A study on the automatic classification of songs into Spotify playlists based on different datasets

Guillermo Ventura,
Technical University of Munich,
Chair for Data Processing,
Department of Electrical and Computer Engineering,
Munich, Germany, gvm104@gmail.com

April 6, 2020

Abstract

Listening to music has become a prominent part of people's life. Due to infinite cloud storage and instant content streaming, the number of songs users have in their library is ever increasing. This makes organizing a music library a difficult task. Since the arrival of portable music players, playlists have been used to ease the music listening experience. In Music Streaming Services (MSS) such as Spotify, playlists are either created by professional playlists curators or by the users themselves. There is a need to organize these large libraries into automatic generated playlists to make the overall listening experience in MSS more pleasurable. Some MSS automatically curate a large set of playlists to suit users' tastes, moods, activities etc. In this reasearch different datasets and machine learning approaches are used to automatically classify songs into a given set of playlists.

Keywords: ***Machine Learning, Ensemble Learning, Voting Classifier, Spotify, Playlists***

1 Introduction

The amount of music society consumes has been increasing exponentially in the last 20 years due to the internet. This increment has been particularly notable in the last years with the appearance of Music Streaming Services (MSS) such as Spotify, Tidal, Apple Music etc. Spotify counts as one of the most successful MSS, with over 270 Million monthly active users in December 2019[1].

This is why MMS platforms such as Spotify have become crucial for music distribution. Record labels and artists do not intend to sell thousand of albums, instead their aim is to reach as many users as possible on online streaming services.

Spotify's music library is over 35 million songs big. It is not unusual that a Spotify user goes to the searchbar and has to stop for a few seconds to think what song/album/artist do they really want to listen to. More than 4 million of the songs on Spotify have never even been played[2]. This is an effect of The Long Tail described in 2004 by Chris Anderson [1]. The best example of how music can be lost in The Long Tail is shown in the documentary "Searching for Sugar Man". This documentary tells the story of Sixto Rodriguez, an artist from Detroit who in the seventies released two albums. These albums never made it to any radio station in the United States, but without him (or anyone else in the US) knowing, his music became extremly popular in South Africa. It is in Spotify's interest that good music does not vanish in the Long Tail like Rodriguez. That is why Spotify fights this effect with playlists.

There are over 3 billion playlists in Spotify [3]. These are either created by users or by Spotify itself. Playlists personally curated for an user (*"Specially for you"* in Spotify) are not taken into account here, since these use data about the users streaming behaviour. Some playlists are curated by

[1] https://newsroom.spotify.com/company-info/
[2] https://gizmodo.com/more-than-4-million-spotify-songs-have-never-been-playe-1444955615
[3] https://review42.com/spotify-statistics/

playlists curators, but others are mantained and updated by machines.

Users can subscribe to playlists (by pressing the "follow" button) and this way, they will add that playlist to their library. These playlists tend to change daily, weekly or monthly, and have nowadays adopted the role of radio stations in the past: Users tune in regularly to check what new music the playlist they are subscribed to has added. The most subscribed playlist on Spotify is "Today's Top Hits" by Spotify, with 26 million followers. Songs get added to this playlist every week and change position on a daily basis. The position of the song in this playlist represents the popularity of the song in Spotify. Spotify pays approximately $0.004 royalties per reproduction, which is not a large quantity. Although, if a song gets featured in a playlist with a large audience, such as "Today's Top Hits", the amount of reproductions that song has, will rapidly increase.

"I saw the song go from 8 million streams
to over 100 million streams, it's insane."
– Quote from artist "Lauv", when 2 years after the release of his single "The Other" the song was featured in "Today's Top Hits" [4].

