# 5. Results

In the last chapter of the thesis the results concerning the quality of the recommendations are shown. At the end of the chapter the work of the thesis is summarized and the open task for future work are outlined. An attempt to quantify the results and the quality of the recommendations is made by choosing objective tests like genre recall and cover song recognition. In the second part a few subjective impressions are given, including personal taste and listening preferences.

## 5.1 Objective evaluation

For the scientific/ objective evaluation at first the resulting distances are analyzed and visualized in section 5.1.1. To test the quality of the resulting song recommendations returned by the Spark application, some tests were made. To test the quality of the timbre and rhythm based distances the genre recall rate is examined in section 5.1.3. Another indicator for the quality of rhythm features is the ability to recommend songs around the same BPM count (see section 5.1.4). As mentioned in section 3.2.4 a way of evaluating the quality of the melodic similarities is to test the ability to detect cover songs. This will be examined in section 5.1.2.

### 5.1.1 feature correlation and distance distribution

This section analyzes the results from the similarity analysis to find out how the distances from different feature sets are correlated to each other and how they are distributed over the unit interval [0*,*1]. To analyze this, a test dataset consisting of distances coming from the Spark framework had to be created. 95 songs (five songs for every genre) were randomly chosen from the 1517 artists dataset and the distances to all other songs of the 1517 artists dataset were calculated. The dataset contains 3180 songs evenly distributed over 19 different genres (see figure 2.13b).

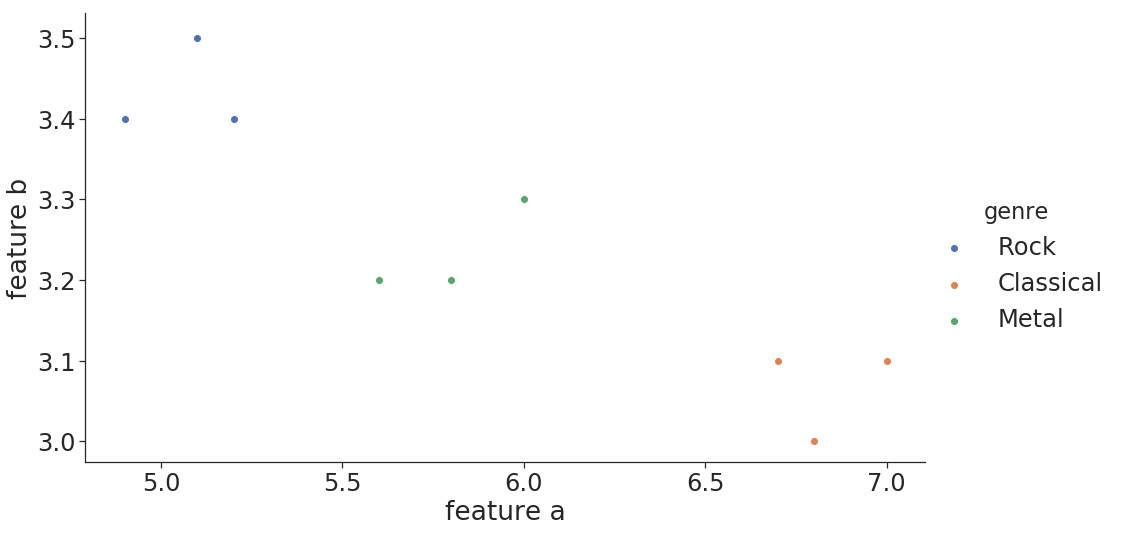


Figure 5.1: Feature space example

The sampling of distances from different genres is important for the analysis of the distribution of the distances, because the distances and their distribution vary, depending on where in the feature space the actual song is located. A song taken from the edge of the distribution of the feature space will end up with different distances than a song taken from the center. To visualize this figure 5.1 shows a minimal example. While the distances from songs tagged with ”Metal” to the songs tagged with ”Rock” and ”Classical” are about the same, the distances from a song taken off the Classical genre to the ”Rock” or ”Metal” songs are different in this example.

Figure 5.2 shows the correlation of the distances from the various feature types. The eight different distances for each song pair are summed up into one new combined distance (following formula 4.5 with all weights *w* = 1 ). This combined distance is labeled as ”agg” in the following plots.

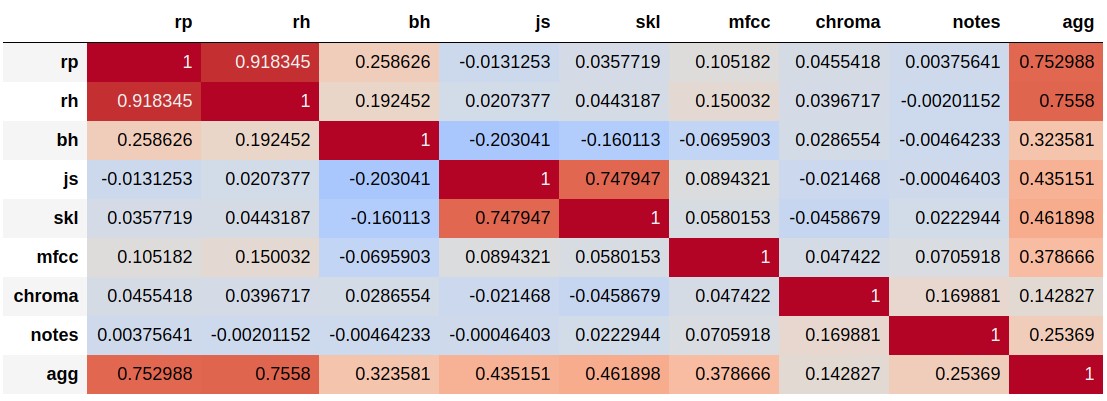
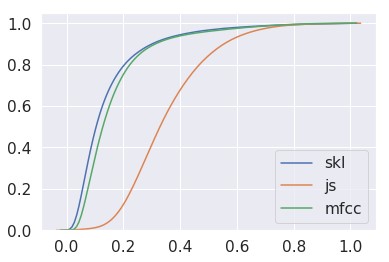


Figure 5.2: correlation 95 songs, 19 genres (5 each), 1517 artists

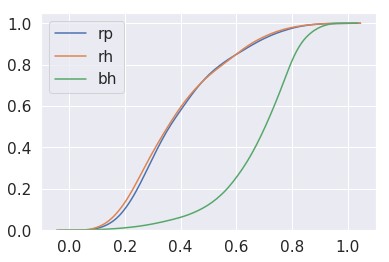
The correlation of a feature type with the overall distance is a sign of the impact of the feature type on the overall distance from the weighted sum. But because not all distances are equally distributed over the unit interval the different feature types have different impacts on the sum of distances. This problem was already mentioned in section 4.2.5 and section 4.2.4. Figure 5.3 shows how the distances are distributed with the cumulative histograms over the unit interval. It is clearly visible that the cross-correlation distances are not evenly distributed. In section 4.2.5 a few proposals were already given how this problem could be solved in the future.



(

a) cumulative distribution

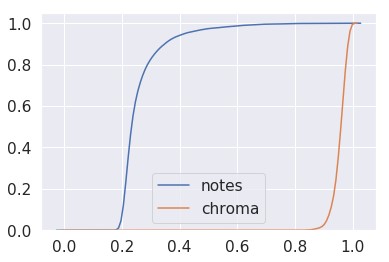
1



(

b) cumulative distribution

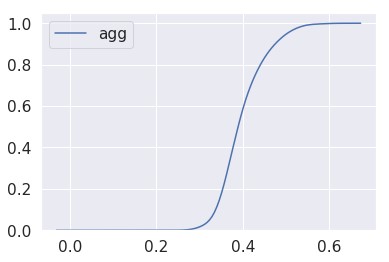
2



(

c) cumulative distribution

3



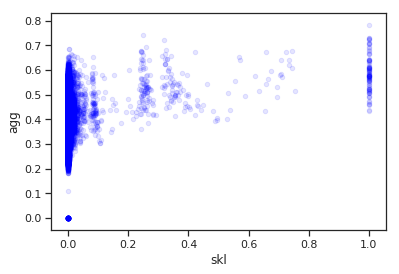
(

d) cumulative distribution

4

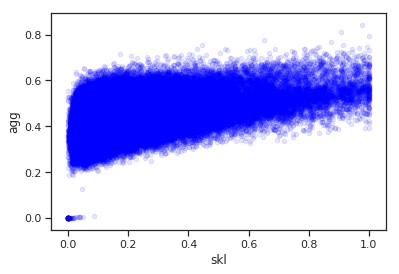
Figure 5.3: Cumulative distributions

As mentioned in section 4.2.4 the SKL divergence was also prone to outliers and had issues when scaling distances to the unit interval. The solution was to filter out all song pairs with an SKL divergence larger than a certain threshold before scaling the distances. If this filter operation is left out nearly all distances calculated with the symmetric Kullback-Leibler divergence are close to zero after the scaling. The impact can be seen in figure 5.4 and figure 5.5 If the outliers are not filtered, the correlation between the distances of the different feature types change drastically. The correlation of the unfiltered SKL distances with the combined distance (”agg”) decreases significantly (see figure 5.5). Interestingly also the correlation of the Jensen-Shannon like divergence and the combined distance (”agg”) is dropping.



