many metadata fields (like codec) are non-continuous, a decision tree algorithm that is able to handle non-continuous data is required. For this a Conditional Inference Tree [67] (ctree[5]) was used. An example of a resulting ctree can be seen in Figure 5.2. The maximum depth of the resulting ctree was set to 5 nodes to prevent the ctree from becoming too large.

4. Inspect the resulting ctree to see which metadata fields appear first in the tree. Since these metadata fields are used for the early splits in the data, these fields give more information about a submission belonging to the class "anomaly" than the metadata nodes lower in the tree. Since the resulting classification trees are too large to display elegantly here, the metadata in the nodes will be scored according to the highest level of the tree they are in. If the tree has N levels, then metadata in level i for $N \geq i \geq 1$ will get a score of $N + 1 - i$. For each metadata field, the highest score is kept. As an example, the ctree in Figure 5.2 would result in the following scores: $vari = 2, vasg = 1, tms = 1$ indicating that $vari$ contained the most information for classification, followed by $vasg$ and tms.

The resulting scores of the different metadata fields as obtained from the trained ctrees are presented in Table 5.7. The results largely agree with those in Table 5.5 with the strongest support for theory 1 since again the essentia version has the highest score, followed by the codec metadata field which provides support for theory 3. Notable differences between the tables are that the decision approach only has one of the three metadata fields corresponding to version with a high score, while Table 5.5 shows high scores for all version fields. This can be explained by how the classification trees are

[5] https://www.rdocumentation.org/packages/partykit/versions/1.2-6/topics/ctree

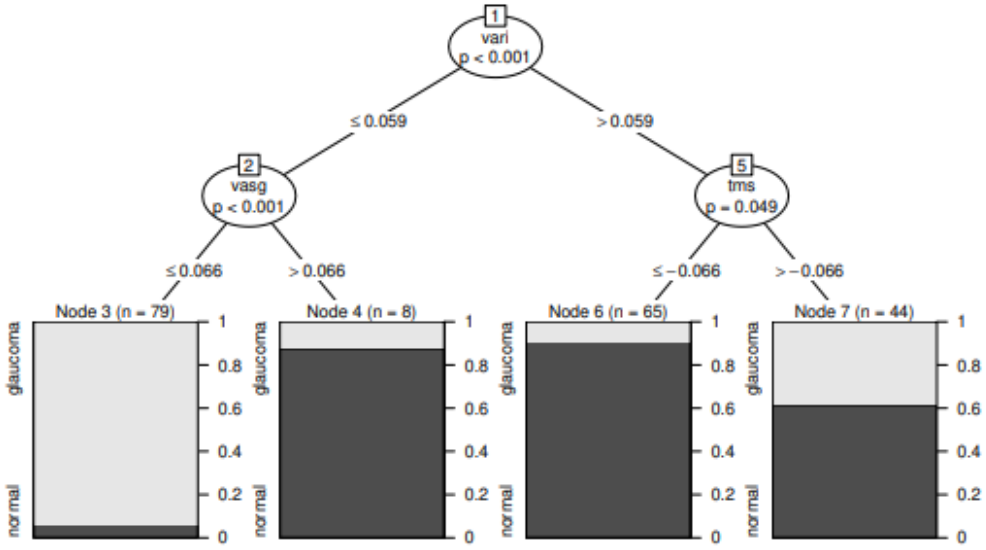


Figure 5.2: Example of a Conditional inference tree trained for classifying glaucoma. Nodes 1, 2 and 5 split the data based on the variable in the node. End nodes show the distribution of classified data [67]. The variable in node 1 gives the most information about the classification, since nodes higher up in the tree represent larger splits in the data.

built, if the top node already splits a large amount of data based on the `essentia_build_sha_low` field, then it becomes less necessary to also split on the more general version fields (since this split already happened implicitly higher up in the tree). The same can be said about the missing `lossless` field in this analysis, since splitting on if a submission is lossless already happens by splitting on codecs like FLAC and ALAC.

5.2. Spotify label distributions

As an extra way of checking if the underlying data is to blame for the anomalous peaks, a distribution graph similar to Figure 5.1 can be made for the Spotify data that was collected in Chapter 4. Studying this graph can give us the following insights:

- If similar large peaks are observed in the Spotify data as well, then this gives more evidence for the underlying data causing these peaks.
- If no peaks are observed in the Spotify data, then this gives more evidence of the AcousticBrainz classifiers behaving strangely and thus causing the peaks. This would further support the hypothesis that the representation and codec of the songs plays a large role in the high-level music classifier performance.

The distribution of all Spotify songs resulting from the mapping described in Section 4.1.2 for each of the high-level Spotify features is shown in Figure 5.3. The distribution of the Spotify data shows less of these anomalous peaks, with only the valence classifier having two relatively large peaks at low and high values. Note that there is less Spotify data available due to having to use the mapping which is dependent on the Million Song Dataset. The Spotify classifiers do not seem to exhibit the extreme peaking behavior observed with the AcousticBrainz classifiers, however they do show other anomalies like the smaller dips in the danceability classifier.

5.3. Conclusions

The peaks observed in this Chapter can be assumed to play a role in the low stability and agreement scores observed in the previous Chapters, given that these large peaks might highly skew the label probabilities to certain values. What exactly causes the high-level classifiers to display this behaviour is hard to say given the uncontrolled nature of the dataset under study. The results presented in this

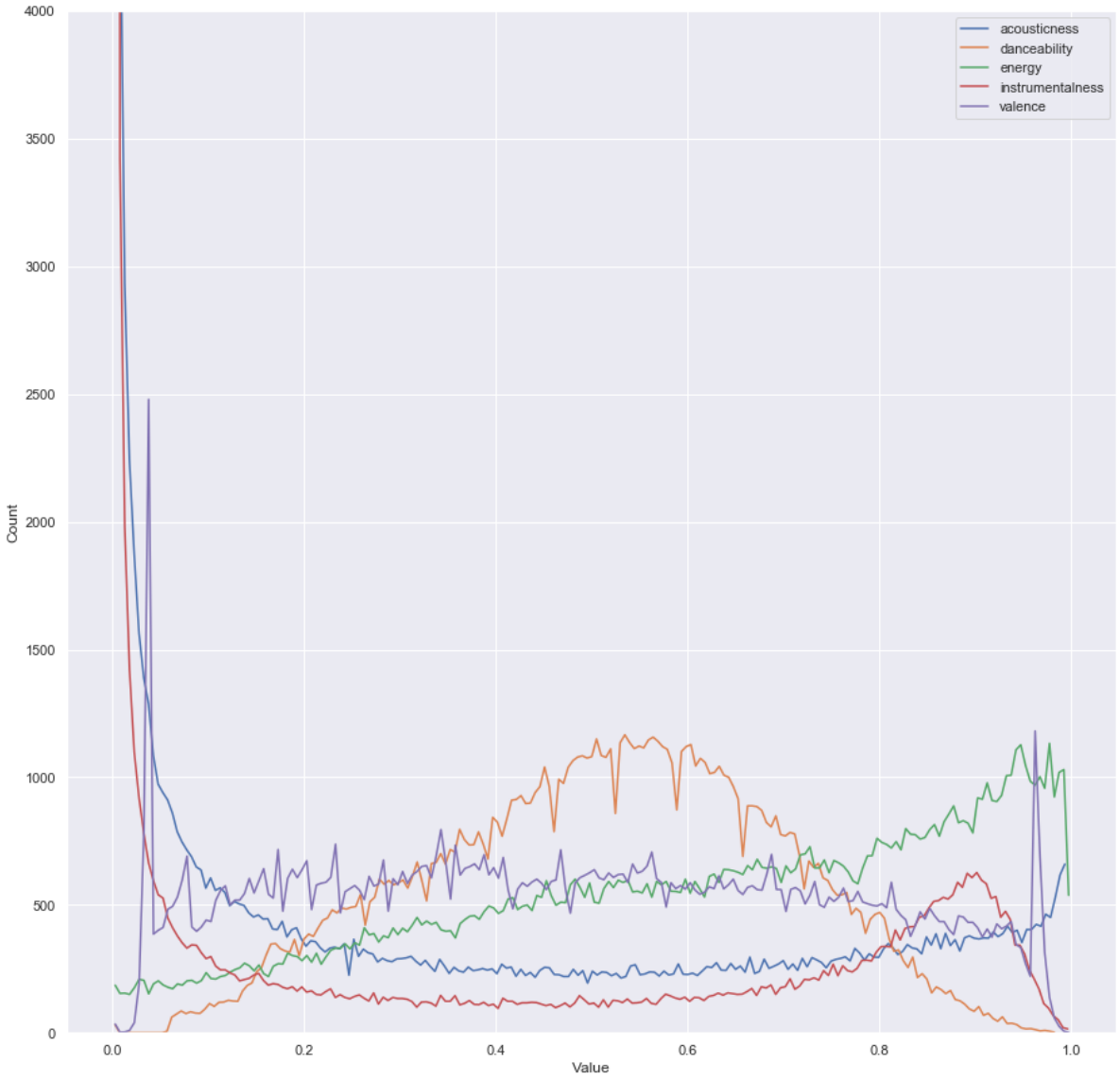


Figure 5.3: Distribution plot of label probability distributions in the Spotify data by means of binned counts using 200 bins.

Chapter do, however, indicate that it is likely that the low-level feature extractor plays a large role in the formation of these distributional peaks, with JSDs between the baseline and anomalous data being relatively high (≈ 0.60). The decision tree approach confirms these finding with the `essentia_build_sha_low` field appearing the highest on average in the constructed trees.

From these results we can also see that bitrate and codec seem to play a role, with JSDs of ≈ 0.30 indicating that certain audio quality differences might also lead to the observed anomalous behaviour. This further adds to what was observed in Chapters 3 and 4, where the audio encoding also seemed to be an important factor for stability and agreement metrics. It might be the case that these low scores are partially caused by these anomalous distributions. However, due to the uncontrolled nature of this dataset this cannot be proven statistically. Future research into the effects of audio quality parameters on both the label probability distributions and the label stability and agreement, controlling for all different variables at play could be very valuable, and the first steps towards this will be discussed in Chapter 7.

The question as to why these peaks are present at all is also worth investigating. It might be the case that the classifiers simply learned to always give submissions the value corresponding to their peak when they simply do not have a better guess. In such cases, it might be better to simply guess an average value to artificially score higher when standard metrics like classification accuracy are used. If this behaviour was baked into the classifier in lab conditions, then this would explain the rather large peaks for these values when they are released 'in-the-wild', since it is likely that many of these classifiers will encounter unseen submissions which they do not know how to classify — especially given the relatively small datasets used at training time and the low stability and agreement scores observed in the previous chapters.

In conclusion, while it is possible to speculate as to why these classifiers behave like this on unseen data, given the current data and the nontransparent nature of many of the current classifiers, it is very difficult to understand exactly why this is happening. However, we can be almost certain that this unexpected behaviour affects the quality of the results obtained from such classifiers negatively and deserves future research. Furthermore, an analysis as presented here could easily be applied to classifiers in other domains, and would provide yet another way of probing classifier performance in a way that does not depend on any ground-truth labels.

6

Controlling the data

So far we have seen that metrics based on reliability and agreement can be used to gain more insight into how well classifiers are performing on in-the-wild data, and we observed lower than desirable stability of high-level music classifiers in Chapter 3, relatively low agreement in Chapter 4 and strange distributional anomalies in Chapter 5. While Chapter 5 presented some evidence that genre-skew is probably not the main cause for the anomalous peaks observed in the AcousticBrainz data, there are many other properties of the recordings used to calculate these label probabilities. The full AcousticBrainz dataset includes recordings from many different artists, resulting in a wide variety of vocal styles, instruments and audio mastering techniques used. It might be valuable to see how the stability and agreement metrics change when the underlying data is made to be more homogeneous.

6.1. Controlling the underlying data

While it would, in theory, be possible to filter the dataset based on properties like the instrumentation used, this information is not available in the AcousticBrainz dataset. A more implicit way of restricting the variability in the properties like audio mastering and instruments is to select only the submissions in the AcousticBrainz dataset that belong to one specific artist. While it might be the case that this artist has experimented with different recording techniques or instruments over the course of his career, this variance will always be lower then when *all* data is used. Thus, by selecting only submissions from the AcousticBrainz dataset belonging to one specific artist, we reduce the variability in all properties like mastering techniques when compared to the entire dataset, without having this information explicitly available to us.

To get more accurate estimates for our performance metrics, it is desirable to select an artist for which there are many submissions available in the AcousticBrainz dataset. Looking at the top ten recordings in the dataset with the most submissions (see Table 6.1, column #submissions), The Beatles are the artist with the most submissions per recording. One interesting observation is that this top ten does not overlap with the most streamed songs by The Beatles on for example Spotify[1].

Filtering the data The data was filtered using the 'artist' metadata field present in the AcousticBrainz dataset. For every submission, it was checked if the artist field was equal to 'The Beatles'. If this was the case, then the submission was kept, otherwise, it was discarded. The resulting dataset contains 22,325 submissions across 3,582 recordings for an average of around 6.2 submissions per recording. Note that this is double that of the entire dataset, which had around 3.1 submissions per recording on average (see Section 3.2). This means the following:

- The data is more homogeneous than when using the entire AcousticBrainz dataset.
- On a per recording level we have a better estimate of the classifier variance, since on average the sample size per recording has doubled.

[1] At the time of writing, the most streamed song by The Beatles on Spotify is "Here Comes The Sun" with 447,635,009 reported streams.

Artist	Release	Title	#submissions	#submissions grouped
The Beatles	Rock 'n' Roll Music	I'm Down	126	118
The Beatles	Something New	Slow Down	108	97
The Beatles	Can't Buy Me Love	Can't Buy Me Love	96	-
The Beatles	A Hard Day's Night	A Hard Day's Night	95	-
The Beatles	Abbey Road	Octopus's Garden	94	52
The Beatles	Abbey Road	Something	93	-
The Beatles	A Hard Day's Night	And I Love Her	92	58
The Beatles	The Beatles' Greatest	Ticket to Ride	91	63
The Beatles	Abbey Road	Come Together	89	-
The Beatles	Beatles for Sale	Eight Days a Week	88	53

Table 6.1: Top ten recordings with the most submissions in the AcousticBrainz dataset.

The following sections will explore how the stability metrics as defined in Chapter 3 change when the underlying data used to calculate these metrics changes. The change in agreement could not be calculated with the The Beatles data, since the Spotify mapping constructed in Section 4.1.2 only includes the song "Ain't She Sweet" with nine submissions, which means that the agreement would be calculated over just one recording.

Effect on stability For measuring the effect on the stability of the classifiers, the same Pooled Variance metric as in Section 3.3.2 is used. It is hard to hypothesise what these results *should* look like. It could be that because there are now more submissions available per recording, and thus the variance is more accurate, that the variance might be higher or lower for all classifiers. Classifier performance might also be higher or lower across all classifiers since we are now limited to one genre. This would then depend on the data the classifiers are trained on: if this is more similar to The Beatles songs then the performance will probably be better and vice versa. Unfortunately, due to the classifiers being developed in-house, the training data is not available but the stability analysis presented here might give some insight into this underlying training data.

The Pooled Variance results on the entire AcousticBrainz dataset compared to the results on just the subset of The Beatles songs is presented in Figure 6.1. Interesting to note is that for some classifiers the variance increased and for some the variance decreased. Since the classifiers are trained on different datasets it might be the case that `danceability`, `mood_aggressive` and `mood_party` are trained on data which has characteristics similar to songs by The Beatles since these classifiers had a lower overall pooled variance score for the The Beatles songs than for the full dataset. For all other classifiers, the subset resulted in a higher Pooled Variance.

Bitrate effects Since we now have a more homogeneous dataset (at least when considering genre) it is interesting to see if the observed effects of Chapter 3 are still observable using this new subset, since for the bitrate analysis presented in that chapter the underlying data also changed a lot, resulting in it being difficult to attribute the decrease in stability solely to the changing bitrate. Unfortunately, due to the relatively smaller amount of submissions left in the data after filtering, an extensive codec comparison as in Chapter 3 is not possible. However, we can check if the effect of the bitrate is the same on the pooled variance metric, with higher bitrates having generally lower pooled variance, especially when the submission is lossless.

To do this, the The Beatles subset (22,325 submissions) was grouped by the most common bitrates — those with at least 100 submissions each — on which the pooled variance was then calculated. The sample sizes for each bitrate group are reported in Table 6.2, Figure 6.2 shows the resulting pooled variance values. If we compare this Figure with the one on the complete dataset, Figure 3.5, we can see that while the general trend of higher bitrate submissions resulting in lower pooled variance remains intact — indicating that this observed effect was probably not just because of the change in the underlying data — lossless seems to give less of an improvement on the The Beatles subset when compared to the same analysis on the entire AcousticBrainz dataset. Additionally the submissions with bitrate 256000 have a relatively low pooled variance, something which was also observed in the analysis on the entire dataset, although there the difference was smaller.

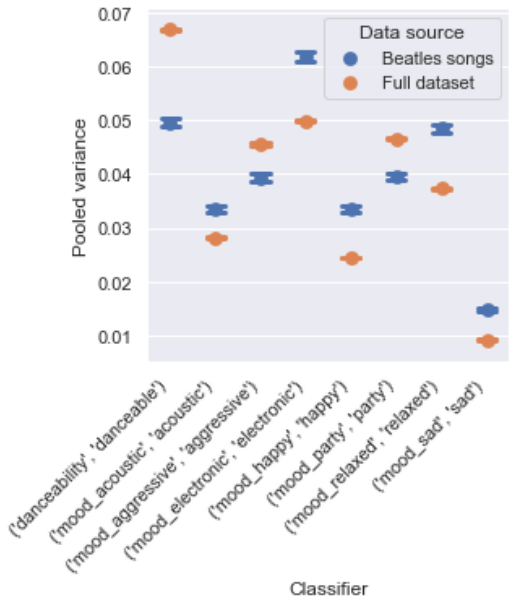


Figure 6.1: Pooled variance metric on the whole AcousticBrainz dataset compared to the The Beatles subset. Lower values are better.

Bitrate	Count
128000	479
192000	396
256000	161
320000	181
448000	4042

Table 6.2: Sample sizes for each of the bitrate groups within the The Beatles subset.

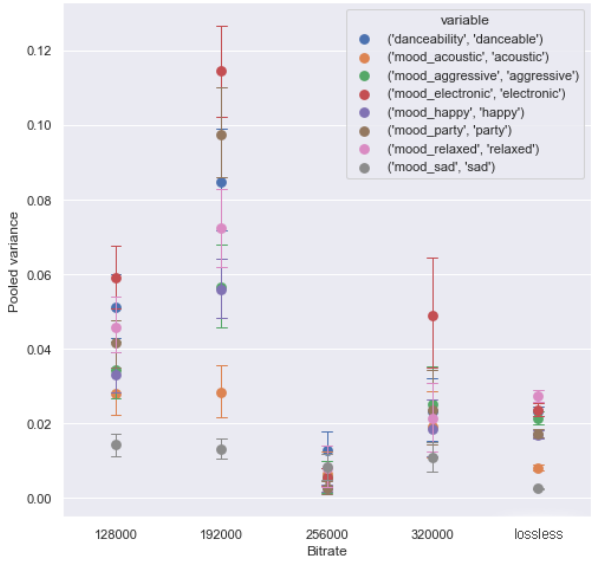


Figure 6.2: Pooled variance calculated over groups with the specified bitrate in the The Beatles subset. Lower values are better, bars indicate 95% confidence intervals for the estimates.