Artists need to label their music as a specific genre when they upload it to Spotify. This label will be crucial for the distribution of the songs. Based on this, the song will get added to playlists with similar content. Music nowadays evolves at a hard pace, and it is difficult to categorize some songs to be a specific genre. Some artists, like Grimes are even told that their contant does not fit into any existing format or genre and can therefore not be featured on playlists[5]. This makes it hard for artists to promote their music. Upcoming artists typically have to contact YouTube channels, bloggers, radio station DJs and also playlists curators to get their music *out there*. Artists would benefit of a way to automatically filter playlists where their music fits. This would speed the process of distributing new songs and has also the potential to directly reaching the listeners library.

A person may listen to a playlists and be able to easily tell which song would fit that playlists. In this research, possible approaches to the automatic classification of songs in playlists are presented. Different feature sets will be explored in section 3 Datasets. Diverse machine learning approaches will be discussed in section 4 Models and the results of my experiments will be presented in section 5 Results.

[4] https://www.rollingstone.com/music/music-news/how-spotify-playlists-create-hits-200277/

[5] https://twitter.com/Grimezsz/status/1204515163438645249?s=20

2 Related Work

In the past years, several studies have been conducted in the music information retrieval field specially focused on playlists.

In a study conducted by the Univesity of Modena [2], the authors present a system that automatically generates user-tailored and time-sensitive playlists which suit the users streaming habits. Low-level audio descriptors from songs contained in the user's listening history are input to two clustering algorithms which generate different playlists. These playlists contain known songs as well as undiscovered songs to the users.

In 2019, I. Rosilde Tatiana *et.al* in "Automatic playlist generation using Convolutional Neural Networks and Recurrent Neural Networks" [3], proposed a model able to generate or continue a playlist based on a CNN and a RNN. Here, the mel-spectogram of the audio files is given to a five-layer CNN (which was proved to effectively predict mood, genre and audio events [4]). Then, a RNN orders the songs in the playlist based on the evolution over time of its feature representation. The authors evaluated their approach based on The Art of Mixing [5] under two scenarios: playlist generation with hidden songs, playlist continuation with seed songs.

On my previous work [6], MIR audio features were first computed. Then, these features were utilized to calculate a higher level feature set. This feature set was composed of eight weighted genre labels (computed using a self established database, labels and a Support Vector Machine) and the detected emotion of the songs (based on a statistical approach). Then a SVM was trained with all low level and high level features to categorize songs into certain playlists. This research only took into account audio data. In the hereby presented approach to the playlist classification problem, further datasets and machine learning models, as well as possible options to combine these will be studied.

3 Datasets

3.1 Playlists

During the time of this research, I periodically checked the status of several Spotify playlists. The Spotify API was used to request the playlists status[6]. These playlists were managed either by Spotify itself (divided by country, e.g.: *"Spotify UK"*), by playlist curators (like *"Chillhop Music"*) or by record labels (*SubPop Records*, *WarnerBros Music*). After inspecting several playlists I decided to have a closer

[6] https://developer.spotify.com/documentation/web-api/

look at the most popular playlists of each user. A list of all the analyzed playlists can be found in the GitHub[7].

Approximately $\frac{2}{3}$ of all Spotify songs have a 30-second-long `.mp3` preview of the track available online. The artist has the right to not allow the distribution of this 30-second long preview. This happens in approximately $\frac{1}{3}$ of all the cases. I temporary downloaded the available files. This allowed me to analyze the audio of the songs by extracting the audio features offline.

3.2 Audio Feature Set

Audio features are acoustic descriptors that can be extracted from audio signals with signal-processing tools. The motivation for the audio features extraction is to be able to mathematically describe audio. Audio features have been proven to work for voice detection and recognition [7], and are also widely used in the Music Information Retrieval (MIR) field. A lot of research has been conducted in the last years in this field. The manner of extracting the audio features may vary depending on the approach used. In Figure 1 the mechanism for extracting audio features with the Matlab MIRToolbox is depicted.

