UTRECHT UNIVERSITY

Utrecht University

BACHELOR THESIS

Automatic Categorization of Electronic Music Genres

Author:
Noah Dani KREBBERS

Student Number:
6215327

Supervisor:
Prof. dr. ir. Jan BROERSEN

Second reader:
Gijs WIJNHOLDS

A 7.5 ECTS thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in

Artificial Intelligence
at the
Faculty of Humanities

June 26, 2020

Abstract

Noah Dani KREBBERS

Automatic Categorization of Electronic Music Genres

In this research we have used three different machine learning approaches on the automatic categorization of electronic music genres. We used Spotify for the collection of the data set, together with their custom track features. The selection of algorithms consists of K-nearest neighbors (KNN), Support vector machine (SVM) and K-means. The comparison between the supervised methods (KNN & SVM) was done using confusion matrices. We achieved an accuracy 70.1% and 75.5% respectively. For the unsupervised method (K-means) the performance was measured using the purity (49.7%) and Silhouette score (0.283). Principal Component Analysis (PCA) was used to visualize the clustering of K-means. We compared this to the original visualization of the data set to find differences and similarities. After this comparison, we came to the conclusion that unsupervised machine learning methods find different ways of genre categorization compared to our current classification. We can use these findings to improve our current ways of automatic genre classification. In addition to it, noticeable differences in both supervised and unsupervised categorization provide new grounds for more detailed comparison between certain genres.

Contents

Chapter 1

Introduction

The interdisciplinary science of music information retrieval (MIR) focuses on the understanding and practical uses of music data, including for example recommendation systems, instrument recognition and music creation. A particular field within MIR aims at improving the automatic categorization of genres, which is the topic of this research. At the annual Music Information Retrieval Evaluation eXchange (MIREX)[1], genre classification is one of the subjects which is largely discussed. Each year, they hold multiple genre classification contests, which shows that this topic is of great relevance in the world of MIR. Since this genre classification is an automated process, it relies heavily on machine learning models. The performance of such a model relies on the given input data. That, in combination with the massive amount of different kinds of genres, results in classifiers which are mostly limited to a certain subset of genres.

Music genre categorization is a way to differentiate and group songs, or on a larger scale albums and artists. However, the way we decide what song belongs to what genre is not always clear (Scaringella, Zoia, and Mlynek, 2006). Oftentimes songs could fit in multiple genres, which adds another layer of difficulty to automatic categorization. An additional way genre categorization is troublesome, is because the classification of a song is not always based on the auditory features. In those cases, genres are defined by certain time periods, such as baroque music for example, or by the culture of the musicians who created it (Mörchen et al., 2005).

To cope with the ever growing amount of songs, it is almost a necessity to have automatic systems for genre categorization, since these machines will have a more objective view. With automatic systems we might open doors which lead to a better understanding of our own ways of categorization, and perhaps find different ways to classify music. Therefore, automatic genre categorization is an interesting scientific research field.

A lot of research has been conducted on this topic. Here, the variance mostly lies within the used genres and the classification methods being used. Furthermore, most research use a set of upper level genres, one example of this is a research that used a classification between rock, classical and jazz (Deshpande, Singh, and Nam, 2001). Another example used a categorization between rock, classical and new wave (Mostafa and Billor, 2009). Several studies found classifiers which could differentiate well between multiple genres. Some studies even found recognition rates as high as 86.7 % on three genre classification (Classical, Hip-hop and Rock) (Kim, Yun, and Kim, 2015). Or even higher: 96% accuracy using neural networks on 4 genre classification (Haggblade, Hong, and Kao, 2011). However, with the ever growing collection of music, we are in need of more refined systems, which can distinguish sub-genres within genres and more genres at a time. This opens a whole new world for research, waiting to be explored.

[1] MIREX The annual exchange for music information retrieval enthusiasts.

1.1 Research questions

A particular genre which holds many sub-genres, is electronic music. This type of music is defined by its usage of mostly electronic instruments. The several sub-genres within electronic music however, are often distinguished from one another by their signature sounds, tempo and type of beat.

Since most genre categorization research has been done on distinctive genres, such as rock, classical and jazz, more intertwined genres remain to be explored for scientific research. Electronic music has a substantial amount of sub-genres, each in relatively close relation to each other, making it an excellent candidate for research in sub-genre categorization. By using machine learning methods, we try to find out whether these sometimes subtle differences could be distinguished automatically and also provide a reliable solution for automatic categorization within this genre. In addition, this research differentiates itself from related work by attempting to automatically categorize a larger set of genres. Therefore the main research question that we would like to answer is:

How do different machine learning approaches compare with each other, regarding the automatic categorization of electronic music genres?

Within machine learning there is a distinction between supervised and unsupervised methods, which we will explain in 2.2. As we are using methods of both of these categories, it would be interesting to find out whether there are significant differences between these methods. Unsupervised methods could provide us with new ways to classify music, since it produces a categorization completely on its own. Hence, interesting sub questions that we would like to answer are:

How do supervised genre categorization methods differ from unsupervised genre categorization methods?

and

Do unsupervised machine learning methods find new ways of classifying electronic music?

1.2 Academic relevance

By comparing the results of the different machine learning techniques, we can gain more understanding of the compatibility of certain approaches. Not only in the discipline of genre classification, but also in the wider sense of machine learning. This will be useful for further research, especially for optimizing genre classification systems.

Another interesting part of this study, is that since we are using unsupervised methods as well, we might find other ways in which genres are classified. This could potentially provide new insight in our current ways genre classification and perhaps makes us reconsider the way we define genres.

This research also broadens the perspective of genre classification within the scientific field of music information retrieval. By exploring more and more sub-genres we could create more unified categorization systems. These systems could also be of use in music generation, which is another field within MIR and is closely related.

1.3 Overview

In the next chapter we first elaborate on the the concept of genres. Followed by the sub-genres of electronic music that have been chosen for this research and their characteristics. Apart from the different genres we will also talk about the differences between supervised and unsupervised machine learning 2.

Next we discuss the data set that has been used for this research, followed by the feature extraction method. In this chapter a detailed explanation is given of the machine learning methods that have been used in this project. And finally the implementation of the algorithms is discussed 3.

The results of the experiment will be given in chapter four, where each of the machine learning methods are analyzed independently 4.

In chapter five we first discuss the results in respect to each other. Thereafter, answers to the research questions and the implications of this research are discussed. And we finish this research by discussing the possibilities for future research 5.

Chapter 2

Background

2.1 Genres

It is no secret that music is embedded in our everyday life, we listen to it while traveling, doing chores, studying and going to live performances (Sloboda, Lamont, and Greasley, 2009), just to name a few examples. Even though genres are extremely useful for the distinction of music themes, getting clear definitions of them seems to be quiet hard. Not surprisingly, because we essentially have to translate from one sense to another. What makes it even harder, is that music is not only a stimulus for our ears, it is connected to our emotions as well (North, Hargreaves, and Hargreaves, 2004; Shifriss, Bodner, and Palgi, 2015). Thus, the following descriptions only barely grasp what these genres truly hold. Nonetheless, they do provide a way to distinct genres from one another.

2.1.1 Genres used for this research

In total, nine different sub-genres of electronic music are used. The selection of these genres is mostly based on the magnitude of the genres, the popularity and the amount of genres within these sub-genres.