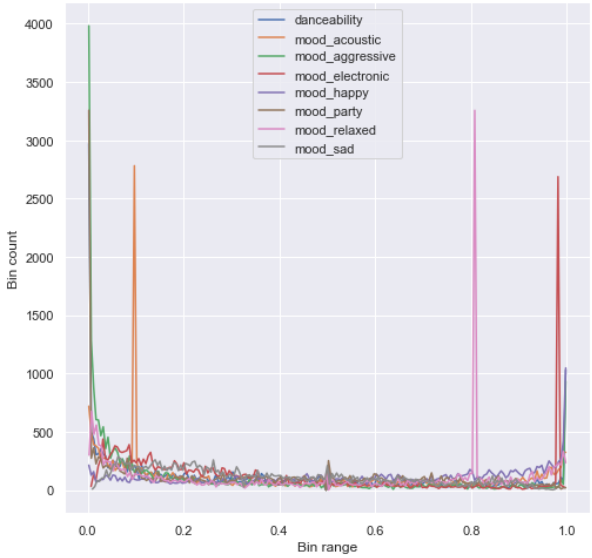


Figure 6.3: Distribution plot of label probability distributions by means of binned counts using 200 bins on the The Beatles subset. Note that the peaks in mood_acoustic, mood_relaxed and mood_electronic are present here as well as in the entire dataset (Figure 5.1).

Classifier distributions By plotting the label probability distributions of the The Beatles subset and comparing this to the distribution plots in Chapter 5 we can get some additional information about what might cause these peaks. If the peaks are *not* visible on the The Beatles data, then the chance that the underlying data plays a large role in the formation of these peaks is larger since then the anomalies would not be observed on more homogeneous data.

The distribution plot can be seen in Figure 6.3, it is clear that the largest peaks present in Figure 5.1, namely those in `mood_acoustic`, `mood_relaxed` and `mood_electronic` are also present in this limited subset, further indicating that distributional differences, which are still present in the The Beatles subset, have the largest effect on the anomalous data. The other peak corresponding to `mood_sad` is *not* discernible in this distribution plot. This corresponds with the findings in Table 5.5 which showed that for the three largest peaks, the representation of the data had the largest effect and for the sad peak this had almost no effect with the most important effect being the length of the songs. This sad peak is not visible here, indicating that it might be caused by the underlying data since it is not visible when the underlying data is more homogeneous.

Recording-level stability The filtered subset constructed in this section can serve another useful role. So far in this thesis the variance has always been calculated in a pooled manner as described in Section 3.3.2 to maximize the utilized sample size. However, given that the ten recordings specified in this chapter also have a moderately high sample size ($n \approx 90$), we can also look at the label stability on a per-recording level. The main advantage of this approach when compared to the pooled approach is that this gives us more insight regarding the 'spread' of these label probability variances over the different songs: does the same classifier generally result in the same stability per song or are there major differences between certain songs? If there are major differences then an analysis of these individual songs might give more insight into which kinds of input are more difficult to classify.

The label probability variances for the 10 recordings with the most submissions (as specified in Table 6.1) are shown in Figure 6.4. Interesting to note is that the label probability variance seems to vary quite a lot on the song level, with 'And I Love Her' having relatively high label probability variance on most classifiers, especially with `mood_acoustic` and `mood_sad` and other songs like 'Come Together' having generally lower variances for all classifiers. However, due to the organic nature of the dataset, it is difficult to attribute this higher variance purely to the nature of the song itself, since it might be the case that the submissions for 'And I Love Her' might have been made with a relatively unstable version of Essentia or an unfavourable codec.

To minimize these effects given the data under study, the submissions in Table 6.1 were further

grouped by the metadata fields `codec`, `essentia_low`, `essentia_git_sha_low` and `essentia_build_sha_low`. Only those recordings for which at least 50 submissions remained were kept, resulting in a group where `codec` = aac, `essentia_low` = 2.1-beta2, `essentia_git_sha_low` = v2.1_beta2-1-ge3940c0 and `essentia_build_sha_low` = 2d9f1f26377add8aeb1075a9c2973f962c4f09fd. These sample sizes of this group, which is also controlled on these representational fields are presented in Table 6.1 under the column '#submissions grouped'. Unfortunately additionally grouping on bitrate was not possible, as this resulted in too few remaining submissions. The label probability variances for these filtered submissions are presented in Figure 6.5. When looking at this Figure, outliers can still be observed with "Octopus's Garden" being relatively unstable when compared to the other songs on `mood_happy, happy`. This shows us that the classifier performance is probably not homogeneous over different recordings, since even on relatively similar songs — all by The Beatles — the label variances still vary quite a lot even when we control for most of the representational differences.

6.2. Controlling the representation

It seems that, even when the underlying data is controlled to be more homogeneous, the effects of audio representation like bitrate on the presented metrics are still observable. If classifier instability really is mostly due to the differences in representation as previous chapters have shown, then we would expect classifiers to be more stable if we only consider inputs with the exact same representation. But will classifiers also score better on the agreement metric when the inputs are controlled in this way? In short we wish to answer the following question: can the performance metrics of the high-level classifiers be improved by making the data more homogeneous by controlling the metadata such as codec, bitrate and Essentia version used? And if so, how big is this improvement?

Effect on classifier stability How much can the stability metric be improved, given all that we know about the effects of bitrate, codec and low level extractor versions? Creating this 'best-case scenario' for these classifiers is difficult, considering the many effects of all of the metadata. Thus, to calculate this 'best-case scenario' the following approach is taken:

1. Group the matched AcousticBrainz dataset on *all* different combinations of the metadata fields `bit_rate, codec, essentia_low, essentia_git_sha_low` and `essentia_build_sha_low`.
2. For each group, filter out recordings that have only one submission, since these recordings give no information about the label stability.
3. Then, check the amount of submissions in each group that remain. If there are less than 1000 submissions in a group, this group is discarded to ensure that the metrics have a large sample size to work with. This results in a total of 22 groups.

Then, for each of these 22 groups the pooled variance metric is applied. Since presenting pooled variance values for all 8 classifiers under study times 22 groups would result in a table with $8 \times 22 = 176$ items, the descriptive statistics for the pooled variance metric over all these 22 groups are presented per classifier in Table 6.3. From these results, it seems that grouping on *any* combination of codec, bitrate and version greatly improves the stability, since the descriptive statistics show that all groups then have relatively low pooled variance scores, with a relatively low mean value of the pooled variance of the 22 groups and relatively low standard deviation values. Table 6.4 shows how big this improvement in stability is when taking the most optimal group for every classifier compared to calculating the stability metric on the entire dataset. It is clear that the classifiers benefit greatly from being presented with data without representational differences, with stability scores greatly increasing for all classifiers.

While it is clear that it is beneficial for the stability to group on *any* combination of bitrate, codec and version, it might be interesting to see if one of these combinations consistently results in the lowest overall stability. Table 6.5 shows the groups that resulted in the lowest variance that correspond to the best stability scores of Table 6.4. Some interesting observations to note:

- Of the eight best groups, only four groups were lossless FLAC, the other four were MP3 with a bitrate of 128000. If codec and bitrate were the only factor influencing the stability score, we would expect higher quality MP3 to do better since previous chapters indicated that higher bitrates often resulted in better scores.

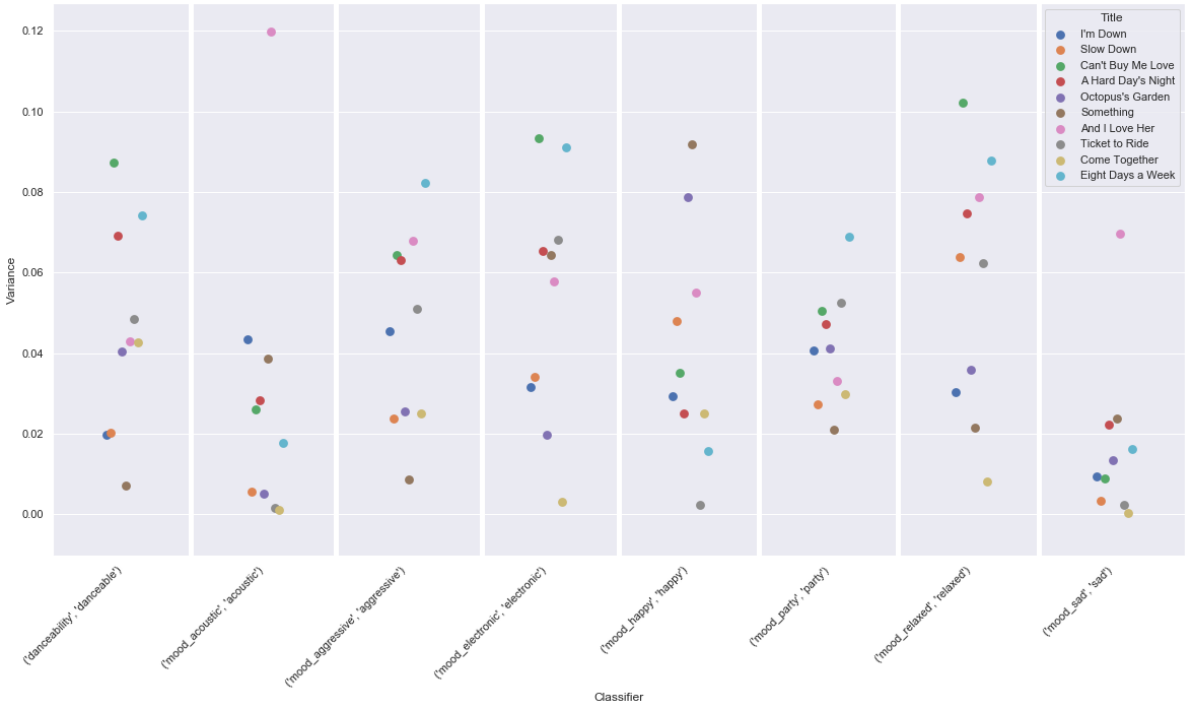


Figure 6.4: Label probability variances for the ten recordings with the most submissions.

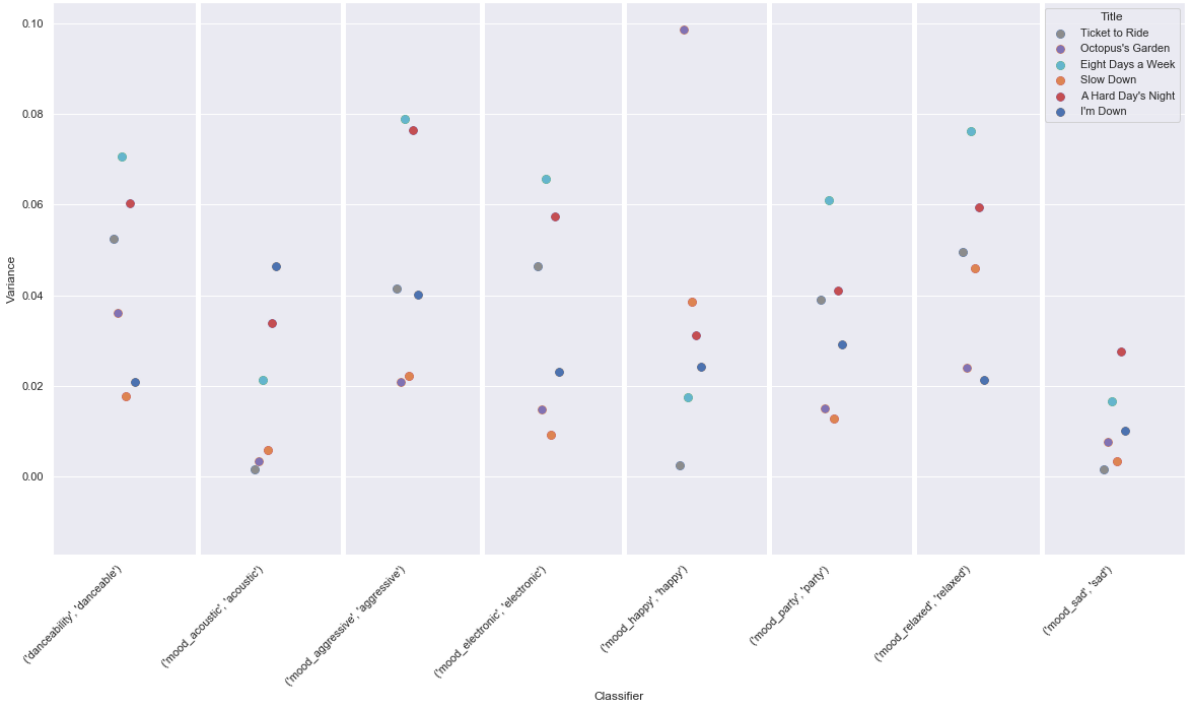


Figure 6.5: Label probability variances for the ten recordings with the most submissions, controlled for codec and essentia version.

	A	B	C	D	E	F	G	H
count	22	22	22	22	22	22	22	22
mean	0.007435	0.002758	0.005524	0.003488	0.005321	0.004239	0.004471	0.000859
std	0.006176	0.002186	0.006631	0.003380	0.003634	0.004734	0.004128	0.000438
min	0.000019	0.000002	0.000016	0.000191	0.000317	0.000124	0.000029	0.000031
25%	0.003979	0.001611	0.001969	0.000850	0.002117	0.000399	0.000714	0.000539
50%	0.006660	0.002521	0.003750	0.002856	0.004801	0.002684	0.003683	0.000952
75%	0.009612	0.003199	0.006411	0.003893	0.007142	0.006890	0.006970	0.001142
max	0.027992	0.010810	0.030432	0.014357	0.012597	0.017684	0.014066	0.001773

Table 6.3: Descriptive statistics of the stability (pooled variance) calculated over 22 different groups based on the metadata. Columns indicate classifiers:
A = `(danceability, danceable)`
B = `(mood_acoustic, acoustic)`
C = `(mood_aggressive, aggressive)`
D = `(mood_electronic, electronic)`
E = `(mood_happy, happy)`
F = `(mood_party, party)`
G = `(mood_relaxed, relaxed)`
H = `(mood_sad, sad)`

Classifier	Stability full	Stability best	Improvement
(danceability, danceable)	0.066811	0.000019	0.066792
(mood_acoustic, acoustic)	0.028181	0.000002	0.028179
(mood_aggressive, aggressive)	0.045529	0.000016	0.045513
(mood_electronic, electronic)	0.049802	0.000191	0.049611
(mood_happy, happy)	0.024462	0.000317	0.024145
(mood_party, party)	0.046484	0.000124	0.046360
(mood_relaxed, relaxed)	0.037327	0.000029	0.037298
(mood_sad, sad)	0.009161	0.000031	0.009130

Table 6.4: Stability score improvement using the best group for each classifier compared to using the full dataset with all metadata fields.

Classifier	Codec	Bitrate	Git SHA	Build SHA
(danceability, danceable)	flac	0	v2.1_beta1-6-g5578087	5a55..
(mood_acoustic, acoustic)	flac	0	v2.1_beta1-6-g5578087	5a55..
(mood_aggressive, aggressive)	flac	0	v2.1_beta1-6-g5578087	5a55..
(mood_electronic, electronic)	flac	0	v2.1_beta1-6-g5578087	5a55..
(mood_happy, happy)	mp3	128000	v2.1_beta1-28-g21ef5f4-dirty	ca57..
(mood_party, party)	mp3	128000	v2.1_beta2	70f2..
(mood_relaxed, relaxed)	mp3	128000	v2.1_beta2	70f2..
(mood_sad, sad)	mp3	128000	v2.1_beta1-28-g21ef5f4-dirty	ca57..

Table 6.5: Metadata groups that resulted in the lowest pooled variance for each classifier. Build SHAs have been shortened for presentation purposes.

	A	B	C	D	E	F
count	26	26	26	26	26	26
mean	0.597379	0.265644	0.468363	0.297532	0.235755	0.187254
std	0.207730	0.148028	0.180931	0.170126	0.136819	0.147677
min	0.082376	-0.041901	0.043146	-0.065932	-0.066955	-0.213750
25%	0.643201	0.187062	0.491885	0.168761	0.219966	0.130945
50%	0.693239	0.311724	0.544237	0.359096	0.279706	0.217505
75%	0.728838	0.350417	0.562564	0.427923	0.313485	0.262585
max	0.761782	0.552018	0.672652	0.522287	0.476783	0.481604

Table 6.6: Descriptive statistics of the agreement (correlation) between the AcousticBrainz and Spotify classifiers calculated over 26 different groups. Columns indicate the classifier pairs:
A = `acousticness, (mood_acoustic, acoustic)`
B = `danceability, (danceability, danceable)`
C = `energy, (mood_relaxed, not_relaxed)`
D = `instrumentalness, (voice_instrumental, instrumental)`
E = `valence, (mood_happy, happy)`
F = `valence, (mood_sad, not_sad)`

- Beta1 was the most stable in six of the eight groups, with git commits v2.1_beta1-6-g5578087 and v2.1_beta1-28-g21ef5f4-dirty. A Google search on the second SHA resulted in a Github issue page[2] on which a user noticed that this version resulted in slight differences on different architectures (32-bit vs. 64-bit).

- The valence related classifiers (mood_sad and mood_happy) both had the same group that resulted in the lowest pooled variance, which could be explained by the fact that both classifiers are trained on similar data [85]. The same holds true for mood_party and mood_relaxed.

Effect on classifier agreement Does performance on the agreement metric also improve when the inputs are controlled over these representational parameters? Which combinations of bitrate, codec and extractor versions result in the highest agreement between the AcousticBrainz dataset and the Spotify dataset? We take the following approach to find the best case scenario for agreement:

1. Match the AcousticBrainz and Spotify datasets using the mapping constructed in Section 4.1.2.

2. Group the matched AcousticBrainz dataset on *all* different combinations of the metadata fields `bit_rate, codec, essentia_low, essentia_git_sha_low` and `essentia_build_sha_low` that have at least 1000 submissions in total, resulting in 26 groups total. This size restriction is taken to keep the metrics as accurate as possible.

3. For every group, calculate the agreement metric (`corr_1`) as defined in Section 4.3.2

Since the full correlation table includes 26 groups defined by 5 metadata fields and for each of these groups 6 correlation metrics are reported (as defined in 4), it is too large to show in full. Thus, again we show the descriptive statistics to see the general distribution of the metric scores after controlling the data. The descriptive statistics are presented in Table 6.6. From these results, it is clear different groups result in significantly different agreement scores, with large differences in the minimum and maximum value for the agreement metric across groups and relatively high standard deviations. This would suggest that, unlike with the stability metric where consistently picking *any* specific representation resulted in better stability, specific representations seem to perform much better on the agreement metric than others. Table 6.7 shows how big this improvement in agreement is when taking the most optimal group for every classifier compared to calculating the agreement metric on the entire dataset. Note that agreement scores between `valence` and `(mood_sad, not_sad)` seemed to benefit the most from controlling the representation of the data, and classifier pairs that performed the worst on the full dataset improved most when the data was controlled in this way.

[2] `https://github.com/MTG/essentia/issues/179`

Spotify	AcousticBrainz	Agreement full	Agreement best	Improvement
acousticness	(mood_acoustic, acoustic)	0.65	0.76	0.11
danceability	(danceability, danceable)	0.26	0.55	0.29
energy	(mood_relaxed, not_relaxed)	0.45	0.67	0.22
instrumentalness	(voice_instrumental, instrumental)	0.28	0.52	0.24
valence	(mood_happy, happy)	0.24	0.48	0.24
valence	(mood_sad, not_sad)	0.15	0.48	0.33

Table 6.7: Agreement score improvement using the best group for each classifier pair compared to using the full dataset with all metadata fields.

