Music and Sport: An Explorative Study using Unsupervised Machine Learning

Marius Alexander Sandberg

Thesis submitted for the degree of
Master of science in Informatics: Programming and Networks
60 credits

Institutt for informatikk
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2019

Music and Sport: An Explorative Study using Unsupervised Machine Learning

Marius Alexander Sandberg

Music and Sport: An Explorative Study using Unsupervised Machine Learning

http://www.duo.uio.no/

Abstract

Music is often used together with activities, such as exercise or relaxing, and is usually associated with a certain emotion, such as energetic or chill. There is little prior research to what constitutes the emotions associated with music, and if the music used in activities are of a certain emotion. In this thesis, we will explore the correlation between music and mood, and work towards finding out what the emotions in music used in activities are. The overall goal is to make a model that can find songs for certain activities, such as exercise or relaxing, and that it distinguish songs based on the emotion in these songs. At the MediaEval Benchmark, researchers have made a publicly available dataset [50] which consists of 1,000 songs annotated with emotions on the valence-arousal dimensional model [38]. Machine learning is an effective way of predicting patterns in large datasets, where clustering is a type of unsupervised learning to predict data without labels. With the dataset from MediaEval and a variety of clustering algorithms, we have extract features from the songs in the dataset using audio signal processing, and clustered these features to explore the correlation of the annotated emotions and the features of music. Features from 45-second clips of the songs have been used with Principal Component Analysis before clustering, and features from clips of the whole songs has been summarized statistically before clustering. The clustering experiments were used with eight different algorithms, but four algorithms were used further as the other algorithms were not deemed useful or did not have a parameter for specifying cluster numbers. These four algorithms are Mini Batch K-means, Gaussian Mixture, Spectral Clustering and Birch. These algorithms has been used with cluster numbers from 2 to 10. This has resulted in 72 different configurations for clustering algorithms with number of clusters. The clustering procedures from the statistically summarized features shows good results compared to the clustering procedures from the features with Principal Component Analysis. The clustering procedures from the former experiment are good in separating clusters of songs into the quadrants of the valence-arousal dimensional model, especially quadrants 1 and 3, which represent emotions such as excited, happy and pleased for quadrant 1, and sad, bored and sleepy for quadrant 3.

Acknowledgements

First and foremost, I would like to thank my supervisors Pål Halvorsen and Michael Riegler for their amazing support and feedback. They have been essential to the development of my master thesis, and I would not have made it without them.

I would also like to thank PhD student Hanna Borgli for her support and feedback throughout the tough period of writing this thesis.

Furthermore, I would like to thank my family for their kind words and caring actions. They have supported me for the longest period of time, and kept doing so when I needed it the most.

Finally, I would like to thank all my friends for being such beautiful people. You truly make the hard times better and the good times exceptional.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Characterizing the emotional qualities of music and the impact on the mood is a subjective experience. Although, there are some indications that music is grouped into moods based on the listeners experience, and that they are overall represented by the correct mood across a wide variety of audience. With the help of machine learning, this thesis explores the possibilities of there being certain characteristics in music of a certain mood that attributes for their very mood classification.

1.1 Background and Motivation

In 1959, the term "Machine Learning" was coined by Arthur Samuel who was a pioneer in the field of machine learning [40]. Machine learning has now become the foremost and most flexible tool for prediction based on already existing data.

Machine learning has the ability to generalize data and to react accordingly to trained boundaries. The main types of machine learning are supervised learning, unsupervised and reinforcement learning. Unsupervised learning is used in the case where there is no labeled data and the boundaries are set based on the nature of the data. Clustering data is a common approach to learn more about unlabeled data.

There are some research into the connections between music and mood [33, 61]. The existing ways of finding music to a certain mood are limited to that of listener-made playlists. Often times, music is found by sharing playlists or listening to a specific artist that you know has the correct mood properties. If music can be heard as 'sad', 'happy', 'energetic' or 'relaxed', there are most certainly elements like key, tempo and melody that play into their mood classification.

The dimensional approach to mood categorization proposed by J. A. Russel [38], with dimensions valence and arousal, has been used in a lot of research [33, 50, 61]. In the research done by MediaEval, they made a dataset consisting of 1,000 songs, annotated with emotions on the valence-arousal dimensional model, crowdsourced by using Amazon Mechanical Turk (MTurk) [50]. This dataset will be the baseline for this thesis, together with clustering, when exploring the correlation between music and mood.

A lot of people also listen to music while doing an activity, such as sports, relaxing, studying, etc. Research suggests that the correct music coupled with exercise can even enhance the workout session [2, 7, 24–26]. The research in this thesis will focus on furthering the development of music analysis and music-to-mood classification, where the music fits the mood of an activity.

1.2 Problem Statement

The research questions for this thesis are the following:

1. *Can songs, based on extracted features, be grouped into clusters that correlate with their annotated emotion, and what is the quality of the clusters in terms of cluster separation and statistical summary?*

2. *How can we evaluate the cluster homogeneity?*

3. *Is it possible to pick reliable models that can be applied to find songs for certain moods?*

The goal of this thesis is to explore the correlation between music features and emotion, and to further the research on the correlation between music and mood. To achieve this goal, a variety of clustering algorithms and feature vectors will be applied to the MediaEval dataset [50].

Further goals would be finding out if the clustering can automatically find music that fits best to a certain activity, like sports, relaxing, studying, etc. These activities are often represented by songs in different playlists on Spotify and other media, including some research papers that use songs for motivational qualities in physical exercise [2, 7, 26] that have listed what songs are used in their study. Using the songs in these playlists and papers, it is possible to use audio signal processing and predict the extracted features with the trained clustering model.

1.3 Scope and Limitations

The clustering algorithms that have been used in this thesis were limited to the ones where we could specify the number of clusters we wanted to have. Some algorithms rely on analyzing the data and makes clusters based on the feature-space. Mean Shift uses a bandwidth parameter to specify the size of the area to search through, and makes cluster based on the bandwidth parameter. Affinity Propagation uses exemplars to represent samples and makes clusters from a final analysis of exemplars, where the parameter for preference in this algorithm can make samples be more likely to become exemplars, and in turn affect the number of clusters.

The features has been pre-processed in two ways, where it is possible to utilize other ways to structure the data before clustering. We initially made plans to extract features from the songs into time frames that was the song

length divided by a certain number, but pyAudioAnalysis [16] was could not extract large enough time frames.

The scope of this thesis will be limited to testing all the features extracted from the songs together with clustering algorithms. It is possible to exhaust all combinations of features extracted from the songs to get a clearer picture of what features best contribute to the optimal clustering, but this thesis will focus on the contribution of all the features together.

There are limitations to the number of clusters that are reasonable to use. To determine the effectiveness of the algorithms, we used cluster numbers from 2 to 10.

1.4 Research Method

The ACM Education Board created in 1989 by a task force on the core of computer science, determines and characterizes the structure of how research in computing, should be approached. This report [9] defines computer science in its essence as an intersection between several central processes. The central processes are applied in mathematics, science and engineering. These central processes are basically reflected in the paradigms of (i) theory, (ii) abstraction and (iii) design.

- **Theory:** The theory process is responsible for defining and characterizing the objects under study by formulating and hypothesize possible relationships. Furthermore, it is characterized by determining the relationships among objects, verifying their correctness and interpreting the results.

  For the theoretical part, we reviewed a handful of literature to build a theory of how the correlation between music and mood could be measured. We touched upon spectrogram representation of audio signals with convolutional neural networks for image processing.

- **Abstraction:** The abstraction process is used for modeling and emerges from experimental scientific methods. While a researcher is investigating a problem, a hypothesis is formed, a model created, experiments designed and finally data collected and analyzed.

  The abstraction process consisted of making experiments to test the clustering algorithms and how to build the system. We also made different metrics for evaluating the performance of the clustering algorithms.

- **Design:** The last process is design, which is closely related to engineering. This involves researchers to state requirements and solutions, followed by designing and implementing a system. This process is concluded with an evaluation of the system.

  The main work falls in the category of design, where we have made a running prototype and performed experiments.

1.5 Main Contributions

In this thesis, we have presented the work we have done in exploring the correlation between music and mood. We used various ways of clustering features from songs, in order to find optimal clustering algorithms and parameters for answering the research questions in the problem statement 1.2. The summarized main contributions of this thesis are:

(i) Development of an extended dataset that are pre-clustered based on number of clusters and clustering algorithms. The code and additional data is available at GitHub[1].

(ii) Literature based analysis of connections between music and exercise performance.

(iii) Development of a metric that allows to chose the most homogeneous clustering algorithms within a set of algorithms.

(iv) Identifying clustering algorithms and number of clusters that can represent the emotions of the valence-arousal dimensional model.

(v) Creating a design of a system that is ready to use for further research on the topic of music and exercise.

1.6 Outline

The thesis is organized as follows:

Chapter 2: Background: In this chapter, we provide background information to what this thesis is built around. Firstly, we summarize what research has been done in terms of the connection between music and mood, mainly in the field of computer science. We also include what types of tools are available and will be used in this thesis.

Chapter 3: Methodology: This chapter describes the methodology we used for researching the problem statements listed in section 1.2. To research this problem, we used different approaches to feature extraction of the songs in the dataset from MediaEval [50]. Furthermore, we describe the procedure of testing the dataset on different clustering algorithms and how to best represent the data we collected from these experiments. We also list the different metrics that are used to determine the performance of the clustering algorithms.

Chapter 4: Experiments, Results & Discussion: In this chapter, we present the design of the experiments as well as the system flow. We also discuss the results that appeared from the clustering experiments, and what the results can be used for. The chapter also contains discussion around the topic of best performing clustering algorithms between experiments, the use of trained models, and theorizing what can be changed to get better results.

[1] `https://github.uio.no/marasand/thesis`

Chapter 5: Conclusion: Finally, we present what parameters and clustering algorithms worked best, and what the properties and the statistical summary of the data tell us for the further research in machine learning for music and mood.

Chapter 2

Background

This chapter presents the background and motivation for the thesis. Relating to section 1.2, we are going to find and review literature that relates to machine learning, and how it can be utilized in mood categorization. Starting off, we will provide some background in the studies of music and mood, and what approaches to this concept can be used in this thesis. Most importantly, we will provide background and motivation for using the MediaEval dataset [50]. The classification of emotion and the connection between music and mood will require some research in order to come up with a concrete strategy to tackle the problem statements we have formulated in section 1.2. In terms of applicability of the work in the thesis, the emotion classification can be used to categorize music for an activity, such as sports, relaxing, studying, etc., where an activity is represented by a mood. For this thesis, we will be reviewing literature where music has been used in sports and physical activity, because there are some benefits of exercise accompanied by music, as we will also present in this chapter.

2.1 Studies in music and mood

To be able to find reliable correlations between music and mood, it is important to have a good way of classifying emotion. The dataset that is available from MediaEval [50], will be reviewed in this section, and we will focus on how the categorization process for emotion in music has been done before.

2.1.1 1,000 Songs for Emotional Analysis of Music [50]

This paper is important to the thesis because the authors made a publicly available dataset for emotional analysis of music, which can be used in this thesis for a baseline of emotion in music. In this paper, they wanted to make dataset for music emotion recognition research, where they used 1000 audio clips from Free Music Archive and crowdsourced emotion annotation, using Amazon Mechanical Turk, for these clips using a parametric model with valence and arousal as the emotional

dimensions [38]. The annotations were made for the whole song (statically) and during the song (dynamically) using an online annotation interface. The static annotations of the whole clips were averaged to serve as the ground truth for static emotion estimation, and the dynamic annotations of the last 40 seconds of the clip, containing 40 values corresponding to each second, were averaged to serve as the ground truth for the dynamic emotion estimation. 700 clips were chosen as the training set and the remaining 300 clips as validation/test set. The estimation of arousal was far better than that of valence. The dataset has a total of 744 songs that have an emotion annotations, where the remaining 256 songs had unstable annotations and were discarded. This work can be used in this thesis as they have established a ground truth dataset for the valence-arousal dimensional model, where it can be used in the context of clustering features from these songs to see if there is a correlation between the features in music and the assigned emotional values. This again can be used to cluster new songs where the applicability of the new songs should have some affiliation to an activity, with for example selecting music from playlists designated to exercise or relaxing, or selecting music from papers listing songs as motivational for training sessions. The dataset from this paper will henceforth be referred to as the MediaEval dataset.