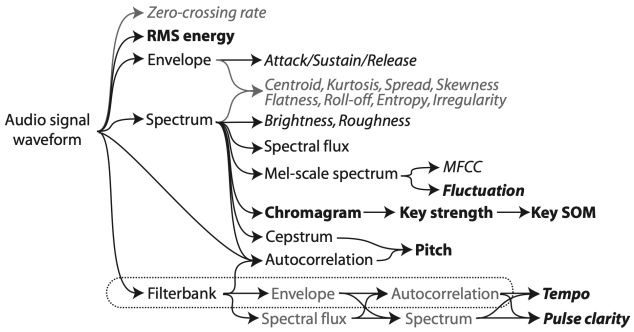


Figure 1: Audio Feature Extraction in the MIR Matlab Toolbox [7]

This toolbox developed by the University of Jyväslylä in Finland is based on a modular structure, where different algorithms can be decomposed and manipulated. This toolbox, together with the AuditoryToolbox (ver. 2) and the Signal Processing Toolbox (ver. 7.5), extracted 35 audio features from the audio files. The MATLAB audio feature extraction process is automatized to a daily routine. The extracted audio features are described in table 1 divided in 4 categories: Timbre, Dynamic, Rhythmic and Tonality Features.

These are 21 different low level audio features. Since the use of mean and standard deviation of the features had been shown to positively influence the outcome of the classifiers [8], I decided to add these to the total number of audio features.

Table 1: Computed Audio Features

Feature Set	Numb.	Features
Timbre	13	spectral centroid, rolloff (85) mean + std, brightness mean + std, spectral flux mean + std, MFCC mean + std, roughness mean + std, cepstrum mean + std
Dynamic	5	RMS Energy std + mean, zero-cross-rate, lowenergy, ASR
Rhythmic	7	tempo, pulse clarity, event density, onsets mean + std + peak pos. + peak val.
Tonality	10	key, key mode, key strength std + mean, pitch mean + std, chromagram std + mean, tonal centroid std + mean

Of the total 35 Features, 13 belong to the spectral (*Timbre*) feature set. These were: spectral centroid and the mean and standard deviation of rolloff (85), brightness, spectral flux, MFCC, roughness, and cepstrum.

The dynamic feature set was formed by 7 features which represented the energy of the audio signal: lowenergy, zero-cross-rate, average silence ratio (ASR) and the mean and standard deviation of the RMS Energy (Root-Mean-Square Energy) and of the envelopes of the music audio signal.

The third feature set, the rhythmic features, was formed by: tempo, pulse clarity, onsets (mean, standard deviation, peak position and peak value of the onset) and the event density.

The last set of features represented the tonality of the music audio signals. This features set was composed by: key, key mode, tonal centroid and the mean and standard deviation of the key strength and of the chromagram.

One must note that these features were extracted from a 30 second-long part of the song. These 30 seconds given by the Spotify preview also seem to be picked from a random moment of the song. Due to this, in some cases, the 30 second-long preview would not reflect the overall ambience of the songs. This limitation was (to some extend) overcomed with the Spotify Feature Set.

3.3 Spotify Feature Set

The Spotify API also provides a very useful feature set. All the features taken from the Spotify API for this investigation are described in Table 2.

[7] https://github.com/GuilleVENT/Auto-PL-classification/blob/master/All-PL.pdf

Some of these features are low level audio descriptors (like *Loudness*), but the Spotify API also provides other, less common, features. These are features like *Energy, Danceability, Instrumentalness, Liveness, Speechiness*. The Spotify API documentation barely defines how these features are computed. It defines the perceptual features that contribute to the *Energy* feature extraction as: dynamic range, perceived loudness, timbre, onset rate, and general entropy. *Danceability* is computed from a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. These features, mostly `floats` between 0.00 and 1.00, are probability scores. The features provided by the Spotify API should be taken into consideration with a grain of salt, since there is no empirical way of reproducing these values. It makes sense to add this feature set since Spotify most probably uses that same feature set to automatically curate machine-made playlists. Spotify automatically generated playlists (like *Daily Mix, Specially For You*) are extraordinarly good understanding users preferences. Training a model with the same feature set might allow me to reproduce a similar behaviour. This feature set is also used in nearly all research projects done in the Spotify platform [2; 9; 10]. These features are readily available for all songs in the Spotify library.