Spotify	Acousticbrainz	Codec	Bitrate	Git SHA	Build SHA
acousticness	(mood_acoustic, acoustic)	vorbis	320000	v2.1_beta2-1-ge3940c0	cead..
danceability	(danceability, danceable)	flac	0	v2.1_beta1-6-g5578087	7f15..
energy	(mood_relaxed, not_relaxed)	flac	0	v2.1_beta1-6-g5578087	8593..
instrumentalness	(voice_instrumental, instrumental)	flac	0	v2.1_beta2-1-ge3940c0	2d9f..
valence	(mood_happy, happy)	flac	0	v2.1_beta1-6-g5578087	7f15..
valence	(mood_sad, not_sad)	flac	0	v2.1_beta1-6-g5578087	7f15..

Table 6.8: Metadata groups that resulted in the highest agreement for every AcousticBrainz-Spotify classifier pair. Build SHAs have been shortened for presentation purposes.

Again, we can take a look at which groups of metadata fields resulted in the highest agreement metrics to get some insight into which metadata correspond to the 'best-case scenario' for these classifiers with regards to the agreement metric. Table 6.8 shows these groups that resulted in the highest agreement between the AcousticBrainz and Spotify classifiers. Some interesting observations:

- While lossless FLAC was only present in half of the groups in the stability analysis presented in Table 6.5, here lossless FLAC is a clear winner, being the codec in five of the six groups that resulted in the best agreement scores.
- Again, beta1 seems to be common in these best case scenario groups, with git SHA v2.1_beta1-6-g557808 making up four of the six groups. This is the same version SHA as the one in Table 6.5 that resulted in the lowest variance for four of the eight groups. Further analysis into the source code of this specific version might give insight into why this version seems to perform better than the other, newer, versions.
- Git SHA v2.1_beta2-1-ge3940c was not present in any of the best case scenarios for the stability analysis, but is present in two out of the six agreement groups, even though this git SHA corresponds to the recommended win 32 bit extractor static binary[3] listed on the AcousticBrainz downloads page[4]

6.3. Conclusions

The results in this chapter further reinforce the findings of the previous chapters. Controlling the underlying data allowed us to rule out the possibility that the effects of audio representation like the codec or bitrate used on the defined metrics were solely due to the change in the underlying data which could not always be controlled in those analyses. Even when the underlying data is kept relatively homogeneous, the same effects on the defined metrics can be observed, showing that bitrate of the input data has an effect on classifier performance and that the distributional anomalies are probably caused by these same differences in representation. However, the results presented in the recording-level stability section show that, while representation certainly plays a large role on the performance of the classifiers, this performance is also dependent on aspects of the recordings themselves.

Furthermore, it is interesting that for some classifiers the controlled dataset of The Beatles songs lowered the variance, while for others this resulted in higher variance. If variance scores were lower

[3] ftp://ftp.acousticbrainz.org/pub/acousticbrainz/essentia-extractor-v2.1_beta2-1-ge3940c0-win-i686.zip

[4] https://acousticbrainz.org/download

across the board for all classifiers on the The Beatles subset this would indicate that the classifiers were probably trained on data similar to songs by The Beatles, since classifiers perform better on data similar to the training data, resulting in lower performance on different data [141] or real world applications [40]. However, this is not the case. Instead we can only assume that *some* aspects of the controlled dataset resulted in improved or decreased stability in the classifiers. While these aspects can be attributed to the data itself — for example, perhaps some instrumentation often used in songs by The Beatles confuses the `mood_electronic` classifier, resulting in higher pooled variance — due to the nature of the dataset this might also be attributed to the metadata of the submissions (since many of the classifiers respond differently to the metadata, as was also shown in Chapters 3 and 4.

The visibility of the same peaks in the distribution in Figure 6.3 as in the distribution of the entire dataset as shown in Figure 5.1 further shows that the underlying data is not the only factor: even when the underlying data used to generate the distribution is made more homogeneous, the three anomalous peaks are still observed, suggesting that these peaks are caused by other properties of the data like the Essentia version used for calculating the low-level representations of the data. The smaller peak corresponding to the `mood_sad` classifier is not visible in this distribution, suggesting that this peak might be caused by the underlying data instead of the representation of this data. This conclusion is in line with the results presented in Table 5.5, which showed that the length of the songs had the highest effect on this anomalous peak, while the other peaks had higher scores for metadata field corresponding to representation such as codec and Essentia version used. Thus, some of the classifiers might show anomalous behaviour due to the underlying data while most seem to be more susceptible to the representation of the audio. This might also be due to differences in the training process of the different classifiers.

The song-level stabilities showed that the stability is also greatly dependent on the input, even when controlling for many of the metadata fields like codec and Essentia version. While a more controlled setup is needed to make conclusive claims, it seems that the classifiers are sensitive to some aspects of the songs themselves, in addition to the metadata and encoding parameters. This could be explained by these songs being less similar to data shown to the classifier in the training phase, and further provides some evidence that these high-level classifiers might not generalize as well as we would hope.

Finally, by controlling the data representation it was shown that these different representations greatly influence the resulting metric scores, with stability and agreement showing improved scores when only presented with data encoded using a specific representation. Interestingly enough, it seemed that while most representations resulted in improved stability scores only some of the representations improved the agreement, with some other representations actually resulting in lower or even negative improvements. This provides even more evidence that trained classifiers might be lacking in construct validity, instead capturing properties related to audio quality or might only be able to grasp the underlying concept if the audio representation is the exact same as the one the classifier was presented with in the training phase.

7

Towards controlled experiments and evaluation

This thesis has identified and explored the issue of a general lack of stability in the label probabilities produced by many high-level music classifiers by utilizing metrics that do not depend on any ground-truth information. Results on the agreement between classifiers and a more in depth analysis of the label distributions raised more concerns about the stability and correctness of these classifiers. These results generally indicate that the labels and their probabilities obtained from these classifiers might be unreliable, and additionally showed that many factors play a role in this instability and probable lack of construct-validity: from the digital encoding used to encode the audio (in terms of codec and bitrate) to the low-level feature extractor version used (in terms of Essentia software versions) to the content of the data itself (with Chapter 6 showing that even when controlling for some of these metadata effects, label probabilities still vary wildly between inputs).

This exploration using the AcousticBrainz dataset has given valuable insights into factors that influence the stability of any (high-level) music classifier and highlighted the need for more in-depth analyses and validation methods to ensure that such classifiers can be trusted. While the AcousticBrainz dataset was perfect for such an analysis — it allowed for a thorough exploration within the timeframe of this thesis by utilizing submissions that were already available — the nature of the dataset did not allow for a thorough, statistical analysis of the effect of every individual effect. However, the insights presented in this thesis are very valuable for the field of MIR and the validation of machine learning pipelines in general, highlighting current problems with these classifiers and making the first steps towards more comprehensive evaluation methods. To make this scientific contribution more concrete, this chapter will present an additional literature study on top of the initial literature study presented in Chapter 2 and the analyses on the AcousticBrainz data, resulting in four valuable research directions that have emerged from the preceding analysis by incorporating ideas and methodologies from the software testing field, addressing **RQ3:** How can evaluation methods for classifiers be improved using techniques from other disciplines like psychology and software testing?

7.1. Controlled effect study

The research direction that follows directly from the work presented in this thesis is to study the effect of the different factors like audio encoding and bitrate on the high-level label probabilities in a more controlled manner. This would allow for a better understanding of how much each of the factors that were observed to have some effect (like bitrate and codec) individually contribute to a change in the stability or agreement of these high-level classifiers.

To facilitate such a controlled experiment a large dataset of audio files is needed. While the analysis in this thesis was able to skip the relatively time consuming process of gathering and processing data by using pre-computed low level features that were already available in the AcousticBrainz dataset, for a more controlled analysis a large amount of audio files is required, preferably in a losslessly encoded form like FLAC or WAV so that it can be encoded to lossy codecs like MP3 without introducing encoding artefacts twice. Due to legal reasons, gathering such a dataset can be troublesome [79] since direct

sharing of copyrighted works is illegal.

7.1.1. Gathering the data

There are three possible approaches to getting the data required for such an analysis. Their advantages and disadvantages will be briefly discussed.

Using Creative Commons-licensed music: This seems to be the easiest and legally 'safest' option. Since the Creative Commons-licenses allow for the copying and distribution of the licensed works for non-commercial purposes[1] the data can be made available publicly and is easy to gather and share. Due to this, datasets of Creative Commons-licensed music is readily available, the FMA dataset, for example, provides 106,574 tracks of MP3-encoded audio data, including metadata like artist and genre [37].

Unfortunately, to the best of my knowledge, such a dataset of lossless audio does not exist. We would like to encode the source audio using a multitude of different codecs and bitrates, and by having a lossy source file we would be encoding the file twice, losing the ability to fully control the input data. Additionally, while non-commercial music might be freely available, it is probably not mixed and mastered in the same way many large-scale commercial music is. Thus, we can not be sure that effects of codec and bitrate observed using non-commercial music translate to effects on commercial music while many of the applications of such high-level music classifiers might be commercial in nature, e.g. music recommendation on a streaming platform.

Using commercial music previews: If we wish to use commercial music to validate the trained music classifiers, the 30-second music previews made available by many streaming platforms like iTunes[2] or Spotify[3] could be used. Unfortunately, since we are now dealing with commercially-licensed music, this data can not be directly shared. Additionally, there are issues concerning the legality of storing this data locally for research purposes:

> "Developers may use certain promotional content (...) for the purposes of promoting the subject of the Promo Content; provided such Promo Content: (...) is streamed only, and not downloaded, saved, cached..."[4]

Moreover, it might be the case that using such previews for measuring the stability of high-level classifiers would give back incorrect results, since the previews often only include the chorus of a song while the classifiers are trained to work on the entire song.

Using commercial music: The best approach then seems to be using full-length, commercial, CD quality music. However, due to the copyright issues mentioned previously, this might be hard to obtain. While it might be possible that any research team wishing to go forwards with this research has an extensive collection of music, this collection also needs to be diverse: if the collection is skewed towards one or more specific genres, then the observed effect might only hold for those genres in particular.

Taking some inspiration from the AcousticBrainz project, it might then be more efficient to crowdsource these recordings. By setting up a similar system where users can use their personal music collection and send in the low-level features computed over these recordings — or better yet, utilizing the existing AcousticBrainz architecture — it becomes possible to collect a large enough dataset for such a controlled study. Such a data-collection project is quite extensive, however, and it might be worth it to first run a smaller scale experiment with locally available recordings while being aware of the limitations induced by the possible genre skew and low sample size.

7.1.2. Processing the data

To properly study the effects of certain (encoding) properties of the input on the output of the high-level music classifiers the following approach might be taken. Given that we have collected some baseline

[1] https://creativecommons.org/licenses/
[2] https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/
[3] https://developer.spotify.com/documentation/web-api/reference/tracks/get-track/
[4] See footnote 2

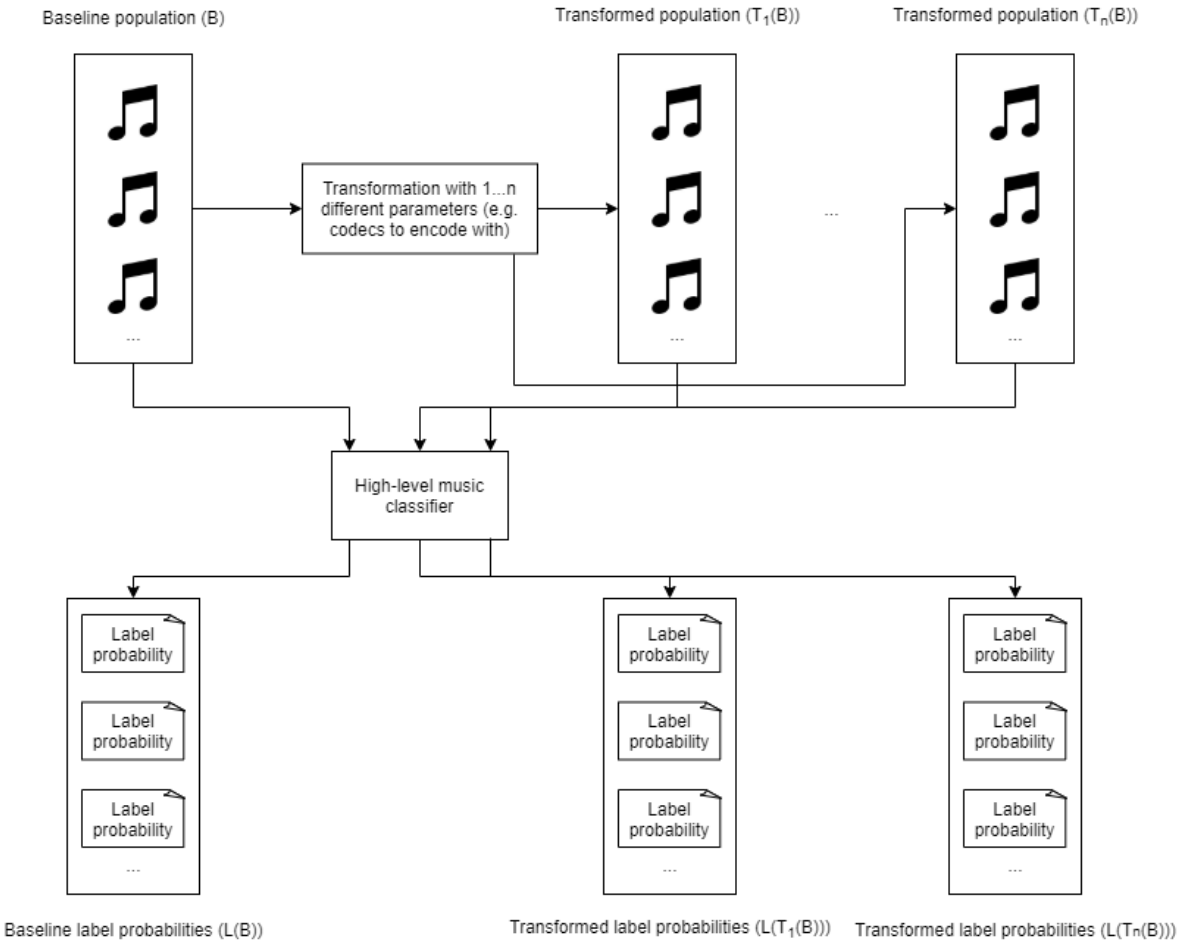


Figure 7.1: Visual representation of how the baseline, CD-quality audio can be processed to facilitate quantifying the effect of each transformation.

population B of CD-quality recordings, preferably all encoded using the same lossless codec for which it holds that:

- The sample size is large enough to draw statistical conclusions
- The samples are (close to) randomly sampled from all music, meaning that all genres are represented somewhat evenly

both of which might be achieved by crowdsourcing similar to the AcousticBrainz project, we can apply a multitude of transformations to the input data (of which a list is presented in Table 7.1). By keeping all other factors constant — making sure that the baseline is all lossless audio encoded with the same codec and bitrate and ran through the same (latest) low-level version of Essentia — we can change one of the factors to see the direct effect it might have on the label outputs.

Visually, this process of applying different transformations on the input data resulting in different high-level label probabilities is shown in Figure 7.1. These transformed label probabilities can then be compared with the label probabilities of the baseline for each transformation and each classifier we wish to study. It should be noted that since all described transformations will result in different low-level feature results which are used as input for the high-level music classifier, the transformations need to be processed locally if the data are to be crowdsourced, given that only these low-level features are legally allowed to be shared over the internet.

[5] https://musicbrainz.org/release-group/a5550169-ec99-45b1-827b-2d79c0c0fbb8

Transformation	Examples	Reasoning
codec:bitrate	mp3:320000, mp3:128000, ...	The AcousticBrainz data showed changes in classifier stability both for different codecs and bitrates within these codecs. Defining bitrate as a sub-transformation of codec allows us to see if there are differences in output between bitrates in the same codec and between codecs.
length	[+1, 0] (add a second of silence to the beginning), [0, -1] (crop a second of audio from the ending)	The AcousticBrainz data indicated that length had an effect on the distributional anomalies observed in the label probability distribution. By creating different length versions of the same input we can check if the observed effect was due to the length of the input or was a secondary effect linked to for example genre.
essentia_git	v2.1_beta1-6-g5578087, ...	The AcousticBrainz data indicated that the software version of the low-level feature extractor used to process the input has a significant impact both on the stability and agreement of the subsequent high-level classifier, but also on the distributional anomalies. By calculating the low-level features using the different git commits which are available on the Essentia GitHub, we can study the effect of these software versions on the high-level classifiers in isolation.
channels	2 (stereo), 1 (mono), ...	Some older songs (like those on the early The Beatles albums[5]) were recorded and mastered in mono, while almost all modern releases are recorded in stereo. It might be the case that classifiers trained on stereo music perform poorly on mono recordings. By converting the baseline recordings to mono we can see study the effect of the amount of channels in isolation.
loudness	+1dB (louder), -1dB (quieter), ...	Due to the 'loudness war' some recordings are mixed to be louder under the assumption that this makes them sound better [39]. It is conceivable that a classifier trained on recordings that are 'brickwalled' to be as loud as possible might underperform on recordings with more dynamic range and vice-versa. By manipulating the loudness on the same recording we can see if the average loudness of the input has an effect on the output.

Table 7.1: List of possible transformations to apply to the baseline, CD quality audio input. Transformations above the middle horizontal line directly follow from the effects observed in this thesis while those underneath this line explore other possibly interesting directions.

7.1.3. Analysis

Given the baseline population B and the corresponding labels $L(B)$ of size n, and a transformation T_x, we can now quantify the effect of this transformation on the labels. Since we are interested in seeing which transformations on the input data have the largest effect on the output, we can quantify the difference by calculating the Mean Absolute Error (MAE) between the baseline and the transformed population as follows:

$$MAE(L(B), L(T_x(B))) = \frac{\Sigma_{i=1}^{n} |L(T_x B)_i - L(B)_i|}{n}$$

For any transformation T_x, a higher value of $MAE(L(B), L(T_x(B)))$ indicates that the average difference in label probabilities was higher, and since all other variables unrelated to the transformation were controlled to remain the same, we can attribute this change in label probabilities to the applied transformation. Furthermore, if the MAE of T_1 is twice that of T_2 then this can directly be interpreted as the effect of the first transformation being twice that of the second, unlike when using other difference measures like the Root Mean Squared Error which grows faster [150] due to the errors being squared before being averaged.

In addition, we can also consider the distribution of the absolute errors for the different transformations to compare the sample means to see if these changes are significant. To gain an even deeper understanding of the stability of these error distributions for the different transformations a random effects model [99] can be fitted, similar to the analysis of the effect of audio quality on low-level features by Urbano et al. [143].

7.2. Differential testing on the Essentia codebase

While the data explored in this thesis is very valuable for the field for MIR, techniques from Software Engineering / Testing could also prove useful for a more thorough analysis of classifier performance in the AcousticBrainz data. Given the results presented in this thesis, which show that different low-level (Essentia) extractor features result in differences in the stability and agreement of subsequent high-level classifiers utilizing this data, the following question emerges:

Is there an uncaught bug in the Essentia codebase that results in this unexpected behaviour like lower label stabilities in the high-level classifiers?