2.1.2 Music mood recognition: State of the art review [33]

This paper is related to the master thesis, as it summarizes the attempt to classify songs into one mood category based on audio features and social tags. The paper summarizes the research done by Bischoff et al. [4], and their attempt to classify songs into a mood category with a hybrid approach by using audio features classified by a Support Vector Machine, and social tags classified by a Naive Bayes Classifier. Both the audio features and the social tags used two different mood categorization models: Categorical models (MIREX mood clusters) and Parametric models (Thayer's Stress-Energy model). The MIREX Mood Clusters model presented in this paper was proposed by Hu, Xiao, et al. [19], with 5 mood clusters. The parametric model measure mood in a continuous multidimensional space or simple multidimensional metrics and not in discrete categories, where Thayer's Stress-Energy model [52] is a different version of the original model of valence-arousal proposed by Russel [38]. For audio feature extraction, Bischoff et al. [4] used audio file excerpts corresponding to the definition of *music piece*, also stated by Kim Youngmoo E., et al. [29], which is an entire song, a section of a song, a fixed length clip or a short term segment. In this case, a 30 second clip in MP3 format with 192 kbps was used. The results show that the classifiers using socials tags dominate the results of the audio-based classifier, although, the combination of both results shows improved overall results. In the context to this thesis, the mood classification models are relevant as they provide the context to how mood classification can be performed, and that there are different ways of grouping songs into moods. The classification of social tags and audio features relates to the work that has been done by MediaEval in

the previously reviewed paper [50]. Furthermore, this paper also lists some valuable points to take away, where among others, point two in the conclusion states that "the vast majority of approaches assume one mood per track, which does not correspond to the reality" [33] also stated by Wu, Bin, et al [58]. This is important to take note of, as the development of a reliable model would have to include the potential for varying mood in a songs, or simply more moods at once in a song.

2.1.3 Machine Recognition of Music Emotion: A Review [61]

This paper is related to the thesis because it tries to recommend how further research in Music Emotion Recognition should be conducted, and emotion in music is a part of the work that this thesis will work with. This paper tries to summarize the different methods used for Music Emotion Recognition (MER) and what the advantages and disadvantages of the different methods are, and how to conduct further research on the topic of MER based on reviewing a handful of literature. The second section of the paper is the conceptualization of emotion and how it is done in the categorical and dimensional model. As for the categorical model, there is no consensus among researchers as to what constitutes the right amount of categories for emotions, and that a vast number of categories could overwhelm the subject. The dimensional approach, as discussed in other papers, is the placement of emotion perceived on an emotion dimension with the commonly used Valence-Arousal axes [38]. The dimensional approach seems to be better than the categorical because the primary emotions that arise when listening to music can be placed on the valence-arousal scale, however, this approach seems to undermine some significant emotions such as anger or fear, where some researchers have adopted a third dimension, potency (dominant, submissive), but this also makes the task of emotion annotation more complicated. Further, the papers summarizes the different techniques used for extracting music features for energy, rhythm, melody, and timbre. Data collection of emotion annotation is complicated because of copyright issues so most of the researchers have made their own databases. For data processing, audio clips of 30 seconds compiled to a standard format have been used. The element of subjective annotation poses some problems as the emotion annotations will vary a lot, so music pieces that have no clear distinction are discarded. The subjective part also poses more problems such as culture, generation, sex, and personality. Different approaches are used for model training, but the best classification algorithms are Support Vector Machines and Gaussian Mixture Models. A fuzzy approach has been proposed to measure the strength of emotion, and to make an emotion have a likelihood of occurring in a music piece [62]. In the context of the thesis, it provides more evidence of using the valence-arousal dimensional model for finding correlations in music and mood. The paper also mentions some important points in the subjective manner that music is categorized. This will make it harder to make datasets that are reliable because of the subjective nature of emotion in music. This paper also contains a visualization of the valence-

arousal dimensional model with emotions placed around the axes to show what types of emotions can be described by positive/negative valence and high/low arousal. The emotions described within this model are not exact, but a fuzzy categorization of emotions. This type of model will be adopted in the thesis to be used as a metric in terms of categorizing songs into emotions.

2.2 Music and Exercise

A good way of finding songs for an activity would be to know the types of emotion that the very activity is classified by. We are going to use exercise and physical activity as the proposed first goal of utilizing a model that can find music of a certain mood. In this section, we will provide some related work on the topic of music that has been used in exercise, and what the types of approaches are for using music in exercise and why it has been used. It is easy to argue that music has the capability of improving exercise, but we are going review some literature on this area that tries to explore why music can have a positive effect when exercise is accompanied by the right type of music.

2.2.1 Music in the Exercise Domain: A Review and Synthesis (Part I and II) [24, 25]

These papers are related to the thesis because they explore a variety of research that investigates the effects of music in the exercise domain. They summarize research on the effects of music for exercise with varying parameters, consisting of three main categories: pre-task, in-task and post-task use of music. Further parameters are the task itself in terms of intensity, duration and mode, whether participants are trained or untrained, if the music is sedative or stimulative and what the measurement techniques are. For pre-task music, it has been shown that music can prepare exercisers both mentally and physically for the upcoming exercise. Most of the research summarized in the paper has focused on the asynchronous use of in-task music. Research shows that music lowers the rate of perceived exertion at low-to-moderate intensities of exercise by ~1̃0%, but not for high intensity exercises [36,51]. Self-selected or experimenter-selected music both results in reduction of perceived effort. Music appears to enhance affect at all exercise intensities but the studies are not consistent in the magnitude of enhancement [11, 20]. Music also appear to have an ergogenic (work-enhancing) effect at all exercise intensities, but especially at low-to-moderate intensities [12, 35]. The preferred tempo of music ranges from 125-140 bpm [22].

The second part of the paper covers the remaining research on in-task synchronous use of music, the longitudinal effects of music and post-task music. Music tempo in a treadmill-based experiment [42] showed that 2.7-2.8 Hz, or 162-168 bpm, was the optimally efficient frequency for running at various intensities, and participants' choice of music to

accompany the treadmill test ranged from 2.6-2.8 Hz, or 156-168 bpm. Both synchronization and the motivational qualities of music seems to be of benefit to high-intensity exercises according to a study [48]. A study [23] showed that music condition did not affect rate of perceived exertion at higher intensity levels, but the motivational condition had a strong influence on feeling states. Music may moderate how one feels, not what one feels, and therefore music can make high-intensity exercise more pleasurable [18]. Shaulov and Lufi [46] supported this when they found that music did not lead to greater energy expenditure, but participants reported higher states of pleasure during a music condition than that of a no-music control condition. A study from Karageorghis et al. [27] used motivational music, oudeterous (neutral) music, and a control condition (metronome) on exercisers during circuit-type exercises. Separating the genders, the study revealed that motivational music made both male and female exercisers perform better than when they listened to oudeterous music, but under the control condition, men performed as good as during the motivational music, and women as bad as the oudeterous music. In this study, women also reported more positive feeling states than men for both music conditions.

Part two of the paper [25] has general discussion section, which covers some important points to take away from this two-part paper. The majority of the studies report an ergogenic effect when music is used in-task. The increase in positive emotion due to music during high-intensity exercises may alter the exercise experience, and in turn increase attendance to exercise [1, 28, 44, 53]. Music appears to have greater effect when either the task or music selection is self-regulated. Music should have a tempo greater than 120 bpm and possess prominent percussive and rhythmical feature in order to have a stimulative effect.

This two-part paper provides a good review of a lot of research done on the effects of music in the exercise domain, and there are a few key points that would be relevant for this thesis. Overall, music seems to yield the best results at low-to-moderate exercise intensities. Arousal, the increase in heart rate, from motivational music pre-task might be an indication of an emotional state that is relevant when doing experiments, and it can be theorized that it is linked to the arousal dimension in the valence-arousal dimensional model for emotion in music. Overall, this paper gives a good review of the fact that music actually improves exercise, but especially at low-to-moderate exercise intensities. The notion that positive emotion can increase adherence to exercise is an important point and it would be beneficial for the thesis if we could find what type of music gives rise to this emotion.

2.2.2 Redesign and initial validation of an instrument to assess the motivational qualities of music in exercise: The Brunel Music Rating Inventory-2 [26]

The original Brunel Music Rating Inventory [28] is a tool for assessing the motivational qualities of music to exercise, where only experts could select

music for exercise. The BMRI-2 would allow both exercise instructors and participants to select music for exercise. The BMRI-2 is comprised of 6 items corresponding to the motivational qualities of the rhythm, the style, the melody, the tempo, the sound of the instruments and the beat of a music piece. These items are rated on a Likert scale from 1 to 7 based on how motivating they are. The sum of the perceived motivational qualities put on the Likert scale is the motivational quotient, which ranges from 6 to 42 where. Music with a motivational quotient from 36 to 42 is rated highly motivational, 24 to 35 is rated moderate, and under 24 is oudeterous [21]. Appendix B in this paper has a summary of how to perform the rating of a song using BMRI-2 [26]. This paper also lists three music pieces were rated using the original BMRI and later validated by test subjects using BMRI-2. The range of scores for the original BMRI ranges from 3.33-33.33, where scores below the middle range, 18.33, represent music as oudeterous (neutral) rather than demotivational. The first piece ”Out of Space” by The Prodigy was rated as oudeterous with a motivational quotient of 16.95, ”Back in My Life” by Alice Deejay was rated as moderately motivational with a motivational quotient of 22.15, and ”Set You Free” by N-Trance was rated as highly motivational with a motivational quotient of 26.18. These music pieces can be of use in terms of testing them on a trained machine learning algorithm to see if the arousal dimension in the valence-arousal model is related to music used in exercise.

2.2.3 The Brunel Music Rating Inventory-2 is a reliable and valid instrument for older cardiac rehabilitation patients selecting music for exercise [7]

This paper tests the reliability and validity of the BMRI-2 for older cardiac rehabilitation patients, where the motivational qualities of 100 popular songs were assessed by an expert panel using the BMRI-2. The music pieces consisted of 20 tracks from each of the 1940s, 1950s, 1960s, and 1970s, and also 20 classical pieces. Two lists were generated based on the four highest and lowest rated music pieces from each decade and classical selections, with 25 oudeterous and 25 motivating songs. The participants examined the 25 pieces in each of the motivating and oudeterous lists, and selected one motivating piece and one oudeterous piece for use in the study. Appendix A and B each contains a table with the 25 highest and lowest rated pieces respectively, along with the motivational quotient, with mean rating by experts and participants, and how many participants selected the piece for use in the study. The songs that are listed in the appendices of this paper can serve as part of a test set when looking for the emotional qualities in music for exercise, when used with a machine learning algorithm.

2.2.4 Summary

The literature we have reviewed in this section provides some solid background into the use of the valence-arousal dimensional model. The MediaEval dataset [50] has 744 songs annotated with emotion according

to this model, so this can be used with a machine learning approach to find correlations between the features of the songs and the moods, to try to answer the first question in the problem statement 1.2. The applicability of music categorized by mood is review by the literature on music used in the exercise domain, where the BMRI-2 can be used further to answer the third question in the problem statement 1.2. As mentioned, a machine learning approach can be used with the MediaEval dataset [50], and we are going to look at the options for this approach in the next section.

2.3 Machine Learning

The term Machine Learning was originally coined by Arthur Samuel who created an algorithm for playing checkers [40]. Machine Learning is the scientific study of using algorithms and statistical models to perform certain tasks without specific instructions, and has the ability to learn features from sample data in order to make a model that has the ability to perform predictions. Unsupervised learning is a sub-category of machine learning, where it is focused around discovering patterns in data that has no labels [5, 14].

2.3.1 Dataset

To answer the first question in the problem statement 1.2, we are going to explore the correlations between the songs in a dataset and their annotated emotions, by applying a variety of clustering algorithms to the features extracted from each song in the dataset. The dataset we are using is from MediaEval [50] and will be explored with different unsupervised clustering approaches. The data is annotated with emotions on the valence-arousal dimensional model, which is also reviewed in the related works section, with both valence and arousal values ranging from 0 to 10. More detail on the dataset can be found in section 2.1.

2.3.2 Clustering

Cluster analysis is a type of unsupervised learning, where a clustering algorithm groups similar data into clusters. In this section, we are going to present a number of clustering algorithms that each will be applied to the dataset in order to see if there is a correlation between music and mood. We will summarize the basic principles for these clustering algorithms and how they work. The clustering algorithms presented in this section are all found in the scikit-learn[1] machine learning library for Python. We chose the clustering algorithms based on different properties such as if they can perform hierarchical clustering or if the number of clusters has to be predefined [60].

[1] https://scikit-learn.org/

Mean Shift

Mean shift is an algorithm that analyses the feature-space for density of samples. It works by shifting candidates for centroids towards the maximum density of samples withing a given region based on the mean of the data points, and it stops when the shift in centroids is small enough. The algorithm have no parameter for specifying cluster numbers, but the bandwidth parameter determines the size of the region to search through [6]. Mean Shift is mostly used for cluster analysis in computer vision and image processing [8].