3.4 Lyrics Feature Set

I decided to also add one more feature set describing the songs lyric content. This would give me more insights on the songs subject matter. I used the lyrics from the Genius website[8] since Spotify and Genius are currently partners and provide realtime lyrics in the Spotify app. I used web-scraping tools and the Genius API to get the lyrics of the songs. If the lyrics of the songs were available in Genius, these would be downloaded into a `txt`-file and later analyzed. It is important to not that some songs, do not have lyrics. This is specially notable in relaxing playlists like *Peaceful Piano* or *Yoga & Meditation*. These features will be left blank in such cases.

Some of lyrics features describe the overall structure of the song, these are: average lines per stanza, average line length, longest line length, shortest line length, maximal number of lines per stanza, minumum lines per stanza, number of repeated stanzas, total number of words, porcentage of repeated words and the total number of stanzas. I expected the model to understand the songs' structure in more detail. These feautes vary between genres (*e.g.* rap music tends to have more words and longer stanzas than, say blues). My assumption was that training the models with these may improve the playlist classification accuracy.

Table 2: Spotify Features

Name	Description
Duration	The duration of the track in milliseconds.
Key	The estimated overall key of the track. Standard Pitch Class notation
Mode	Mode indicates the modality (major or minor) of a track.
Time Signature	The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
Danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements.
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
Instrumentalness	Predicts whether a track contains no vocals. The likelihood the track contains vocal content.
Liveness	Higher liveness values represent an increased probability that the track was performed live.
Loudness	The overall loudness of a track in decibels (dB).
Speechiness	Speechiness detects the presence of spoken words in a track.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
Tempo	The overall estimated tempo of a track in beats per minute (BPM).

A bag-of-words approach was used to define 5 types of word-bags: happy words (words that denote happyness), rock words (popular words in rock songs), pop words (popular words in pop songs), slang and swear words (a collection of words normally used in HipHop/Rap music) and stop words. These collections of words can be visited in the GitHub[9].

Genius also describes all the stanzas (of most songs) with a label *e.g. [Verse 1: Kendrick Lamar]*. The singer of each stanza is always defined in this label. When more than one author sings a stanza, this will be signalized in this label as well. This allowed me to include a "Duet" feature

[8] http://www.genius.com

[9] https://github.com/GuilleVENT/Auto-PL-classification/tree/master/dictionary

to the lyrics feature set. This bool variable defined when there was more than one singer present in a given song.

The language of the song (and if the song was sung in more than one language) was also part of the lyrics feature set. This will be very important when classifying playlists from specific countries, like *e.g.* user: *"spotify_france"* playlist: *"Fresh Rap"*. We will see later that the language of the lyrics plays a major role when classifying songs in popular Spotify playlists.

4 Concept

4.1 Support Vector Machines (SVM)

SVMs were first introduced by Vladimir N. Vapnik in 1995 [11], and have nowadays become one of the most popular machine learning models when working with small to medium datasets. SVMs use optimal hyperplanes to classify elements in a feature space. SVM have been proven to work very accurately when using audio features to classify music genres [12; 13] and emotions [14]. In the past, I had used SVMs to succesfully classify eight genres: rock, pop, country-folk, metal, reggae-ska, disco-house-techno, hiphop-rap and classical with a classification accuracy varying between 65% and 72.5%. The SVM had also proofed to be effective when classifying songs to playlists. More information on the classification of songs in playlists through a SVM can be found in experiment 1 and in [6].