While the results from the research proposed in Section 7.1 will give more insight into the exact effect of the Essentia version on the robustness of high-level classifiers, given the results presented in this thesis it seems likely that the software version has an effect. To be able to mitigate this effect it then becomes important to find and fix the cause in the codebase itself. This section will propose how to use an approach inspired by inter-release differential testing on the AcousticBrainz data to locate the potential bug.

7.2.1. Software testing methodologies

A common way of testing a codebase as it continually grows is to write test cases for the functionality of the code. To determine whether a test case passes or fails, a mechanism called a test oracle [68] is used. For tests like unit tests which test single units of code in isolation and integration tests which test if the units of code are functional when put together, *specified* oracles can be used: these oracles are based on the requirement specifications of the software to check if the code is working *as specified*.

However, such specified oracles are only as good as the specifications they are based on. When they are manually developed alongside the software they are subject to the individual interpretation of a developer implementing the test case, they may contain errors or have a limited amount of tested inputs. While certain methods — like the automatic generation of test cases from the documentation [112] — aim to overcome such problems, specified oracles might simply not exist for many test cases like software crashes on unexpected inputs or if the software is a machine learning classifier, the correct label probability for a specific input is unknown.

Derived oracles When such specified oracles are not available so called derived oracles can be used to assess if tests pass or fail. These derived oracles assess if a test is correct by using derived information from artefacts of the system under test (e.g. results of an execution on a specific input) [11].

One such approach often taken during the continuous development of a software product is regression testing [7]: the outcomes of tests on the previous version are used as a derived oracle to assess the results of the same tests on the new version. Using the previous version as a derived oracle allows the pinpointing of the problem if a certain test fails that passed in the previous version, or when a performance test gives back worse results then this can be investigated and bugfixed to prevent this issue from remaining in the codebase in future versions.

(Inter-release) differential testing Regression testing is often employed during the development cycle of the software to make sure that all parts of the software remain functional while the codebase is still evolving. However, it might still be the case that certain logical or semantic errors remain present in the codebase if these are not explicitly tested using a specified oracle, and since the amount of paths and conditions to test explodes when software grows, especially in integration testing [106] it becomes impossible to test for all of these cases.

Differential testing complements regression testing by being able to find possible semantic and logic errors in the Software Under Test (SUT) [96]. The main idea is relatively simple: given that there are n different versions or implementations of the SUT and we present an input x to all n versions of the SUT. If the output for any of the versions of the SUT differs for x, then we have found a potential bug in the code, and the differences in the source code for the version that resulted in a different output for x can be analyzed to find it. While differential testing can be applied to different implementations of the same system (as in McKeeman [96] where bugs were found by comparing the outputs of several different C-compilers), different versions of the same systems can also be used. When utilizing different versions of the same software this is called inter-release differential testing.

The generation of the inputs to use for such a differential test generally requires deep knowledge of the SUT, since inputs should be valid but diverse enough as to hopefully cover the entire codebase. Additionally, the generated inputs should resemble real-world input. For many systems, this search space is massive, and generating large amounts of quality data for differential testing can be difficult.

7.2.2. Testing Essentia using the AcousticBrainz data

Now that it is clear how inter-release differential testing works and how it can be valuable for finding uncaught bugs, the parallel to the AcousticBrainz data becomes clear. From the results presented in this thesis it appears likely that there is some undesired behaviour present in the Essentia codebase which causes some of the instability, and this might be due to either an uncaught bug, logical or semantic error or a change in the implementation of one of the low-level feature extraction algorithms present in Essentia. The submissions in the AcousticBrainz data can be viewed as similar to test cases in inter-release differential testing, since for every submission the version label (`essentia_git_sha_low`, which specifies which commit of Essentia was used to calculate the low-level features) is present, allowing us to see if there were any differences between these version on a larger scale (in terms of stability and agreement). However, these inputs were not 'pure' differential test cases due to the following:

- In many cases, one specific input (recording) was only ran on one specific version. In pure inter-version differential testing, the same input would be ran on *all* (or a specific subset) versions of the SUT. Again, this is due to the organic nature of the dataset.
- In differential testing the SUT would be one system (in this case Essentia), however in this thesis the 'systems' were the combinations of Essentia and the high-level feature extractors. This also allowed for the testing of non software related factors like data respresentation and emotion modeling (which also resulted in interesting results), but for a proper differential testing setup of Essentia you would look at the direct output of the system which in this case would be the low-level features.

This thesis has shown, however, that it might be valuable to perform such a differential test in a more controlled manner on the Essentia codebase. The infrastructure for this is already largely in place: since the Essentia codebase is open source it is possible to run and compile any of the previous versions of the software by pulling those specific commits[6]. The results presented in this thesis also provide a good

[6] `https://github.com/MTG/essentia/commits/master`

starting point for selecting a subset of Essentia versions to use for the differential testing. For example from the results in Chapters 3 and 4 we have seen that `v2.1_beta2` seems to perform best on average, however in the best cases scenarios presented in Chapter 6 one specific git SHA of the older beta1 version, `v2.1_beta1-6-g5578087`, seemed to perform even better on the stability and agreement metrics when presented with flac input and `v2.1_beta1-28-g21ef5f4-dirty` seemed to perform best on lower bitrate mp3 input on the stability metric for some of the high-level classifiers. Performing differential testing by feeding these different versions CD-quality inputs as well as input encoded using various codecs (the same data that is needed for the future research proposed in Section 7.1) and seeing how the low-level feature outputs differ between these versions can give us valuable insights. For example, there might have been a bug that was only introduced sometime after these commits or the algorithm itself might have changed. Performing such tests can both help improve the Essentia software but perhaps more importantly also allow us to better understand how to represent low-level music data in a stable way, which is very valuable for further machine learning research in musical contexts due to the dependence on using these low-level features as input, owing to the legal issues concerning the sharing of commercial music datasets.

7.3. Leveraging metamorphic relations

Instead of presenting the same input to many different versions of the SUT, another testing approach might present many different versions of the input to the (latest version of the) SUT. In essence, many of the inputs from the AcousticBrainz dataset analyzed in this thesis followed this concept, with one recording containing multiple submissions with different audio representations. The analysis was based on the criticisms by Sturm [137] about many MIR systems being horse systems, picking up on irrelevant confounds in the data resulting in poor construct validity. Again, there is a parallel to a technique in the software testing field: *metamorphic testing* [28].

Metamorphic testing The idea of metamorphic testing is as follows, and again stems from the oracle problem where it might be difficult or impossible to construct a specified oracle [11]: if it is impossible to get an exact oracle truth, we might be able to derive an oracle by applying some transformation on a given input and then, using our knowledge of the data relationships between the transformed and non-transformed inputs, gain partial oracle truth for the original input. Chen et al. [28] demonstrate this with the following example: Say we wish to test if our implementation of an algorithm that returns the shortest path given a weighted graph G and source and destination nodes x and y. Testing if the shortest path $x, v_1, v_2 ..., v_k, y$ of length p is actually the shortest path is difficult, especially if G is particularly large and complex. However, we can use the output corresponding to input (G, x, y) as an oracle for a derived test case. Using what we know about any shortest path, namely that if the shortest path from x to y is of length p then the shortest path from y to x should also be of length p, we run the algorithm on the transformed input (G, y, x) and check if the length is equal. If the lengths are different, then our derived test case has exposed an error in the implementation. If the lengths are the same, then we can be *more certain than before* that the implementation is correct.

While metamorphic testing was originally developed for software testing, it can also be very valuable for the validation of machine learning applications. With any machine learning algorithm there is no oracle for the correctness of the output produced by the algorithm for unseen data apart from ground truth labels that are held out of the training process, however these need to be collected manually *and* are subject to interpretation issues of an expert labeler or crowd bias if the labels are crowdsourced. Xie et al. [151] present case studies on the kNN and Naïve Bayes Classifier showcasing the feasibility of metamorphic testing for the validation of machine learning algorithms.

Applicability to music data Using this framework, many of combinations of submissions on a recording-level already present in the AcousticBrainz dataset can effectively be seen as derived test cases with some domain specific metamorphic relations under the following assumption: It is impossible to assess if the output of the high-level music classifier, $f(x)$, corresponding to any input x is correct since there is no oracle for $f(x)$ due to the lack of the exact ground-truth. However, if x is a CD-quality recording, and $T(x)$ is the transformation on x defined by, for example, the encoding using a codec like mp3 with a high enough bitrate, then we know that by design $T(x)$ and x should sound (almost) the same to the human ear due to the use of perceptual coding algorithms. Then, the output of the classifier should

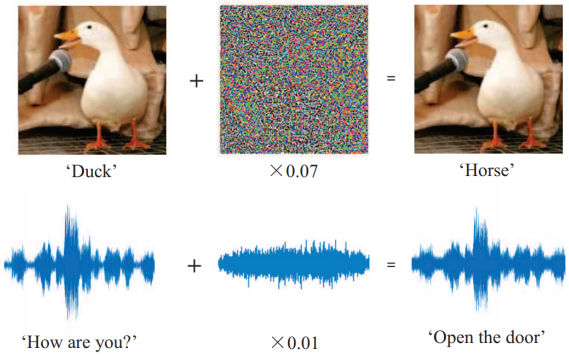


Figure 7.2: Imperceptibly small but targeted perturbations to the input provided to many machine learning models can lead the model to make wrong predictions. Such perturbations exist for for the visual as well as the auditory domain. [53]

also be same, since it should be based on human perception. Thus, for any input x we can use $f(x)$ as an oracle for the derived test case $f(T(x))$ by checking if the two outputs are the same.

Essentially, the transformations as described in Section 7.1 are a small subset of the full set of transformations that can be applied to any music input x, so long as the transformation is small enough that is is imperceptible to the human ear (otherwise, if the perception changes then so can the 'correct' label, and then the metamorphic relation no longer holds). Evaluating these high-level music classifiers using a large amount of such transformations can give us a better insight in their performance than when using traditional cross-validation approaches, since metamorphic testing might detect faults missed by cross-validation [151]. All that remains to be done to facilitate this is to:

- Make sure that such transformations actually exist. While the results presented in this thesis suggest that they do — different codecs which are perceptually similar resulted in significant differences in the classifier output — Section 7.3.1 will explore this further from the perspective of adverserial attacks employed in the field of deep learning.

- Find these transformations. As will be described in Section 7.3.2, adversarial examples provide one such transformation. Finally, Sections 7.3.3 and 7.3.4 will propose the use of a Genetic Algorithm (GA) to find such adversarial examples.

.

7.3.1. Adversarial examples

One comparable approach, often encountered in Deep Learning literature, is that of finding *adverserial examples*: inputs that intentionally break the classifier, causing it to misclassify or misinterpret the input. One category of these approaches is the white-box approach, which requires full knowledge of the model under study and often use gradient descent to find an optimal pertubation of the input [139]. While these white-box approaches have been shown to be successful, effectively being able to trick an image classifier into misclassifying a panda as a gibbon [139], due to the fact that they require knowledge of the model they are not applicable in all scenarios.

More directly applicable to our music data, and more in line with the idea behind metamorphic testing are black-box adverserial approaches. These approaches do not utilize the model under study, and instead only focus on the input-output pairs. Such attacks might utilize the output class probabilities [59] or try and construct a substitute model which serves as a 'copy' of the model under study on which gradient ascend can be applied [107] to effectively find an optimal transformation of the input data able to fool the classifier into misclassifying the given input.

While much of the research into finding adverserial examples has focused on computer vision [139] [83] [105], adverserial examples do not seem to be limited to the image domain, with adverserial attacks on voice-controlled systems [53] [54] (see also Figure 7.2) or even copyright detection systems [128] being possible. And while many of these approaches employ white-box techniques, it seems highly likely that such adversarial examples will also exist for the high-level music classifiers under study.

7.3.2. Finding metamorphic test cases for music classifiers

To quickly summarize, we wish to evaluate the performance of music classifiers beyond simple cross-validation approaches. The metamorphic testing framework can be valuable for this: since we cannot directly and accurately evaluate if the model output $f(x)$ is correct for x we wish to generate a derived testcase on a transformation on the input, $T(x)$ for which $f(x)$ serves as an oracle. As described in Section 7.3.1, finding an adversarial example based on x provides such a test case where we check if $f(x) \approx f(T(x))$, trying to find a T such that $f(x) \neq T(f(x))$, i.e. try to find an adversarial example that slightly perturbs the input but maximizes the error (deviation from $f(x)$).

The amount of such transformations that can be applied to any audio input is extremely large, effectively turning the problem into a search-space problem: we wish to navigate the search space of all possible transformations on a given audio input in an efficient way since it is impossible to test all possible transformations. This section proposes to use a Genetic Algorithm (GA) to effectively find these adversarial examples, since GAs have been shown to be effective in finding adverserial examples in the image domain [26] and work on input-output pairs, using a fitness function to navigate the search space. Such an approach can provide metamorphic test cases for any music classifier due to the black-box nature of the approach, effectively providing a way of 'stress-testing' a trained classifier beyond the traditional cross-evaluation approaches that are often used.

7.3.3. Genetic representation of the transformation

To effectively use a genetic algorithm, the transformation needs to be represented as a set of 'genes' on which the evolutionary operations can be run. This section will describe two possible ways of modeling these transformations.

As noise The results in Chapters 3 and 4 indicate that the codec used to encode the audio can have a significant impact on the output of a high-level music classifiers. However, as could also be seen in Figure 2.9, the majority of the audio signal is roughly the same between different codecs of the same bitrate. Thus, only a slight transformation on the input data seems to be enough for the classifier to start making errors.

These small transformations can then effectively be seen as 'noise' which is added to the original, clean CD quality signal (ignoring the cut-off of higher frequencies often employed by codecs). Given PCM encoded audio, which is the standard for music CDs, this noise can simply be added to the quantized samples resulting in a new, noisy audio file. This addition operation is shown visually in Figure 7.3. The resulting samples should be clamped to the maximum or minimum possible value for the bit-depth of the file after adding the noise, so that the audio file remains valid.

While it might seem unintuitive that adding values directly to the PCM data would result in an audio file which is perceptually indistinguishable from the original input, anecdotally it seems that PCM audio is very resillient to such perturbations. This was tested by adding randomly generated integers to the samples of a WAV file of the song 'Octopus's Garden' by The Beatles. Through trial and error it seemed that adding random integer values from $[-100, 100]$ resulted in clean sounding audio with noise being barely audible while the actual audio signal is greatly different (see Figure 7.4). This indicates that a GA has plenty of 'room' to manipulate the audio without actually perceptually changing the audio.

A GA could then find the specific set of 'genes' or in this case integers that result in the classifier making a large error. Note that because the 'noise' is directly added to the samples of the audio, many different transformations can be modeled given the complete freedom the GA has to manipulate the audio through this transformation method. The example presented here simply added random integers in a range (effectively resulting in the addition of white noise).

Theoretically, by giving the GA direct control over the sample of the audio, it can explore the full transformation search space, even transformations unrelated to codec artefacts. However, it might be the case that this gives the GA *too* much freedom, resulting in it taking a very long time to converge to a solution. Additionally, it might be the case that the genes are simply too large if every sample is individually perturbed since most music consists of millions of samples (standard CD-quality audio often consists of 44.100 samples per second [72]). One possible fix might be to limit the perturbation to a smaller size and repeat it as often as needed to cover the entire input, this approach would also have the additional benefit of being independent of the length of the input. Another benefit of this approach is that it allows us to check if the resulting perturbation generalizes to other audio.

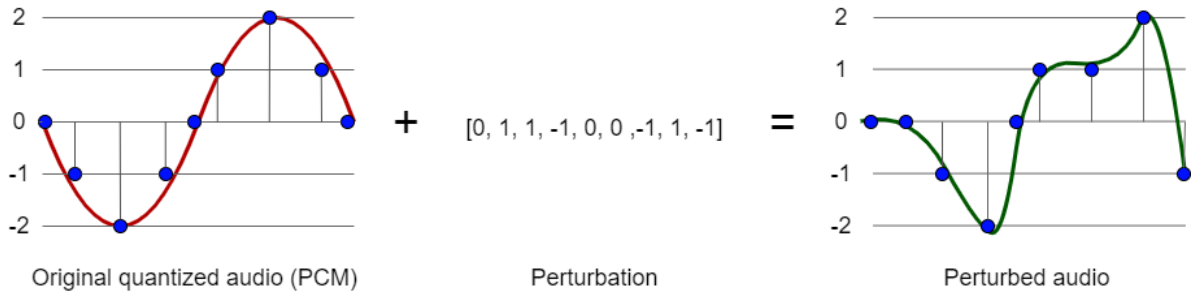
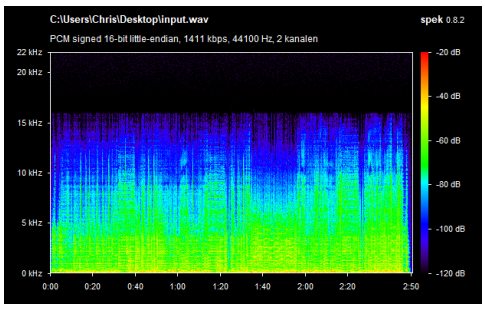
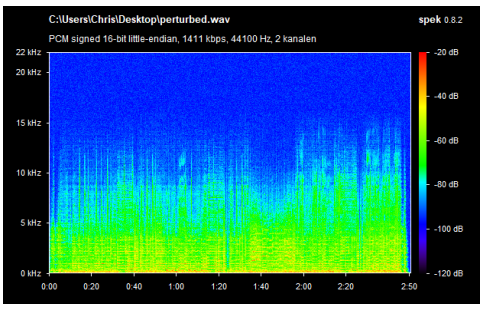


Figure 7.3: Using a perturbation to add noise to PCM encoded audio. The elements in the perturbation are added to the quantized samples of the audio, resulting in a noisy representation of the original input. This perturbation is to be optimized by the described GA.



(a) Original PCM encoded audio.



(b) Perturbed audio obtained by adding random integers in $[-100, 100]$ directly to the samples.

Figure 7.4: Perturbing PCM encoded audio by directly adding integers to the samples. Note that the audio depicted by the two spectrograms sounds nearly identical, while the audio signal is noticeably different. A GA would be able to find more elaborate perturbations, the random integer pertubation employed here merely serves as an example.

As encoding parameters A more limited representation might be beneficial if the noise representation takes too long to converge. Since the results in this thesis showed that codec parameters like bitrate also influenced classifier output, the transformation could also be represented as a list of encoding parameters. Then, if these encoding parameters are 'added' to the original audio, it is simply re-encoded using those parameters and the resulting audio is used in the next iteration of the GA.

While this approach will probably converge faster since the total amount of possible transformations is greatly reduced, the downside of using this representation is this same lack a generalizability: the transformation is only really applicable to a specific codec. However, this approach *does* generalize to different length inputs, since by design any encoder can handle different length inputs and we are only changing the parameters of the encoder.

7.3.4. Using a GA to navigate the search space

Regardless of the chosen representation, the setup for the GA remains largely the same. This section will briefly discuss the general proposed pipeline for setting up a GA which is able to effectively find adversarial examples for any music classifier. The proposed setup is presented visually in Figure 7.5 and the following paragraphs will describe how the individual blocks in the GA should be implemented.

Initialization In the initialization phase n random genes which describe different transformations will be generated. Depending on the chosen representation, these should either be:

- An array of integers within the bit-range of the given input audio if the noise representation is to be used. The length of the array can either be the same as the amount of samples s in the input audio (exploring a larger space, converging slower, not generalizing for other input audio) or of size l where $l < s$, tiling the array over the original audio when the addition operator is reached.
- An array of valid parameters for the given input audio if the encoding parameter representation is to be used. The parameters will be used to re-encode the original audio when the addition operator is reached.