Mini Batch K-means

Mini Batch K-means is an optimized version of the standard K-means clustering algorithm in terms of computation time. Mini Batch K-means needs a specified number of clusters before running the algorithm. K-means operates on the whole dataset for each iteration of the algorithm, where Mini Batch K-means uses a random subset, hence mini batch, from the dataset of a fixed size for each iteration. The are mainly two steps for Mini Batch K-means. In the first step, the random samples are drawn from the dataset and assigned to the nearest centroid. In the second step, for each sample in the mini batch, the centroids are updated using a learning rate that decreases with the number of iterations, which in turn decreases the rate of change for a centroid over time. These steps are repeated until the rate of change for centroids is small enough, or a predetermined number of iterations is reached. Mini Batch K-means saves a lot of computation time, but have slightly worse results than the standard K-means [3, 45].

Agglomerative Clustering

Agglomerative Clustering is a type of hierarchical clustering where the observations starts within each cluster, and then merging clusters up the hierarchy [37]. The result of running this algorithm is a dendrogram, or a tree, of subclusters. The merging of clusters will stop when the specified number of clusters is reached. When this algorithm merges clusters it uses a linkage criterion that is specified before running the algorithm. The criteria are various types of distances between a pair of observations, and are of the following types:

Single linkage: combines clusters based on the minimum distance of the closest elements between clusters.

Complete linkage: combines clusters based on the minimum distance of the elements farthest away between clusters.

Average: combines clusters based on the minimum distance of the average distances of elements in a cluster, between clusters [49].

Ward: combines clusters so that the variance within the combined cluster is the minimum of all pairs of clusters [56].

Spectral Clustering

Spectral clustering performs dimensionality reduction on the data before clustering, and the clustering is done by using a standard clustering algorithm, such as K-means. The procedure for this algorithm consists of computing the Laplacian matrix on the similarity matrix of the data. The clustering is performed on relevant Eigenvectors of the Laplacian matrix [55]. The similarity matrix is provided as a parameter to the algorithm. This clustering algorithm also needs cluster numbers to be specified. In general, Spectral Clustering is useful when the data points are structured in a non-convex way. This algorithm also has a variety of applications in image segmentation.

Affinity Propagation

Affinity Propagation has no parameter for specifying number of clusters, but the algorithm creates clusters by sending messages between pairs of samples, where the samples in a cluster is represented by an exemplar. The algorithm iteratively send messages of two different categories between each other. The first is the responsibility message, which is accumulated evidence that one sample k should be the exemplar of another sample i. The second is the availability message, which is accumulated evidence that sample i should choose sample k as its exemplar. Matrices for these two types of messages are updated based on the similarity between points, which is usually the negative squared euclidean distance. The iterations are performed until there are no change in the cluster boundaries, or after a predetermined number of iterations. At the end, the exemplars are then chosen based on the final matrices. The parameter for preference specifies how likely a data point is to be chosen as an exemplar, which in turn can influence the number of clusters the algorithm creates. The damping factor is a parameter that is there to avoid numerical oscillations when updating the messages between samples, and is usually between 0.5 and 1 [13].

Birch

The Birch algorithm builds a Clustering Feature (CF) tree, where the input data is compressed to a set of CF nodes that hold the necessary information. These nodes have the information on number of samples in a subcluster, the linear sum of the samples in the subcluster, the squared sum of the samples in the subcluster, centroids and the squared norm of the centroids. The algorithm operates in four stages, where the second and fourth stages are optional. The first stage builds a height-balanced tree data structure out of the data points. The CF tree is specified by the parameters threshold and branching factor. The threshold limits the distance between an entering sample and the existing subcluster and the branching factor is the maximum number of subclusters in each node. The second stage is optional, which includes scanning the leaf entries in the CF tree and building a smaller CF tree, removing outliers and grouping

crowded subclusters into larger ones. Stage three consists of using an existing clustering algorithm to cluster the subclusters from the leaves, treating the subclusters as new samples. Agglomerative Clustering is used in the third stage when number of clusters is specified before running the Birch algorithm. The fourth stage, which is optional, uses the centroids of the clusters produced in the third stage as seeds to redistribute the data points according to their closest seed to form new clusters [64].

Gaussian Mixture

A Gaussian Mixture model is a probabilistic model that assumes all data points are generated from a mixture of a set of Gaussian distributions. The clusters in this algorithm are defined as components, where the components are a the Gaussian distributions. However, this algorithm does not know which component the data points belongs to. The expectation-maximization (EM) algorithm is an algorithm that assumes which component a data point belongs to. The EM algorithm is an iterative process that finds the maximum likelihood of a data point belonging to a component. The algorithm first computes the probability that a data point belongs to a component. Then it sees what points influence the distribution the most, based on the probability values. These steps are repeated until convergence. The Gaussian Mixture algorithm has parameter for different types of covariances [10, 63].

2.4 Summary

This section summarizes research that has been done in the area of music and mood, as well as research for music used in the exercise domain. The literature on mood classification suggests that the valence-arousal dimensional model is a properly devised model, and that it has the capability of representing emotions in fuzzy categories, where the quadrants of the valence-arousal dimensions are representative of distinct emotions based on the parameters. To answer the problem statements in section 1.2, the valence-arousal dimensional model will be used to determine the classification properties of an unsupervised approach to machine learning, where the valence-arousal dimensions, and the quadrants therein, will serve as the determinant metric for clustering features from songs. Even though the approaches to using the valence-arousal dimension in this way, the fuzzy categories of the emotions can represent more emotions than one, and the quadrants of the dimensional model will not be fully discriminant this way, but will base the statistical summary of their contents to look for optimal clusters with a likelihood of an emotion of occurring. We have suggested an unsupervised learning approach to finding correlation between music and mood, where the clustering algorithms that has been provided in this chapter will be used for this purpose. The MediaEval dataset [50] will also be used in various ways with the clustering algorithms to cover as many approaches to the

problem statement 1.2 as possible, and we will talk more about this in the following chapter. The literature on music in the exercise domain is a further goal for utilizing the model we will create, that is also mentioned in the problem statement 1.2. An activity is usually associated with a type of mood, where the mood is represented by the valence-arousal dimensions of the model, and therefore music for an activity, such as exercise and physical activity, can validate the clustering approach to songs. The BMRI-2 which is reviewed, can be of utility when trying to categorize new songs, where it can be theorized that highly motivational songs would be high in the arousal dimension and an oudeterous, or neutral songs, would be low in arousal.

Chapter 3

Methodology

Considering the problem statement 1.2 with the related work from the previous chapter 2.1 in mind, this chapter is the procedure to find the solution. In this chapter, we will discuss the different approaches used to finding solutions to the problem statement 1.2. We will focus on the implementation and the design of the system, and what libraries, parameters and metrics that has been used.

3.1 Preparing Data

The first objective in this thesis includes finding data that we can use to explore the correlation between music and mood, and to theorize how this data can be processed and used. As mentioned in the previous chapter 2.1, MediaEval has crowdsourced a dataset that includes 1,000 songs with annotated emotion on a valence-arousal dimensional model [50]. Going forward, this dataset will serve as the baseline for the work conducted in this thesis.

3.1.1 Dataset

The dataset made by MediaEval [50] is publicly available[1]. As the content was used in a different way in the paper, this dataset is suitable for the research that will be conducted in this thesis, as it serves as a ground truth dataset for the valence-arousal dimensional model. Out of the 1000 songs that was included in the crowdsourcing experiment performed by MediaEval, there are 744 songs that has emotion annotation attached to them.

The dataset includes dynamic annotations and static annotations for emotions. For the purpose of this thesis, the static annotations will be used, as they account for the overall emotional quality for each song, and can therefore be used as a dimensional representation of the songs. The emotional annotations consists of valence and arousal values ranging from 0 to 10, where valence is usually represented by the x-axis, and arousal by the y-axis. The static annotations are listed as follows:

[1] http://cvml.unige.ch/databases/emoMusic

- Song ID
- Mean value for arousal
- Standard deviation for arousal
- Mean value for valence
- Standard deviation for valence

Aside from the full clips and the 45-second clips, there is info about all the songs in the dataset. The info is listed as follows:

- Song ID
- Filename
- Artist
- Song title
- Start of the segment (minutes.seconds)
  For the 45-second clips
- End of the segment (minutes.seconds)
  For the 45-second clips
- Genre

3.1.2 Feature Extraction

To use the dataset from MediaEval, we had to assess what it was going to be used for. We touched on the topic of using audio signal processing to make a visual representation of each song, a spectrogram, and then use convolutional neural network for image processing to make a machine learning model to recognize emotion in music. After some discussion on the topic, we decided to try to find inherent correlation between music and mood based on the features in music, and from that, the idea of using clustering as a means to explore correlation between music and mood arose.

Different tools were considered for audio feature extraction of the MediaEval dataset, namely OpenSMILE and pyAudioAnalysis [16]. The latter was chosen as our means of extracting features from the dataset. pyAudioAnalysis allowed us to extract features into different matrices. The matrices are ordered according to their corresponding time-frame and feature, with the signature described in table 3.1.

Zero Crossing Rate is the number of times a signal changes sign, i.e., going from positive to negative or vice versa, for a given time frame [34]. Zero Crossing Rate is a feature that can been used to identify percussive sounds [17] and detecting human speech in audio clips [47].

Row	Feature Name
1	Zero Crossing Rate
2	Energy
3	Entropy of Energy
4	Spectral Centroid
5	Spectral Spread
6	Spectral Entropy
7	Spectral Flux
8	Spectral Rolloff
9-21	MFCCs
22-33	Chroma Vector
34	Chroma Deviation

Table 3.1: Description of feature vectors when extracting features using pyAudioAnalysis.

Energy is the energy of the signal. This is computed using the sum of squares on the signal values, averaged by the specified time frame [34].

Entropy of Energy is defined as the measurement of abrupt changes in a signals given time frame [16].

Spectral Centroid is the center of gravity of the spectrum. Spectral centroid is often used in audio signal processing to automatically identify the timbral brightness, or tone quality [34, 43].

Spectral Spread is associated with the bandwidth of the signal. Noisy signals have a larger spectral spread than that of isolated tonal sounds. The spectral spread is the spread of the spectrum around the mean value, i.e., the variance around the spectral centroid [34].

Spectral Entropy is the entropy of the normalized spectral energies for a set of sub-frames [16]

Spectral Flux is defined as the average difference of the power spectrum of a signal between consecutive frames [30]. Spectral Flux can be useful in distinguising music and speech signals, as there is more change in music [41]. In pyAudioAnalysis, Spectral Flux is calculated by the square difference between the normalized magnitudes of the spectra of the two successive time frames [16]

Spectral Rolloff defines the frequency of below which a percentage of the total spectral energy is concentrated. The percentage of spectral energy below the rolloff point varies with different implementations

of this feature, but is usually from 85% [54] to 95% [41]. This feature is good for distinguishing voiced from unvoiced speech, where the percentage of signal energy for unvoiced speech and music are contained in the lower bands [41]. The rolloff point that is used in pyAudioAnalysis is the frequency where 95% of the singals energy is below [16].

Mel-Frequency Cepstral Coefficients (MFCCs) are coefficients that represent the mel-frequency cepstrum [59]. The mel-frequency cepstrum is the cepstrum computed on the frequencies on a mel-scale instead of a fourier spectrum [34]. The mel-scale is a logarithmic representation of frequencies, where the frequency bands are equally spaced [59]. The mel-frequency cepstrum is the discrete cosine transform of the log powers of the mel-frequencies [34, 39]. MFCCs are often used in speech recognition [15], and have uses in music information retrieval for finding timbral characteristics as well as genre classification [31]. pyAudioAnalysis has a vector of 13 coefficients representing the mel-frequency cepstrum [16].

Chroma Vector is a 12-element feature vector representing the twelve pitch classes of western-type music, where each element represents how much energy of the signal is present in that pitch class in a given time frame [16]. A pitch class, or a chroma, is the set of all octaves for a given tone. Chroma features are frequently used tool for music data processing and analysis because of its robustness in identifying harmonic features in music [32].

Chroma Deviation is the standard deviation of the 12 chroma coefficients [16].

The feature extraction was performed on whole folders at a time. Feature extraction was performed on WAV files. The feature extraction needed to be used on WAV files, and since the dataset comes in MP3 file format, it was simply converted using FFMPEG as shown in listing. After the conversion of the folder was complete, the conversion from MP3 to WAV had to be performed. See listing 3.2.

```
1    for i in *.mp3;
2        do name=‘echo $i | cut -d’.’ -f1‘;
3        echo $name;
4        ffmpeg -i "$i" "${name}.wav";
5    done
```

Listing 3.1: FFMPEG conversion of all files in a folder from MP3 to WAV

```
1    python audioAnalysis.py featureExtractionDir -i ../
    clips_full/
2        -mw 1.0 -ms 1.0 -sw 0.050 -ss 0.050
```

Listing 3.2: Feature extraction of all WAV-files in a folder

3.1.3 Data Preprocessing

After the feature extraction has been performed on the songs in the dataset, the idea is to use the data with the clustering algorithms in different ways. There were mainly three ways that we planned using the data with the clustering algorithms:

- (i) Clustering features from 45-second clips.
- (ii) Clustering statistical summary of features from whole clips.
- (iii) Clustering features from whole song divided into equal sample size based on the length of the song.