4.2 K-Nearest Neighbors (kNN)

In 1951, Fix and Hodges [15] introduced a non-parametric method for pattern classification that has since become known the k-nearest neighbor rule [16]. K nearest neighbors store all available information of previous cases and classifies new cases based on a similarity measure (*e.g.,* distance functions: Euclidian, Manhattan, Minkowski, Hamming). KNN models have been used since the 1970's for statistical estimation and pattern recognition. These models have also shown positive results in other MIR scenarios [17; 2]. Generally, kNN work good in adapting and changing scenarios.

4.3 Random Forest (RF)

Random Forests are an ensamble learning method for classification or regression. In these models, the feature space is partitioned multiple times based on a series of rules. These models are specially efficient due to their speed, simplicity and the ability to automatically detect relevant features. There are certain features that are key to creating a cohesive playlist, like: Tempo, Language, Danceability *etc.*. This made me think the RF was apt for this classification task.

4.4 Ensemble Models

Furthermore, I trained three other different types of ensamble models: AdaBoost, Bagging and Majority Voting. Ensemble learning approaches, generally speaking, do not intend to train the best model to better understand the data, instead they build a combination of different models and then decide between them trough e.g.: majority voting. They are also referred in the literature as meta-learners or meta-algorithms. Ensemble approaches, have normally better predictive performance than standalone learning algorithms.

The AdaBoost algorithm is an iterative procedure that combines many weak classifiers. Starting with the unweighted training sample, the AdaBoost builds an initial classifier, that produces class labels. If a training data point is misclassified, the weight of that training data point is increased (boosted). A second classifier is built using the new weights, which are no longer equal. This is recursively repeated. Then, each classifier has a score, and the final classifier is defined as the linear combination of the classifiers from each stage [18]. A connection has been made between suppport vector machines and boosting methods for classification [19]. In this approach the initial classifier for the AdaBoost and Bagging classifier is a SVM.

Bagging (Bootstrap aggregating) was proposed by Leo Breiman in 1994 [20] as a method of improving classification accuracy by combining classifications of randomly generated training sets. It aims to produce an ensemble model that is more robust than the individually trained models it is compound of. The final meta-prediction is then based on all the predictions of the individual model.

I also wanted to combine the feature sets and models to see which combination of these would give the best classification results. For this I developed a majority voting classifier with either two or three voters. The feature sets presented in section 3. were used to train individual machines (SVM, kNN and RF). The groups of classifiers, democratically voted what song would fit more into what playlists. The potential of training such an ensemble model is promising, since some machines seem to have very good accuracy results when trained with certain features and playlists. The results of this approach will be discussed in experiment 3.

5 Results

5.1 Experiment 1.-

In this first experiment, the accuracy results of the SVM when using the different feature sets is compared. Here, the amount of playlists used for training is stepwise increased. The data was first processed through a singular value decomposition and a principal component analysis. The features were also scaled using a standard scaler. The pre-processing impacted the computation time positively. The parameters used for the SVM and the size of the reduced feature sets would vary depending on the playlist set used for training. A 5-Fold Cross-Validation was performed before training. The playlist depicted in Table 3 were picked to study the classification accuracy. These playlists were selector since they seem to span the feature space extensively. These playlists contain songs with different genres, languages, moods etc. Some playlists do contain similar songs (like *"Massive Dance Hits"* and *"Mint"*, or like *"¡Viva Latino!"* and *"Mint Latin"*). To keep a perspective on the audiences of each playlists, the amount of followers each playlist has is also depicted in Table 3.

Table 3: Playlists used for Experiment 1.