Fitness function Since we are using a black-box approach, every time the audio is perturbed it needs to be run through the classifier again to observe the corresponding output. Thus, the fitness function block consists of three parts:

- LL: the Low-Level feature extractor. Since we are perturbing the audio and *not* the low-level features, each time a different transformation is applied to the audio the low-level features need to be recalculated.
- HL: the High-Level classifier. This can be any music classifier since our approach is a black-box one which is only concerned with input-output pairs, as long as it outputs label probabilities.
- F: the fitness function calculation. This function is very important and will steer us in the right direction through the search space. We essentially have two objectives: to find a perturbation which does not make the original audio sound significantly different and to make the classifier under test make an error which is 'as large as possible'.

  Since the output of the fitness function should be higher for more desirable solutions (these will have a higher probability of being selected in the evolution process) we can model the fitness of the solution for the first objective as the audio similarity between the original audio and the perturbed audio.

  Such audio similarity calculations might be costly if we are to run many generations of the GA to converge to a solution as this similarity would have to be calculated for each solution in the current population during each iteration of the GA. As such, it might be beneficial to use a simpler metric. For the noise based approach we could simply look at the perturbation integers, since these are the only changes made to the input. The similarity of the original and perturbed audio could then be modeled as:

$$sim(p_i) = 1 - \frac{\Sigma_{j=0}^{n} |p_{i_j}|}{p_{max}}$$

  where p_i is the perturbation array corresponding to solution i and p_{max} would be the maximum value of the perturbation for the input audio calculated as:

$$p_{max}(n, bitdepth) = \frac{2^{bitdepth}}{2} n$$

  where n is the length of the perturbation array and $bitdepth$ is the bit-depth of the input audio.

  The second objective, maximizing the error, could then be calculated by looking at the difference in label probabilities on the original audio and the perturbed audio:

$$\delta(f(O), f(O + p_i))$$

  where O is the original audio and p_i is the perturbation array corresponding to solution i.

  The full fitness function for solution s_i with corresponding perturbation array p_i would then be:

$$sim(p_i) + \delta(f(O), f(O + p_i))$$

  with optional weighting of the two objectives in the fitness function if one is deemed to be more important than the other or if convergence seems difficult using a strict similarity rule.

Convergence Convergence can be set to either a specific fitness value or a number of iterations depending on how well the GA performs and how much time each iteration takes. If this convergence criterium is reached, then we are 'happy enough' with the found solution and return it.

Selection, Crossover, Mutation Selection, crossover and mutation can now work as in any GA, given the representation of the transformation on the audio and the fitness function. The selection operator, which can be roulette wheel or tournament selection for example, will make sure that better solutions are selected with a higher probability without completely disregarding lower fitness solutions since they might still be viable after evolving further. The crossover operator will then combine the

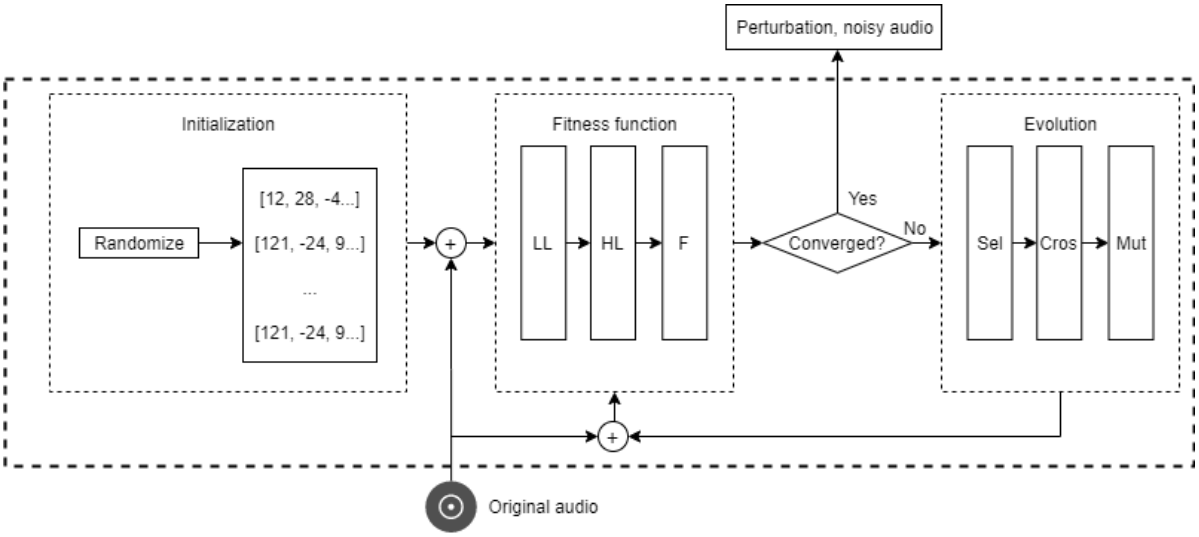


Figure 7.5: General setup of the proposed Genetic Algorithm (GA) approach to finding perturbations that can be used for the testing of (high-level) music classifiers. Both the perturbation as well as the noisy audio are returned, since the found perturbation might generalize to other audio inputs. Illustration inspired by [26]
LL = Low-Level feature extractor
HL = High-level classifier
F = Fitness function
Sel = Selection
Cros = Cross-Over
Mut = Mutation

genes (integers of the perturbation or codec parameter values) of the selected individuals, creating new solutions in the form of offspring. Each of these offspring then has a chance to mutate, where one or more values in the solution are changed randomly to promote exploration of the search space without getting stuck in a local optimum.

Output The GA should output both the perturbation — either the noise perturbation or the codec parameters, depending on the chosen representation — and the perturbed audio itself. While the perturbed audio, if found, serves as a direct adversarial example that is able to break the classifier, the perturbation might not be limited to just the given input audio or even music classifier given that adversarial examples often seem to generalize [55][93]. If the found pattern is able to break classifiers across multiple inputs or even multiple different classifiers, then it can be used as a very effective tool to generate new adversarial examples that can serve as metamorphic test cases for a wide variety of music classifiers *without* having to run the GA for each test.

7.4. Classifier retraining

While the abovementioned approaches aim to analyze *if* there is a problem with the classifier under test and can help identify where the current approaches fall short, they do not describe how to fix these potential issues.

One potential approach inspired by the image classification literature in the Deep Learning field would be to encode the CD-quality audio using different encodings and include these re-encoded items in the training dataset as a form of data augmentation. While data augmentation has been shown to improve the performance of image classifiers [110], performance on audio data is mixed [81][108], effectively turning an overfitting problem into an underfitting problem requiring larger networks and more training time to gain better performance.

Adverserial training — including the generated adverserial examples as training data — has also been shown to increase robustness against such adversarial attacks [55] but similarly shows that such training might actually hurt the overall performance of the trained classifier by limiting generalizability [121]. Thus, it might not be a good idea to include adversarial examples generated by the GA with the noise representation, since such perturbations might never be found in real-world data. Training

on the adversarial examples with the encoding parameter representation might limit this decrease in generalizability, since these perturbations should also be present in data in-the-wild, however this will have to be investigated further.

Developing new approaches to increase the robustness of classifiers still is an open problem, and the exact implementation is highly dependent on the results of the previously proposed research directions. Gaining a better understanding into why music classifiers seem to relatively unstable will be very valuable in steering this research in the right direction: without such deeper understanding, every shot we take will essentially be a shot in the dark.

8

Conclusions

With the results presented in the two literature studies as well as the analyses of the AcousticBrainz data, we can now summarize the majority of the analysis regarding the stability and agreement of the music classifiers in the AcousticBrainz dataset by answering the first two research questions posed in the introduction of this thesis.

RQ1: How can multiple representations of the same input be leveraged to quantify the performance of a classifier in terms of stability?

By utilizing multiple representations of the same input we can quantify the performance of any classifier based on input and output pairs, irrespective of any ground truth labels by employing metrics like the pooled variance to quantify the stability of such classifiers. Applying this metric to the AcousticBrainz data showed that small changes in the input audio due to for example the bitrate or codec used result in small changes in the low-level feature representation of this audio [143]. However, Chapter 3 demonstrated that, at least for the AcousticBrainz data, these small changes can have a significant effect on the output of any classifier trained to use to features in a high-level context. While the representation of the audio can have a relatively large impact on the stability of the output labels of such high-level classifiers (answering **RQ1.1**), representation of the audio seems to not be the only factor influencing this stability, with differences in label stability also being observed for different versions of the underlying Essentia feature extractor (answering **RQ1.2**).

RQ2: How can outputs from multiple implementations of classification tasks be leveraged to quantify the performance of a classifier in terms of agreement?

Utilizing outputs from multiple implementations of classification tasks effectively allows us to gain some insight into the construct validity of a classifier, again based only on input-output pairs when applying the specified agreement metric. If classifiers score low on this agreement metric, then there is a lower likelihood that such classifiers are actually able to properly capture the underlying constructs. Applying this agreement metric on the AcousticBrainz data in Chapter 4 seems to indicate insufficient construct validity in the trained high-level classifiers, with many of the label correlations between related classifiers in the AcousticBrainz data being worryingly low, especially for some of the genre classifiers which sometimes even showed negative correlations. Further analysis using the Spotify dataset provided more evidence that it is unlikely that the classifiers were properly able to capture the underlying constructs, especially for the valence (or mood_happy, mood_sad) classification. This is in line with the analysis presented in Chapter 2 which highlights the many uncertainties in assigning mood labels to audio.

The answers to **RQ2.1** and **RQ2.2** again are that the representation of the audio and the software versioning can influence the agreement results. In the case of agreement this might be due to the perceived label instability for some audio representations or simply due to the inability of the classifier to properly capture the underlying constructs.

8.1. The current state of high-level music classifiers

The results presented in this thesis provide reasons to be critical of the labels produced by high-level music classifiers such as those present in the AcousticBrainz dataset. It might be the case that the

construct underlying some of these high-level descriptors is hard to capture, as demonstrated in the literature study presented in Chapter 2 and as implied by the low agreement scores between related concepts. However, a lot of the results can be interpreted as evidence that while these classifiers were validated in lab conditions, they display anomalous and unstable behaviour on in-the-wild data due to this data being too different from the data used to train these classifiers. In-the-wild, these classifiers are presented with different representations of the audio regarding codec or bitrate, representations produced by different versions or builds of the low-level feature extraction software calculated on different operating systems using different CPU architectures or simply with audio that is simply different from the training set in terms of genre or instrumentation. It seems that the classifiers under study are not robust enough to all these minor variations in the input data. The anomalous behaviour can manifest itself as low label stability, low agreement scores or even distributional anomalies as analyzed in Chapter 5.

However, as the results of Chapter 6 show us, even if we control for many of these small variations of the input data by only considering data encoded using the same codec and extracted using the exact same software build of the low-level feature extractor, the label stability can still vary significantly between different songs even when these songs are relatively similar (being performed by the same artist). This shows us that even if the classifier would be robust enough to deal with slight variations in the input due to codec or software differences, stable results might still not be guaranteed. If one were to use the data provided in AcousticBrainz *as is*, then it would probably be best to only look at labels calculated on lossless, flac audio since this often resulted in better stability and agreement scores. However, given the many uncertainties about the validity of the output labels and the general lack of proper, extensive validation these labels should be interpreted very carefully.

The main takeaway then is that results from such high-level music classifiers should be taken as *anything but* the ground truth. These labels are produced in a process with cascading uncertainties: can the concept be captured with certainty at all? Is the representation of the audio used as input for any classifier stable enough? Does the classifier properly capture the desired concept? Is the classifier robust enough to deal with all different representations of the input present in-the-wild? Does the classifier generalize well enough to unseen data? The results presented in this thesis show that these are all questions that should be explored in more detail before doing any further interpretation of the mood labels produced by such high-level classifiers.

8.2. Towards better validation

This leads us to the answer to **RQ3: How can evaluation methods for such classifiers be improved using techniques from other disciplines like psychology and software testing?** Metrics that do not rely on ground truth data, but instead on for example different representations of the same input to quantify stability (bearing a resemblance to metamorphic testing) or the same input to different versions of a classifier (bearing a resemblance to differential testing) can prove useful in quantifying the performance of classifiers in a way that goes beyond simple cross-validation. The second literature study presented in Chapter 7 suggested several possible strategies to apply this idea on a larger scale, inspired by the psychological and software testing literature. Traditional cross-validation approaches are highly dependent on validation data, for which no objective, specified oracle exists. Thus, the presented approaches utilize derived oracles from different underlying software versions (in the case of differential testing) or different representations of the data (in the case of metamorphic testing). The manual analysis of the classifiers present in the AcousticBrainz data demonstrates the need for such validation methods, with many of the classifiers displaying anomalous behaviour while being validated to be 'correct' using the classical cross-validation approaches.

The high-level description of the Genetic Algorithm for finding these metamorphic transformations demonstrates the feasibility of the automatic metamorphic testing approach for aiding classifier validation. However, since the problem can essentially be cast into one of search space navigation, other search algorithms could be applied as well.

8.3. Shifting focus

These results indicate that a shift in focus for MIR research might be in order. While hunting for the perfect model and perfecting the machine learning part of the pipeline to maximize the reported accuracy might seem like a good idea, it is questionable if this actually helps to progress the field of MIR [35].

In the case of the high-level classifiers present in the AcousticBrainz set, many of them have reported accuracies of over 90% while displaying highly questionable results in real-world settings. It seems that these reported accuracies offer little guarantees on how well the model will actually perform. Instead, we should focus on the less studied parts of the pipeline: the data representation, software quality and reproducability (as indicated by the diffent outputs across Essentia versions and in line with the argument made by McFee et al. [95]) and evaluation methods.

8.4. Beyond the field of MIR

It should be noted that it would be naive to assume that MIR is the only field plagued by issues of interpretation, data collection and representation and a lack of proper validation methods. The described oracle problem is not unique to music data and oracle issues can also be present in for example the image domain [91] or clustering algorithms in general [151]. Essentially, *any* machine learning approach will be dependent on the data and labels used and such labels will always be subject to some interpretation, except for more trivial cases where there is a provably correct answer.

This highlights the need, not just for MIR but for machine learning in general, to move beyond validation using such 'ground-truth' labels to deal with the oracle problem. The GA described in Section 7.3.4 serves as an example of one such approach, and was purposefully described in such a way that it is able to generate test cases in a black-box manner, only looking at input-output pairs. Thus, such approaches could be translated to other domains without having to make any major changes to the algorithm, only the possible transformations and fitness function would have to be adapted using domain knowledge. Additionally, many of the analyses presented in this thesis only considered input-output pairs in the stability and agreement metrics, and thus these metrics could also be applied to data in other domains. Such metrics and approaches are the first steps towards better validation of machine learning classifiers, not just for the field of MIR but for any domain where such classifiers are applied.

Figure 8.1: Are high-level music classifiers actually able to model the abstract concepts we try to teach them, or do they follow in Clever Hans's hoovesteps?

Remember the criticism by Sturm [137] who likened some machine learning-based systems to the horse Clever Hans (see Figure 8.1) who appeared to be capable of extraordinary human feats like arithmetic, but turned out to rely on irrelevant confounds? Much of the anomalous behaviour of the studied classifiers can be interpreted as 'horse-like'. And, if it looks like a horse, walks like a horse and neighs like a horse, then maybe it *is* a horse.

Placing our blind trust in the output of systems with such symptoms would be akin to believing our horse really does have a gift for arithmetic. Instead, we should do something similar to what Dr. Oskar Pfungst did back then: take a critical look at the entire pipeline, from input to output. Run experiments in controlled conditions to see if our system depends on confounds in the data and focus on developing methods for extensive evaluation. Only then could we conclude that we have developed a classifier, and not a horse.

Bibliography

[1] The Echo Nest's Rosetta Stone: Unlocking Social Music. https://blog.echonest.com/post/66963888889/the-echo-nests-rosetta-stone-unlocking-social, last accessed on 06-03-2020.

[2] Fixing matching errors | Million Song Dataset. http://millionsongdataset.com/blog/12-2-12-fixing-matching-errors/, last accessed on 06-03-2020.

[3] Get Audio Features for a Track | Spotify for Developers. https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/, last accessed on 06-03-2020.

[4] Spotify — Company Info. https://newsroom.spotify.com/company-info/, last accessed on 06-03-2020.

[5] MIDI Association. https://www.midi.org/, last accessed on 26-02-2020.

[6] How Cancer Is Diagnosed. https://www.cancer.gov/about-cancer/diagnosis-staging/diagnosis, last accessed on 27-01-2020.

[7] H. Agrawal, J.R. Horgan, E.W. Krauser, and S.A. London. Incremental regression testing. In *1993 Conference on Software Maintenance*, pages 348–357, Montreal, Que., Canada, 1993. IEEE Comput. Soc. Press. ISBN 978-0-8186-4600-3. doi: 10.1109/ICSM.1993.366927.

[8] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, January 1991. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00153759.

[9] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, August 1992. ISSN 0003-1305, 1537-2731. doi: 10.1080/00031305.1992.10475879.

[10] Heike Argstatter. Perception of basic emotions in music: Culture-specific or multicultural? *Psychology of Music*, 44(4):674–690, July 2016. ISSN 0305-7356, 1741-3087. doi: 10.1177/0305735615589214.

[11] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, May 2015. ISSN 0098-5589, 1939-3520. doi: 10.1109/TSE.2014.2372785.

[12] Dominikus Baur, Jennifer Büttgen, and Andreas Butz. Listening factors: A large-scale principal components analysis of long-term music listening histories. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems - CHI '12*, page 1273, Austin, Texas, USA, 2012. ACM Press. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2208581.

[13] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105, 2004.

[14] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. 2011. doi: 10.7916/D8NZ8J07.

[15] Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of political Economy*, 100(5):992–1026, 1992.

[16] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R. Zapata, and Xavier Serra. Essentia: An Audio Analysis Library for Music Information Retrieval. In *ISMIR*, 2013.

[17] Dmitry Bogdanov, Alastair Porter, Hendrik Schreiber, Julián Urbano, and Sergio Oramas. The AcousticBrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale. In *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019): 2019 Nov 4-8; Delft, The Netherlands.[Canada]: ISMIR; 2019.* International Society for Music Information Retrieval (ISMIR), 2019.

[18] Marina Bosi and Richard E Goldberg. *Introduction to Digital Audio Coding and Standards*. Springer US, Boston, MA, 2003. ISBN 978-1-4615-0327-9. OCLC: 852779153.

[19] Gordon H. Bower. Mood and memory. *American Psychologist*, 36(2):129–148, 1981. ISSN 1935-990X, 0003-066X. doi: 10.1037/0003-066X.36.2.129.

[20] C. E. Brodley and M. A. Friedl. Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research*, 11:131–167, August 1999. ISSN 1076-9757. doi: 10.1613/jair.606.

[21] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, and Marcus Liwicki. DeXpression: Deep Convolutional Neural Network for Expression Recognition. *arXiv:1509.05371 [cs]*, August 2016.

[22] Donald Byrd and Tim Crawford. Problems of music information retrieval in the real world. *Information Processing & Management*, 38(2):249–272, March 2002. ISSN 03064573. doi: 10.1016/S0306-4573(01)00033-4.

[23] Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco, editors. *A Practical Guide to Sentiment Analysis*, volume 5 of *Socio-Affective Computing*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-55392-4 978-3-319-55394-8. doi: 10.1007/978-3-319-55394-8.