As stated before, 744 of the 1,000 songs were annotated with emotions, where the rest had too unstable emotion annotation and were discarded.

(i) Out of the remaining 744 songs, 6 songs had a marginal difference in length. These songs had more or less than 50 milliseconds in length and were not included in further research for this method. The structure of the data from the feature extraction can be seen in table 3.2. The total number of features for this is 34 features $*$ 900 samples $=$ 30600. This is too many features and results in too high dimensionality when clustering. When clustering this data, we employed Principal Component Analysis, which reduces the number of samples to avoid the curse of dimensionality. PCA reduces the number of samples to Principal Components that accounts for the variability of a percentage of samples [57].

	Feature 1	$\cdots$	Feature 34
Sample 1	*data*	$\cdots$	*data*
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sample 900	*data*	$\cdots$	*data*

Table 3.2: Data structure of the extracted features of the used 45-second clips. Features correspond to table 3.1, and samples per 50 milliseconds of the 45-second clips.

```
#Apply PCA to dataset
pca = PCA(.95)

X = pca.fit_transform(X)
```

Listing 3.3: Principal Component Analysis where the Principal Components account for 95% of the sample variance

(ii) In this method, the features are extracted from audio clips of the whole songs. The songs are corresponding to the 45-second clips in method (i). The features extracted from the whole songs are structured as shown

in table 3.3. The statistics that are going to be used per feature vector are average, mean, standard deviation and skewness. Each feature vector are summarized according to the statistics mentioned, and then concatenated into a single vector, as shown in listing 3.4. This results in a vector of 34 features $*$ 4 different statistics $=$ a 136 long vector of statistical summarized features that are on the form shown in table 3.4.

	Feature 1	$\cdots$	Feature 34
Sample 1	*data*	$\cdots$	*data*
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sample $l/50ms$	*data*	$\cdots$	*data*

Table 3.3: Data structure of the extracted features of whole songs. Features correspond to table 3.1, and samples per 50 milliseconds of the whole song. l is the duration of the song.

```
# n annotated songs = 744
# 34 features * (mean + average + stdev + skewness)
dataset = np.empty([744, 136])

for i, j in enumerate(annotations[:, 0]):
    path = 'clips_full/{0}.wav_st.npy'.format(str(int(j)))
    data = np.load(path)

    averages = np.average(data, axis=1)
    means = np.mean(data, axis=1)
    stdevs = np.std(data, axis=1)
    skewness = scipy.stats.skew(data, axis=1)

    dataset[i] = np.concatenate((averages, means, stdevs,
    skewness))
```

Listing 3.4: Code for making statistical summary of features for mean, average, standard deviation and skewness.

(iii) This method was discarded due to insufficient support for large time-frames in pyAudioAnalysis' method for feature extraction.

3.2 Clustering

Clustering is used in this thesis as a means to explore the correlation between features in music and mood in music. As the already processed data was done using python, the scikit-learn library for python has been used for clustering.

The work that was done with clustering is inspired by the tutorial on

how to perform clustering with scikit-learn for python[2]. Furthermore, methods for processing data and plotting figures has been added to the code template. All code for the clustering and visualization produced for this thesis is made open available at GitHub[3].

The clustering in itself has been conducted with a variety of clustering algorithms, and these algorithms have been reviewed in the background chapter in section 2.3.2. These clustering algorithms have a set of parameters that are configured within the code. The most important parameter we used when experimenting with these algorithms were the number of clusters. The parameter for specifying the number of clusters are provided as input when running the scripts. Some of the clustering algorithms that have been used have no option for pre-defining a number of clusters, but rather make the clusters based on the data it observes. These clustering algorithms have been included in the experimentation as we wanted to see the metrics, in terms of statistics, cluster separation and genre inclusion. For the final prototype, these clustering algorithms have been discarded as they do not provide a means to predict clusters for new data.

In terms of parameters, the values that have been changed before clustering is the number of clusters the algorithm creates. Plots of the clustering has also been done. These plots are based on the valence-arousal emotional annotation for the songs. This has been done to illustrate how well the clustering algorithms perform in making clusters of songs that are relatively separated according to the valence and arousal of the songs in the cluster. We are going to go through the clustering algorithms and their parameters used in the methods presented in section 3.1.3:

Mean Shift: This clustering algorithm has no option for defining clusters, but it makes clusters based on ”blobs” in the data. The initiation of Mean Shift uses the parameters for estimating bandwidth and if it uses bin seeding. Initiation can be viewed in listing 3.5. Parameters to this class can be viewed on the scikit-learn homepage[4].

```
1         ms = cluster.MeanShift(
2             bandwidth=bandwidth,
3             bin_seeding=True)
4
```

Listing 3.5: Initiation of cluster object for Mean Shift. This includes parameter for estimating bandwidth, which can be viewed in listing 3.6. Parameter for bin seeding has been set to true, which sets the initial kernel location to the discretized version of points binned onto a grid corresponding to the bandwidth's coarsness.

[2] http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

[3] https://github.uio.no/marasand/thesis

[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html#sklearn.cluster.MeanShift

```
        # estimate bandwidth for mean shift
        bandwidth = cluster.estimate_bandwidth(
            X,
            quantile=params['quantile'])

```

Listing 3.6: Estimation of bandwidth to use with Mean Shift clustering. X is defined as the dataset. Quantile is set to 0.3, which is also the default quantile for this function.

Mini Batch K-means: The initiation for Mini Batch K-means only has parameter for number of cluster. Initiation can be viewed in listing 3.7. Parameters to this class can be viewed on the scikit-learn homepage[5].

```
        mini_batch = cluster.MiniBatchKMeans(
            n_clusters=params['n_clusters'])

```

Listing 3.7: Initiation of cluster object for Mini Batch K-means. n_clusters is the only hyperparameter and it indicates the number of clusters to form as well as the number of centroids.

Agglomerative Clustering: Agglomerative clustering has parameters for number of clusters, linkage, affinity and connectivity. Initiation can be viewed in listing 3.8. Parameters to this class can be viewed on the scikit-learn homepage[6]. Parameters for Kneighbors graph can also be viewed on the scikit-learn homepage[7]

```
        average_linkage = cluster.
    AgglomerativeClustering(
            linkage="average",
            affinity="cityblock",
            n_clusters=params['n_clusters'],
            connectivity=connectivity)
```

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans

[6] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering

[7] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.kneighbors_graph.html#sklearn.neighbors.kneighbors_graph

```
6
```

Listing 3.8: Initiation of cluster object for Agglomerative Clustering. n_clusters is the number of clusters to find. Linkage determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion. average uses the average of the distances of each observation of the sets. Affinity is a metric used to compute linkage, which is set to ”cityblock”. Connectivity sets for each sample the neighboring samples based on the structure of the data, where connectivity is set as shown in listing 3.9

```
1         # connectivity matrix for structured Ward
2         connectivity = kneighbors_graph(
3             X,
4             n_neighbors=params['n_neighbors'],
5             include_self=False)
6
7         # make connectivity symmetric
8         connectivity = 0.5 * (connectivity +
    connectivity.T)
9
```

Listing 3.9: Connectivity matrix formed giving the dataset to a kneighbors_graph. X is the dataset. Parameter n_neighbors are defined as 2. include_self is used for marking if a sample is the first nearest neighbor to itself, which in this case it is set to False.

Ward: Ward is a linkage type of Agglomerative Clustering which is defined by the linkage parameter to the class. ”Euclidean” is the only accepted affinity parameter when the linkage type is ward. This clustering algorithm also has a parameter for the number of clusters. Initiation of this clustering algorithm can be viewed in listing 3.10. Parameters to this class can be viewed on the scikit-learn homepage[8]

```
1         ward = cluster.AgglomerativeClustering(
2             n_clusters=params['n_clusters'],
3             linkage='ward',
4             connectivity=connectivity)
5
```

Listing 3.10: Initiation of cluster object for Ward. n_clusters is the number of clusters to find. Ward linkage minimizes the variance of the clusters being merged. When ward is used for the linkage parameter, the only affinity parameter that is allowed is ”euclidean”, which is the default parameter. Connectivity is the same as for the Agglomerative Clustering object, as shown in listing 3.9

[8] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering

Spectral Clustering: This clustering algorithm has parameters for number of clusters, affinity and eigenvalue decomposition strategy. Initiation can be viewed in listing 3.11. Parameters to this class can be viewed on the scikit-learn homepage[9].

```
        spectral = cluster.SpectralClustering(
            n_clusters=params['n_clusters'],
            eigen_solver='arpack',
            affinity="nearest_neighbors")

```

Listing 3.11: Initiation of cluster object for Spectral Clustering. n_clusters is the dimension of the projection subspace. eigen_solver is the eigenvalue decomposition strategy, which in this case is ARPACK. Affinity is set to "nearest_neighbors", which is a string argument accepted by this class.

Affinity Propagation: The parameters that has been used for this clustering algorithm are damping and preference. Affinity Propagation has no parameter for defining number of clusters. However, this algorithm makes exemplars based on the input data, which then is influenced by the preference value. Initiation can be viewed in listing 3.12. Parameters to this clustering algorithm can be found on the scikit-learn homepage[10].

```
        affinity_propagation = cluster.
    AffinityPropagation(
            damping=params['damping'],
            preference=params['preference'])

```

Listing 3.12: Initiation of cluster object for Affinity Propagation. Damping can be set from 0.5 to 1 and is a factor to what extent the current value is maintained relative to incoming values. Incoming values are weighted $1 -$ damping. In this case, damping is set to .9. Preference is set to -200.

Birch: This algorithm constructs a tree data structure with subclusters centroids at the leaf. The parameter that has been used for this clustering algorithm is the number of clusters. Initiation can be viewed in listing 3.13. Parameters to this clustering algorithm can be found on the scikit-learn homepage[11].

[9] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering

[10] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html#sklearn.cluster.AffinityPropagation

[11] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html#sklearn.cluster.Birch

```
        birch = cluster.Birch(
            n_clusters=params['n_clusters'])

```

Listing 3.13: Initiation of the cluster object for Birch. The parameter n_clusters is the number of clusters this algorithm uses with a new algorithm, namely Agglomerative Clustering, after its inital clustering steps.

Gaussian Mixture: This clustering algorithm uses the parameters for number of clusters and the covariance type. Initiation can be viewed in listing 3.14. Parameters to this clustering algorithm can be found on the scikit-learn homepage[12].

```
        gmm = mixture.GaussianMixture(
            n_components=params['n_clusters'],
            covariance_type='full')

```

Listing 3.14: Initiation of the cluster object for Gaussian Mixture. n_components specifies the number of components, i.e. clusters. covariance_type is set to 'full', which specifies that each component has its own general covariance matrix.

After the clusters have been initiated, the objects of the clustering algorithms are put in tuples with a descriptive name that matches the algorithm. All the tuples of algorithms are then put in an outer tuple, as shown in listing 3.15, to simplify the iteration of clustering algorithms.

```
    clustering_algorithms = (
        ('MiniBatchKMeans', mini_batch),
        ('AffinityPropagation', affinity_propagation),
        ('MeanShift', ms),
        ('SpectralClustering', spectral),
        ('Ward', ward),
        ('AgglomerativeClustering', average_linkage),
        ('Birch', birch),
        ('GaussianMixture', gmm)
    )
```

Listing 3.15: Tuple of objects of clustering algorithms.

The procedure of clustering uses the fit function of the algorithms, as seen in listing 3.16

```
for name, algorithm in clustering_algorithms:

    # catch warnings related to kneighbors_graph
```

[12] https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture

```
    with warnings.catch_warnings():
        warnings.filterwarnings(
            "ignore",
            message="the number of connected components
of the " +
            "connectivity matrix is [0-9]{1,2}" +
            " > 1. Completing it to avoid stopping the
tree early.",
            category=UserWarning)
        warnings.filterwarnings(
            "ignore",
            message="Graph is not fully connected,
spectral embedding" +
            " may not work as expected.",
            category=UserWarning)
        algorithm.fit(X)
```

Listing 3.16: Code for interating through clustering algorithms and performing clustering with the fit function to the algorithms. X is the dataset being given to the algorithms. This code also catches warnings when using kneighbors_graph for Agglomerative Clustering.

3.3 Metrics

A variety of metrics has been used in this thesis to determine how the clustering algorithms perform. Cluster separation on the valence-arousal dimensional model has been considered as the most apparent metric in this thesis. Other metrics have been chosen after some clustering has been performed to determine what parameters to look at. Info given by the MediaEval dataset [50] on the songs included has provided a lot of parameters for evaluation. The number of clusters will be a parameter together with the different clustering algorithms to see what combination of algorithms and cluster numbers perform best in terms of the metrics described in this section.