(

a) SKL unscaled

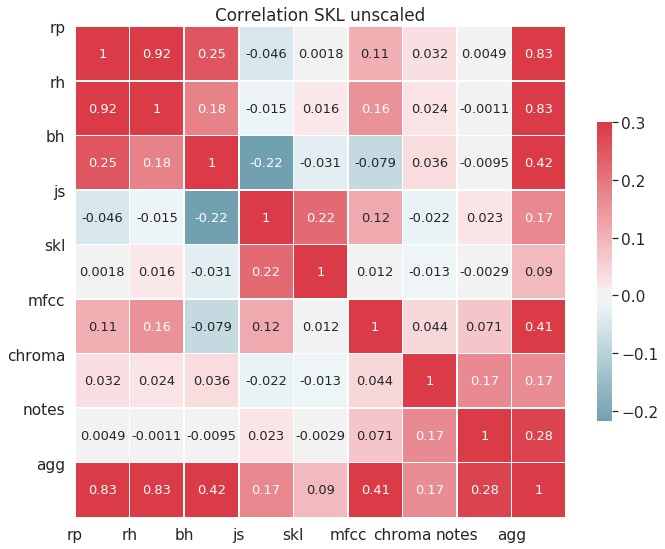


(

b) SKL scaled

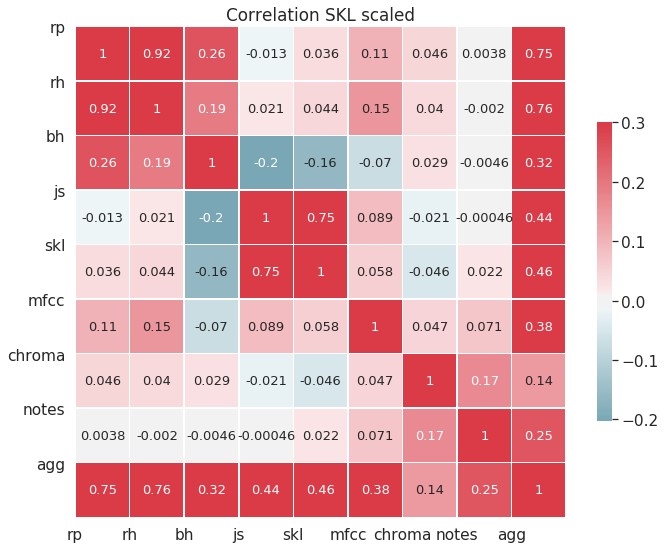
Figure 5.4: Correlation of features depending on SKL scaling

A possible explanation could be that the SKL and JS distances are indeed highly correlated but due to the bad scaling the SKL has no impact on the overall distance. The results from the JS divergence alone are not able to impact the weighted sum of the combined distance in the same way, both features together could.



(

a) SKL unscaled



(

b) SKL scaled

Figure 5.5: Correlation of features

The last plot (figure 5.6) shows the full scatter matrix of the various distances. The main diagonal shows the histograms of the distances from the belonging unique feature-sets.

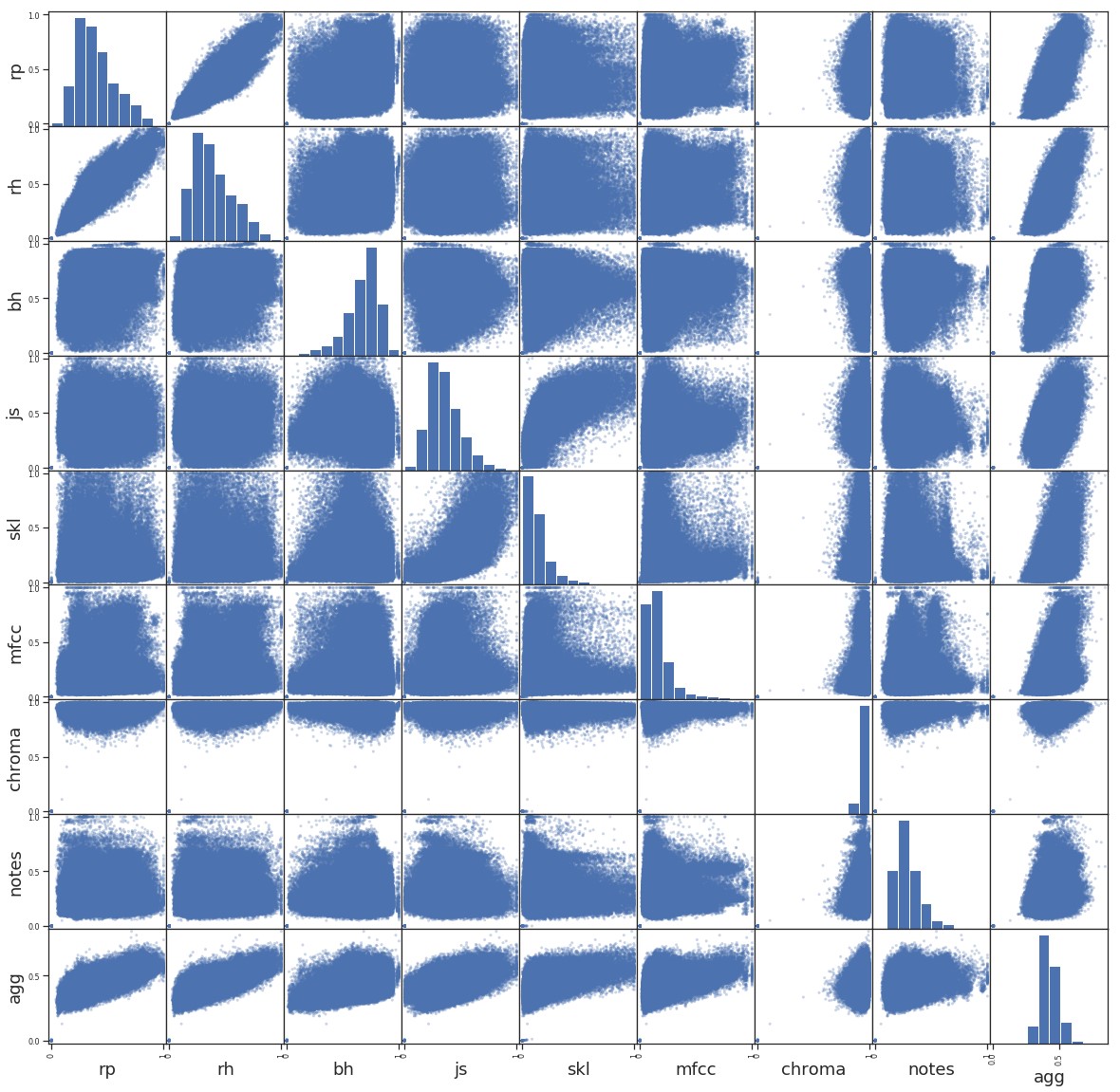


Figure 5.6: correlation 95 songs, 19 genres (5 each), 1517 artists *TODO: describe correlation of melodic features and AGG*

### 5.1.2 Cover song identification

As mentioned in section 3.1.4, purely MFCC based recommender systems lack the ability to detect cover songs. Melody based similarity algorithms like the cross-correlation approach by Ellis and Poliner (see section 3.2.3) and the approach of using the Levenshtein distance by Xia (et al.) in section 3.2.3 on the other hand were especially implemented to detect cover songs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chroma | 30 |  | chroma | 33 |
| chroma + notes | 27 | chroma + notes | 31 |
| chroma + skl | 26 | chroma + notes + rp | 30 |
| chroma + notes + rp | 24 | chroma + skl | 29 |
| chroma + rp | 22 | chroma + rp | 29 |
| chroma + skl + rp | 22 | chroma + skl + rp | 26 |
| chroma + mfcc | 19 | chroma + mfcc + rp | 24 |
| chroma + js + rp | 17 | notes | 23 |
| chroma + js | 17 | all | 23 |
| notes | 17 | chroma + mfcc | 22 |
| chroma + mfcc + rp | 15 | chroma + js + rp | 22 |
| all | 15 | chroma + js | 21 |
| notes + rp | 13 | notes + rp | 19 |
| mfcc + notes + rp | 7 | rp | 15 |
| rp | 7 | mfcc + notes + rp | 14 |
| mfcc + js + skl | 3 | mfcc + js + skl | 10 |

Table 5.1: Cover recognition - Top 1 Table 5.2: Cover recognition - Top 5

Running the first tests on the full dataset consisting of 114210 songs, the Spark implementation was able to find the cover of ”Rock you like a Hurricane” by the Scorpions and covered by Knightsbridge as top recommendation when using the cross-correlation. The application was also able to find an alternative recording of the piece ”Fur Elise”¨ cover as a top recommendation in over 114210 songs, even when using the filter and refine algorithm (starting with chroma features) presented in section 4.2.7.