	Playlist	User	Followers
1	**Peaceful Piano**	spotify	5.8 M
2	**Mint**	spotify	5.6 M
3	**Rock Classics**	spotify	7.5 M
4	**¡Viva Latino!**	spotify	10.5 M
5	**Pop Remix**	spotify_germany	3.1 M
6	**Techno Bunker**	spotify_germany	3.1 M
7	**Fresh Rap**	spotify_france	600 k
8	**Massive Dance Hits**	spotify_uk_	1.1 M
9	**Yoga & Meditation**	spotify_uk_	1.3 M
10	**Mint Latin**	spotify_españa	1.8 M

In Figure 2, the accuracy results of the SVM when classifying the validation set is plotted in the y-axis. The red line represents the classification accuracy when the model were trained with the Audio Features computed using the MIR Matlab Toolbox. The green line defines the accuracy of the classifier when trained using the Spotify API feature set. The blue line shows the accuracy when using the lyrics feature set and the black line depics the accuracy when using all feature sets combined. The playlist numeration is in accordance with the indexing in Table 3.

Training the support vector machine using the audio feature from the 30-second long preview seems to give very good results when using a small playlist set (less than 6 playlists). More experiments of the SVM using these audio features are presented in [6]. It is interesting to note

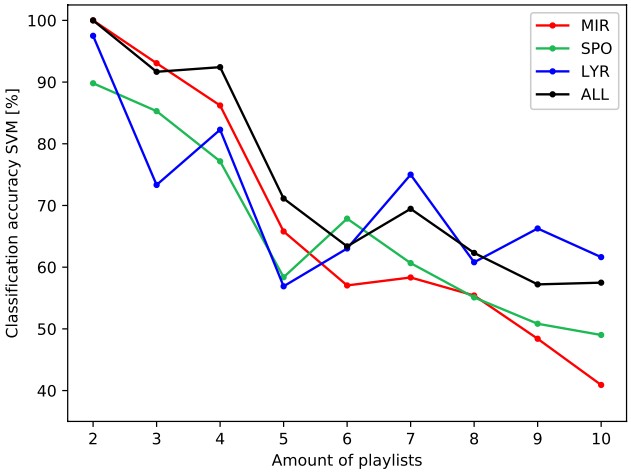


Figure 2: Classification accuracy of the SVM trained using the playlists from Table 3

that when the SVM was trained with the lyric feature set, the classification accuracy increases notable with playlists sung in different languages. This is the case in playlist 4. "¡Viva Latino!" (spanish) and in playlist 7. "Fresh Rap" (french). The lyrics features, although being rudimental, seem to provide the highest accuracy results when the SVM was trained with a high number of playlists. The SVM trained with the Spotify feature set seems to be effectiv when classifying dance playlists (see playlist 6. Techno Bunker), but not so on Pop/Rock playlists (playlists #3, #4 and #5). The accuracy results of the SVM trained using all three feature sets combined, gives overall very good results.

5.2 Experiment 2.-

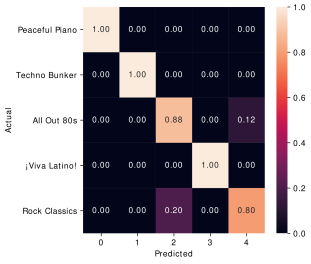
In the second experiment, the playlist used for training were: Peaceful Piano, Techno Bunker, All Out of the 80's, ¡Viva Latino! and Rock Classics. Here, the relevance of performing a principal component decomposition in the different datasets and models is studied. The accuracy results of the different models trained with the various features sets (scaled and with/without a previous PCA) is depicted in Table 4:

In this experiment, the PCA reduced the Spotify features set from 12 features to two principal components. The lyrics and audio feature sets were reduced to eight principal components each and the combined feature set was reduced to eleven features. It is interesting to note that the PCA didn't increase the validation accuracy in any of the models. Only on the lyrics feature set the validation accuracy stayed practically equal with and without a PCA. On the Spotify and audio feature sets the validation accuracy throughout most models decreased by $\sim$10% when

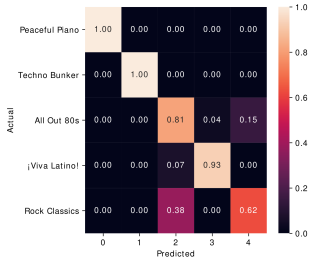
Table 4: Validation accuracy of the models trained with different datasets