Disco is a genre that emerged in the nightlife scene of the 1970s, with influences of soul and funk. It has a lively and positive feel to it, which stimulates dancing. 4/4 beats and groovy bass lines are highly present in this genre.

Drum and Bass is characterized by its break-beat rhythm with a focus on the bass and sub-bass lines, with an energetic feel to it. The general tempo lies in the higher range of around 165 to 185 beats per minute.

Dubstep is somewhat related to drum and bass regarding the emphasis on bass-lines. The rhythm however, is generally more of a half-step tempo. A large part of dubstep is the wobble bass, where a bass note is extended in a rhythmic fashion.

Downtempo is more or less a mixture of different genres (soul, ambient, jazz, etc.). While there is an emphasis on the beat, it's not as insistent as trance. The overall mood is mellow and relaxed. A mix between acoustic and electric instruments is also typical for this genre. As the name suggests, the tempo is on the lower side, compared to other genres.

Hardcore is characterized by its cold and minimalist sounds, with powerful and oftentimes aggressive constant beats. With tempos from around 160 to 200 bpm and up, it is one of the faster types of genres.

House is a very broad genre within electronic music with lots of sub-genres. In essence, it's characterized by repetitive 4/4 rhythms with off-beat hi-hats, typically between 115 and 130 bpm, making it a very dance-able genre.

Techno is a genre with long and repetitive songs, between 120 and 140 bpm. Typically, it uses harsh mechanical beats at its base, accompanied with repetitive simple melodies. Usually, the beat remains audible trough out the song, making the sections within a song more overlapping.

Trance music has melodic and repetitive phrases, typically between 120 and 140 bpm. It focuses on the build-up, having beat-less sections and then slowly gaining more depth. Because of this, the tracks are often on the longer end of the timescale.

Trap is a genre which is a combination of hip-hop and dance music. Hi-hat triplets are present in the majority of these kinds of songs, together with loud low-end kicks. The tempo is more on the lower side of the spectrum, making it a more relaxing type of genre.

2.2 Supervised versus Unsupervised learning

Machine learning is a well-known concept within Artificial Intelligence. Over the last two decades, machine learning has developed into a wide-spread, commercially available tool, for both research and other uses (Jordan and Mitchell, 2015). The main distinction between the algorithms used in this research is whether it is a supervised or unsupervised machine learning algorithm. Both have their strengths and weaknesses, making it interesting to explore the differences and similarities.

Supervised algorithms have labeled data, which is used for the verification of the algorithm's performance (Bhavsar and Ganatra, 2012). An advantage of this, is that the models can make classifications using the input data and their respective labels, often resulting in better performance. An issue however, is that real-world problems do not always have labeled data available.

For these supervised algorithms, we need to split our existing data set if we want to determine the classifying power. One part of the set will be used of the actual training of the model, while the other part is used for measuring the performance of the model. This performance measure can be done by comparing the the model's labels to the actual labels of the input. This is a very simple process since the labels are already at hand.

Unsupervised algorithms on the other hand, do not require labeling for the development of the model. The most commonly used unsupervised algorithms rely on clustering techniques. The goal of an unsupervised clustering method is to discover natural groupings within a set of points or objects (Jain, 2010). Finding clusters might not be that hard when we work in a 3-dimensional space, working in considerably higher dimensions calls for sophisticated methods.

Clustering algorithms can be divided into hard- and soft margin methods (McGregor et al., 2004). Hard clustering algorithms categorize data points to only one cluster, while soft clustering methods can assign data points to multiple groups. The clustering methods of both of these types can be used on data sets with large amounts of dimensions. Though, the soft-margin clustering methods are more flexible and might provide more logical clustering.

Chapter 3

Methodology

3.1 Data set

For this research, no suitable data set was already openly available. Therefore, we created a data set specifically for this project with the help of Spotify. By combining numerous playlists, made by users and Spotify themselves, we were able to create a labeled data set which we could expand when needed. See Appendix A for the complete list of playlists used.

Initially a set of around 150 songs per genre was chosen, which turned out to be too little after initial results. Ultimately we gathered a set of 300 songs per genre. By distributing the data evenly, we hoped on creating machine learning algorithms that were less prone to uneven classification.

The selection of the playlists used for the data set is based on the title of the playlist and the amount of followers the list has. By using this selection criteria, we can assume that the data is mostly labeled correctly. However, we are aware that the labeling might not be 100 percent correct. The human-made playlists might contain songs from different genres than what the playlist is intended for.

Spotify already has song features available for development and research purposes[1]. By requesting this for all the songs from the playlists we were able to generate a data set of our own, in a fairly reasonable amount of time. This process of feature extraction would have definitely taken more time, if we were to create a data set where we had to manually gather tracks and their features.

3.2 Feature extraction

Using Spotify's developer tools we can extract song features easily by simply requesting song features online. On the upside, it saves us a lot of time regarding the assembly of the data set. Features are directly available, instead of analyzing all of the songs in the data set, which would take a lot of time. Another positive note, is that Spotify provides almost perfectly clean data. The only thing we need to correct, is that the beats per minute of a song is halved, sometimes.

On the downside, by using Spotify's self-made features, we are limiting ourselves to features that are not available to everyone. That is because you need permission from Spotify to make use of those features. Though, we can assume that one of the largest music streaming services in the world, has carefully crafted the algorithms for these features, probably resulting in better performing machine learning models.

[1] Spotify provides special development/research tools on request

Spotify provides both objective and subjective features (Spotify, 2020). Some of the objective features however, might not be useful for our research, because they have little meaning for genres and have low variance. The available objective features are as follows:

Duration (ms) Duration in milliseconds
Key Estimated overall key
Mode Indicates modality (major or minor)
Time signature Estimation of how many beats are in a bar
Tempo Overall estimated tempo of a track in beats per minute

Features such as the key and mode are most likely not really useful, because there is very little distinction within these features. The key of a song is represented as an integer between 0 and 11 and the mode is represented as 1 (major) or 0 (minor). Because of this low variance, and the high number of genres, we expect that the impact of these features will be small. Though, we will initially be using these features for the models.

The following features are made by Spotify, and unfortunately it is not known how these features work exactly. Nevertheless, they have been crafted for data analysis and most of them already have normalized values (floating point numbers between 0 and 1).

Acousticness Confidence measure whether a track is acoustic or not
Danceability Describes how suitable a track is for dancing
Energy Perceptual measure for intensity and activity, perceptual features to this include dynamic range, loudness and timbre
Instrumentalness Predicts whether a song does not contain vocals
Liveness Detects presence of an audience
Loudness Overall loudness of a track in decibels
Speechiness Detects presence of spoken words, mostly used to distinguish talk-shows/podcasts from songs
Valence Describing the musical positiveness (high valence indicates positive and happy songs, while low valence indicates negativity)

Initially, all mentioned features will be used for the models. During training however, we found that certain features were of negative impact for the performance. These features were therefore removed, this will be discussed in 4.1.3. See Appendix B for a sample of the feature data set.

3.3 Machine learning methods

3.3.1 Supervised learning

K-Nearest Neighbors

The K-nearest neighbors algorithm uses instance-based learning where the generalization process is delayed until classification is performed (Kotsiantis, Zaharakis, and Pintelas, 2007). These type of algorithms require less computation time in the training phase than eager-learning algorithms, but more time during the classification stage.