As a third example the famous ”Rondo Alla Turca (Allegretto)” also known as the

Turkish March by Mozart was tested. This song was also used in chapter 3.1.4 where the ability of the Musly toolkit to detect cover songs was tested. At first a combination of js, chroma and rp features was used. The top five results are listed below.

*Song request: 100 Meisterwerke der Klassik/ Mozart - Alla Turca (Allegretto) (private collection), JS + RP + CHROMA*

* 1) Piano Perlen/ Mozart - Turkischer Marsch (private collection)¨
* 2) FRITZ STEINEGGER - RONDO ALLA TURCA KV 331 (1517 artists)
* 3) 136071 (fma dataset)
* 4) Sean Bennett - Variations on the Turkish March (1517 artists)
* 5) Mozart - Fantasie in D minor (1517 artists)

Two different versions were detected as the top results and the fourth recommendation even listed a variation of the original song theme. Although the private music collection used contains two additional versions of this song (see section 3.1.4), the other versions could not be detected because the rp extract tool failed during the extraction of the features from these songs due to file format issues. So for the second attempt the RP were left out. The six top recommendations are again listed below:

*Song request: 100 Meisterwerke der Klassik/ Mozart - Alla Turca (Allegretto) (private collection), JS + CHROMA*

* 1) Mozart Collection/ CD31/ KV331-3 Alla turca allegretto (private collection)
* 2) Piano Collection/ CD25 - Mozart - Alla Turca Allegretto (private collection)
* 3) Piano Perlen/ Mozart - Turkischer Marsch (private collection)¨
* 4) FRITZ STEINEGGER - RONDO ALLA TURCA KV 331 (1517 artists)
* 5) 136071 (2Kutup - We Shall Cuddle Up And Sleep) (fma dataset)
* 6) Sean Bennett - Variations on the Turkish March (1517 artists)

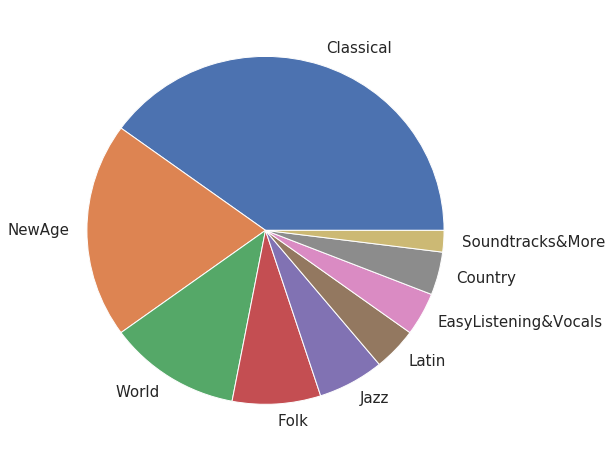
Even if the JS features are left out and only the cross-correlation is computed, the top 6 results don’t change. But on the other hand even when the JS features are included, they aren’t interfering with the cover song recognition. In a third request where only the Jensen-Shannon-like divergence was tested to detect the alternative recordings, the first version only appeared as the 13th recommendation. This confirmed the presumption, that timbral features and the Jensen-Shannon-like divergence and the symmetric Kullback-Leibler divergence are not appropriate for cover song recognition. But there are also song requests where the cross-correlation fails to detect the cover song, one example being the song Chandelier by Sia and the cover version by Pvris that was used in chapter 3.2.2 to explain the computation of the chroma features. To further quantify the ability to detect cover songs after the promising first tests, the covers80 dataset introduced in section 2.4.1 was loaded onto the cluster. The 80 ”A-versions” songs were passed to the Spark application as song requests and the resulting nearest neighbors were analyzed. Table 5.1 counts the appearance of the ”B-version” songs as the first recommendation while table 5.2 lists the count of the recommended cover versions in the top five results of the 80 requested songs when using different combinations of feature sets. As expected the approaches using melodic similarity features perform best. The combination of the different timbre based features performs worst. Surprisingly the distances based on rhythm patterns are also able to detect cover songs.

Although 30 out of 80 on the first doesn’t seem like a surprisingly good hit rate and is not quite as good as the results from the original paper, it has to be mentioned that most of the cover versions in the cover80 dataset differ a lot in musical style, instrumentation, rhythm and even genre from the original recordings. These differences in musical style were also mentioned in the original paper from Ellis and Cottin [66, p. 3]. As an interesting side note it has to be mentioned that the detected cover versions of the ”chroma-” and ”notes-only” requests were mostly the same. Besides two songs, the chroma feature cross-correlation approach detected all of the cover songs that the Levenshtein distance also detected.

### 5.1.3 Genre similarity

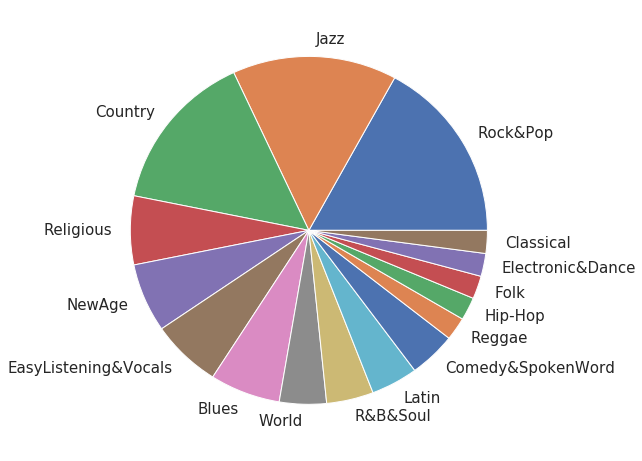
Another way to quantify the quality of the distances and therefor the quality of the music recommendations is to measure the genre recall rate. In a simple test on the 1517 artists dataset, five classical songs are passed to Spark and the nearest neighbors based on the rhythm and timbre features (skl, js, mfcc, rp, rh and bh) are calculated. Then the genres of the top ten recommendations from all five song requests are analyzed.

The result is pictured in figure 5.7a.



(

a) 1517 artist classical recommendations



(

b) 1517 artist rock recommendations

Figure 5.7: genre recall

Although not all recommendations are classical songs, all recommended other genres like New Age, Wold, Folk and Jazz music are somewhat closely related to classical music.

Not a single song from more ”modern” genres like Hip-Hop, Rock & Pop, Electronic & Dance or Reggae appears. The same was tested with five songs from the Rock & Pop genre (see figure 5.7b). The results are scattered across 16 out of 19 different genres from 1517 artists dataset. A possible explanation for this is, that the songs annotated wit ”Rock & Pop” in this dataset actually come from a wider variety of sub-genres. When taking a closer look at the dataset it shows, that e.g. also Metal songs are tagged as Rock & Pop.