[24] Pedro Cano, Emilia Gómez Gutiérrez, Fabien Gouyon, Herrera Boyer, Markus Koppenberger, Bee Suan Ong, Xavier Serra, Sebastian Streich, and Nicolas Wack. ISMIR 2004 audio description contest. 2006.

[25] Davide Castelvecchi. Can we open the black box of AI? *Nature*, 538(7623):20–23, October 2016. ISSN 0028-0836, 1476-4687. doi: 10.1038/538020a.

[26] Jinyin Chen, Mengmeng Su, Shijing Shen, Hui Xiong, and Haibin Zheng. POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm. *Computers & Security*, 85:89–106, August 2019. ISSN 01674048. doi: 10.1016/j.cose.2019.04.014.

[27] Pei-Chun Chen, Keng-Sheng Lin, and Homer H. Chen. Emotional Accompaniment Generation System Based on Harmonic Progression. *IEEE Transactions on Multimedia*, 15(7):1469–1479, November 2013. ISSN 1520-9210, 1941-0077. doi: 10.1109/TMM.2013.2267206.

[28] T. Y. Chen, S. C. Cheung, and S. M. Yiu. Metamorphic testing: A new approach for generating next test cases. Technical report, Hong Kong University of Science and Technology, 1998.

[29] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, March 1989. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00116835.

[30] M.A. Conway and H.L. Williams. Autobiographical Memory. In *Learning and Memory: A Comprehensive Reference*, pages 893–909. Elsevier, 2008. ISBN 978-0-12-370509-9. doi: 10.1016/B978-012370509-9.00135-2.

[31] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00994018.

[32] Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3852–3856, Brighton, United Kingdom, May 2019. IEEE. ISBN 978-1-4799-8131-1. doi: 10.1109/ICASSP.2019.8682475.

[33] L. J. Cronbach and P. E. Meehl. Construct Validity in Psychological Tests. *Psychological Bulletin*, 52:281–302, 1955.

[34] Stuart Cunningham and Iain McGregor. Subjective Evaluation of Music Compressed with the ACER Codec Compared to AAC, MP3, and Uncompressed PCM. *International Journal of Digital Multimedia Broadcasting*, 2019:1–16, July 2019. ISSN 1687-7578, 1687-7586. doi: 10.1155/2019/8265301.

[35] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *arXiv:1911.07698 [cs]*, November 2019.

[36] Charles Darwin and Paul Ekman. *The Expression of the Emotions in Man and Animals*. Oxford University Press, Oxford; New York, 1998. ISBN 978-0-19-511271-9 978-0-19-515806-9 978-0-00-255866-2. OCLC: 37862652.

[37] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A Dataset for Music Analysis. In *18th International Society for Music Information Retrieval Conference*, 2017.

[38] Rémi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, and Manuel Moussallam. Music Mood Detection Based On Audio And Lyrics With Deep Neural Net. *arXiv:1809.07276 [cs, stat]*, September 2018.

[39] Kyle Devine. Imperfect sound forever: Loudness wars, listening formations and the history of sound reproduction. *Popular Music*, 32(2):159–176, May 2013. ISSN 0261-1430, 1474-0095. doi: 10.1017/S0261143013000032.

[40] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, Miami, FL, June 2009. IEEE. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206631.

[41] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008. ISSN 1347-5177, 1346-3969. doi: 10.1250/ast.29.247.

[42] Tuomas Eerola and Jonna K. Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):18–49, January 2011. ISSN 0305-7356, 1741-3087. doi: 10.1177/0305735610362821.

[43] P. Ekman, E. R. Sorenson, and W. V. Friesen. Pan-Cultural Elements in Facial Displays of Emotion. *Science*, 164(3875):86–88, April 1969. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.164.3875.86.

[44] Paul Ekman. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200, May 1992. ISSN 0269-9931, 1464-0600. doi: 10.1080/02699939208411068.

[45] Daniel Ellsberg. Risk, ambiguity, and the Savage axioms. *The quarterly journal of economics*, pages 643–669, 1961.

[46] D.M. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, July 2003. ISSN 0018-9448. doi: 10.1109/TIT.2003.813506.

[47] Michael Fingerhut. Music Information Retrieval, or how to search for (and maybe find) music and do away with incipits. In *IAML-IASA Congress, Oslo*, 2004.

[48] David Freedman, Robert Pisani, and Roger Purves. *Statistics*. W.W. Norton & Co, New York, 4th ed edition, 2007. ISBN 978-0-393-92972-0 978-0-393-93043-6. OCLC: ocm76142955.

[49] Joe Futrelle and J. Stephen Downie. Interdisciplinary Research Issues in Music Information Retrieval: ISMIR 2000?2002. *Journal of New Music Research*, 32(2):121–131, June 2003. ISSN 0929-8215. doi: 10.1076/jnmr.32.2.121.16740.

[50] David Galvin. Three tutorial lectures on entropy and counting. *arXiv:1406.7872 [math]*, June 2014.

[51] Alex Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by Transduction. *arXiv:1301.7375 [cs, stat]*, January 2013.

[52] Sandra Garrido and Emery Schubert. Individual Differences in the Enjoyment of Negative Emotion in Music: A Literature Review and Experiment. *Music Perception: An Interdisciplinary Journal*, 28(3):279–296, February 2011. ISSN 07307829, 15338312. doi: 10.1525/mp.2011.28.3.279.

[53] Yuan Gong and Christian Poellabauer. An Overview of Vulnerabilities of Voice Controlled Systems. *arXiv:1803.09156 [cs]*, March 2018.

[54] Yuan Gong and Christian Poellabauer. Crafting Adversarial Examples For Speech Paralinguistics Applications. *arXiv:1711.03280 [cs, eess, stat]*, January 2019. doi: 10.1145/3306195.3306196.

[55] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, March 2015.

[56] Fabien Gouyon, Herrera Boyer, Emilia Gómez Gutiérrez, Pedro Cano, Jordi Bonada, Alex Loscos, Xavier Amatriain, and Xavier Serra. Content processing of music audio signals. *Polotti P, Rocchesso D, editors. Sound to sense, sense to sound: a state of the art in sound and music computing. Berlin: Logos Verlag; 2008.*, 2008.

[57] Dara Greenwood. Of Sad Men and Dark Comedies: Mood and Gender Effects on Entertainment Media Preferences. *Mass Communication and Society*, 13(3):232–249, June 2010. ISSN 1520-5436, 1532-7825. doi: 10.1080/15205430903186526.

[58] Enric Guaus. *Audio Content Processing for Automatic Music Genre Classification: Descriptors, Databases, and Classifiers*. PhD thesis, PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2009.

[59] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple Black-box Adversarial Attacks. *arXiv:1905.07121 [cs, stat]*, August 2019.

[60] Prakash C. Gupta. *Data Communications and Computer Networks*. Eastern Academy Edition. PHI Learning Private Limited, New Delhi, 2011. ISBN 978-81-203-2846-4. OCLC: 930795200.

[61] Byeong-jun Han, Seungmin Rho, Sanghoon Jun, and Eenjun Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, May 2010. ISSN 1380-7501, 1573-7721. doi: 10.1007/s11042-009-0332-6.

[62] Wendy Heller. Neuropsychological mechanisms of individual differences in emotion, personality, and arousal. *Neuropsychology*, 7(4):476–489, 1993. ISSN 0894-4105. doi: 10.1037/0894-4105.7.4.476.

[63] Leah Henderson. The Problem of Induction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition, 2020.

[64] Perfecto Herrera, Zuriñe Resa, and Mohamed Sordo. Rocking around the clock eight days a week: An exploration of temporal patterns of music listening. In *RecSys 2010*, 2010.

[65] James Hiller and Susan C Gardstrom. The Selection of Music Experiences in Music Therapy. *Music Therapy Perspectives*, 36(1):79–86, April 2018. ISSN 0734-6875, 2053-7387. doi: 10.1093/mtp/miy001.

[66] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A Benchmark Dataset for Audio Classification and Clustering. In *ISMIR*, volume 2005, pages 528–31, 2005.

[67] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Ctree: Conditional inference trees. *The Comprehensive R Archive Network*, pages 1–34, 2015.

[68] W.E. Howden. Theoretical and Empirical Studies of Program Testing. *IEEE Transactions on Software Engineering*, SE-4(4):293–298, July 1978. ISSN 0098-5589. doi: 10.1109/TSE.1978.231514.

[69] Xiao Hu and J. Stephen Downie. Exploring Mood Metadata: Relationships with Genre, Artist and Usage Metadata. In *ISMIR*, pages 67–72, 2007.

[70] David Huron. Why is sad music pleasurable? A possible role for prolactin. *Musicae Scientiae*, 15(2):146–158, July 2011. ISSN 1029-8649, 2045-4147. doi: 10.1177/1029864911401171.

[71] Myra Interiano, Kamyar Kazemi, Lijia Wang, Jienian Yang, Zhaoxia Yu, and Natalia L. Komarova. Musical trends and predictability of success in contemporary songs in and out of the top charts. *Royal Society Open Science*, 5(5):171274, May 2018. ISSN 2054-5703, 2054-5703. doi: 10.1098/rsos.171274.

[72] International Electrotechnical Commission. *Audio Recording: Compact Disc Digital Audio System*. Geneva: International Electrotechnical Commission, 1999.

[73] Peter Jackson. *Introduction to Expert Systems*. International Computer Science Series. Addison-Wesley, Harlow, England ; Reading, Mass, 3rd ed edition, 1999. ISBN 978-0-201-87686-4.

[74] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, editors. *An Introduction to Statistical Learning: With Applications in R*. Number 103 in Springer Texts in Statistics. Springer, New York, 2013. ISBN 978-1-4614-7137-0. OCLC: ocn828488009.

[75] Petr Janata, Stefan T. Tomic, and Sonja K. Rakowski. Characterisation of music-evoked autobiographical memories. *Memory*, 15(8):845–860, November 2007. ISSN 0965-8211, 1464-0686. doi: 10.1080/09658210701734593.

[76] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, Oct./1993. ISSN 00189219. doi: 10.1109/5.241504.

[77] George H. John. Robust Decision Trees: Removing Outliers from Databases. In *KDD*, volume 95, pages 174–179, 1995.

[78] George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. *arXiv:1302.4964 [cs, stat]*, February 2013.

[79] Dimitra Karydi, Ioannis Karydis, and Ioannis Deliyannis. Legal issues in using musical content from iTunes and YouTube for music information retrieval. In *International Conference on Information Law*. Citeseer, 2012.

[80] Ai Kawakami, Kiyoshi Furukawa, Kentaro Katahira, and Kazuo Okanoya. Sad music induces pleasant emotion. *Frontiers in Psychology*, 4, 2013. ISSN 1664-1078. doi: 10.3389/fpsyg.2013.00311.

[81] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[82] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729694.

[83] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*, February 2017.

[84] Pierre Simon Laplace. *Théorie Analytique Des Probabilités*. Courcier, 1820.

[85] Cyril Laurier, Owen Meyers, Joan Serra, Martin Blech, and Perfecto Herrera. Music Mood Annotator Design and Integration. In *2009 Seventh International Workshop on Content-Based Multimedia Indexing*, pages 156–161, Chania, Crete, June 2009. IEEE. ISBN 978-1-4244-4265-2. doi: 10.1109/CBMI.2009.45.

[86] Kate Lawrence, Ruth Campbell, and David Skuse. Age, gender, and puberty influence the development of facial emotion recognition. *Frontiers in Psychology*, 6, June 2015. ISSN 1664-1078. doi: 10.3389/fpsyg.2015.00761.

[87] Quoc V. Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8595–8598, Vancouver, BC, Canada, May 2013. IEEE. ISBN 978-1-4799-0356-6. doi: 10.1109/ICASSP.2013.6639343.

[88] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14539.

[89] Cornelius T. Leondes, editor. *Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century*. Academic Press, San Diego, 2002. ISBN 978-0-12-443880-4.

[90] Lie Lu, D. Liu, and Hong-Jiang Zhang. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):5–18, January 2006. ISSN 1558-7916. doi: 10.1109/TSA.2005.860344.

[91] Cynthia C. S. Liem and Annibale Panichella. Oracle Issues in Machine Learning and Where to Find Them. In *Proceedings of the 8th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, ICSEW' 20, 2020. doi: 10.1145/3387940.3391490.

[92] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, Jan./1991. ISSN 00189448. doi: 10.1109/18.61115.

[93] Jiajun Lu, Hussein Sibai, and Evan Fabry. Adversarial Examples that Fool Detectors. *arXiv:1712.02494 [cs]*, December 2017.

[94] Omar Adil Mahdi, Mazin Abed Mohammed, and Ahmed Jasim Mohamed. Implementing a novel approach an convert audio compression to text coding via hybrid technique. *International Journal of Computer Science Issues (IJCSI)*, 9(6):53, 2012.

[95] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M. Bittner, and Juan Pablo Bello. Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research. *IEEE Signal Processing Magazine*, 36(1):128–137, January 2019. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2018.2875349.

[96] William M. McKeeman. Differential testing for software. *Digital Technical Journal*, 10(1):100–107, 1998.

[97] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, December 2014. ISSN 20904479. doi: 10.1016/j.asej.2014.04.011.

[98] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, New York, NY, international ed., [reprint.] edition, 20. ISBN 978-0-07-115467-3. OCLC: 846511832.

[99] Douglas C. Montgomery. *Design and Analysis of Experiments*. Wiley, Hoboken, NJ, tenth edition edition, 2020. ISBN 978-1-119-49247-4 978-1-119-49244-3.

[100] G.E. Moore. Cramming More Components Onto Integrated Circuits. *Proceedings of the IEEE*, 86 (1):82–85, January 1998. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.1998.658762.

[101] Sreerama K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998. ISSN 13845810. doi: 10.1023/A:1009744630224.

[102] Muyan Wang, Naiyao Zhang, and Hancheng Zhu. User-adaptive music emotion recognition. In *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.*, volume 2, pages 1352–1355, Beijing, China, 2004. IEEE. ISBN 978-0-7803-8406-4. doi: 10.1109/ICOSP.2004.1441576.

[103] Brady Neal. On the Bias-Variance Tradeoff: Textbooks Need an Update. *arXiv:1912.08286 [cs, stat]*, December 2019.

[104] David F. Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33(4): 275–306, April 2010. ISSN 0269-2821, 1573-7462. doi: 10.1007/s10462-010-9156-z.

[105] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[106] Annibale Panichella. Beyond Unit-Testing in Search-Based Test Case Generation: Challenges and Opportunities. In *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*, pages 7–8, Motreal, QC, Canada, May 2019. IEEE. ISBN 978-1-72812-233-5. doi: 10.1109/SBST.2019.00010.

[107] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. *arXiv:1602.02697 [cs]*, March 2017.

[108] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Interspeech 2019*, pages 2613–2617. ISCA, September 2019. doi: 10.21437/Interspeech.2019-2680.

[109] Minsu Park, Jennifer Thom, Sarah Mennicken, Henriette Cramer, and Michael Macy. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature Human Behaviour*, 3(3):230–236, March 2019. ISSN 2397-3374. doi: 10.1038/s41562-018-0508-z.

[110] Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621 [cs]*, December 2017.

[111] Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, 4(Jun):211–255, 2003.

[112] D.K. Peters and D.L. Parnas. Using test oracles generated from program documentation. *IEEE Transactions on Software Engineering*, 24(3):161–173, March 1998. ISSN 00985589. doi: 10.1109/32.667877.

[113] Oskar Pfungst. *Clever Hans (the Horse of Mr. Von Osten): A Contribution to Experimental Animal and Human Psychology*. Forgotten Books, Place of publication not identified, 2015. ISBN 978-1-332-47137-9. OCLC: 975964540.

[114] Rosalind W. Picard. Affective computing: (526112012-054), 1997.

[115] John Platt. Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods—Support Vector Learning (pp. 185–208). *AJ, MIT Press, Cambridge, MA*, 1999.

[116] Robert Plutchik. Emotions: A general psychoevolutionary theory. *Approaches to emotion*, 1984: 197–219.

[117] Robert Plutchik. The Nature of Emotions. *American Scientist*, 89(4):344, 2001. ISSN 0003-0996, 1545-2786. doi: 10.1511/2001.4.344.

[118] Alastair Porter, Dmitry Bogdanov, Robert Kaye, Roman Tsukanov, and Xavier Serra. AcousticBrainz: A Community Platform for Gathering Music Information Obtained from Audio. In *ISMIR*, 2015.

[119] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 2014. ISBN 978-0-08-050058-4. OCLC: 953284835.

[120] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3): 221–234, September 1987. ISSN 00207373. doi: 10.1016/S0020-7373(87)80053-6.

[121] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial Training Can Hurt Generalization. *arXiv:1906.06032 [cs, stat]*, August 2019.

[122] David Reinsel, John Gantz, and John Rydning. The digitization of the world from edge to core. *IDC White Paper*, 2018.

[123] Peter J. Rentfrow and Samuel D. Gosling. The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, 84 (6):1236–1256, 2003. ISSN 1939-1315, 0022-3514. doi: 10.1037/0022-3514.84.6.1236.

[124] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. *arXiv:1412.6550 [cs]*, March 2015.

[125] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x.

[126] James A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

[127] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, N.J, 1995. ISBN 978-0-13-103805-9.

[128] Parsa Saadatpanah, Ali Shafahi, and Tom Goldstein. Adversarial attacks on Copyright Detection Systems. *arXiv:1906.07153 [cs, stat]*, June 2019.

[129] Klaus R. Scherer. Which Emotions Can be Induced by Music? What Are the Underlying Mechanisms? And How Can We Measure Them? *Journal of New Music Research*, 33(3):239–251, September 2004. ISSN 0929-8215, 1744-5027. doi: 10.1080/0929821042000317822.

[130] Hendrik Schreiber. Improving Genre Annotations for the Million Song Dataset. In *ISMIR*, pages 241–247, 2015.

[131] Emery Schubert. Enjoyment of Negative Emotions in Music: An Associative Network Explanation. *Psychology of Music*, 24(1):18–28, April 1996. ISSN 0305-7356, 1741-3087. doi: 10.1177/0305735696241003.

[132] Matthew D. Schulkind, Laura Kate Hennis, and David C. Rubin. Music, emotion, and autobiographical memory: They're playing your song. *Memory & Cognition*, 27(6):948–955, November 1999. ISSN 0090-502X, 1532-5946. doi: 10.3758/BF03201225.

[133] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27 (3):379–423, July 1948. ISSN 00058580. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[134] Kyung-Shik Shin, Taik Soo Lee, and Hyun-jung Kim. An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, 28(1):127–135, January 2005. ISSN 09574174. doi: 10.1016/j.eswa.2004.08.009.

[135] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, January 1974. ISSN 00359246. doi: 10.1111/j.2517-6161.1974.tb00994.x.

[136] Simone Stumpf, Vidya Rajaram, Lida Li, Margaret Burnett, Thomas Dietterich, Erin Sullivan, Russell Drummond, and Jonathan Herlocker. Toward harnessing user feedback for machine learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, pages 82–91, 2007.

[137] Bob L. Sturm. A Simple Method to Determine if a Music Information Retrieval System is a “Horse”. *IEEE Transactions on Multimedia*, 16(6):1636–1644, October 2014. ISSN 1520-9210, 1941-0077. doi: 10.1109/TMM.2014.2330697.

[138] Larry Sullivan, editor. *The SAGE Glossary of the Social and Behavioral Sciences*. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States, 2009. ISBN 978-1-4129-5143-2 978-1-4129-7202-4. doi: 10.4135/9781412972024.n2493.