In principle, the idea behind this algorithm is very simple. Given an input, we look for the k nearest already identified input(s), and classify it as such. To determine

which neighbor is most nearby, commonly the Euclidean distance is used between the test and training samples (Peterson, 2009).

The distance is calculated as follows. We define $\boldsymbol{x}_i$ as an input sample with p features. The input sample will then be: $(x_{i1}, x_{i2}, ..., x_{ip})$. The total number of input values is n $(i = 1, 2, ..., n)$ and the total number of features is p $(j = 1, 2, ..., n)$. The distance between two points x_i and x_k is then defined as:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt{(x_{i1}, x_{k1}))^2 + (x_{i2}, x_{k2}))^2 + ... + (x_{ip}, x_{kp}))^2}$$

A drawback of the nearest neighbor approach, is when the class distribution is skewed, the algorithm tends to classify more input as the more represented class (Coomans and Massart, 1982). However, since we chose to use an evenly distributed data set, we were able to avoid this problem.

Support Vector Machine

Support vector machine can be used for both regression and classification tasks. While regression is about predicting a quantity, classification is about predicting a label. Since we are trying to find labels for our inputs, we will be using SVM for classification. The main goal of this machine learning technique is to find hyperplanes in a d-dimensional space that distinctly classify data points, where d is the number of features being used. Commonly, there are multiple hyperplanes available that separate classes, SVM works by searching for the maximum margin for these hyperplanes. In other words, we want to define our hyperplanes with maximum distance to the separating classes. By defining these margins, we can classify new data according to on which side of the hyperplane the new point is (Gandhi, 2018). The data points on which the margins rely are called support vectors, hence the name of the algorithm.

For this research we use a linear SVM using soft-margins, to counteract for the probably not linearly separable data being used, caused by errors in the data set. If we were to use hard-margins, the classification would then fail. Soft-margin SVMs use hinge loss functions to maximize the margins. The hinge loss function is defined as:

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

If both the predicted label $f(x)$ and the actual label y are of the same value, then the cost function is equal to zero, since $f(x)$ and y are either 1 or 0. This is because originally the Support vector machine is a binary classifier, but can be used as multi-class classifier by solving individual "one-vs-many" problems. After adding a regularization parameter to balance the margin maximization and loss, the loss function is as follows (Gandhi, 2018):

$$min_w \lambda \left\| w \right\|^2 + \sum_{i=1}^{n} (1 - y_i \left\langle x_i, w \right\rangle)_+$$

Updating the weights depends on whether the classification of a point is correct. When no mistake is made, only the regularization parameter is used for updating:

$$w = w - \alpha * (2\lambda w)$$

When we do make a mistake however, the weights are updated by including the loss with the regularization parameter:

$$w = w + \alpha * (y_i * x_i - 2\lambda w)$$

Compared to KNN this machine learning model is more complex, although it is more robust regarding high number of features (Kotsiantis, Zaharakis, and Pintelas, 2007). However, since we are using a rather small set of features, this advantage will most likely not be seen in our results.

3.3.2 Unsupervised learning

K-Means

We define our data in the same fashion as in K-NN. This results in our set $X = \{x_i\}$ where $i = (1, 2, ..., n)$ for all n data points. Where we want to group our set in K clusters $C = (1, 2..., K)$. With the K-means algorithm the goal is to create clusters where the error between the mean of a cluster and all the points within the cluster is as little as possible (Jain, 2010). We define μ_k as the mean of cluster C_k. The squared error of a cluster is then defined as:

$$J(C_k) = \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

By minimizing the the squared error for all the clusters we get the objective function of the algorithm:

$$J(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

The squared error uses the distance between a certain point and the mean of the corresponding cluster. This distance can be measured in multiple ways. The most common way for this is by using the Euclidean distance, as we have used in K-nearest neighbors. Even though the Euclidean distance is used in both these methods, it unfortunately does not provide a direct way of comparison between them, since the distance is used for different purposes in these different methods.

Since the minimization of this function has proven to be an NP-hard problem (Drineas et al., 2004), the algorithm converges to a local minimum. Therefore, the outcome of the algorithm is not deterministic, but will be slightly different each time we run it.

3.4 Implementation

Before implementing the machine learning methods, we first gathered our data set. By using the Spotify tools for developers mentioned earlier, and combining this with the Spotipy python library (Lamere, 2014) we created a data set which could be extended easily by simply adding more Spotify playlist IDs.

Next, some analysis of the data set was done using R. Here we noticed that the tempo of the tracks was sometimes off. An explanation for this, is that Spotify assigns these features automatically. By doing so, the automatic system wrongly halved or doubled tempos, resulting in skewed representations for the genres. We corrected this by doubling/halving the tempo below/above a certain threshold. The rest of the data set was left as it was. Thanks to Spotify's sophisticated system, no

missing data was found and the data set was easily applicable for machine learning purposes.

For the actual different machine learning algorithms, we started off by converting the data set to a pandas[2] data frame. For the different machine learning algorithms we used implementations provided by the Scikit-learn library (Pedregosa et al., 2011). The data was split into train and test sub-sets for the supervised methods and was scaled for all algorithms.

After testing we found that most of the parameters of the different methods were best kept to their default values, provided by Scikit-learn. We did use certain alterations that improved the performance. For K-nearest neighbors the amount of neighbors was set to 9. For Support vector machine we used a linear kernel with regularization parameter of 8.1. For K-means we used 9 clusters and the number of time the k-means algorithm will be run with different centroid seeds, was set to 1000, to find a near optimal convergence.

Even though the inclusion of the parameters might come across as unnecessary, we provide this to make the research as open as possible and in support of the Open Science[3] initiative of Utrecht University.

The representation of the supervised methods performance was done using confusion matrices, the formatting of these matrices was done using the Seaborn[4] library. For the unsupervised method we used Python's plotting library to visualize the data, after using Principal Component Analysis (Jolliffe and Cadima, 2016) on the data, to reduce the amount of dimensions to a more easily interpretable fashion. PCA translates high-dimension data sets to data sets of lower dimensions, making it possible to visualize a d-dimensional set as a 2-dimensional graph.

[2] Pandas Python data analysis library
[3] Open Science Utrecht University's initiative to promote open and reliable research
[4] Seaborn Statistical data visualization

Chapter 4

Results

4.1 Supervised performance

4.1.1 K-Nearest Neighbors

When we initially ran the K-Nearest Neighbors algorithm with the standard parameters, the results were promising, given we achieved an accuracy 67.7%. By altering the amount of neighbors that the algorithm uses, and removing certain features (see 4.1.3), we were able to achieve an accuracy of 70.1%. We used confusion matrices to visualize the results for the supervised algorithms, making it easy to identify notable mix-ups.