3.3.1 Valence-Arousal Dimensional Model

The valence-arousal dimensional model will serve as a baseline for discovering correlations between music and mood. This model is also included as a way of visualizing the data from the clustering experiments. The valence-arousal model can be viewed in terms of quadrants, as shown in one of the papers [61] reviewed in section 2.1, where we have adopted the model with fuzzy emotion categories that can be seen in figure 3.1. This way of and we will use the quadrants as a metric to see how many songs in a cluster is in one quadrant compared to other quadrants, where the quadrants serve as a representation for these emotions. Quadrant 1 represent emotions such as excited, happy and please, and have high arousal and positive valence. Quadrant 2 represents emotions such as annoying, angry and nervous, and have high arousal and negative valence.

Quadrant 3 represents emotions such as sad, bored and sleepy, and have low arousal and negative valence. Quadrant 4 represents emotions such as relaxed, peaceful and calm, and have low arousal and positive valence.

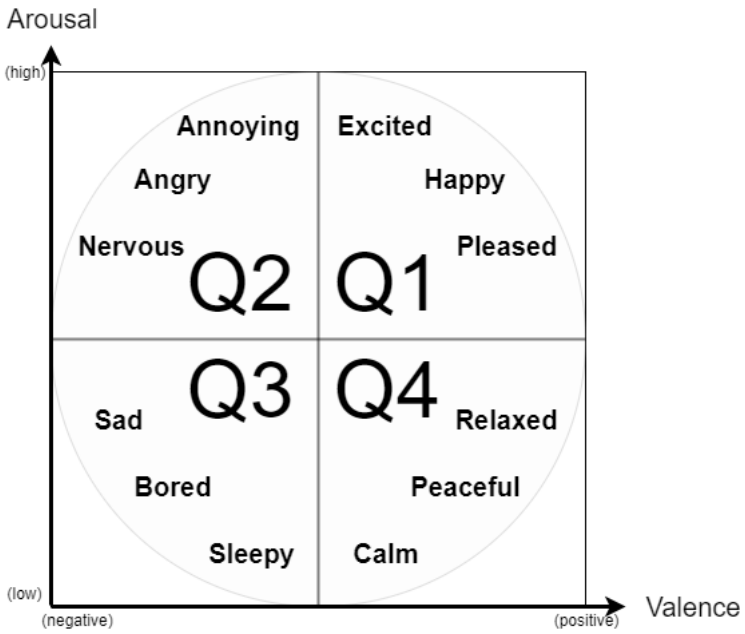


Figure 3.1: Overview of quadrants for the valence-arousal dimensional model. Quadrant 1: Top-right, high arousal/positive valence, 5-10 arousal/5-10 valence, excited, happy, pleased. Quadrant 2: Top-left, high arousal/negative valence, 5-10 arousal/0-5 valence, annoying, angry, nervous. Quadrant 3: Bottom-left, low arousal/negative valence, 0-5 arousal/0-5 valence, sad, bored, sleepy. Quadrant 4: Bottom-right, low arousal/positive valence, 0-5 arousal/5-10 valence, calm, peaceful, relaxed.

3.3.2 Music Genres

The MediaEval dataset [50] has info on genres with each song in the dataset. Music in general is grouped into genres because of variety in musical components, therefore genres may provide valuable as a metric to find correlation between music and mood. The music genres that are included in the dataset are as follows:

- Blues
- Classical
- Country
- Electronic
- Folk

- Jazz
- Pop
- Rock

3.3.3 Number of clusters

The number of clusters will be the primary parameter for testing clustering algorithms with the extracted features from the songs. The outcome of the clustering will vary when we test the different numbers of clusters, as the clustering algorithms may need a high or low amout to perform better in separating clusters into quadrants of the valence-arousal dimensional model. If the cluster number is four, the clustering algorithm may be able to separate songs into quadrants entirely, or maybe the a higher number of clusters are needed. The number of clusters may also be useful in representing genres. There are eight genres in the dataset we use, so if we for example use eight clusters, the clustering algorithm may separate genres entirely. For the experiments, the number of clusters that will be used are ranging from 2 to 10.

3.3.4 Post-processing

After the clustering has been performed on the data in the different ways, specified in section 3.1.3, the info from the clustering is gathered. For each clustering algorithm used per cluster number, the info is stored in a CSV file on the format shown in table 3.5. The information will be saved on this format so it is easy to compare the metrics and clustering algorithms to determine the performance of the experiments.

3.4 summary

In this chapter, we have come up with different methods for approaching the problem statements in section 1.2. The first part of this chapter gives an overview of the approaches when using the MediaEval dataset [50] in terms of feature extraction and ways of applying the features to clustering algorithms. The data pre-processing is the first step towards making the various clustering approaches, where the data has been used, as mentioned in section 3.1.3, with bullet point (i) for clustering features of 45-second clips with PCA, and with bullet point (ii) for clustering features of whole clips using statistically summarized features for clips of whole songs. We have also listed the different clustering algorithms that will be used on the different pre-processed data and their parameters. Furthermore, the metrics that are going to be used have been defined, where the quadrants of the valence-arousal dimensional model will be the primary metric for determining the performance of the clustering algorithms. In the next chapter, we will describe the design of experiments based on the methods of pre-processing data, clustering procedures and metrics we have described in this chapter.

	Average Feature 1	...	Average Feature 34	Mean Feature 1	...	Mean Feature 34	Standard Deviation Feature 1	...	Standard Deviation Feature 34	Skewness Feature 1	...	Skewness Feature 34
Summary of samples	*data*	...	*data*	*data*	...	*data*	*data*	...	*data*	*data*	...	*data*

Table 3.4: Structure of the dataset containing the statistical summary of each feature of the whole songs. Features are listed according to table 3.1.

Song number	Filename	Artist	Song Title	Start of segment	End of segment	Genre	MediaEval 2013 set	Song number	Mean arousal	Standard Deviation arousal	Mean valence	Standard Deviation valence	Cluster number
data	...	...	...	...	...	...	...	...	...	...	...	...	*data*

Table 3.5: Structure of the CSV file that is saved to disk after the clustering procedures.

Chapter 4

Experiments, Results & Discussion

The previous chapter contained procedures on the design of the system to explore solutions to the problem statement in section 1.2. In section 3.1.3, we defined the some methods for approaching the problem statement. In this chapter we will discuss the approaches to the design of these experiments, and also look at the final design of the prototype for this thesis. Furthermore, we will discuss the results that emerge from the clustering experiments in term of the metrics defined in section 3.3.

4.1 Design of Experiments

The main goal of the experiments is to gather as much empirical data as possible to find what combination of parameters results in the best clustering of data. The experiments are designed to provide a good amount of statistics, given the dataset from MediaEval [50] and the clustering algorithms. There are mainly two categories of experiments derived from the approaches mention in section 3.1.3:

- (i) Clustering features from 45-second clips with Principal Component Analysis
- (ii) Clustering statistical summary of features from whole song clips

4.1.1 Clustering procedures

The two main categories of experiments are split into two different python files for each clustering procedure:

feature_clustering.py: This file is created according to bullet point (i) in 4.1. The pre-processing of data is handled together with Principal Component Analysis first [57]. After the pre-processing, the data is clustered according to the clustering procedure mention in section 3.2.

stats_clustering.py This file is created according to bullet point (ii) in 4.1. The pre-processing consists of summarizing the features of each song statistically. After the pre-processing, the data is clustered according to the clustering procedure mention in section 3.2.

For each iteration of the experiments above, the number of clusters is changed from 2 clusters up to 10 clusters. After the clustering, the clustered data, the emotion annotations and the information on the songs are saved in a CSV file. Plots of the clustering are saved as SVG files. The experiments mentioned above results in 18 folders containing plots and post-processed data from the clustering. Each of the files above results, when running experiments from cluster numbers 2 to 10, results in 9 folders each, with one folder per number of clusters specified before running the script. The CSV files in these folder are saved on the format specified in table 3.5.

The following python files implements processing of data files to summarize statistics for genres and clusters, and uses the CSV files created from the clustering procedures.

parse_quadrants.py: This file parses the information of the clustering and makes a table of songs in each quadrant per cluster. The information is stored in a CSV file, on the form shown in table 4.1, per cluster in the respective experiments with varying cluster numbers.

parse_genres.py: This file parses the information of the clustering and makes a table of songs in each genre per cluster. The information is stored in a CSV file, on the form shown in table 4.2, per cluster in the respective experiments with varying cluster numbers.

parse_summary.py: This file takes both the quadrant and the genre info, and makes a new CSV file that has info for all clusters and clustering algorithms in it. There will be generated one CSV file per experiment to compare the experiment specific parameters and metrics. The output is designed to be easy to read and process when applying statistical analysis to it. The structure of the CSV file can be seen in table 4.3.

Songs in cluster			
Q1	Q2	Q3	Q4
Songs in Q1	Songs in Q2	Songs in Q3	Songs in Q4
Percentage in Q1	Percentage in Q2	Percentage in Q3	Percentage in Q4

Table 4.1: Structure of CSV file for one cluster with info on songs per quadrant in this cluster. Q's are the respective quadrants of the valence-arousal dimensional model.

Songs in cluster							
Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock
Songs in genre	Songs in genre	Songs in genre	Songs in genre	Songs in genre	Songs in genre	Songs in genre	Songs in genre
Percentage in genre	Percentage in genre	Percentage in genre	Percentage in genre	Percentage in genre	Percentage in genre	Percentage in genre	Percentage in genre

Table 4.2: Structure of CSV file for one cluster with info on songs per genre in this cluster.

The functionality of parse_quadrants.py and parse_genres.py could have been implemented directly in parse_summary.py. These files were made in an ad-hoc manner to investigate the results and to find a way to make sense of the data. Although, it was easier implementing the code for parse_summary.py, as the data already had a good structure. An overview of the system flow can be seen in figure 4.1.

4.2 Running the Experiments

We will cover the different libraries and dependencies that is needed in order to run the experiments. First and most importantly, the MediaEval dataset [50] has to be acquired[1]. The next step after the dataset is downloaded is converting the files from MP3 format to WAV format. This can be done using FFMPEG, as mentioned in section 3.1.3, or other ways if desired.

Next, the library for pyAudioAnalysis has to be acquired from their GitHub page[2]. After pyAudioAnalysis is aquried, the commands for extracting features are shown in listing 4.1 for use with experiment (i) and listing 4.2 for use with experiment (ii).

The code that is made for this thesis has to be acquired, and it is available at GitHub[3]. Scikit-learn is also a prerequisite for running the experiments, as the clustering algorithms belong to this library.

The feature_cluster.py and stats_clustering.py files have command line input for specifying cluster number to have with the experiments.

Files parse_genres.py and parse_quadrants.py have to be configured internally to which folders, and in turn experiments, they are configured to read from. These files take command line input for number of clusters of the algorithms to parse information from.

The file parse_summary.py will summarize information specified by the path to the information generated by parse_genres.py and parse_quadrants.py, and store it to the specified path.