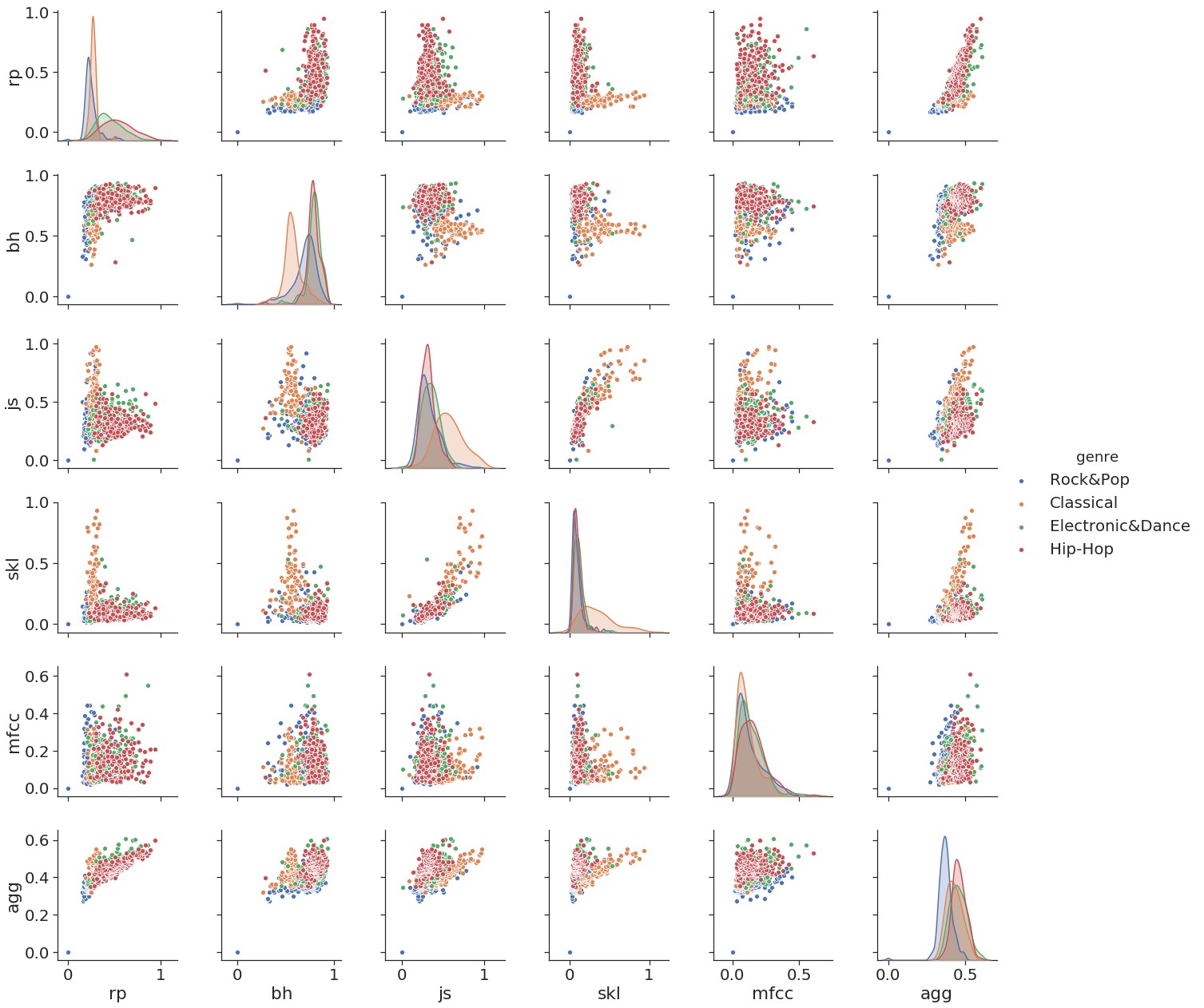


Figure 5.8: distances 1 song, Rock&Pop, 1517 artists, 4 genres

To investigate the impact of different feature types on the overall recommendations and to visualize the distribution of distances for different genres, another test was performed. For single song requests all distances to the songs from a subset of the 1517 artists dataset containing the genres ”classical music”, ”hip-hop”, ”electronic & dance” and ”rock & pop” were computed. Figure 5.8 shows the scatter matrices of all distances from one song request from the genre Rock & Pop. The different distances of the recommendations are colored by the genre of the recommended song. On the main diagonal the Kernel Density Estimation of the respective feature-type is shown. One interesting detail that should be pointed out is that the JS distance alone is unable to distinguish between Rock/Pop songs and Hip-Hop songs but is able to separate between classical music and the rest. On the other hand the rhythm patterns alone can’t separate classical music from Rock/Pop. But when both feature-types are combined all three genres can be separated. The scatter plot of the distances from the Rhythm Patterns and Jensen-Shannon-like divergence in combination shows three clusters of songs belonging to different genres. The fourth genre ”Electronic & Dance” however can not be separated from Hip-Hop songs no matter what feature-set is used. But it has to be kept in mind that all these distances are distances between a song request from the Rock/Pop genre.

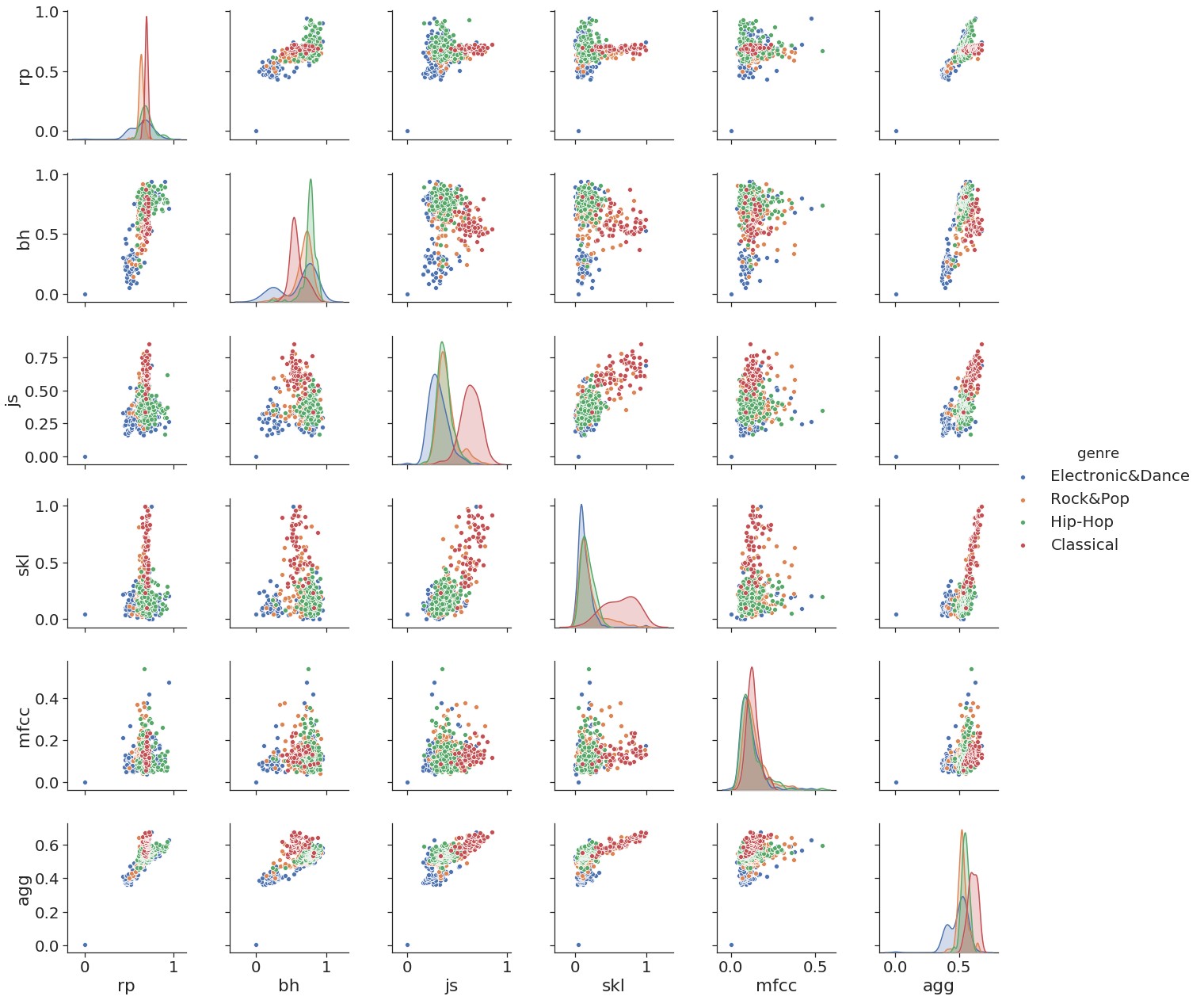


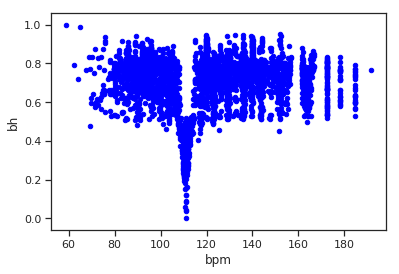
Figure 5.9: distances 1 song, electronic, 1517 artists, 4 genres

As mentioned in section 5.1.1 and visualized in figure 5.1 the distribution of the distances vary depending on where in the feature space the song request is located. Apparently the songs of the Hip-Hop and Electronic/Dance genre are on average all about the same distance away from the requested Rock/Pop song. When requesting a song from the genre Electronic/Dance the distribution of the distances look completely different (see figure 5.9). The ”agg” - plots represent the weighted sum of all features combined (also including cross-correlation and levenshtein distances not shown in the plots). After the combination of all feature-types the returned results primarily recommend other Rock & Pop songs in figure 5.8 and Electronic & Dance songs in figure 5.9.