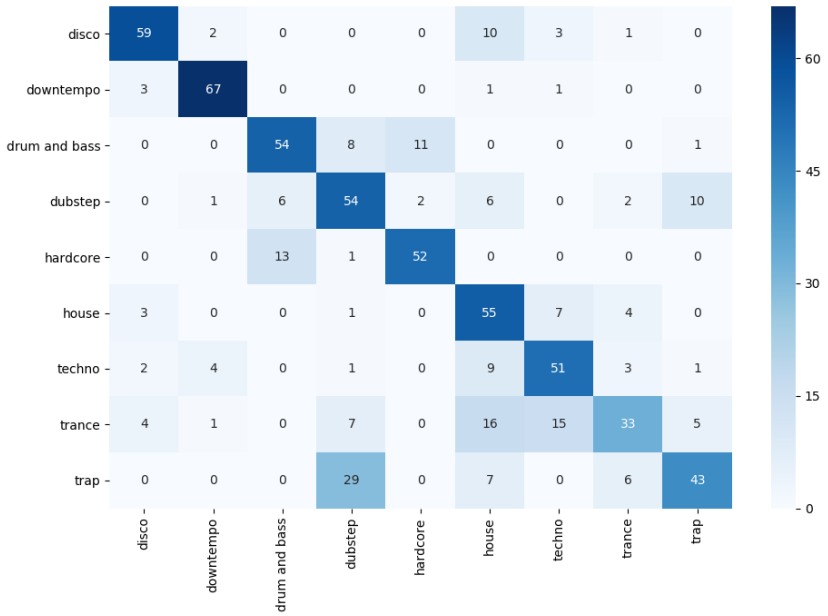

FIGURE 4.1: Confusion matrix for K-Nearest Neighbors

As seen in figure 4.1 (See Appendix C for enlarged versions of the figures), overall the algorithm performs quite well for all genres. There are some genre misclassifications which are notable. There seems to be a large part of trap which is identified as dubstep. Which is quite remarkable, since by ear these genres are very different. However, since there is also a part dubstep which is wrongly identified as trap, it is an indication to further examine these genres.

Another mix-up between genres is notable between hardcore - drum and bass. As seen in the figure, this mix-up happens both ways. Which shows that these genres do have similarities according to the features that we used.

Furthermore we can see some smaller misidentifications such as between disco - house, dubstep - trap and trance - house. Since these are one way misinterpretations, they draw less attention for further investigation.

4.1.2 Support Vector Machine

The support vector machine algorithm performed better than the KNN algorithm, with an initial accuracy of 74.5%. After some alterations an accuracy 75.5% was achieved. For the sake of time preservation we only used linear classification, i.e. no polynomial functions were applied. The increase in accuracy was achieved by adjusting the penalty parameter and performing feature selection (see 4.1.3). The results led us to the following confusion matrix.

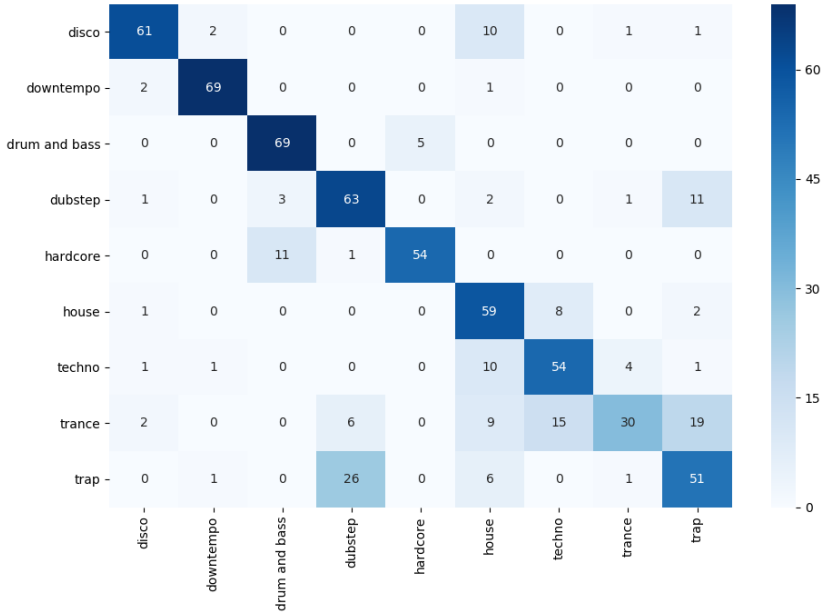

FIGURE 4.2: Confusion matrix for Support Vector Machine

If we compare both the confusion matrices, we can see that they are quite similar. However, the most noticeable difference is the misidentification of trance which is identified as trap. Another interesting difference is that the drum and bass - hardcore mix-up is not as prevalent as in the KNN matrix.

We do see however, that the amount of trap wrongly identified as dubstep, is still highly present. Therefore, it might be worth looking into the similarities and differences between these genres.

4.1.3 Feature selection

After obtaining the initial results from both classification algorithms, we decided to perform a feature selection method to increase the scores and remove unnecessary features. Feature selection provides a way to select the best contributing features, and remove the features that are irrelevant and negatively impacting the accuracy. Though results may improve using feature selection, it is not guaranteed that the performance will be better. This procedure was done using the *SelectKBest* algorithm provided by Scikit-learn. The algorithm uses ANOVA F-values to determine which features have the most impact, and selects the K best features.

By selecting the desired number of features, we ran the algorithms multiple times, in order to determine the optimal amount of features. In the following table we can see the accuracy of both KNN and SVM classifiers on the amount of features used.

TABLE 4.1: The effects of the amount of features(K) on the accuracy of the classifying algorithms (SVM & KNN)

K (n)	SVM (%)	KNN (%)
13	75.8	64.0
12	75.5	63.7
11	75.4	64.1
10	75.8	67.1
9	75.5	69.3
8	74.5	67.8
7	72.7	69.3
6	71.5	69.6
5	70.8	70.8
4	67.5	66.8
3	64.7	65.6
2	62.9	65.9

By choosing the amount of features according to the best combined result, the optimal value for K seemed to be 9. The features removed by the algorithm were *liveness* and all of the categorical variables: *key, mode* and *time signature*. We have chosen for this best combined result, in order to provide an optimal feature set for the unsupervised algorithm.

4.2 Unsupervised performance

4.2.1 K-means

For the unsupervised method we used a different technique to determine the accuracy, since we are clustering instead of classifying. First, let us visualize the data set by using Principal Component Analysis to reduce the amount of features to a two-dimensional graph. During this process however, we lose our original variables, which are replaced by newly created functions. Hence the axes of the plot are not clearly defined.

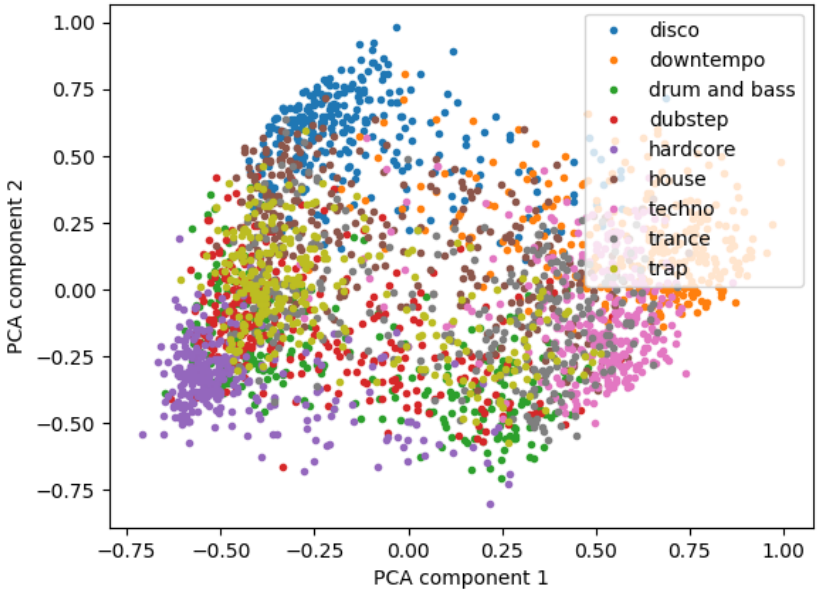

FIGURE 4.3: Visualization of data set using PCA

It seems that some genres do in fact have some sort of cluster formation. However, this is not the case for all genres. For example, drum and bass is spread across the graph.