[1] `http://cvml.unige.ch/databases/emoMusic/`

[2] `https://github.com/tyiannak/pyAudioAnalysis`

[3] `https://github.uio.no/marasand/thesis`

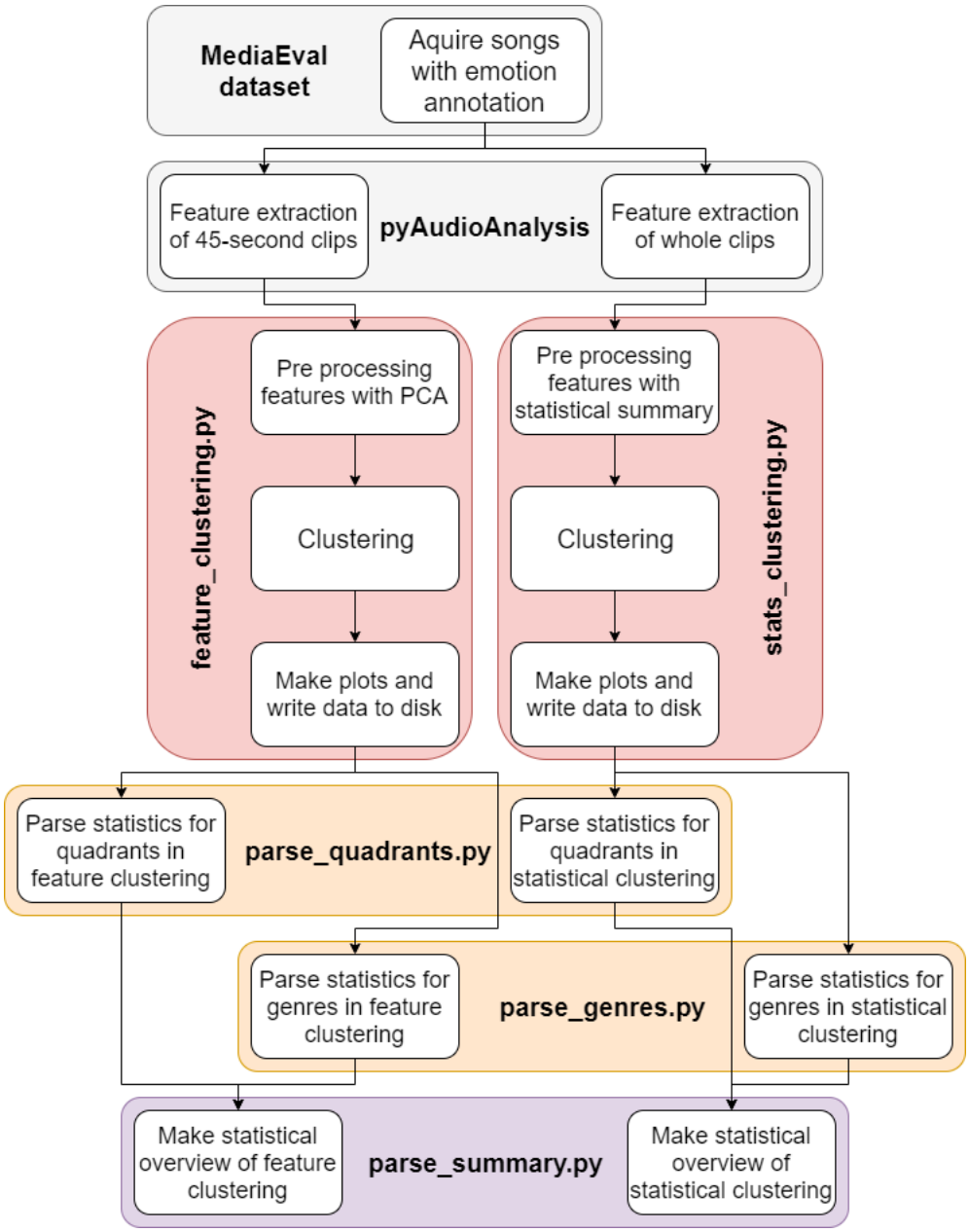


Figure 4.1: Overview of the system flow for the experiments. This flow chart shows all the necessary files for running the experiments.

```
python audioAnalysis.py featureExtractionDir -i ../clips_45sec/clips_45seconds/ -mw 1.0 -ms 1.0 -sw 0.050 -ss 0.050
```

Listing 4.1: Command for feature extraction of 45-second clips using pyAudioAnalysis.

```
python audioAnalysis.py featureExtractionDir -i ../clips_full/ -mw 1.0 -ms 1.0 -sw 0.050 -ss 0.050
```

Listing 4.2: Command for feature extraction of audio clips of whole songs using pyAudioAnalysis.

4.3 Results

The experiments have been tested a variety of times to achieve the output that provides the best information. At the start, the clustering was performed with only plots to see how the algorithms performed. The results that have emerged from the experiments results in a total of 18 folders containing plots and CSV files with structured data. The output files has later been processed to give a different structure containing all information of the experiments. In this section, we will present some of the results that has emerged from the experiments for early iterations of the experiments, to the final results and the prototype for these experiments.

4.3.1 Early experiments

The experiments that were made in the early stage of this thesis consisted of getting a visual representation of the clustering, as it would make sense to look at the clusters and the separation according to the quadrants. Metrics that were used in the early stage of the experiments was number of clusters and the valence-arousal dimensional model for visual representation.

Clustering features with PCA of 738 songs yielded no significant result when plotting the clusters. The clusters are scattered across the map with no apparent correlation between the emotion annotation and the features extracted from the 45-second clips, as can be seen in figure 4.2. Although the visualization of the clusters did not show any significant results, the statistical summary for genres within each cluster yielded some interesting results. Table 4.4 is an example of the genre content within each cluster for the clustering algorithm Birch with 4 clusters. Looking at this table, reading first the Blues songs across the clusters, the two first clusters have an average percentage of blues songs, but the third cluster have a very low percentage of blues songs, and the fourth has a somewhat high number of blues songs in it. Looking at the table, this algorithm does not seems to cluster blues songs in a good way. For classical songs, there is an even more evident separation. Looking at the first cluster, the percentage of classical songs are very low, even for the second cluster. The third cluster has the highest percentage of all genres in classical songs, and the fourth cluster is a bit high as well, but not in a significant way. Of all the genres, the classical songs have the highest percentage, across all the clusters, in the third cluster. The inference from this is if we cluster new songs with the Birch algorithm and 4 clusters, if a song ends up in the third cluster it is likely a classical song. The results that have been discussed are just an

example of what we found early on, and steps was taken to find a more appropriate approach to finding the optimal clustering.

Clustering statistical summary of the full length of the songs resulted in somewhat good separation of the data points. As can be seen in figure 4.3, there are two clusters, orange and blue. The orange data points seem to occupy the upper end of the valence-arousal dimensions, with high arousal and positive valence, and the blue data points seem to occupy the lower end, with low arousal and negative valence. However, the clusters have some overlap as there are some data points that have the same valence-arousal values, so the clusters needs to be separated to avoid confusing perspective. The proposed way of separating the clusters for visualization purposes is to make subplots that are organized next to each other with subplots being each own cluster.

The data points have some overlap as the emotion annotation for some songs are the same (more songs have the same values for valence/arousal). This can be avoided by separating overlapping points in a third dimension. Overall, adding a third dimension to separate overlapping data points as well as getting a clearer visualization of the clusters will be good for extracting relevant correlations between features and emotion.

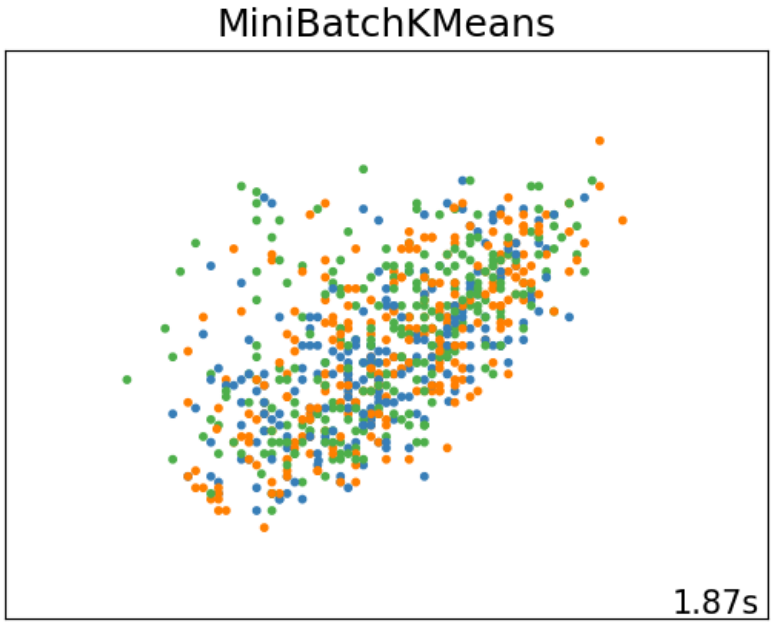


Figure 4.2: Example of early results when clustering features with PCA using Mini Batch K-means and 3 clusters.

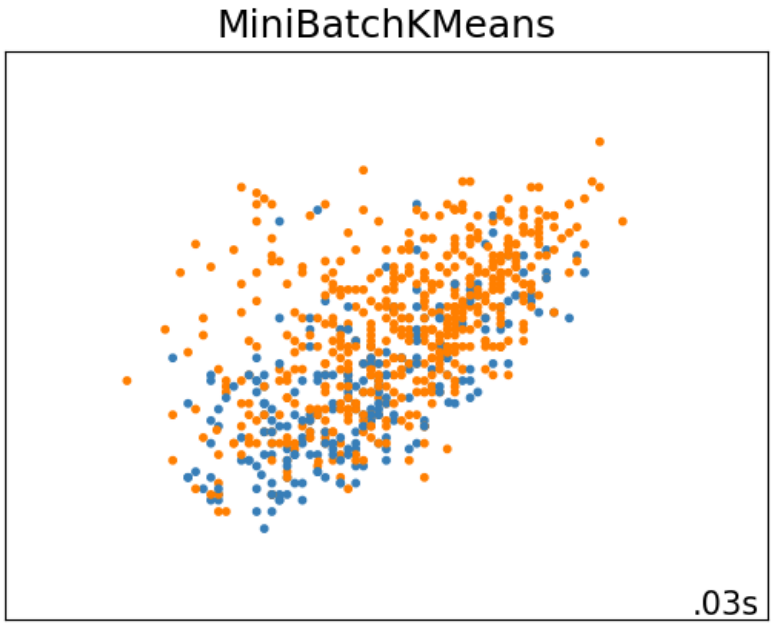


Figure 4.3: Example of early results when clustering features from whole clips with statistical summary using Mini Batch K-means and 2 clusters.

4.3.2 Results from final prototype

After the early iterations of the experiments, the design of the prototype was completed, shown in 4.1. The main focus of the experiment was to have the best clustering in terms of clusters representing the quadrants of the valence-arousal dimensional model in the best possible way. There has been little to no focus on the genre aspect of the clustering, but as mentioned in 4.3.1, there are indications to good genre clustering.

Plotting the clustering was done using one subplot per cluster. An example of this can be seen in the figure 4.4, where the clusters are separated into two subplots so there is no overlapping data points. This figure shows cluster one on the left and cluster two is on the right, where cluster one has a lot fewer data points than cluster two. In terms of representing quadrants using the clusters, this figure is a bad example because cluster two seems to occupy the same area as cluster one. Figure 4.4 was generated using stats_clustering.py, and serves as an example of the clustering procedures that was performed using the statistical summary of the feature extraction. Looking at the plots generated by feature_clustering.py, figure 4.5 will serve as an example plot using this clustering procedure. This figure shows that the first and third clusters can somewhat distinguish songs that are high and low in arousal. The first cluster representing the songs that are high in arousal are mostly occupying quadrants 1 and 2, where the third cluster representing the songs that are

low in arousal are mostly occupying quadrants 3 and 4. In comparison, the second and fourth clusters have no indication in terms of separating clusters into quadrants. Although the first and third clusters seem to somewhat represent the quadrants, in term of high and low arousal, we need to look more into the clustering algorithms and see if there are better clustering algorithms to represent each quadrant.

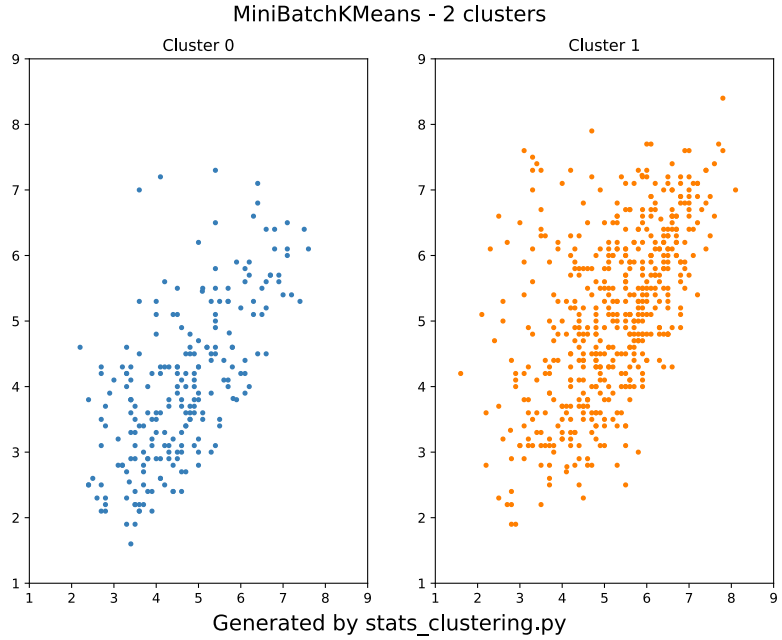


Figure 4.4: Plot of the clustering procedure for stats_clustering.py, using Mini Batch K-means and 2 clusters. X-axis is valence and y-axis is arousal. This is the same clustering algorithm and parameters as the plot shown in figure 4.3, but with separate subplots for each cluster.

In general, the visualization of the clusters does not seem to serve as a good indicator of performance quality, but serves as an accompanying feature to the data itself, that can be used to present the data that yields to good results.

Utilizing data from experiment (ii) as an example

The final step in the system flow figure 4.1 is to use parse_summary.py to summarize the data that emerged from the clustering experiments. The data from parse_summary.py is structured according to table 4.3, and we will use the information generated by this file with the clustering procedure of experiment (ii), that is clustered data using stats_clustering.py, to find suitable clustering algorithms and parameters for representing quadrants of the valence-arousal dimensional model.