	Spotify		Audio		Lyrics		All	
PCA	No	Yes	No	Yes	No	Yes	No	Yes
SVM	91.2%	75.6%	90.8%	79.3%	78.2%	76.8%	92.3%	82.3%
RF	88.7%	71.3%	86.2%	80.5%	76.8%	72.4%	78.9%	76.9%
kNN	88.7%	67.5%	85.0%	73.5%	75.3%	73.9%	86.5%	73.0%
Bagging	90.0%	76.8%	86.2%	78.2%	76.8%	75.4%	92.3%	88.5%
AdaBoost	86.4%	75.6%	87.3%	72.4%	73.9%	73.9%	84.6%	80.7%

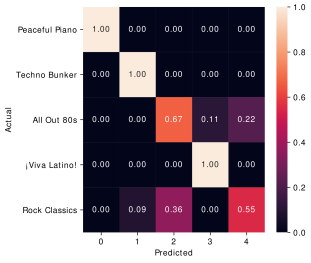
pre-processed through a PCA. This decrease in accuracy when using a PCA might be due to some features in the dataset only being relevant for the classification of a particular playlist, and missing that feauture leads to information loss. The playlists used for this experiment are not sung in a particular lenguage. Morover, most of the songs in Techno Bunker and Peaceful Piano do not have lyrics. This made classification when only using the lyrics feature set a hard task. The combined feature set (All) does not take into account songs with missing Audio or Lyrics data. The best accuracy results (marked above in green) were given by the SVM and the Bagging classifier.



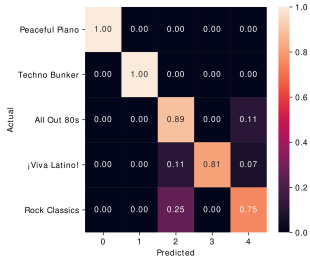
(a) Bagging - trained with all features



(b) Bagging - trained with Spotify features



(c) SVM - trained with audio features



(d) SVM - trained with Spotify features

Figure 3: Confusion Matrices of the best classifiers from Table 4

In Figure 3 the Confusion Matrices of the classifiers with the highest validation accuracy (Bagging and SVM) are shown. There seems to be some confusion between "Rock Classics" and "All Out of the 80's". No song in these playlists is present in both. However, the classifiers seems to find a relationship between these playlists. The sound of the 80's is closely related to the classic rock sound of the 60's and 70's. Arguably, a person might also find it difficult to distinguish a song from the 80's from a classic rock song.

5.3 Experiment 3.-

In the third experiment, the same playlists as in the previous experiment was used. No PCA was used in this case since it does not seem to increase accuracy results. Here, the aim was to developed a voting classifier. This was done twice, once with two and once with three voting classifiers. These models (trained with either the audio, Spotify or lyrics data) would vote to classify what song fit what playlists the most. The results of this democratic approach to the playlist classification problem were quite good. In total, 119 voting classifiers were trained, 36 with two voters and 83 with three voters. This accounts to all the possible combinations of models and data sets. The speed to train these classifiers was also quite remarkable when compared to the AdaBoost and Bagging classifiers training speed.

With two voters, 18 classifiers produced validation accuracy beyond 90%. Surprisingly the SVM was only present in six of these 18 voters and was always trained with the lyrics dataset. The kNN was present in 14 out of the 18 voters that performed over 90% validation accuracy. The RF voted on 13 of the best 18 classifiers. The dataset used the most in these voting classifier was the lyric set. The best validation accuracy (96.2%) was voted by a kNN trained with the Spotify data (with a test accuracy of 84%) and a RF trained with the lyrics data (with a test accuracy of 76%).

Out of the 86 voting classifiers with three voters, 13 voting classifier had over 96% validation accuracy. The SVM trained with the lyrics dataset voted in five out of these 13 voting classifiers. The kNN was present in all of these voting classifiers except for a voting classifier fully composed by RFs trained with the three datasets. The lyrics and Spotify datasets were the most prominant in the 13 voting classifiers with over 96% accuracy.