By starting with the same feature set used in the the supervised methods, we hoped to maximize our results. By comparing the purity and the Silhouette score, we determined the model's performance.

The purity score is defined as the percentage correctly classified data (Christopher and Schütze, 2008). This was possible for us, since we were using labeled data. The Silhouette score defines how well the clusters are defined in a range from -1 to 1. Scores below zero indicate wrong type of clustering. The closer to one, the better the seperation between clusters.

We further increased results by performing PCA on the unsupervised data set. Since PCA uses dimensionality reduction, it can be useful for increasing performance, apart from it being able to visualize data. By reducing the amount of features to 5 using PCA, we were able to get a purity of 49.7 % and a Silhouette score of .283. This resulted in the following visualization.

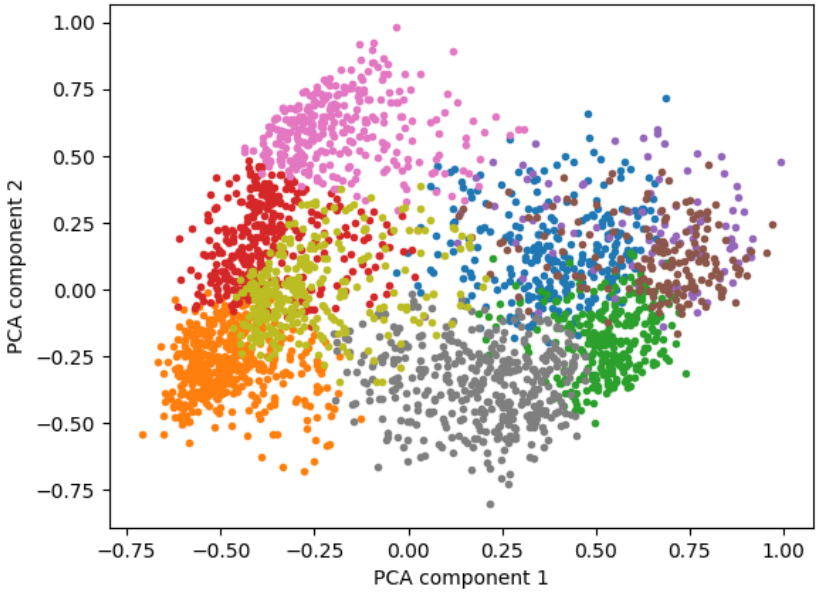

FIGURE 4.4: Visualization of data after clustering

Since the clustering algorithm does not define the genres, we cannot create a legend for this plot. Disregarding the color however, we can directly compare the two figures since the data points in the two figures are the same.

After the initial clustering using the desired amount of genres. We compared different amounts of genres to their respective scores, in hope of finding a better way to categorize genres.

TABLE 4.2: The effects of the amount of clusters(C) on the purity and Silhouette score

C (n)	Pur. (%)	Sil.
14	54.3	0.229
13	53.6	0.226
12	53.0	0.260
11	51.2	0.234
10	50.3	0.253
9	49.7	0.283
8	48.5	0.258
7	48.7	0.278
6	44.3	0.286
5	38.4	0.318
4	38.4	0.337

Using the table, we can see an increase in purity if we were to use more different genres, but this would be of negative impact for the Silhouette score. Nevertheless, The Silhouette scores are all fairly similar, which can be explained by the spread of the genres in the data set before K-means has been applied.

Chapter 5

Discussion

The results of our research are very promising. The fairly high scores using the supervised machine learning algorithms, indicate that the data set and features were usable for this project, even though some of the features were better off removed. It gave us a good indication whether the unsupervised method results are accurate.

The confusion matrices provided insight in the supervised classification algorithms. Some notable anomalies such as the trap - dubstep misclassification and the drum and bass - hardcore misclassification might be worth looking into. It might be that these genres have more in common than we think. Nevertheless, we are aware of the fact that the data set is probably not perfectly categorized, which might also be a cause for genre mix-ups.

Using the feature selection method, we found that certain track features were negatively impacting the performance of the models. *Liveness* and the categorical variables *key*, *mode* and *time signature* do not seem to be of influence on the type of genre, meaning that there is no relation between the distribution of these values and the genres.

The clustering method using K-means resulted in an interesting graph, looking vastly different than the initial PCA reduced visualization. Since it is a clustering algorithm, we can not use the model to classify what cluster belongs to what genre. However, a side-by-side comparison of the two graphs gives some interesting cluster information.

The first point of interest is that the trance genre (lower left corner) is very well defined. Being almost totally similar in both graphs. The same goes for downtempo and techno, in the upper right corner. The same is true for the upper-left corner, which takes up a cluster similar to the disco cluster in the real graph. Dubstep and trap seem to be split in a different fashion by K-means. This different categorization found by K-means might be interesting for further exploration, to find out what the similarities and differences are exactly. Interestingly, more evenly spread genres (in the real representation), such as house and hardcore, are hardly noticeable in the K-means graph.

The initial representation of the data set using PCA showed that the genres were greatly intertwined, which shows that the electronic music genres used are closely related to each other. That comes to no surprise, since they are all sub-genres of an overarching genre. Therefore, the results obtained by all three methods are to our satisfaction, though we hoped the supervised methods would obtain higher results.

Some annotations about our own research are the following. First, the labeling of the tracks in the data set is most likely not 100 percent correct. Even though the playlists used for the data set were carefully selected, some mix-ups were almost inevitable. Especially because of the close relations between the genres used for the research. Secondly, the features developed by Spotify were not completely accurate. We noticed some errors in the assignment of tempo to the tracks. The automated

process sometimes wrongly halved our doubled track tempos, resulting in skewed values. Though we have tried to rectify this, it is not certain that this process worked perfectly. In addition to this, different undetected errors might also be present, since the data set's validity was not manually checked for the entire data set.

Since this field of research has been closely studied, it might be interesting to compare our results to similar research. However, because we used Spotify for feature extraction and a custom data set, comparison might not be totally significant. Nevertheless, our accuracy shows great resemblance to earlier research, mentioned in the introduction. This suggests that the feature set provided by Spotify, is of high value for genre classification.

This research contributes to the development of machine learning methods, by showing where the strengths and weaknesses of certain methods are. We can see that the supervised methods are useful for correctly classifying genres to our current standards. Unsupervised methods might not be as useful for this. It might however, show us to think in different ways to classify our music. In addition, this research contributes to the greater, more specified field of genre classification, by using genres that were not thoroughly examined.