[139] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, February 2014.

[140] Christa L. Taylor and Ronald S. Friedman. Sad Mood and Music Choice: Does the Self-Relevance of the Mood-Eliciting Stimulus Moderate Song Preference? *Media Psychology*, 18(1): 24–50, January 2015. ISSN 1521-3269, 1532-785X. doi: 10.1080/15213269.2013.826589.

[141] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, Colorado Springs, CO, USA, June 2011. IEEE. ISBN 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995347.

[142] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002. ISSN 1063-6676. doi: 10.1109/TSA.2002.800560.

[143] Julián Urbano, Dmitry Bogdanov, Herrera Boyer, Emilia Gómez Gutiérrez, and Xavier Serra. What is the effect of audio quality on the robustness of MFCCs and chroma features? In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014); 2014 Oct 27-31; Taipei, Taiwan.[Place Unknown]: International Society for Music Information Retrieval; 2014. p. 573-578.* International Society for Music Information Retrieval (ISMIR), 2014.

[144] Paul E. Utgoff, James Cussens, Stefan Kramer, Sanjay Jain, Frank Stephan, Luc De Raedt, Ljupčo Todorovski, Pierre Flener, Ute Schmid, Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, Carlos Soares, Eamonn Keogh, William D. Smart, Pieter Abbeel, and Andrew Y. Ng. Inductive Learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 529–529. Springer US, Boston, MA, 2011. ISBN 978-0-387-30768-8 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_395.

[145] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, New York, 2nd ed edition, 2000. ISBN 978-0-387-98780-4.

[146] Jonna K. Vuoskoski and William F. Thompson. Who Enjoys Listening to Sad Music and Why? *Music Perception: An Interdisciplinary Journal*, 29(3):311–317, February 2012. ISSN 07307829, 15338312. doi: 10.1525/mp.2012.29.3.311.

[147] Geoffrey I. Webb, Johannes Fürnkranz, Johannes Fürnkranz, Johannes Fürnkranz, Geoffrey Hinton, Claude Sammut, Joerg Sander, Michail Vlachos, Yee Whye Teh, Ying Yang, Dunja Mladeni, Janez Brank, Marko Grobelnik, Ying Zhao, George Karypis, Susan Craw, Martin L. Puterman, and Jonathan Patrick. Deductive Learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 267–267. Springer US, Boston, MA, 2011. ISBN 978-0-387-30768-8 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_206.

[148] Tony Wigram, Lars Ole Bonde, and Inge Nygaard Pedersen. *A Comprehensive Guide to Music Therapy: Theory, Clinical Practice, Research, and Training*. Jessica Kingsley Publishers, London; Philadelphia, 2002. ISBN 978-1-84642-349-9 978-1-84310-083-6. OCLC: 896940900.

[149] Patrik Wikström. *The Music Industry: Music in the Cloud*. Polity, 2009. ISBN 978-0-7456-4390-8.

[150] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, 30(1): 79–82, 2005.

[151] Xiaoyuan Xie, Joshua W.K. Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software*, 84(4):544–558, April 2011. ISSN 01641212. doi: 10.1016/j.jss.2010.11.920.

[152] Dan Yang and Won-Sook Lee. Music Emotion Identification from Lyrics. In *2009 11th IEEE International Symposium on Multimedia*, pages 624–629, San Diego, California, USA, 2009. IEEE. ISBN 978-1-4244-5231-6. doi: 10.1109/ISM.2009.123.

[153] Fan Yang, Mengnan Du, and Xia Hu. Evaluating Explanation Without Ground Truth in Interpretable Machine Learning. *arXiv:1907.06831 [cs, stat]*, August 2019.

[154] Yi-Hsuan Yang, Chia-Chu Liu, and Homer H. Chen. Music emotion classification: A fuzzy approach. In *Proceedings of the 14th Annual ACM International Conference on Multimedia - MULTIMEDIA '06*, page 81, Santa Barbara, CA, USA, 2006. ACM Press. ISBN 978-1-59593-447-5. doi: 10.1145/1180639.1180665.

[155] Yi-Hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer H. Chen. Music Emotion Classification: A Regression Approach. In *Multimedia and Expo, 2007 IEEE International Conference On*, pages 208–211, Beijing, China, July 2007. IEEE. ISBN 978-1-4244-1016-3 978-1-4244-1017-0. doi: 10.1109/ICME.2007.4284623.

[156] Lotfi A. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(1):28–44, 1973. ISSN 0018-9472. doi: 10.1109/TSMC.1973.5408575.

[157] Eva Zangerle, Ramona Huber, Michael Vötter, and Yi-Hsuan Yang. Hit Song Prediction: Leveraging Low-and High-Level Audio Features. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019.

[158] Marcel Zentner, Didier Grandjean, and Klaus R. Scherer. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion*, 8(4):494–521, 2008. ISSN 1931-1516, 1528-3542. doi: 10.1037/1528-3542.8.4.494.

[159] Dolf Zillmann. Mood management: Using entertainment to full advantage. In *Communication, Social Cognition, and Affect (PLE: Emotion)*, pages 163–188. Psychology Press, 2015.

A

ISMIR 2020 Paper

During the thesis process me and my thesis supervisor, C.C.S. Liem, wrote a paper presenting part of the results from Chapters 3-5, which is included on the following pages. This paper has been accepted to the 21st International Society for Music Information Retrieval (ISMIR) conference[1] and will be presented at the conference between the 11th and 15th of October 2020 and be published in the conference proceedings.

[1] https://ismir.github.io/ISMIR2020/

CAN'T TRUST THE FEELING? HOW OPEN DATA REVEALS UNEXPECTED BEHAVIOR OF HIGH-LEVEL MUSIC DESCRIPTORS

Cynthia C. S. Liem
Delft University of Technology
Delft, The Netherlands
c.c.s.liem@tudelft.nl

Chris Mostert
Delft University of Technology
Delft, The Netherlands
chrismostert@outlook.com

ABSTRACT

Copyright restrictions prevent the widespread sharing of commercial music audio. Therefore, the availability of resharable pre-computed music audio features has become critical. In line with this, the AcousticBrainz platform offers a dynamically growing, open and community-contributed large-scale resource of locally computed low-level and high-level music descriptors. Beyond enabling research reuse, the availability of such an open resource allows for renewed reflection on the music descriptors we have at hand: while they were validated to perform successfully under lab conditions, they now are being run 'in the wild'. Their response to these more ecological conditions can shed light on the degree to which they truly had construct validity. In this work, we seek to gain further understanding into this, by analyzing high-level classifier-based music descriptor output in AcousticBrainz. While no hard ground truth is available on what the true value of these descriptors should be, some oracle information can still be derived, relying on semantic redundancies between several descriptors, and multiple feature submissions being available for the same recording. We report on multiple unexpected patterns found in the data, indicating that the descriptor values should not be taken as absolute truth, and hinting at directions for more comprehensive descriptor testing that are overlooked in common machine learning evaluation and quality assurance setups.

1. INTRODUCTION

In many music information retrieval (MIR) applications, it is useful to include information related to music content. However, many large-scale music audio collections of interest cannot legally be shared as-is. As a compromise, efforts have been undertaken to locally pre-compute music audio descriptors and make these available through APIs or as part of research datasets. Parties without in-house access to large audio corpora need to rely on such data for subsequent use. Indeed, large-scale pre-computed descriptor corpora have been feeding into further machine learning pipelines, empowering music applications, facilitating benchmarking initiatives [1, 2], and leading to inferences and statements about the nature of music preferences and listening behavior at an unprecedented scale [3–6].

Audio-based music descriptors are commonly divided into low- and high-level descriptors. Low-level descriptors can closely be related to the audio signal, while high-level descriptors are more semantically understandable to humans. This does not make high-level descriptors easier to extract; many of them cannot objectively and directly be measured in the physical world, and thus consider *constructs* rather than physically measurable phenomena.

The performance of automated music descriptor extraction procedures is reported according to the common evaluation methodologies in the field. For descriptors based on supervised machine learning, this normally includes a performance report on a test set that was partitioned out of the original dataset and not seen during training, or on cross-validation outcomes. However, descriptors that are reported and assumed to be successful may still be prone to sensitivities not explicitly accounted for in their design and evaluation. In lower-level music descriptors, implementations of MFCC and chroma descriptors showed sensitivities to different audio encoding formats [7], while common textual descriptions of audio extractor pipelines turned out insufficiently specific to yield reproducible results [8]. For higher-level descriptors, seemingly well-performing trained music genre classifiers turned out to be unexpectedly sensitive to subtle, humanly interpretable audio transformations [9]. Such sensitivities are not restricted to music genre classification; for example, trade-offs between accuracy and semantic robustness have also been observed in deep music representations [10]. Generally, in many MIR tasks, ground truth relies on human judgement and labeling. This may be imprecise and subjective, leading to low inter-rater agreement. In its turn, this leads to questions on whether a clear-cut ground truth exists at all, while this often is fundamental to machine learning techniques and their evaluation [11–14].

Can we tell whether automated descriptors are as trustable as initially assumed? Do they truly measure what they are intended to measure? Do they match broader, less explicitly encoded assumptions we have on them? These are important questions to ask: in case of negative an-

swers, the descriptors may not provide a valid basis for subsequent work to build upon. However, finding sensitivities that were unnoticed in original evaluation contexts is non-trivial, requiring a broader, more meta-analytic perspective. In this work, we focus on this, by providing an analysis of music descriptor values obtained through the AcousticBrainz [15] platform. By soliciting community-contributed submissions of locally run, but largely standardized music feature extractors, the platform offers a large-scale perspective on music that 'people felt worth the upload'. As such, it offers a more ecological 'in-the-wild' data perspective than what was studied in the lab, when the descriptors were originally designed. Indeed, through cross-collection evaluation procedures employing independent ground truth validation sets, several well-known genre classification models were shown not to generalize well beyond their original evaluation datasets [16].

The AcousticBrainz data is unusually transparent and rich: more so than e.g. the popular Million Song Dataset [17]. Many descriptor fields are available for each submission, multiple submissions can be added for the same MusicBrainz recording, each submission is encoded with additional metadata on characteristics of the input audio and the extractor software, and the extractor software is open source [18]. We use this richness to comprehensively analyze existing computed descriptor values in AcousticBrainz. Rather than relying on explicit and clear-cut ground truth, we look at the data through a meta-scientific lens, and impose more general assumptions on descriptor behavior, inspired by psychological and software testing techniques. This way, we will reveal several unexpected patterns in the descriptor values. As original music audio is not attached to the descriptor entries, we will not (yet) be able to fully replicate how descriptors were computed, nor will we be able to recreate experimental conditions on this data, in which possible reasons for unexpected behavior can cleanly be statistically controlled. Still, our analysis will help in pinpointing concrete directions towards future controlled studies.

In the remainder of this paper, we will discuss related work in Section 2. Then, we will introduce the data used for our analyses in Section 3, after which we will present analyses into intra-dataset correlations (Section 4), descriptor stability (Section 5), and descriptor value distributions (Section 6), followed by the conclusion and an outlook towards future work.

2. RELATED WORK

In conducting science, it is non-trivial to assess whether the outcomes we are observing, the inferences we are making and the conclusions we are drawing are truly correct. These questions of *validity* were first acknowledged in the domain of psychological testing, where the focus was on measuring psychological constructs: abstracted human characteristics (e.g. 'conscientiousness') that are not directly and physically observable, but that can still be measured (e.g. through well-designed surveys). Various subcategories of validity exist [19]. Among these, one of the most intuitive to understand, yet hardest to pinpoint, is the notion of *construct validity*: the question whether a measurement procedure can indeed be considered to yield a *"measure of some attribute or quality which is not "operationally defined""* [20].

The traditional viewpoint on ways to assess construct validity, is to consider a measure procedure as part of a *nomological network*, and relate its outcomes to those of other procedures, that have previously been shown to be valid [20]; in practice, in much of psychological research, this is done by assessing correlations between construct measurements that are theorized to have an interpretable relation to one another. This does create dependencies under uncertainty, still boiling down to a philosophical question of 'what the first truth is to start with'—something that may be disproven during the research process, as more evidence will come in and further comparisons are being made. It has therefore been argued that comprehensive inquiry into construct validity will not only lead to better assessments, but also leads to fundamental questionings and improvements of the complete scientific process [21].

Within MIR, while comprehensive meta-scientific questions on this have not been asked, criticisms of current evaluation practices, referring to the notions of both validity and reliability and the way in which they have been used in the Information Retrieval field, have been presented by Urbano et al. [22]. In addition, Sturm's criticisms of 'horse systems' in MIR [9] (machine learning-based systems that performance-wise appear to make humanly intelligent decisions, but that turn out to pick up on irrelevant confounds in data) can again be related to construct validity.

As a method to assess whether a system is a 'horse system', Sturm proposes to investigate how systems react to input data transformations that are considered 'irrelevant' (i.e. imperceptible) to humans. Interestingly, this technique has been used in another research field focused on 'testing': the field of software testing, in which it would be called *metamorphic testing* [23]. While software testing appears to be a much more objective and precise procedure than psychological testing, from a formal, logical perspective, many real-life programs may actually be considered non-testable, and the problem of determining whether a software artefact is bug-free is undecidable [24]. While one cannot pinpoint one exact oracle truth, it still may be possible to *derive* partial oracle truth through transformations based on known data relationships [25], e.g. by applying input transformations that should not change a system's output, which is done in metamorphic testing.

3. ACOUSTICBRAINZ

In our studies, we study descriptor values as found through the AcousticBrainz platform. More specifically, we will depart from the most recent high-level descriptor data dump obtained through the AcousticBrainz website [1]. We are interested in the high-level descriptors, as they should

[1] `https://AcousticBrainz.org/download`.
The data dump used in our analyses is `AcousticBrainz-highlevel-json-20150130.tar.bz2`

mimic humanly understandable semantic concepts, which should be relatable in humanly interpretable ways.

The data dump considers 1,805,912 entries of community-contributed high-level descriptor values, that can be broken down into genres, moods, and other categories (e.g. danceability); a full overview can be found in ([15], Table 4). Unless indicated otherwise, our analyses will consider this full data dump. In all cases, descriptor values consider classification outputs, obtained through machine learning; for each possible class label within a descriptor (e.g., `jazz` in the genre_dortmund classifier), the classifier confidence for that class label is given as a float value. The performance of each of the classifiers is documented on the AcousticBrainz website; where possible, performance is reported on publicly available datasets [2] .

4. INTRA-DATASET CORRELATIONS

Following the psychological concept of the nomological network, one way to assess validity is to assess how the outcomes of related measurement procedures correlate with each other. For this, we take advantage of *semantic redundancy* within the AcousticBrainz high-level descriptors. For example, several musical genres literally re-occur as class labels within the various genre classifiers. Then, it is not unrealistic to assume that, given the same audio input, the output of alternative jazz classifiers should positively correlate. Furthermore, some 'softer' assumptions on meaningful relationships can be made: e.g., aggressive music is likely not relaxed, and happy music is likely not sad. We defined multiple of these relationships for which we would expect to observe (strong) positive correlations between classifier label predictions, and computed their Pearson correlations. The results are displayed in Table 1.

The found correlations were unexpected; we were especially surprised by the very low correlations found for the genre classifiers, while they should target the same concepts. A scatter plot of `rock` classifier confidences in genre_rosamerica and genre_tzanetakis (which yielded a negative correlation) is given in Figure 1. It appears that confidences outcomes do not uniformly distribute over the full [0.0, 1.0] confidence range; we will investigate this further in the following sections.

Out of all 'softer' assumptions that were compared, the lowest correlation (.13) is between `happy` and `not sad`, implying that music classified as happy could be sad at the same time. The classifiers used in AcousticBrainz indeed allow for this, as separate binary classifiers exist for happy and sad moods; however, this contradicts Russell's 2D circumplex model of affect [26], in which happiness and sadness would have opposite scores on the valence dimension.

5. STABILITY

Our correlation analyses showed unexpected results. However, as different classifiers were trained on different datasets, they may have considered different characteristics of the input data. Inspired by the idea of derived oracles,

[2] `https://AcousticBrainz.org/datasets/accuracy`

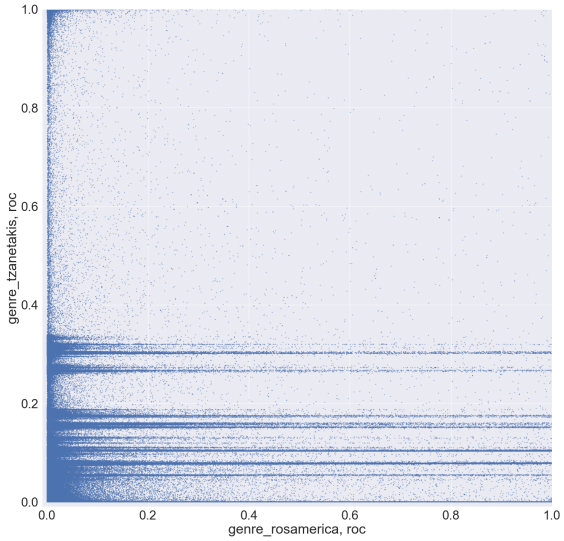


Figure 1: Scatter plot of classifier confidences. Each point indicates an AcousticBrainz submission, with confidences for genre_rosamerica, `roc` and genre_tzanetakis, `roc`.

we can however also consider relationships that should be closer to the identity, and thus should lead to (nearly) identical outcomes.

In AcousticBrainz, multiple submissions can be made for the same MusicBrainz recording ID (MBID). Semantically, a MusicBrainz recording really references one and the same recording. So while users may have encoded the recording audio in different ways, and may be using different versions of the feature extractor, we should intuitively be able to assume that re-submissions of one and the same recording should yield descriptor values that are very close to one another. In other words, we wish for re-submissions for the same MBID to display *stability*.

For this, we need to consider the MBIDs in our data dump that have more than one associated submission. Filtering for this led to a corpus of 941,018 submissions for 299,097 different MBIDs. If n submissions are available for a given MBID, a given classifier c and a given classifier label l, the corresponding classifier confidences for these submissions can now be grouped into a population (MBID, c, l) of size n. Considering we have k unique MBIDs in our dataset (in our case, k = 299,097), we can then enumerate the populations as $[(\text{MBID}_1, c, l), (\text{MBID}_2, c, l), ..., (\text{MBID}_k, c, l)]$, and operate within and/or across them when calculating instability metrics.

We consider two alternative ways to quantify instability. First, for each of the submission populations, we can compute the variance observed for classifier confidences, for each label l in classifier c. As there may be a varying amount of submissions within a population, we normalize for this by computing the *pooled variance* $\overline{\text{var}}(c, l)$ over our filtered corpus as follows:

$$\overline{\text{var}}(c, l) = \frac{\Sigma_{i=1}^{k}(n_i \times \text{var}((\text{MBID}_i, c, l)))}{\Sigma_{i=1}^{k} n_i}$$

Classifier, label A	Classifier, label B	Pearson's r	p
genre_rosamerica, cla	genre_tzanetakis, cla	.29	<.001
genre_dortmund, rock	genre_rosamerica, roc	.24	<.001
genre_dortmund, jazz	genre_rosamerica, jaz	.22	<.001
genre_dortmund, pop	genre_rosamerica, pop	.11	<.001
genre_dortmund, jazz	genre_tzanetakis, jaz	.08	<.001
genre_rosamerica, pop	genre_tzanetakis, pop	.06	<.001
genre_rosamerica, hip	genre_tzanetakis, hip	.05	<.001
genre_rosamerica, jaz	genre_tzanetakis, jaz	.02	<.001
genre_dortmund, blues	genre_tzanetakis, blu	.01	<.001
genre_dortmund, pop	genre_tzanetakis, pop	-.05	<.001
genre_dortmund, rock	genre_tzanetakis, roc	-.06	<.001
genre_rosamerica, roc	genre_tzanetakis, roc	-.07	<.001
mood_aggressive, aggressive	mood_relaxed, not_relaxed	.59	<.001
mood_acoustic, acoustic	mood_electronic, not_electronic	.58	<.001
danceability, danceable	mood_party, party	.53	<.001
mood_electronic, electronic	genre_dortmund, electronic	.48	<.001
danceability, danceable	genre_rosamerica, dan	.33	<.001
mood_happy, happy	mood_party, party	.20	<.001
mood_happy, happy	mood_sad, not_sad	.13	<.001

Table 1: Pearson correlations between high-level classifier outcomes, theorized to positively correlate with another.

where n_i is the sample size of the ith population in our enumeration.