The output CSV file from parse_summary.py was uploaded to google sheets so we could look at the table with some statistical metrics and color scaling of the metrics. Figure 4.6 shows all the clustering algorithms for

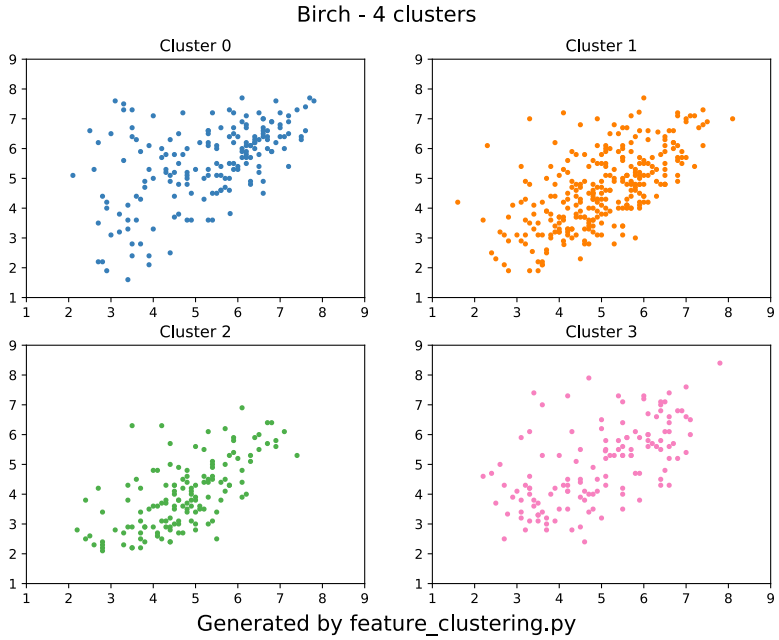


Figure 4.5: Plot of the clustering procedure for feature_clustering.py, using Birch and 4 clusters. x-axis is valence and y-axis is arousal.

cluster numbers 2 to 10, and the percentage of songs in each quadrant. The percentage of songs in each quadrant is the average of the number of clusters that has been assigned to the algorithm. The color scaling of the numbers is done per quadrant, where the lowest number is red, the 50th percentile is yellow and the highest number is green. This is so that we can easily see what numbers have high, low and average amounts of songs in a quadrant.

We want to find a clustering algorithm and cluster numbers that have clusters that cover the quadrants fairly. If there is a high or low percentage of songs in a quadrant, the clustering algorithm is not considered good in distinguishing songs into quadrants, which is what we want the algorithm to do. The column on the right side is a calculation done by summing unit fractions of the quadrants where the denominator is the quadrant percentage. This results in a formula on the form $Distribution = (1/Q1) + (1/Q2) + (1/Q3) + (1/Q4)$, where the Q's represent the respective quadrants. After this calculation, the rows in the spreadsheet is sorted in order of the lowest to highest distribution as seen in figure 4.7. The median of the distributions is calculated to give an indication to what clustering algorithms performed the best in terms of homogeneity. In figure 4.7, the median is highlighted in the color green, but the surrounding clustering algorithms should also perform good. Plots of all the clustering procedures can be seen in appendix B for a reference to figure 4.7. In this example, Mini Batch K-means with 5 clusters and Birch with 5 clusters are highlighted as the median for the distribution values. We are going to present a way of looking at the clustering algorithms that

can be used when clustering new songs.

Clusters	Algorithm	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4	Percent	Distribution
2	Birch	28,501	8,1135	49,2235	14,1615	100,00	0,2492673477
2	GaussianMixture	31,3295	8,9795	44,638	15,053	100,00	0,2321179518
2	MiniBatchKMeans	29,464	8,2955	47,3585	14,882	100,00	0,2427978108
2	SpectralClustering	30,7425	8,571	45,38	15,306	100,00	0,2365707545
3	Birch	31,65	8,091	43,48433333	16,77433333	100,00	0,2378013695
3	GaussianMixture	34,10233333	8,936333333	41,499	15,46266667	100,00	0,22999509
3	MiniBatchKMeans	35,759	10,46	39,02166667	14,75933333	100,00	0,2169478086
3	SpectralClustering	35,42966667	10,23966667	39,872	14,45833333	100,00	0,2201288849
4	Birch	33,368	9,30375	41,4425	15,88575	100,00	0,2245316912
4	GaussianMixture	35,691	9,758	39,291	15,26025	100,00	0,2214791314
4	MiniBatchKMeans	34,2595	9,3015	40,35475	16,0845	100,00	0,2236504118
4	SpectralClustering	40,77475	11,6115	34,776	12,83775	100,00	0,2172972326
5	Birch	34,4478	8,8254	39,1694	17,5572	100,00	0,2248255651
5	GaussianMixture	35,1968	8,724	41,04	15,039	100,00	0,2338982466
5	MiniBatchKMeans	33,7814	9,1632	41,969	15,0868	100,00	0,2288444885
5	SpectralClustering	39,0868	12,0766	34,9346	13,9018	100,00	0,2089468948
6	Birch	32,01016667	8,042	42,83166667	17,11616667	100,00	0,2373587608
6	GaussianMixture	36,70216667	10,04	38,28533333	14,97283333	100,00	0,2197552308
6	MiniBatchKMeans	27,61233333	9,371	45,81233333	17,2045	100,00	0,2228804089
6	SpectralClustering	39,09633333	11,133	36,73216667	13,0385	100,00	0,2193209289
7	Birch	33,70914286	9,065	40,51942857	16,70614286	100,00	0,2245176786
7	GaussianMixture	34,35128571	9,578571429	40,329	15,74142857	100,00	0,2218333819
7	MiniBatchKMeans	31,31985714	7,095571429	48,11814286	13,46642857	100,00	0,2679025228
7	SpectralClustering	40,47628571	10,82171429	35,28271429	13,41928571	100,00	0,2199747306
8	Birch	32,0645	8,605875	43,2135	16,11575	100,00	0,2325788433
8	GaussianMixture	31,83675	8,531875	44,566125	15,065375	100,00	0,2374337113
8	MiniBatchKMeans	29,1315	7,22725	47,89375	15,74775	100,00	0,2570730041
8	SpectralClustering	43,615125	11,917375	31,17925	13,28825	100,00	0,2141659888
9	Birch	34,063	7,677777778	38,48488889	19,77433333	100,00	0,2361582172
9	GaussianMixture	30,66488889	8,336555556	46,38166667	14,61688889	100,00	0,2425384582
9	MiniBatchKMeans	31,015	8,21	45,51944444	15,25555556	100,00	0,2415636674
9	SpectralClustering	41,78855556	13,99877778	31,58077778	12,63211111	100,00	0,2061929674
10	Birch	36,2188	9,1261	35,9714	18,6838	100,00	0,2185079684
10	GaussianMixture	36,014	7,879	37,9512	18,1559	100,00	0,2361147817
10	MiniBatchKMeans	37,5008	7,9395	36,9655	17,5944	100,00	0,2365071312
10	SpectralClustering	38,5276	13,7422	35,9976	11,7327	100,00	0,2117354744

Figure 4.6: List of clustering algorithms and the percentage of songs in each quadrant from experiment (ii). The percent column is to check that they add up to 100%. Distribution is the distribution of songs across the quadrants, calculated by using the formula $Distribution = (1/Q1) + (1/Q2) + (1/Q3) + (1/Q4)$.

Assigning clusters to quadrants

In table 4.5 and table 4.6, we have excerpts of the output CSV file created by parse_summary.py, and after calculating the distributions above with the median of the distribution being on these two clustering algorithms, they will serve as the example for assigning clusters to quadrants. The reason for assigning each cluster to a quadrants, is that each quadrant represents the fuzzy emotion categories that can be seen in figure 3.1, and when we apply new songs without a valence-arousal rating to the trained clustering

Clusters	Algorithm	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4	Percent	Distribution	Median	
9	SpectralClustering	41,78855556	13,99877778	31,58077778	12,63211111	100,00	0,2061929674	0,2268350268	Extreme Q2
5	SpectralClustering	39,0868	12,0766	34,9346	13,9018	100,00	0,2089468948	0,2268350268	
10	SpectralClustering	38,5276	13,7422	35,9976	11,7327	100,00	0,2117354744	0,2268350268	
8	SpectralClustering	43,615125	11,917375	31,17925	13,28825	100,00	0,2141659888	0,2268350268	Extreme Q1
3	MiniBatchKMeans	35,759	10,46	39,02166667	14,75933333	100,00	0,2169478086	0,2268350268	
4	SpectralClustering	40,77475	11,6115	34,776	12,83775	100,00	0,2172972326	0,2268350268	
10	Birch	36,2188	9,1261	35,9714	18,6838	100,00	0,2185079684	0,2268350268	
6	SpectralClustering	39,09633333	11,133	36,73216667	13,0385	100,00	0,2193209289	0,2268350268	
6	GaussianMixture	36,70216667	10,04	38,28533333	14,97283333	100,00	0,2197552308	0,2268350268	
7	SpectralClustering	40,47628571	10,82171429	35,28271429	13,41928571	100,00	0,2199747306	0,2268350268	
3	SpectralClustering	35,42966667	10,23966667	39,872	14,45833333	100,00	0,2201288849	0,2268350268	
4	GaussianMixture	35,691	9,758	39,291	15,26025	100,00	0,2214791314	0,2268350268	
7	GaussianMixture	34,35128571	9,578571429	40,329	15,74142857	100,00	0,2218333819	0,2268350268	
6	MiniBatchKMeans	27,61233333	9,371	45,81233333	17,2045	100,00	0,2228804089	0,2268350268	
4	MiniBatchKMeans	34,2595	9,3015	40,35475	16,0845	100,00	0,2236504118	0,2268350268	
7	Birch	33,70914286	9,065	40,51942857	16,70614286	100,00	0,2245176786	0,2268350268	
4	Birch	33,368	9,30375	41,4425	15,88575	100,00	0,2245316912	0,2268350268	
5	Birch	34,4478	8,8254	39,1694	17,5572	100,00	0,2248255651	0,2268350268	MEDIAN
5	MiniBatchKMeans	33,7814	9,1632	41,969	15,0868	100,00	0,2288444885	0,2268350268	MEDIAN
3	GaussianMixture	34,10233333	8,936333333	41,499	15,46266667	100,00	0,22999509	0,2268350268	
2	GaussianMixture	31,3295	8,9795	44,638	15,053	100,00	0,2321179518	0,2268350268	
8	Birch	32,0645	8,605875	43,2135	16,11575	100,00	0,2325788433	0,2268350268	
5	GaussianMixture	35,1968	8,724	41,04	15,039	100,00	0,2338982466	0,2268350268	
10	GaussianMixture	36,014	7,879	37,9512	18,1559	100,00	0,2361147817	0,2268350268	
9	Birch	34,063	7,677777778	38,48488889	19,77433333	100,00	0,2361582172	0,2268350268	Extreme Q4
10	MiniBatchKMeans	37,5008	7,9395	36,9655	17,5944	100,00	0,2365071312	0,2268350268	
2	SpectralClustering	30,7425	8,571	45,38	15,306	100,00	0,2365707545	0,2268350268	
6	Birch	32,01016667	8,042	42,83166667	17,11616667	100,00	0,2373587608	0,2268350268	
8	GaussianMixture	31,83675	8,531875	44,566125	15,065375	100,00	0,2374337113	0,2268350268	
3	Birch	31,65	8,091	43,48433333	16,77433333	100,00	0,2378013695	0,2268350268	
9	MiniBatchKMeans	31,015	8,21	45,51944444	15,25555556	100,00	0,2415636674	0,2268350268	
9	GaussianMixture	30,66488889	8,336555556	46,38166667	14,61688889	100,00	0,2425384582	0,2268350268	
2	MiniBatchKMeans	29,464	8,2955	47,3585	14,882	100,00	0,2427978108	0,2268350268	
2	Birch	28,501	8,1135	49,2235	14,1615	100,00	0,2492673477	0,2268350268	Extreme Q3
8	MiniBatchKMeans	29,1315	7,22725	47,89375	15,74775	100,00	0,2570730041	0,2268350268	
7	MiniBatchKMeans	31,31985714	7,095571429	48,11814286	13,46642857	100,00	0,2679025228	0,2268350268	

Figure 4.7: List of clustering algorithms and the percentage of songs in each quadrant from experiment (ii). This figure has the same content as figure 4.6, but this is ordered by lowest to highest distribution. Numbers highlighted with green are the median of the distributions. Extremes are the highest percentage of songs in a quadrant.

models, we can say that the new songs have the emotions specified by the quadrants the cluster they get clustered into represents. Generally, the data itself is organized so there is an inherent correlation between low-arousal negative-valence songs and high-arousal positive-valence songs. This makes it harder to find clusters that represent quadrants 2 and 4 as there are more songs overall in quadrants 1 and 3. With this in mind, looking at table 4.5, we can do a step-by-step assigning of clusters to quadrants as follows:

Step 1: Look at the first cluster and see what quadrant it has the highest percentage of songs in. In this case, cluster 0 has the highest

percentage of songs in quadrant 3, so we assign cluster 0 to quadrant 3.