5.1 Conclusions

With the results and the discussion, we were able to answer our research questions.

How do different machine learning approaches compare with each other,
regarding the automatic categorization of electronic music genres?

The supervised algorithms showed reasonable results in classifying the data set. Support vector machine performed slightly better than K-Nearest neighbors, but the results were fairly similar. K-means, on the unsupervised side, showed a purity score noticeably lower than the performance of the supervised methods. This is not a bad thing however, since this supports the hypothesis that unsupervised genres use different ways of classification.

How do supervised genre categorization methods differ
from unsupervised genre categorization methods?

and

Do unsupervised machine learning methods find
new ways of classifying electronic music?

As mentioned above, there is a noticeable difference between the performance of supervised and unsupervised machine learning methods regarding the classification of the electronic music genres used in this research. As seen in 4.4, the clustering is vastly different compared to 4.3, by attempting to create better defined borders between the genres. As a result, the genres that we know, have been combined to form new groups. We can conclude that unsupervised learning does provide new ways of genre categorization.

The categorization is based on the features that we have used in this model, with a different set of features we might find different clusterizations. We compared the feature set, which was determined by the selection algorithm, to the initially found feature set. We found that the subset provided better purity scores. Therefore, the decision to use the subset for the unsupervised algorithm has paid off.

5.2 Further research

From here on, we can continue research on the comparison between supervised and unsupervised machine learning algorithms, within the music industry. First, it would be interesting to recreate this experiment, but with even more attention to the data set. We acknowledge that this is a very time consuming project, though it might be of great importance for the performance of the different algorithms.

Furthermore, we can continue research into two different directions. The first being the alteration of the machine learning algorithms. Since there are more machine learning methods than we have used in this research, it would be interesting to use different methods on both the supervised and unsupervised side. Secondly, we could use the current algorithms on different types of genre sets. Of course we can continue endlessly by combining different kinds of genres. What we have seen in this research however, is that some genre combinations might be exceptionally interesting to explore. We have seen that dubstep and trap are noticeably more mixed up in the confusion matrices, as were drum and bass - hardcore. The clustering method also showed us that dubstep and trap might be worth looking into, since the method categorized this entirely different. Further investigation in these genres might lead to similarities we have not yet considered. Maybe one day, genre categorization could be an entirely automated process.

References

Bhavsar, Hetal and Amit Ganatra (2012). "A comparative study of training algorithms for supervised machine learning". In: *International Journal of Soft Computing and Engineering (IJSCE)* 2.4, pp. 2231–2307.

Christopher Manning, Prabhakar Raghavan and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Coomans, Danny and Désiré Luc Massart (1982). "Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules". In: *Analytica Chimica Acta* 136, pp. 15–27.

Deshpande, Hrishikesh, Rohit Singh, and Unjung Nam (2001). "Classification of music signals in the visual domain". In: *Proceedings of the COST-G6 Conference on Digital Audio Effects*, pp. 1–4.

Drineas, Petros et al. (2004). "Clustering large graphs via the singular value decomposition". In: *Machine learning* 56.1-3, pp. 9–33.

Gandhi, Rohith (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Accessed:2020-13-06. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

Haggblade, Michael, Yang Hong, and Kenny Kao (2011). "Music genre classification". In: *Department of Computer Science, Stanford University*.

Jain, Anil K (2010). "Data clustering: 50 years beyond K-means". In: *Pattern recognition letters* 31.8, pp. 651–666.

Jolliffe, Ian T and Jorge Cadima (2016). "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150202.

Jordan, Michael I and Tom M Mitchell (2015). "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245, pp. 255–260.

Kim, Kyuwon, Wonjin Yun, and Rick Kim (2015). *Clustering Music by Genres Using Supervised and Unsupervised Algorithms*. Tech. rep. Technical report, Stanford University.

Kotsiantis, Sotiris B, I Zaharakis, and P Pintelas (2007). "Supervised machine learning: A review of classification techniques". In: *Emerging artificial intelligence applications in computer engineering* 160, pp. 3–24.

Lamere, Paul (2014). *Spotipy python package*. Accessed:2020-14-06. URL: https://spotipy.readthedocs.io/en/2.12.0/.

McGregor, Anthony et al. (2004). "Flow clustering using machine learning techniques". In: *International workshop on passive and active network measurement*. Springer, pp. 205–214.

Mörchen, Fabian et al. (2005). *MusicMiner: Visualizing timbre distances of music as topographical maps*. Univ.

Mostafa, Mohamed M and Nedret Billor (2009). "Recognition of western style musical genres using machine learning techniques". In: *Expert Systems with Applications* 36.8, pp. 11378–11389.

North, Adrian C, David J Hargreaves, and Jon J Hargreaves (2004). "Uses of music in everyday life". In: *Music Perception: An Interdisciplinary Journal* 22.1, pp. 41–77.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12, pp. 2825–2830.

Peterson, Leif E. (2009). *K-nearest neighbor*. Accessed:2020-13-06. URL: http://scholarpedia.org/article/K-nearest_neighbor.

Scaringella, Nicolas, Giorgio Zoia, and Daniel Mlynek (2006). "Automatic genre classification of music content: a survey". In: *IEEE Signal Processing Magazine* 23.2, pp. 133–141.

Shifriss, Roni, Ehud Bodner, and Yuval Palgi (2015). "When you're down and troubled: Views on the regulatory power of music". In: *Psychology of Music* 43.6, pp. 793–807.

Sloboda, JA, Alexandra Lamont, and Alinka Greasley (2009). "Choosing to hear music". In: *The Oxford handbook of music psychology* 1, pp. 431–440.

Spotify (2020). *Get Audio Features for a Track*. Accessed: 2020-26-05. URL: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/.