As there are multiple possible labels within the same classifier, but we want to discuss outcomes at the classifier level, we then take the mean pooled variance, $\overline{\overline{\text{var}(c)}}$, over all possible labels $l \in L_c$ for classifier c.

When using variances, classifier confidences are considered to be informative. Alternatively, one could choose to rather consider each classifier label as a binary label. To reflect this perspective, for each population and for each classifier, we can compute the normalized information entropy $\hat{H}(\text{MBID}_i, c)$, which uses the Shannon entropy [27], but normalizes by the amount of possible labels $|L_c|$ for c:

$$\begin{aligned}&\hat{H}(\text{MBID}_i, c)\\ &= -\Sigma_{l \in L_c} \frac{P((\text{MBID}_i, c, l)) \log_2 P((\text{MBID}_i, c, l))}{\log_2 |L_c|}\\ &= -\Sigma_{l \in L_c} P((\text{MBID}_i, c, l)) log_{|L_c|} P((\text{MBID}_i, c, l)).\end{aligned}$$

Where $P((\text{MBID}_i, c, l))$ is the probability of label l in classifier c, following the observed empirical distribution within the population corresponding to MBID_i. Then, to have a weighted measure per classifier over the whole filtered corpus, we calculate the pooled normalized entropy $\overline{\hat{H}}(c)$, similarly to how we computed the pooled variance.

While we want for descriptor values to be stable within a submission, it is usually not the intention that for a given descriptor, the classifier would be so stable that it always predicts a single l throughout the whole corpus. This e.g. happens for the genre_dortmund classifier, which unrightfully classifies many AcousticBrainz submissions as electronic music, as also noticed in [16]. To quantify the unbiasedness of a classifier, we compute the normalized entropy for each classifier over our complete (unfiltered) corpus, denoted as $\hat{H}(c)_{all}$. A higher $\hat{H}(c)_{all}$ denotes a more uniform distribution over the different possible class labels for c across the corpus, and thus lower classifier bias.

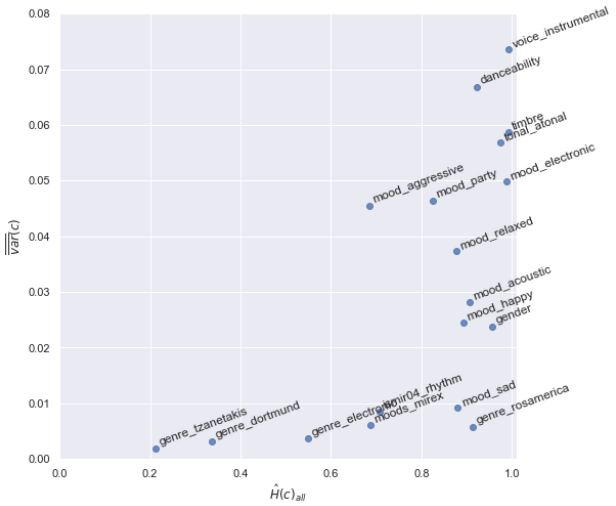
Plots in which we illustrate $\overline{\overline{\text{var}(c)}}$ and $\overline{\hat{H}}(c)$ (pooled with regard to recordings with multiple submissions) vs. $\hat{H}(c)_{all}$ (taken across the whole, unfiltered corpus) are shown in Figure 2. As we can see, indeed, the genre classifiers turn out stable but highly biased. While in most cases, observed trends are comparable for the two possible instability measures, some exceptions are found, most notably on the gender classifier, which is considered stable when using $\overline{\overline{\text{var}(c)}}$, but unstable when using $\overline{\hat{H}}(c)$. Seemingly, confidences for this classifier are close to 0.5, meaning that male/female classifications easily flip within a submission.

6. VALUE DISTRIBUTIONS

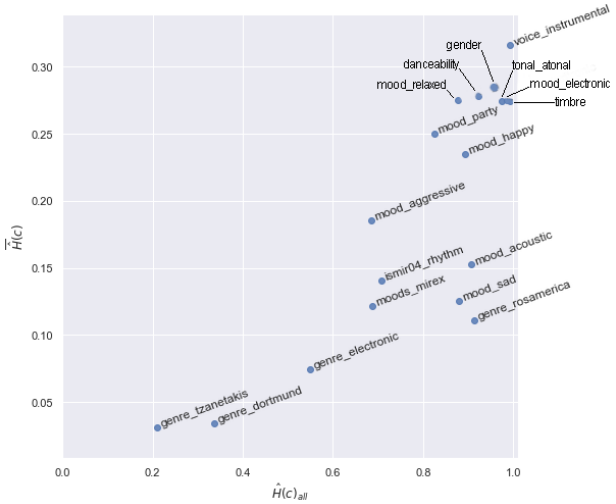
From Figure 1, it was observed that descriptor values clustered together in small bands. This behavior occurs for several genre and mood classifiers. To illustrate this, Figure 3 displays a histogram of descriptor values for the mood_acoustic, mood_relaxed, mood_electronic and mood_sad classifiers, as observed across the complete AcousticBrainz corpus. Some confidence values seem disproportionally represented: in the histogram, sharp spikes occur for mood_acoustic, mood_relaxed, mood_electronic, and a minor spike for mood_sad.

There are various reasons why this may be the case. Possibly, the community may have fed skewed data to the classifier. Alternatively, the feature extractor may have shown anomalous responses to specific inputs. For each submission, we have rich metadata, which e.g. includes information about audio codecs, bit rates, song lengths, and software library versions that were used when the submission was created. While, in the absence of a conscious experimental design underlying the data, we cannot cleanly test for contributions of individual facets, we still can examine *whether major distributional differences occur for submissions with scores within the anomalous-looking spikes, when comparing these to submissions with scores outside of these*.

For this, for each of the classifiers, we manually define range intervals for the classifier confidences, within which



(a) Instability based on mean pooled variance $\overline{\overline{\text{var}}}(c)$.



(b) Instability based on pooled normalized entropy $\overline{\hat{H}}(c)$.

Figure 2: Submission instability vs. corpus-wide unbiasedness ($\hat{H}(c)_{all}$).

we consider a submission to belong to an anomalous classifier confidence value spike. We then compare the metadata value distributions of submissions within each classifier spike to those of submissions that do not occur in any of the four anomalous spikes (1,239,882 submissions for 855,266 unique MBID recordings).

To investigate whether the observed anomalies may have been skewed towards any particular genre, we also study a subset of our corpus, which was cross-matched against the AcousticBrainz genre dataset [28]. More specifically, we only kept MBIDs which also occurred in all three publicly available ground truth sets (Discogs, last.fm and tagtraum) of the AcousticBrainz genre dataset, reducing the corpus to 402,279 submissions for 164,826 unique MBID recordings. Examining confidence value distributions for this filtered dataset, we still observed the same anomalous spikes for the same range intervals. Therefore, we will apply the same range intervals as before to select values associated to anomaly spikes, and will

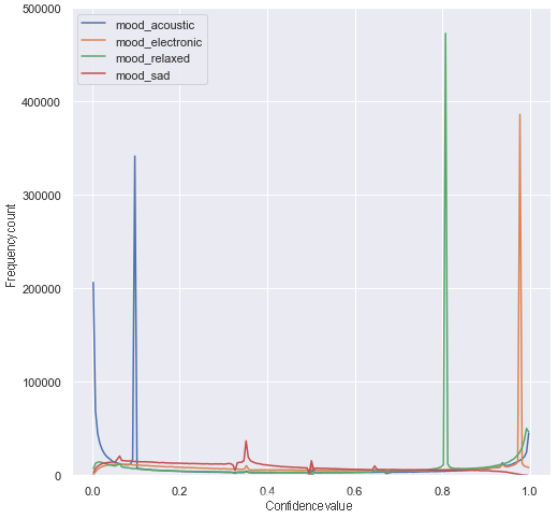


Figure 3: Histogram of descriptor values for several classifiers, considered across the whole corpus.

again compare distributional differences between these and non-anomalous submissions (now amounting to 267,394 submissions for 128,687 unique MBID recordings), in this case to see whether certain genres are overrepresented in the anomalous spikes. For each classifier of interest, an overview of anomalous spike interval ranges and counts of corresponding unique recording MBIDs and submissions is given in Table 2.

To quantify distributional differences, we use the Jensen-Shannon (JS) distance metric:

$$JS\_distance(p, q) = \sqrt{\frac{D(p\|m) + D(q\|m)}{2}}$$

where m is the pointwise mean of p and q and D is the Kullback-Leibler (KL) divergence [29]. The JS distance is based on the JS divergence [30]; as advantages over the KL divergence, the JS divergence is symmetric and always has a finite value within the [0, 1] range [31].

For each metadata category in our overall corpus, and for each genre category in our genre-filtered corpus, we calculate the JS distance between the frequency occurrence profiles of category values, counted over all submissions within an anomalous spike, vs. all submissions without any anomalous spike. As some categories can assume many different values (e.g. replay_gain), we only do comparisons for values that occur at least 10 times in both frequency profiles. JS distance values for the metadata comparisons are listed in Table 3, while JS distance for the genre comparisons are listed in Table 4.

As can be observed in Table 3, comparing submissions within and outside of the anomalous spikes, major distributional differences are found for used extractor software versions. These go up to the level of Essentia Git commit and build versions that were used for low-level feature extraction. In addition, we also observe distributional differences for bit_rate and codec, likely confirming earlier observations [7] that low-level feature extractors may display sensitivities with regard to different audio codecs and

Classifier	Anomalous range	Full		Genre	
		#MBIDs	#submissions	#MBIDs	#submissions
mood_acoustic, `acoustic`	[0.09, 0.10]	282,605	358,747	60,261	94,268
mood_relaxed, `relaxed`	[0.805, 0.815]	373,555	485,184	72,739	119,050
mood_electronic, `electronic`	[0.972, 0.982]	315,626	401,151	64,944	101,915
mood_sad, `sad`	[0.346, 0.362]	57,697	75,688	8,854	14,242

Table 2: Details of anomalous spike data slices used for distributional comparisons. For each classifier of interest, we indicate the classifier confidence range for which a submission was considered to be anomalous. We also list the counts of unique MBID recordings and overall submissions, both for the full corpus and our genre-filtered corpus.

	`acoustic`	`relaxed`	`electronic`	`sad`
bit_rate	.42	.32	.39	.17
codec	.34	.26	.32	.06
length	.15	.15	.15	.32
lossless	.28	.21	.27	.02
essentia_low	.61	.52	.59	.15
essentia_git_sha_low	.67	.58	.66	.23
essentia_build_sha_low	.70	.62	.69	.24

Table 3: JS distances between frequency profiles over metadata categories, for anomalous vs. non-anomalous submissions considering the four classifiers of interest. For metadata categories that are not listed, found JS distances were always 0.

	`acoustic`	`relaxed`	`electronic`	`sad`
Discogs	.12	.09	.11	.11
last.fm	.14	.12	.13	.14
tagtraum	.14	.11	.13	.14

Table 4: JS distances between frequency profiles over genre categories, for anomalous vs. non-anomalous submissions considering the four classifiers of interest.

compression rates. In contrast, Table 4 shows that JS distances are equivalent and low across genre taxonomies and types of anomalies: from this, it seems more likely that the anomalies were caused by submission extraction contexts, rather than the inclusion of anomalous data.

7. CONCLUSIONS AND FUTURE WORK

In this work, we analyzed patterns in high-level descriptor values in AcousticBrainz. As we showed, while the descriptors were successfully validated under lab conditions, they show unexpected behavior in the wild, raising questions on the extent to which they have construct validity.

The unexpected behavior could have two potential causes. First of all, **the construct underlying several high-level descriptors may be conceptually problematic by itself**. For example, the concept of genre [32], as well as its use in machine learning classification tasks [33] has been criticized by musicologists and musicians. Furthermore, within music psychology, there have been findings that sad music does not necessarily elicit sad emotions [34, 35]. Further interdisciplinary research will be needed to better understand these phenomena.

Our current analyses also accumulated evidence that **the AcousticBrainz community confronted the descriptors with audio and extraction contexts that were too different from the contexts on which classifiers originally were trained**. It should be noted that original training datasets for the classifiers were far smaller in size (several hundreds to thousands of data points) than the current scale of AcousticBrainz, and that this logically may not have managed capturing all intricacies of larger-scale, ecologically valid data. However, our analyses suggest that anomalous behavior may also be due to audio codecs, compression rates and different versions of software implementations and builds that were used during extraction, which are rarely explicitly considered and reported in evaluation setups. As for the software versions, it should further be noted that, while we focused on high-level descriptors, all found differences occurred in the extraction procedures of low-level descriptors (feature representations), while the high-level machine learning models stayed constant. Thus, low-level descriptor performance should explicitly stay in scope when studying high-level descriptors.

With this work, we wished to shed light on current challenges regarding the reproducibility and generalizability of research outcomes, and on elements of processing pipelines that are under-represented in applied machine learning and signal processing literature, yet play a critical role for the pipeline's performance [8, 36]. Inspired by literature in both psychological and software testing, we also offered several possible strategies to assess descriptor validity, even in the absence of a clear ground truth.

While we exposed several potentially problematic patterns, we explicitly do not wish for this work to be seen as a criticism of AcousticBrainz and/or Essentia. No other MIR resource or API currently offers similar levels of transparency that allow for analyses like we performed here, and we would like to explicitly thank the teams behind these initiatives for their openness. It also is this openness that will allow for us to perform further research in the near future—with more systematic testing strategies and experimental designs—towards more holistic quality assurance procedures for applied machine learning procedures in the context of humanly-interpretable signal data.

8. REFERENCES

[1] A. Schindler and R. Mayer and A. Rauber, "Facilitating comprehensive benchmarking experiments on the million song dataset," in *Proceedings of the 13th Conference of the International Society for Music Information Retrieval (ISMIR 2012)*, 2012.

[2] D. Bogdanov, A. Porter, J. Urbano, and H. Schreiber, "The MediaEval 2018 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources," in *MediaEval Benchmark Workshop*, 2018.

[3] J. Serrà, A. Corral, M. Boguñá, M. Haro, and J. L. Arcos, "Measuring the Evolution of Contemporary Western Popular Music," *Scientific Reports*, vol. 2, 2012.

[4] M. Interiano, K. Kazemi, L. Wang, J. Yang, Z. Yu, and N. L. Komarova, "Musical trends and predictability of success in contemporary songs in and out of the top charts," *Royal Society Open Science*, vol. 5, no. 171274, 2018.

[5] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy, "Global music streaming data reveal diurnal and seasonal patterns of affective preference," *Nature Human Behaviour*, vol. 3, no. 3, pp. 230–236, 2019.

[6] E. Zangerle, R. Huber, M. Vötter, and Y.-H. Yang, "Hit Song Prediction: Leveraging Low-and High-Level Audio Features," in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[7] J. Urbano, D. Bogdanov, P. Herrera, E. Gómez, and X. Serra, "What is the Effect of Audio Quality on the Robustness of MFCCs and Chroma Features?" in *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[8] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, "Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research," *IEEE Signal Processing Magazine*, vol. 36, 2019.

[9] B. L. Sturm, "A Simple Method to Determine if a Music Information Retrieval System is a "Horse"," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636–1644, 2014.

[10] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "Are Nearby Neighbors Relatives? Testing Deep Music Embeddings," *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 53, 2019.

[11] A. Flexer and T. Grill, "The problem of limited inter-rater agreement in modelling music similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016.

[12] A. Flexer and T. Lallai, "Can we increase inter- and intra-rater agreement in modeling general music similarity?" in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[13] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, "Annotator subjectivity in harmony annotations of popular music," *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.

[14] S. Balke, J. Abeßer, J. Driedger, C. Dittmar, and M. Müller., "Towards evaluating multiple predominant melody annotations in jazz recordings," in *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[15] A. Porter, D. Bogdanov, R. Kaye, R. Tsukanov, and X. Serra, "AcousticBrainz: A Community Platform for Gathering Music Information Obtained from Audio," in *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR 2015)*, 2015.

[16] D. Bogdanov, A. Porter, P. Herrera, and X. Serra, "Cross-collection evaluation for music classification tasks," in *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR 2016)*, 2016.

[17] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proceedings of the 12th Conference of the International Society for Music Information Retrieval (ISMIR 2011)*, 2011.

[18] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: An Audio Analysis Library for Music Information Retrieval," in *Proceedings of the 14th Conference of the International Society for Music Information Retrieval (ISMIR 2013)*, 2013.

[19] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.

[20] L. J. Cronbach and P. E. Meehl, "Construct Validity in Psychological Tests," *Psychological Bulletin*, vol. 52, p. 281–302, 1955.

[21] G. T. Smith, "On Construct Validity: Issues of Method and Measurement," *Psychological Assessment*, vol. 17, no. 4, pp. 396–408, 2005.

[22] J. Urbano, M. Schedl, and X. Serra, "Evaluation in Music Information Retrieval," *Journal of Intelligent Information Systems*, vol. 31, pp. 345–369, 2013.

[23] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Hong Kong University of Science and Technology, Tech. Rep., 1998.

[24] E. T. Weyuker, "On Testing Non-testable Programs," *The Computer Journal*, vol. 25, no. 4, pp. 465—-470, 1982.

[25] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[26] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161—-1178, 1980.

[27] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[28] D. Bogdanov, A. Porter, H. Schreiber, J. Urbano, and S. Oramas, "The AcousticBrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale," in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval (ISMIR 2019)*, 2019.

[29] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[30] D. Endres and J. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[31] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[32] C. C. S. Liem, A. Rauber, T. Lidy, R. Lewis, C. Raphael, J. D. Reiss, T. Crawford, and A. Hanjalic, "Music Information Technology and Professional Stakeholder Audiences: Mind the Adoption Gap," in *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, vol. 3.

[33] B. L. Sturm, "Classification accuracy is not enough: On the evaluation of music genre recognition systems," *Journal of Intelligent Information Systems*, vol. 41, pp. 371–406, 2013.

[34] J. K. Vuoskoski and T. Eerola, "Can sad music really make you sad? Indirect measures of affective states induced by music and autobiographical memories." *Psychology of Aesthetics, Creativity, and the Arts*, vol. 6, no. 3, pp. 204–213, 2012.

[35] A. Kawakami, K. Furukawa, K. Katahira, and K. Okanoya, "Sad music induces pleasant emotion," *Frontiers in Psychology*, vol. 4, 2013.

[36] M. F. Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, "A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research," *arXiv preprint arXiv:1911.07698*, 2019.