When using only one feature-set the Spark recommendation engine would not be able to separate all four of the different genres from each other. Only due to the combination of different rhythmic and timbral features an overall satisfying list of recommendations can be returned.

### 5.1.4 Rhythm Features

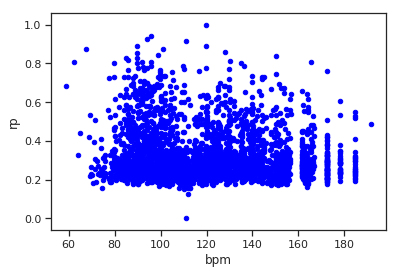
Another critical requirement for the recommendation engine is the ability to return songs that are about the same tempo. To investigate the capabilities of the rhythm features, figure 5.10 shows the resulting distances of two song requests performed on the 1517 artists dataset.



(

a) BPM - BH (Rock&Pop

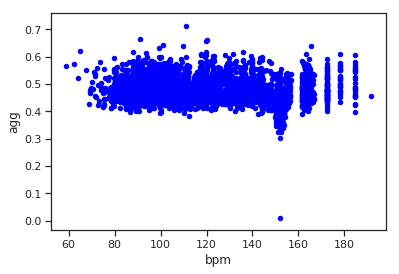
)



(

b) BPM - RP (Rock&Pop

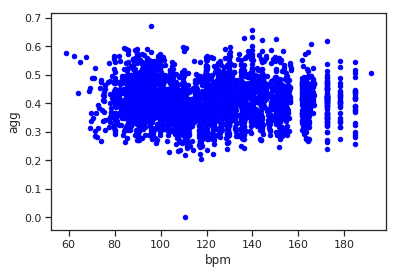
)



(

c) BPM - AGG (Rock&Pop

)



(

d) BPM - AGG (Classical

)

Figure 5.10: Rhythm features/ BPM

The scatter plots show that the beat histogram and the rhythm patterns are closely related to the overall BPM of the songs. The ”agg” value (the weighted sum) includes all eight different feature types so the overall impact of the rhythm features on the recommendations can be seen. All in all the Spark recommendation engine is more likely to recommend songs that have similar BPM when rhythm features are included in the weighted sum. The classical song request in figure 5.10d also shows that the overall distances still are not exclusively dominated by the bpm but rather slightly influenced.

## 5.2 Subjective evaluation

This section includes the personal opinion and music taste of the author. Although these results are not ”scientific”, music taste is something personal and judging music recommendation solely from an objective perspective would be the wrong attempt. The core strength of this Spark-based recommender system is, that its parameters can be used to personalize the music recommendation.

### 5.2.1 Beyond genre boundaries

The main reason for the choice of the topic of this thesis was that recommender systems as they come with streaming platforms like Spotify tend to value. For example the ”Song Radio”- option coming with Spotify stays in the boundaries of genres and is heavily influenced by other peoples listening behavior. Although this is not necessarily a bad thing to do, this thesis tried focus directly on the tonal, rhythmic and properties. As a result, songs from other genres are recommended as can be seen in the following example. When searching for the nearest neighbors of the ”Prelude in C- Sharp Minor (Op. 3 No. 2)” by the Russian composer Sergei Rachmaninoff based on the euclidean distance of MFCCs, the following results were returned:

* Klassik/Rachmaninoff - Piano Concerto No2 In C Minor Op18-1 Moderato
* Klassik/Liszt - Piano Concerto No 1 in E flat major S124(LWH4) Allegro maestoso
* Klassik/Brahms - Piano Sonata No2 in F sharp minor Op2 - III Scherzo allegro
* Metal&Rock/Steve Moore - Intro & Credits
* Klassik/Liszt - Piano Concerto No 1 in E flat major S124(LWH4) Allegro animato

The ”Metal & Rock” recommendation seems out of place at the first glance, but when taking a closer look, the recommended song is called ”Intro & Credits”. When listening to it some similarities are recognizable, it is a calm, dark instrumental piece made of synthesizer sounds. The original requested Prelude is a dark piano piece. Of course this is just one example and the recommendation is arguably not perfect. In general some of the timbre based recommendations seen out of place. This might be due to the choice of 13 MFCC bands over 25 as the musly toolkit uses, or maybe there are some unnoticed issues with the implementation left, which would have to be investigated in future work. But as also stated in section 5.1.3 the overall performance concerning the genre recall rate is reasonably good.

### 5.2.2 personal music taste

As a last side note on personal music taste, a test using one of my favorite songs was made. As already mentioned my private music collection was a part of this thesis. To retain some kind of reproducibility the whole collection is documented and the according list of albums and songs is on a document on the CD in the appendix. On the last pages of this document there is also a list containing my personal song favorites in the metal music genre. One of these songs was chosen and the recommendations based on rhythm patterns was calculated for the private music collection. The song is called ”The Art Of Dying” by the band Gojira. The recommendations are listed below. Another track from my personal list of favorite song appears as well.

* Numenorean - Adore
* Shylmagoghnar - Transience
* Amon Amarth - The Last Stand Of Frej
* Delain - We Are the Others
* Ensiferum - Descendants Defiance Domination

This could be an indication that my personal taste is closely related to the rhythmic properties of the music. An idea for future research could be to reverse engineer a users musical taste by looking at a list of favorite songs. The information which songs a user likes the most is already available to all streaming platforms, because most likely the songs a user listens to the most are also the songs he likes the best. Spark could be used to calculate the similarities between these favorite songs of a user and analyze the distances. Whether or not these songs are more similar in rhythm, melody or in timbre could enhance the parametrization of a recommender engine and further personalize music recommendations by adapting the weights of a recommendation engine. Of course the field of personalized music recommendation is an already existing one, but maybe the addition of Spark and Big Data opportunities of using audio content instead of contextual information and collaborative filtering could enhance these existing system.

# 6. Summary

## 6.1 Conclusion

Looking back at the content of this thesis, the first chapter provided an overview over the field of music information retrieval. Different high- and low-level audio features were explained and various ways to measure the similarities between audio files based on the audio features were introduced. Additionally a short introduction to Big Data frameworks, especially Apache Spark and Hadoop was given and different audio data sources were gathered. The second chapter presented ways to extract and pre-process timbre, rhythm and melodic features from audio files and multiple algorithms for calculating the distances between the extracted features were given. With the theoretic knowledge from the first chapters, the implementation could be planned. The data was collected. Over 1TB of music files containing 114000 different songs were aggregated. In the first part of the implementation the necessary audio features were extracted and pre-processed (e.g. by extractiong the melody estimation) in parallel using MPI on a computer cluster, paving the way for the usage with the Big Data processing framework Spark. The features were loaded into the HDFS of a cluster and multiple similarity measurements were implemented, tested, evaluated and improved using the

Spark framework. With Spark multiple approaches (RDD, DataSet, Filter and Refine, Cluster Configurations) were tested and the runtime was measured. The results of the were presented and visualized.

The final application is able to recommend songs similar to a song request by computing the distances. The recommendations are parameterized, giving the user the option to prioritize different aspects of the music. The system is scalable. More songs can be added, the cluster size can be increased and the possibility to add different kinds of audio features and more state-of-the-art similarity measurements is also given.

## 6.2 Outlook

There are still a few minor flaws, especially when looking at the implementation of the symmetric Kullback-Leibler divergence and the Jensen-Shannon divergence and the scaling of the distances. The different starting points for possible future research were laid out during the whole thesis and are summarized here.