Appendix A

Used playlists

ID	Title	Creator
0HdP9VAO6ydihgNgEN6Cp3	DISCO HITS 70s-80s-90s	Alessio Tamburo de Bella
37i9dQZF1DX2GKumqRIZ7g	Disco Fever	Spotify
37i9dQZF1DXbS8bPVXXR2B	Disco Hi-Life	Spotify
37i9dQZF1DX1MUPbVKMgJE	Disco Forever	Spotify
3AtFItPTNrmxqREWOWZV6e	Disco 70's Hits	Nobby Clarke
068WHS0zOWsqvn2uIBYb5D	Drum and Bass 2020	Elon Musk
3gqEaRQUN0xYi9kHexWQpY	DnB - Best drum and bass songs!	Yung Fortnite God
0TNtOQpT4BP3F1bIMWAAPG	DRUM AND BASS	ReadyToFestival
37i9dQZF1DX5wDmLW735Yd	Massive Drum & Bass	Spotify
4oOZJEq1TBUti6PSouTo5M	UKF Drum & Bass	UKF
3ObJ6Qra3CkV0gNCRTtK0c	Dubstep Bangers 2020	knivgaffel
37i9dQZF1DX4arVIN5Cg4U	Dubstep Classics	Spotify
37i9dQZF1DX5Q27plkaOQ3	Dubstep Don	Spotify
5wNRJwSnBImUuZXJW1TiAj	Dubstep 2020	Filtr Sweden
4oluc1u5nMoUItgaO48OxV	Dubstep 2020 - UKF	UKF
3JCd8rIuqDJQWo5AhTDq71	DUBSTEP	dubstepgutterofficial
2d6sogVjwXrnxC0IAXFwcl	Downtempo Deep Electronica	Joanna Hengstebeck
37i9dQZF1DWWQp0YMTvpD3	Downtempo Beats	Spotify
4JE6MxO5jXQAaeCEp1UR83	Downtempo	Jo Hannes
1tey17jGZg2tp6woRxvolN	HARDCORE 2020	d-ceptor
0tIJcuikyYauclNo6FeQAX	MASTERS OF HARDCORE	Art of Dance
2GEXzPeksIINQMTivWQ2el	HARDCORE	Masters of Hardcore
3dUZ038b8Wew9D2A9tVh1V	Masters of Hardcore 2020	Simon Wyssen
37i9dQZF1DWXDvpUgU6QYl	House is a Feeling	Spotiy
2otQLmbi8QWHjDfq3eL0DC	House Music 2020	Topsify
37i9dQZF1DXa8NOEUWPn9W	Housewerk	Spotify
46GbJCgS1j3YiWTfTs7kCy	HOUSE CLASSICS (1992-2020)	Wolfgang Wee
37i9dQZF1DX6J5NfMJS675	Techno Bunker	Spotify
76TlgpqdBryXFAcrHmiGWH	Techno Bangers	Adam Heaton
2oz1zP2wII4W2bhNYfIUD1	Techno Tuesday	otterkids
5XR0PF0thgGhoWy8Xw47p9	TECHNO 2020	No Mercy - Techno
5JyM6mqPEDLcToVHY5HlY4	TRANCE MUSIC 2020...	UNKNOWN
260cw4PvjDjcWuCi5duiEf	TRANCE CLASSICS (90s/2000)	Wolfgang Wee
37i9dQZF1DXbtYAdenGE9U	Trance Energy	Spotify
78AFAJFvRzboZfEDnAkkFn	Trance Anthems 90s-00s	Shane Codd
2iTMQ36cvjhxgIYGv7uld5	Trance 100 - by Armada Music	Armada Music
0NCspsyf0OS4BsPgGhkQXM	Trap Nation	Trap Nation
37i9dQZF1DX1OIMC8iDi74	Trap Mojito	Spotify
5aPwKjwNHr6dnCejLcPTVx	Trap Nation	1198177587
7Kk0KvfzcaX4QLtli1rIBK	Trap City	Trap City

Appendix B

Feature data set sample

	genre	duration_ms	key	mode	time_signature	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	valence	tempo
1638	downtempo	368219	7	1	4	0.78	0.84	0.448	0.173	0.0998	-9.673	0.0361	0.212	89.997
1639	downtempo	262326	3	0	4	0.21	0.709	0.637	0.704	0.171	-8.392	0.0509	0.817	85.9895
1640	downtempo	451030	9	0	4	0.718	0.72	0.34	0.778	0.344	-14.421	0.0437	0.127	108.989
1641	downtempo	346272	9	0	4	0.519	0.595	0.341	0.924	0.0811	-14.431	0.0487	0.441	96.9935
1642	downtempo	537591	9	0	4	0.301	0.778	0.341	0.845	0.113	-14.108	0.0393	0.0714	105.064
1643	downtempo	393402	2	0	4	0.0294	0.674	0.449	0.804	0.0794	-10.16	0.0623	0.269	97.013
1644	downtempo	441739	6	0	4	0.0544	0.76	0.562	0.546	0.435	-10.387	0.0648	0.0746	95.033
1645	downtempo	412251	2	1	4	0.0127	0.71	0.573	0.788	0.0871	-10.889	0.0498	0.279	99.995
1646	downtempo	416000	9	0	4	0.491	0.756	0.409	0.671	0.0944	-10.088	0.0412	0.0898	104.997
1647	downtempo	484478	9	0	4	0.553	0.795	0.256	0.914	0.0995	-12.622	0.0583	0.336	108.017
1648	downtempo	475429	6	0	4	0.0801	0.782	0.442	0.894	0.124	-13.599	0.0456	0.0398	105
1649	downtempo	332574	4	0	4	0.0495	0.781	0.421	0.621	0.101	-11.231	0.0653	0.159	94.009
1650	downtempo	464176	2	1	4	0.352	0.686	0.555	0.879	0.114	-11.428	0.0647	0.411	91.028
1651	downtempo	282885	10	0	4	0.127	0.738	0.466	0.953	0.0883	-14.394	0.0406	0.126	111.993
1652	downtempo	464341	3	0	4	0.0479	0.707	0.371	0.883	0.0988	-12.501	0.347	0.386	90.9975
1653	downtempo	382689	9	0	4	0.341	0.773	0.383	0.375	0.305	-11.34	0.086	0.396	92.9
1654	downtempo	278569	4	0	4	0.521	0.639	0.641	0.389	0.0901	-8.116	0.06	0.818	80.002
1655	downtempo	300168	4	1	4	0.00344	0.794	0.456	0.238	0.0785	-8.296	0.0786	0.474	90.013
1656	downtempo	363095	8	1	4	0.615	0.913	0.141	0.907	0.084	-14.659	0.105	0.364	108.017