The confusion matrices of the voting classifiers with the best classification results show that these models have the same level of confusion distinguishing songs from "All out of the 80's" and "Rock Classics". More detailed information on these voting classifiers can be seen in GitHub[10]. Overall, the results show improvement when the models are trained with the training sets individually as oposed to a combined feature set.

[10] https://github.com/GuilleVENT/Auto-PL-classification/tree/master/CLF_results/Exp_3

6 Conclusion

In this study, possible approaches to the automatic classification of songs in playlists have been presented. Different feature sets as well as machine learning models have been considered. In the first experiment, it was proved that the amount of playlists used for this classification task is strongly correlated to the overall accuracy of the model. However, using certain feature sets for the classification of specific playlists (like the lyrics feature set on playlists sung in a particular language) was proven to positively impact the accuracy of these classifiers. In experiment 2. the accuracy of these classifiers was benchmarked. Here, it was shown that a feature set reduction actually decreases the classification accuracy in certain cases. In the last experiment, a ensemble voting classifier was put to test. Here, separating the features sets and giving these to different models was proven to be beneficial. The computing time as well as the overall accuracy was positively impacted.

References

[1] C. Anderson and M. P. Andersson, "Long tail," *Bonnier fakta*, 2004.

[2] M. Furini, J. Martini, and M. Montangero, "Automated generation of user-tailored and time-sensitive music playlists," *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019.

[3] R. T. Irene, C. Borrelli, M. Zanoni, M. Buccoli, and A. Sarti, "Automatic playlist generation using convolutional neural networks and recurrent neural networks," *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019.

[4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *arXiv preprint arXiv:1703.09179*, 2017.

[5] D. Gibson, "The art of mixing: A visual guide to recording," *Engineering, and Production*, vol. 236, 1997.

[6] G. Ventura, "An audio feature based approach to reach prediction in spotify," *Technical University of Munich*, 2018.

[7] O. Lartillot and P. Toiviainen, "A matlab toolbox for musical feature extraction from audio," *International conference on digital audio effects*, pp. 237–244, 2007.

[8] Y. Song, S. Dixon, and M. Pearce, "Evaluation of musical features for emotion classification." *ISMIR*, pp. 523–528, 2012.

[9] M. Bakhshizadeh, A. Moeini, M. Latifi, and M. T. Mahmoudi, "Automated mood based music playlist generation by clustering the audio features," *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 231–237, 2019.

[10] T. S. Bohra, V. Kumar, and S. Ganesan, "Segmenting music library for generation of playlist using machine learning," *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pp. 421–425, 2015.

[11] V. Vapnik, "The nature of statistical learning theory," *Springer science and business media*, 1995.

[12] D. P. Kumar, B. Sowmya, K. Srinivasa *et al.*, "A comparative study of classifiers for music genre classification based on feature extractors," *IEEE*, pp. 190–194, 2016.

[13] M. Haggblade, Y. Hong, and K. Kao, "Music genre classification," *Department of Computer Science, Stanford University*, 2011.

[14] S. Pouyanfar and H. Sameti, "Music emotion recognition using two level classification," *2014 Iranian Conference on Intelligent Systems (ICIS)*, pp. 1–6, 2014.

[15] E. Fix and J. Hodges, "Discriminatory analysis, nonparametric discrimination," *Technical Report 4, United States Air Force*, 1951.

[16] B. W. Silverman and M. C. Jones, "E. fix and jl hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951)," *International Statistical Review/Revue Internationale de Statistique*, pp. 233–238, 1989.

[17] A. J. Eronen and A. P. Klapuri, "Music tempo estimation with k-nn regression," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, pp. 50–57, 2010.

[18] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[19] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a regularized path to a maximum margin classifier," *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 941–973, 2004.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.