First of all the file format issues with \*.wav audio files when using the rp extract tool from the TU Wien should be fixed to be able to compute all features from all the songs of a dataset (see section 4.1.2).

The next step would be to re-evaluate the Jensen-Shannon-like divergence and the symmetric Kullback-Leibler divergence and fix the issues with the outliers. Also the issues the issues with non-invertible or non-singular covariance matrices should be investigated (see section 4.2.4 and 4.2.4). Maybe also the proposed enhancements by Schnitzer [22] of reducing the hubness with mutual proximity and by using more MFCC bands would be sufficient to improve the quality of the recommendations (see section 3.1.2). Also the scaling of the different features could be improved in a way, where all feature are evenly distributed over the unit interval (see section 4.2.5).

Tests of the performance on larger clusters and with more data would be critical to assess the scaling of the problem. An implementation of the Spark streaming abilities to enable a real-time computation of similarities instead of using batch-processing job would be the next logical step if the goal is to develop a system able to run with music streaming platforms. When evaluating the genre recall rate with Spark, an issue with the garbage collection running out of memory after 40 subsequent song requests was encountered and should be fixed first.

As another way of improving the presented Spark application, more state of the art similarity measurements like block-level features (section 3.1.3) or the TF-IDF weights (section 3.2.3) for melodic similarity could be added. The most promising enhancement for the developed recommendation engine in this thesis would be the addition of genre and metadata information, genre specific features, collaborative filtering and lyrics (see section 2.3.4). Basically all the music contextual information that would normally be processed by a Big Data framework wasn’t included in this thesis but could greatly enhance the recommendations. Most streaming services already have all the information to add available, like users listening behavior or audio metadata and services like Spotify are also already using Spark for collaborative filtering so the Spark application presented in this thesis could be added and integrated into running streaming systems.

A last, but possibly interesting suggestion for future enhancements is to investigate the proposal from section 5.2.2 of personalized music recommendation based on the audio feature similarities of a users favorite songs made available by the Big Data framework.

# References

1. Peter Knees and Markus Schedl. Music Similarity and Retrieval. An introduction to web audio- and web-based strategies. In: Springer, 2016. isbn:

9783662497203.

1. B. McFee et al. The Million Song Dataset Challenge, AdMIRe ’12. In:
2. Brian McFee et al. LibROSA: Audio and Music Signal Analysis in Python. In:

Proceedings of the 14th Python in Science Conference.

1. C. Weihs; D. Jannach; I. Vatolkin; G. Rudolph. Music Data Analysis. New

York: Chapman and Hall/CRC, 2017. url: [https://doi.org/10.1201/9781315370996.](https://doi.org/10.1201/9781315370996)

1. David Moffat, David Ronan, and Joshua Reiss. An Evaluation of Audio Feature Extraction Toolboxes. In: Nov. 2015. doi: [10.13140/RG.2.1.1471.4640.](https://doi.org/10.13140/RG.2.1.1471.4640)
2. B.Mathieu et al. YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In: Proceedings of the 11th ISMIR conference, Utrecht, Netherlands, 2010.
3. D. Bogdanov et al. ESSENTIA: an Audio Analysis Library for Music Information Retrieval. In: International Society for Music Information Retrieval Conference (ISMIR’13). 2013, pp. 493–498.
4. Michael I. Mandel and Daniel P. W. Ellis. Labrosa’s audio music similarity and classification submissions. In: 2007.
5. Jupyter. In: url: [https://jupyter.org/index.html.](https://jupyter.org/index.html)
6. Dr. Dominik Schnitzer. Audio Music Similarity. In: url: [http://www.musly.](http://www.musly.org/index.html)

[org/index.html.](http://www.musly.org/index.html)

1. M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In: Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR, 2005.
2. D. Schnitzer et al. Using mutual proximity to improve content-based audio similarity. In: Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR, 2011.
3. Olivier Lartillot and Petri Toiviainen. MIR in Matlab (II): A Toolbox for Musical Feature Extraction from Audio. In: Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007.
4. MATLAB. in: url: [https://de.mathworks.com/help/matlab/index.html.](https://de.mathworks.com/help/matlab/index.html)
5. John W. Eaton et al. GNU Octave version 4.2.0 manual: a high-level interactive language for numerical computations. In: 2016. url: [http://www.gnu.org/ software/octave/doc/interpreter/.](http://www.gnu.org/software/octave/doc/interpreter/)
6. Martin Ariel Hartmann. A port of MIRToolbox for Octave. In: 2016. url:

[https://github.com/martinarielhartmann/mirtooloct.](https://github.com/martinarielhartmann/mirtooloct)

1. Pr´elude cis-Moll (Rachmaninow). In: url: [https://imslp.org/wiki/Morceaux](https://imslp.org/wiki/Morceaux/_de/_fantaisie/%2C/_Op.3/_(Rachmaninoff/%2C/_Sergei))\ [\_de\\_fantaisie\%2C\\_Op.3\\_(Rachmaninoff\%2C\\_Sergei).](https://imslp.org/wiki/Morceaux/_de/_fantaisie/%2C/_Op.3/_(Rachmaninoff/%2C/_Sergei))
2. Fur¨ Elise. In: url: [https://upload.wikimedia.org/wikipedia/commons/a/ a9/BH\\_116\\_Vergleich.png.](https://upload.wikimedia.org/wikipedia/commons/a/a9/BH/_116/_Vergleich.png)
3. Paul Brossier et al. aubio/aubio: 0.4.8 (Version 0.4.8). In: 2018. url: [http:](http://doi.org/10.5281/zenodo.1494152)

[//doi.org/10.5281/zenodo.1494152.](http://doi.org/10.5281/zenodo.1494152)

1. J. Salamon and E. G´omez. Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics. In: IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770. 2012.
2. C. Cannam, C. Landone, and M. Sandler. Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files. In: Proceedings of the ACM Multimedia 2010 International Conference. Firenze, Italy, 2010, pp. 1467–1468.
3. Dr. Dominik Schnitzer. Dealing with the Music of the World: Indexing Content-Based Music Similarity Models for Fast Retrieval in Massive Databases, 1st ed. In:

2012. isbn: 9781477494158.

1. Matija Marolt. A Mid-level Melody-based Representation for Calculating Audio Similarity. In: ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006. 2006.
2. Jonathan Foote, Matthew Cooper, and Unjung Nam. Audio Retrieval by Rhythmic Similarity. In: Jan. 2002.
3. Zhao Yufeng and Li Xinwei. Design and Implementation of Music Recommendation System Based on Hadoop. In: 2018. doi: [10.2991/icsnce-18.2018.36..](https://doi.org/10.2991/icsnce-18.2018.36.)
4. B. McFee and G.R.G. Lanckriet. Large-scale music similarity search with spatial trees. In: ISMIR ’11. url: [http://citeseerx.ist.psu.edu/viewdoc/summary](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.226.5060)? [doi=10.1.1.226.5060.](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.226.5060)
5. Thierry Bertin-Mahieux et al. The Million Song Dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011). 2011.
6. Guangyu Xia et al. MidiFind: Similarity Search and Popularity Mining in Large MIDI Databases, 259-276. In: 2013. doi: [10.1007/978-3-319-12976-1\ \_17..](https://doi.org/10.1007/978-3-319-12976-1/_17.)
7. Jong Wook Kim et al. CREPE: A Convolutional Representation for Pitch Estimation. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2018.
8. Guangyu Xia et al. MidiFind: Similarity Search and Popularity Mining in Large MIDI Databases. In: Oct. 2013, pp. 259–276. isbn: 978-3-319-12975-4. doi: [10.1007/978-3-319-12976-1\_17.](https://doi.org/10.1007/978-3-319-12976-1_17)
9. Sankalp Gulati, Joan Serra, and Xavier Serra. An evaluation of methodologies for melodic similarity in audio recordings of Indian art music, 678-682. In: 2015.

doi: [10.1109/ICASSP.2015.7178055..](https://doi.org/10.1109/ICASSP.2015.7178055.)