Appendix C

Figures

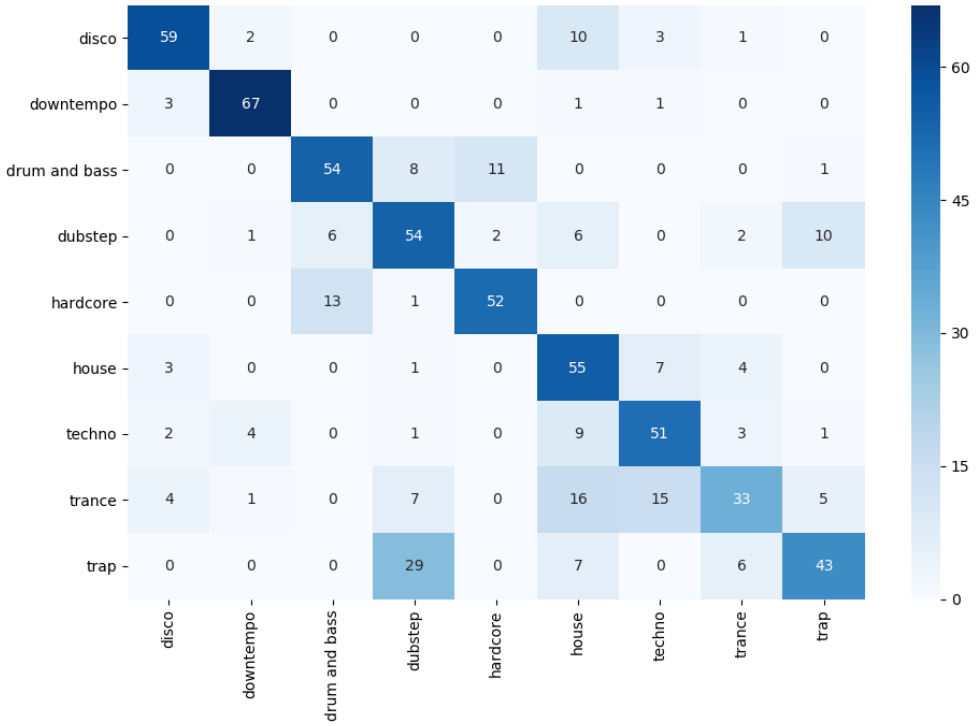

FIGURE C.1: Confusion matrix for K-Nearest Neighbors

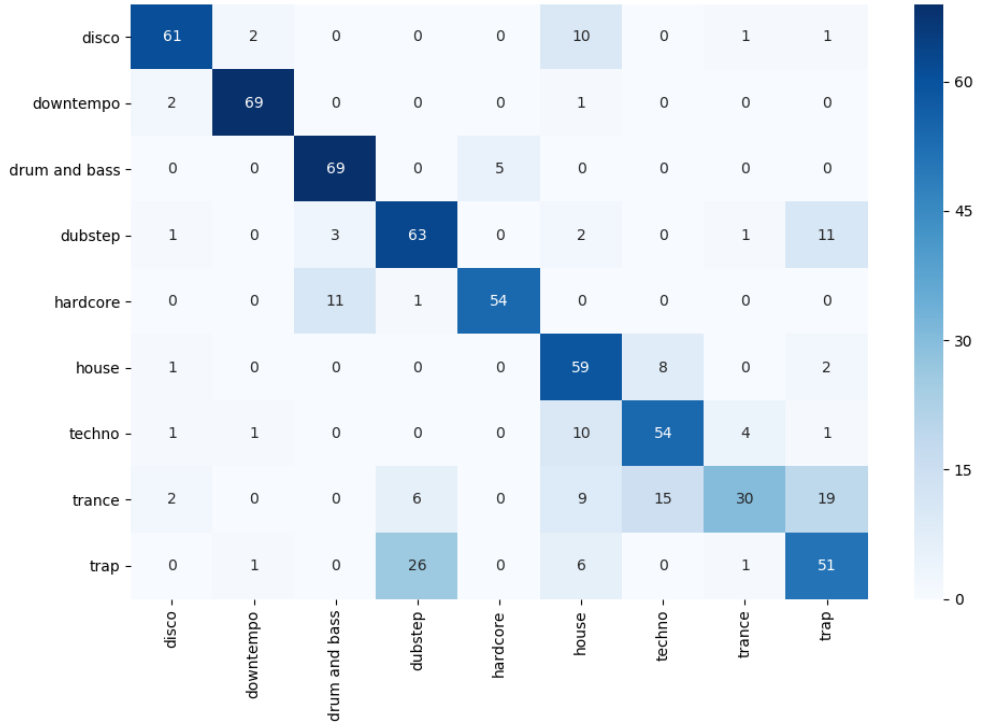

FIGURE C.2: Confusion matrix for Support Vector Machine

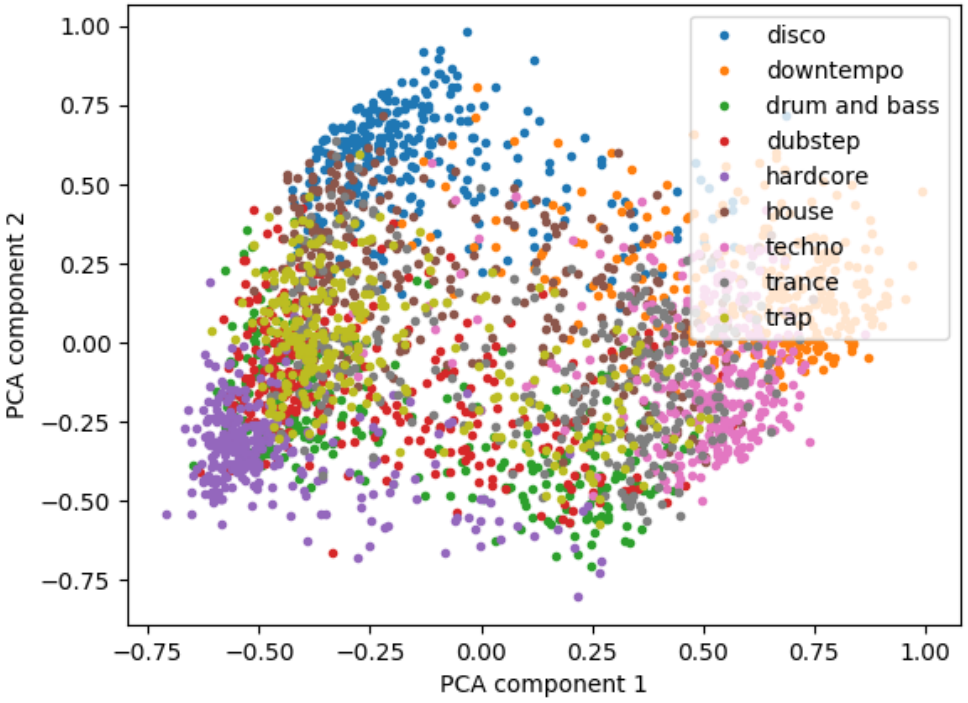

FIGURE C.3: Visualization of data set using PCA

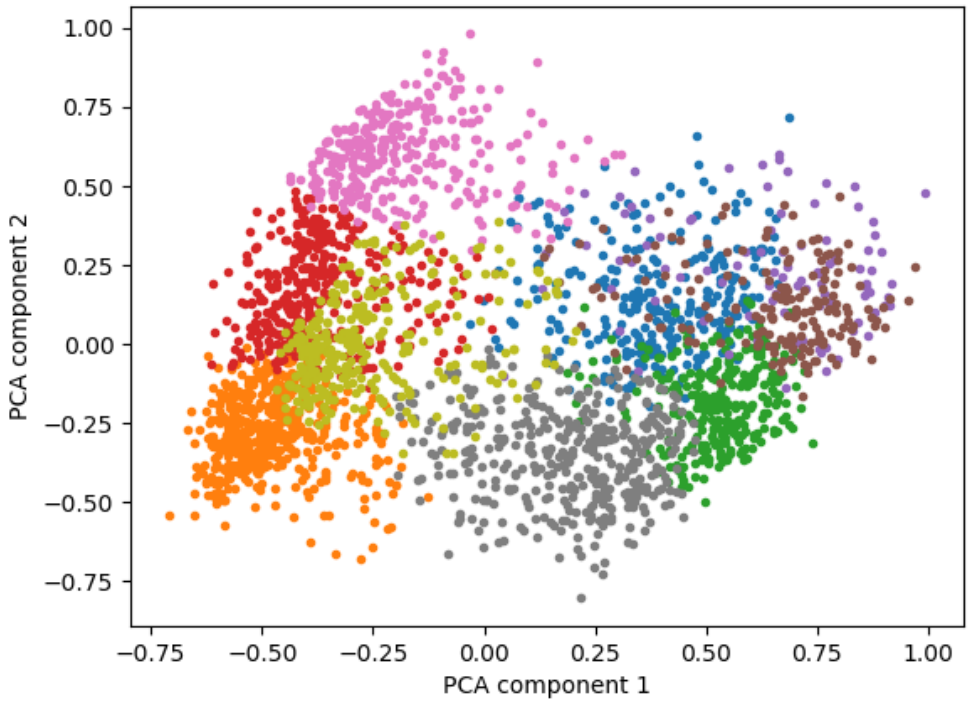

FIGURE C.4: Visualization of data set after clustering