1. Cumhur Erkut et al. Extraction of Physical and Expressive Parameters for Model-based Sound Synthesis of the Classical Guitar. In: 2002.
2. Michael Defferrard et al.¨ FMA: A Dataset for Music Analysis. In: 18th International Society for Music Information Retrieval Conference, 2017. url: [https://arxiv.org/abs/1612.01840.](https://arxiv.org/abs/1612.01840)
3. R. Bittner et al. ”MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research”. In: 15th International Society for Music Information Retrieval Conference, 2014.
4. Soundcloud bqpd. In: url: [https://soundcloud.com/bq\_pd.](https://soundcloud.com/bq_pd)
5. John Thickstun, Zaid Harchaoui, and Sham M. Kakadetitle. Learning Features of Music from Scratch. In: International Conference on Learning Representations (ICLR). 2017. url: [https://arxiv.org/abs/1611.09827.](https://arxiv.org/abs/1611.09827)
6. Klaus Seyerlehner. 1517-Artists Dataset. In: 2010. url: [http://www.seyerlehn](http://www.seyerlehner.info/index.php?p=1\_3\_Download)er.

[info/index.php?p=1\\_3\\_Download.](http://www.seyerlehner.info/index.php?p=1\_3\_Download)

1. D. P. W. Ellis. The ”covers80” cover song data set. In: 2007. url: available: http://labrosa.ee.columbia.edu/projects/coversongs/covers80/.
2. R. Bittner et al. MedleyDB 2.0: New Data and a System for Sustainable Data Collection. In: New York, NY, USA: International Conference on Music Information Retrieval (ISMIR-16). 2016.
3. B. De Man et al. The Open Multitrack Testbed. In: 137th Convention of the Audio Engineering Society, 2014.
4. Spotify API. in: url: [https://developer.spotify.com/documentation/.](https://developer.spotify.com/documentation/)
5. The Echo Nest. In: url: [http://the.echonest.com/.](http://the.echonest.com/)
6. Spotipy - a Python client for The Spotify Web API. in: url: [https://github.](https://github.com/plamere/spotipy)

[com/plamere/spotipy.](https://github.com/plamere/spotipy)

1. Spotify Playlist Miner. In: url: [https://developer.spotify.com/community/ showcase/playlist-miner-spotify/.](https://developer.spotify.com/community/showcase/playlist-miner-spotify/)
2. Spotify Terms and Conditions of Use. In: url: [https://www.spotify.com/ lt/legal/end-user-agreement/plain/#s9.](https://www.spotify.com/lt/legal/end-user-agreement/plain/#s9)
3. Spotify Developer Terms of Service. In: url: [https://developer.spotify.](https://developer.spotify.com/terms/)

[com/terms/.](https://developer.spotify.com/terms/)

1. Spotify Commercial Restrictions. In: url: [https://developer.spotify.com/ legal/commercial-restrictions/.](https://developer.spotify.com/legal/commercial-restrictions/)
2. Hendrik Schreiber. Improving Genre Annotations for the Million Song Dataset. In: In Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR), pages 241-247, Ma´laga, Spain, Oct. 2015.
3. Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset. In: url: [http://labrosa.ee.columbia.edu/millionson](http://labrosa.ee.columbia.edu/millionsong/lastfm)g/ [lastfm.](http://labrosa.ee.columbia.edu/millionsong/lastfm)
4. The Echo Nest Taste profile subset, the official user data collection for the Million Song Dataset. In: url: [http://labrosa.ee.columbia.edu/millionson](http://labrosa.ee.columbia.edu/millionsong/tasteprofile)g/ [tasteprofile.](http://labrosa.ee.columbia.edu/millionsong/tasteprofile)
5. SecondHandSongs dataset, the official list of cover songs within the Million Song Dataset. In: url: [http://labrosa.ee.columbia.edu/millionsong/ secondhand.](http://labrosa.ee.columbia.edu/millionsong/secondhand)
6. Matei Zaharia et al. Apache Spark: A Unified Engine for Big Data Processing.

In: Commun. ACM 59.11 (Oct. 2016), pp. 56–65. issn: 0001-0782. doi: [10.](https://doi.org/10.1145/2934664)

[1145/2934664.](https://doi.org/10.1145/2934664) url: [http://doi.acm.org/10.1145/2934664.](http://doi.acm.org/10.1145/2934664)

1. Apache Hadoop. In: url: [https://hadoop.apache.org/.](https://hadoop.apache.org/)
2. Jeffrey Aven. Data Analytics with Spark Using Python. 1st. Addison-Wesley Professional, 2018. isbn: 013484601X, 9780134846019.
3. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In: vol. 37. Dec. 2003, pp. 29–43. doi: [10.1145/945445.945450.](https://doi.org/10.1145/945445.945450)
4. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In: vol. 51. Jan. 2004, pp. 137–150. doi: [10.1145/1327452.](https://doi.org/10.1145/1327452.1327492)

[1327492.](https://doi.org/10.1145/1327452.1327492)

1. MapReduce. In: url: [https://commons.wikimedia.org/wiki/File:Mapreduc](https://commons.wikimedia.org/wiki/File:Mapreduce.png)e.

[png.](https://commons.wikimedia.org/wiki/File:Mapreduce.png)

1. Butch Quinto. Next-Generation Big Data : A Practical Guide to Apache Kudu, Impala, and Spark. Berkeley, CA: Apress, 2018. isbn: 978-1-4842-3147-0. doi:

[10.1007/978-1-4842-3147-0.](https://doi.org/10.1007/978-1-4842-3147-0)

1. Rau´l Estrada and Isaac Ruiz. Big Data SMACK : A Guide to Apache Spark, Mesos, Akka, Cassandra. Springer Science+Business Media, 2016. isbn: 978-14842-2174-7. doi: [205310.1007/978-1-4842-2175-4.](https://doi.org/2053%2010.1007/978-1-4842-2175-4)
2. Klaus Seyerlehner and Markus Schedl. Block-Level Audio Features for Music Genre Classification. In: 2009.
3. Nicola Orio and Antonio Roda`. A Measure of Melodic Similarity based on a Graph Representation of the Music Structure. In: ISMIR. 2009.
4. David Englmeier et al. Musical similarity analysis based on chroma features and text retrieval methods. In: Datenbanksysteme fur Business, Technologie¨ und Web (BTW 2015) - Workshopband. Ed. by Norbert Ritter et al. Bonn:

Gesellschaft fur Informatik e.V., 2015, pp. 183–192.¨

1. Joan Serra et al. Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification. In: 16 (Sept. 2008), pp. 1138 –1151. doi: [10.1109/](https://doi.org/10.1109/TASL.2008.924595)

[TASL.2008.924595.](https://doi.org/10.1109/TASL.2008.924595)

1. Daniel P.W. Ellis and Graham E. Poliner. Identifying ‘Cover Songs’ with Chroma Features and Dynamic Programming Beat Tracking. In: vol. 4. May 2007, pp. IV–1429. doi: [10.1109/ICASSP.2007.367348.](https://doi.org/10.1109/ICASSP.2007.367348)
2. Mathworks. xcorr2. In: url: [https://www.mathworks.com/help/signal/ ref/xcorr2.html.](https://www.mathworks.com/help/signal/ref/xcorr2.html)
3. Daniel P W Ellis and Courtenay Cotton. The 2007 LabROSA cover song detection system. In: (Jan. 2007).
4. George Tzanetakis and Perry Cook. Musical Genre Classification of Audio Signals. In: IEEE Transactions on Speech and Audio Processing 10 (Aug. 2002), pp. 293 –302. doi: [10.1109/TSA.2002.800560.](https://doi.org/10.1109/TSA.2002.800560)
5. Matthias Gruhne, Christian Dittmar, and Daniel Gartner.¨ Improving Rhythmic Similarity Computation by Beat Histogram Transformations. In: Jan. 2009, pp. 177–182.
6. Thomas Lidy and Andreas Rauber. Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification. In: Jan. 2005, pp. 34–41.
7. Audio Feature Extraction. In: url: [https://github.com/tuwien-musicir/ rp\_extract.](https://github.com/tuwien-musicir/rp_extract)
8. Audio Feature Extraction - Rhythm Patterns. In: url: [http://www.ifs.tuwie](http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html)n.

[ac.at/mir/audiofeatureextraction.html.](http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html)

1. Elias Pampalk. Computational Models of Music Similarity and their Application in Music Information Retrieval. In: 2006.
2. Tim Pohle et al. On Rhythm and General Music Similarity. In: Jan. 2009, pp. 525–530.
3. Johannes Schoder. MusicSimilarity-Spark. In: url: [https://github.com/ oObqpdOo/MusicSimilarity-Spark.](https://github.com/oObqpdOo/MusicSimilarity-Spark)
4. mpi4py. In: url: [https://pypi.org/project/mpi4py/.](https://pypi.org/project/mpi4py/)
5. beegfs. In: url: [http://www.beegfs.io/content/latest-release/.](http://www.beegfs.io/content/latest-release/)
6. Locality Sensitive Hashing. In: url: [https://spark.apache.org/docs/2.4.](https://spark.apache.org/docs/2.4.0/ml-features.html#lsh-operations)

[0/ml-features.html#lsh-operations.](https://spark.apache.org/docs/2.4.0/ml-features.html#lsh-operations)

# 7. Appendix

## 7.1 Feature Analysis

Scatter Matrix, 1 Song (Soundtack) from 50 song sample

Main diagonal = Kernel Density Estimationicked

Subset of 5 genres in figure 7.1

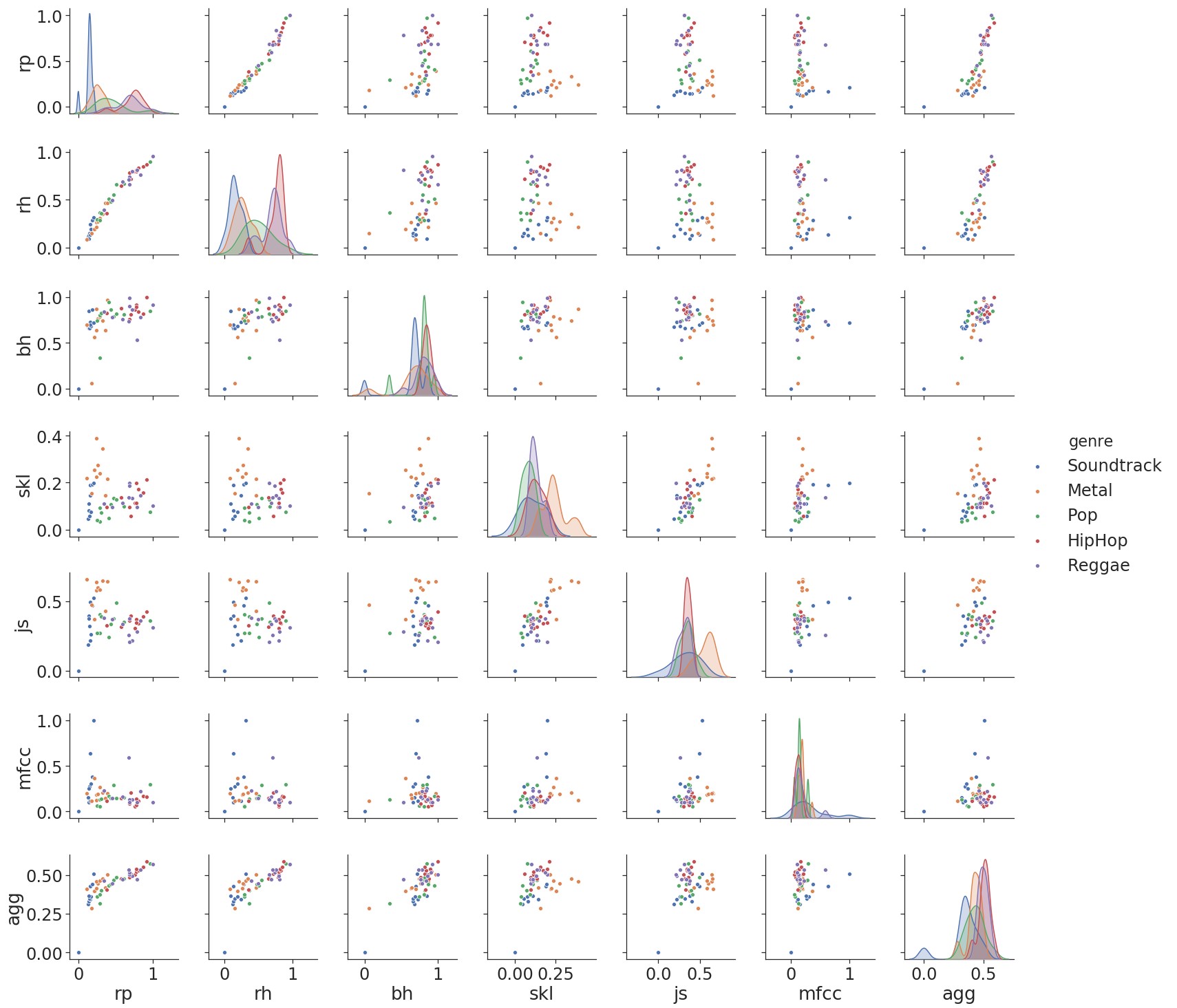


Figure 7.1: distances 1 song (Soundtrack), 5 genres (10 songs each)

## 7.2 Spotipy Data Miner

from \_\_future\_\_ import print\_function from spotipy.oauth2 import SpotifyClientCredentials import json, sys, spotipy, time, os.path import requests, urllib import matplotlib.pyplot as pl import h5json, scipy import numpy as np

from scipy.spatial import distance

reload(sys)

sys.setdefaultencoding(’utf8’)

client\_credentials\_manager = SpotifyClientCredentials()

sp = spotipy.Spotify(client\_credentials\_manager=client\_credentials\_manager) if len(sys.argv) > 1: uri = sys.argv[1]

else:

uri = ’spotify:user:bqpd:playlist:5oF8D71X38WwzeRUdyvpmd’

username = uri.split(’:’)[2] playlist\_id = uri.split(’:’)[4]

playlist = sp.user\_playlist(username, playlist\_id) results = sp.user\_playlists(username, limit=50) playlist\_length = playlist[’tracks’][’total’] path = os.getcwd() path = path + "/crawled\_data" playlist\_name = playlist[’name’] directory = path + "/" + playlist\_name if not os.path.exists(directory):

os.makedirs(directory) t\_start = time.time()

f\_feat = open(path + "/" + playlist\_name + "/featurevector.txt","w") f\_feat.write("Features: \n") f\_feat.close() feat\_vec = [] feat\_num = [] feat\_name = [] for num in range(0, playlist\_length, 100):

results = sp.user\_playlist\_tracks(username, playlist\_id, limit=100, offset=int(num

)) tracks = results for i, item in enumerate(tracks[’items’]):

track = item[’track’] track\_id = str(track[’id’]) path = os.getcwd()

path = path + "/crawled\_data" artist = str(track[’artists’][0][’name’]) songtitle = str(track[’name’]) artist = artist.replace("/", "") songtitle = songtitle.replace("/", "") artist = artist.replace("$", "") songtitle = songtitle.replace("$", "") number = i + num

name = str(number) + " - " + artist + " - " + songtitle directory = path + "/" + playlist\_name + "/" + name prev\_url = track[’preview\_url’] if not prev\_url == None:

if not os.path.exists(directory):

os.makedirs(directory)

filename = directory + "/" + artist + " - " + songtitle + ".mp3" urllib.urlretrieve(prev\_url, filename) tid = ’spotify:track:’ + track[’id’] analysis = sp.audio\_analysis(tid) with open(directory + "/" + songtitle + ’\_analysis.json’, ’w’) as outfile:

json.dump(analysis, outfile)

outfile.close() segments = analysis["segments"] bars = analysis["bars"] beats = analysis["beats"] tid = str(tid)

features = sp.audio\_features(tid) with open(directory + "/" + songtitle + ’\_features.json’, ’w’) as outfile:

json.dump(features, outfile)

outfile.close()

acousticness = features[0][’acousticness’] danceability = features[0][’danceability’] energy = features[0][’energy’]

instrumentalness = features[0][’instrumentalness’] liveness = features[0][’liveness’] loudness = features[0][’loudness’] speechiness = features[0][’speechiness’] valence = features[0][’valence’]

feat\_vec.append(scipy.array([acousticness, danceability, instrumentalness,

liveness, loudness, speechiness, valence])) else:

print("no url - entry: " + artist + " - " + songtitle) print(track\_id + "\n")

t\_delta = time.time() - t\_start

print ("features retrieved in %.2f seconds" % (t\_delta,)) dist = distance.euclidean(feat\_vec[0], feat\_vec[1])

## 7.3 Contend of the CD

#### Code

* mpi4py ara features
* mpi4py ara files
* mpi4py ara rhythm
* ...

**...**

PDF private music collection

...

**Declaration of Authorship**

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. This thesis was not previously presented to another examination board and has not been published in German, English or any other language.

Jena, August 26, 2019

Johannes Schoder