Step 2: Repeat the previous step until all clusters have been assigned to a quadrant. The result for Birch clustering algorithm after the first steps looks like table 4.7.

Step 3: If there is a quadrant that does not have a cluster assigned to it, find the quadrant and then find the cluster with the highest percentage of songs in that quadrant, then reassign that cluster to this quadrant. In this case, quadrant 2 and 4 do not have clusters assigned to them. We start with quadrant 2, where the highest percentage of songs in that quadrant is in cluster 1, so we assign cluster 1 to this quadrant instead of the previous assigned quadrant.

Step 4: Repeat the previous step until all quadrants have been covered. In this case, quadrant 4 have no cluster assigned to it. The highest percentage of songs in quadrant 4 are represented by cluster 4, so we reassign cluster 4 from quadrant 3 to quadrant 4. The resulting assignment of clusters to quadrants can be seen in table 4.8.

Using the steps above with Mini Batch K-means and 5 clusters, we get the table shown in 4.9. The two clustering algorithms that we have used for the example of assigning clusters to quadrants have plots that can be seen in figure 4.8 for Birch with 5 clusters, and in figure 4.9 for Mini Batch K-means with 5 clusters.

The steps we have specified above to assign clusters to quadrants does not work if there are less clusters with the clustering algorithm than there are quadrants. When the songs in the MediaEval dataset were annotated with valence-arousal values for emotions, arousal was annotated more consistently than valence [50]. When we look at the dataset, the songs are also ordered in a way that there are an inherent correlation between low arousal negative valence songs, and high arousal positive valence songs. When we are going to assign clusters to quadrants when there are fewer clusters than quadrants, we are going to utilize the fact that the songs are ordered this way. First, we are going to take a look at experiment (i) and the results from parse_summary.py.

Utilizing data from experiment (i)

Experiment (i) did not show any promising results early on, so we used the results from the early stages of experiment (ii) to determine what to look for, where we ended up with the procedures for discovering the optimal clustering algorithms and their parameters used in the previous sections.

The procedure for discovering clustering algorithms that distribute songs in a fair manner is applied to the data from experiment (ii) that is generated by parse_summary.py. First, the CSV file generated by parse_summary.py is uploaded to google sheets, and the data is ordered according to the distribution formula specified previously. The resulting

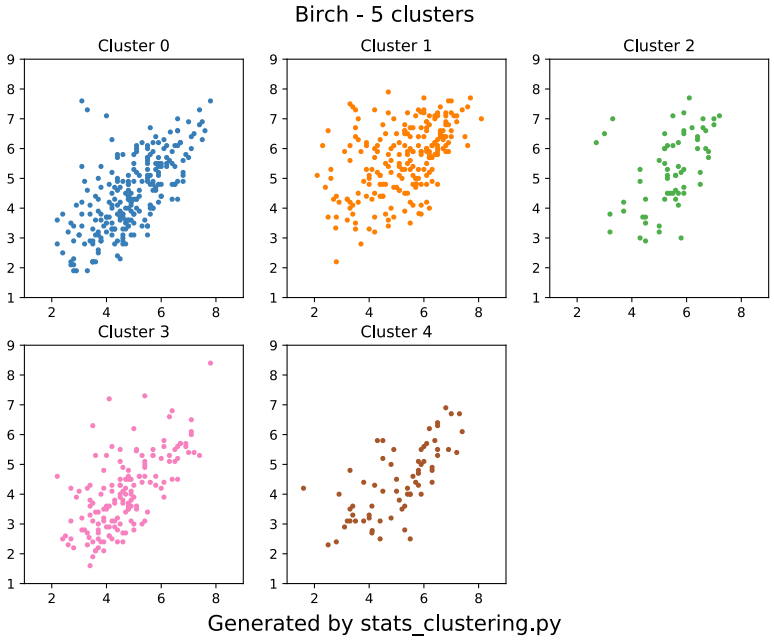


Figure 4.8: Plot of the clustering procedure for stats_clustering.py, using Birch and 5 clusters. X-axis is valence and y-axis is arousal.

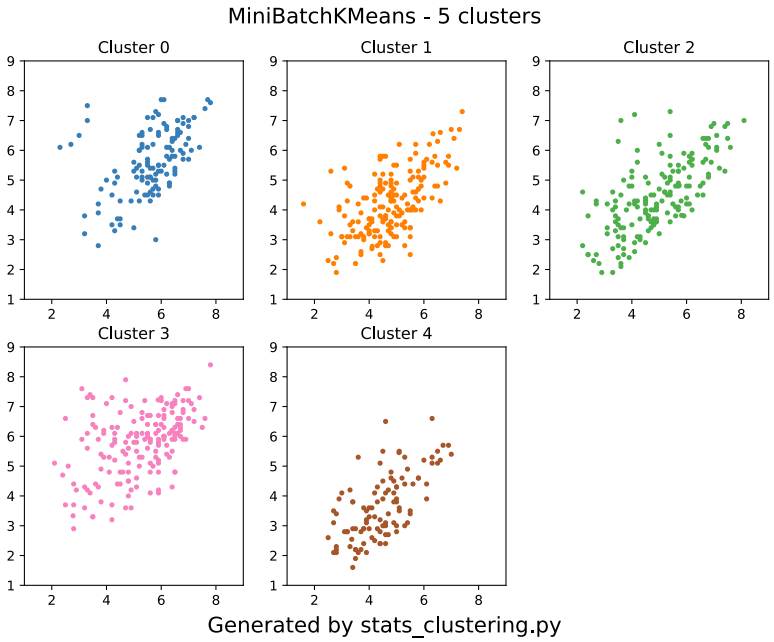


Figure 4.9: Plot of the clustering procedure for stats_clustering.py, using Mini Batch K-means and 5 clusters. X-axis is valence and y-axis is arousal.

spreadsheet can be seen in figure 4.10. The median of the distributions in this experiment is Birch with 10 clusters, and Birch with 3 clusters, with

excerpts from parse_summary.py in table 4.11 and 4.10 respectively. With this experiment, as with experiment (ii), the clustering algorithms that are spread around the median are also noteworthy of their performance, but we will use the clustering algorithms on the median for further examples. Plots from all the clustering procedures can be seen in appendix A for a reference to figure 4.10.

We will use similar steps to the ones described in the first section of assignment steps to assign the clusters of these algorithms to each quadrant. In the examples we provided in the first section for assigning clusters to quadrants, we used clustering algorithms with more clusters than quadrants, but in this example, we have Birch with 3 clusters. As we also stated previously, we are going to base the assignment of clusters to quadrants on the fact that the annotations for arousal is far more consistent than the annotations for valence.

Assigning clusters to quadrants where there are fewer clusters than quadrants

We are going to use a step by step assignment of clusters to quadrants using the Birch clustering algorithm with 3 clusters from the clustering procedures of experiment (i). When there are fewer clusters than quadrants, we want to assign one clusters to two quadrants, where one cluster should be assigned to either quadrant 1 and 2, or quadrant 3 and 4. The reason for this is that based on the nature of the MediaEval dataset, arousal has more consistent annotations than valence [50].

Step 1: Look at the first cluster and see what quadrant it has the highest percentage of songs in. In this case, cluster 0 has the highest percentage of songs in quadrant 1 with quadrant 3 right behind, but we assign cluster 0 to quadrant 1.

Step 2: Repeat the previous step until all clusters have been assigned to a quadrant. The result for Birch clustering algorithm with 3 clusters after the first steps looks like table 4.12.

Step 3: If there is a quadrant that does not have a cluster assigned to it, find the quadrant and then find the cluster with the highest percentage of songs in that quadrant, then assign that cluster to this quadrant as well as the previously assigned quadrant. In this case, quadrant 2 and 4 do not have clusters assigned to them. We start with quadrant 2, where the highest percentage of songs in that quadrant is in cluster 0, right above cluster 1, so we assign cluster 0 to quadrant 2 as well as quadrant 1.

Step 4: Repeat the previous step until all quadrants have been covered. In this case, quadrant 4 have no cluster assigned to it. The highest percentage of songs in quadrant 4 are represented by cluster 2, so we assign cluster 2 to quadrant 4 as well as quadrant 3. The resulting assignment of clusters to quadrants can be seen in table 4.13.

The results of this procedure produces a somewhat reliable model for clustering new songs, however, as we can see in table 4.10 and table 4.13, the percentage of songs in cluster 0 covering quadrant 1 and 2 add up to only 49.355%, which is only around half of the songs in the cluster. This cluster should not be considered for clustering new songs because the cluster is spread too much, as we also can see in figure 4.11. Later in this chapter, we will discuss more around the assignment of clusters to quadrants for clustering new songs.

Assigning clusters to quadrants where there is a high number of clusters

We are going to give an example of assigning clusters to quadrants where there is a high number of clusters. In this example, we have Birch with 10 clusters generated from the clustering procedures of experiment (i), that have an excerpt from the CSV file generated by parse_summary.py in table 4.11. The procedure for assigning high numbers of clusters should be to equally cover the quadrants with clusters. Using the steps in the first section of assignment steps, we can assign clusters to quadrants for the Birch algorithm with 10 clusters, where we end up with the table shown in 4.14. After using these steps, some of the clusters should be reassigned to cover the quadrants that is only covered by one cluster, i.e., quadrant 2 and 4. We can reassign one cluster from quadrant 1 or 3 to quadrant 2 and one to quadrant 4 using the same technique as the third step in the first section for assignment steps, where the highest percentage of songs in quadrant 2 and 4 would be the next highest percentage. After doing this, we end up with a table shown in 4.15. The plot that goes along with the Birch clustering algorithm with 10 clusters can be seen in figure 4.12.

Extreme clusters

In the spreahsheet that are shown in figure 4.7 and 4.10, there are clustering algorithms that are listed as extreme clustering to the quadrants. These extremes are the highest average percentage of all the clustering algorithms in a quadrant. From figure 4.7 corresponding to the clustering procedures from stats_clustering.py, we can look at an extreme example, where for example the Birch algorithm with 9 clusters has the highest percentage for quadrant 4 and is not too far from the median. An excerpt of the CSV file generated by parse_summary.py is shown in table 4.16, with a corresponding plot in figure 4.13, for the Birch algorithm with 9 clusters. In this example, we can see that the extreme average value is attributed to cluster number 8, where is has 50% of its content in quadrant 1 and 50% in quadrant 4. In this table, the number of songs in each cluster have been included, and we can see that cluster 8 only has two songs in it. This is something that the formula for calculating distributions across clustering algorithms does not take into account. The formula for finding good distributions would ideally take into account the number of songs in a cluster to find the best cluster homogeneity. We can look at another extreme example from figure 4.7, where Spectral Clustering with 9 cluster

Clusters	Algorithm	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4	Percent	Distribution	Median	
10	GaussianMixtur	20,4121	16,1013	44,342	19,1447	100,00	0,1858830959	0,2254096019	Extreme Q2
9	GaussianMixtur	15,50144444	15,51755556	51,32733333	17,65344444	100,00	0,2050822228	0,2254096019	
5	Birch	33,3928	10,587	41,1948	14,825	100,00	0,2161305737	0,2254096019	
6	Birch	33,46716667	10,53	41,172	14,83066667	100,00	0,2165629968	0,2254096019	
10	SpectralCluster	40,1712	13,2129	35,2154	11,4004	100,00	0,2166899561	0,2254096019	
9	SpectralCluster	36,95744444	11,97944444	38,66288889	12,40011111	100,00	0,2170435078	0,2254096019	
7	Birch	36,03328571	10,39285714	38,86285714	14,71114286	100,00	0,2176792436	0,2254096019	
7	SpectralCluster	35,50328571	11,04171429	40,03328571	13,42142857	100,00	0,2182189849	0,2254096019	
3	MiniBatchKMea	33,06933333	10,024	41,28266667	15,62366667	100,00	0,2182287746	0,2254096019	
8	Birch	34,276375	9,972875	40,401125	15,34975	100,00	0,2193460283	0,2254096019	
9	Birch	34,31011111	9,929333333	40,41333333	15,34733333	100,00	0,2197598313	0,2254096019	
2	Birch	34,1915	9,9955	40,8615	14,9515	100,00	0,2206478914	0,2254096019	
2	MiniBatchKMea	33,4455	9,887	41,482	15,1855	100,00	0,2210014383	0,2254096019	
3	GaussianMixtur	33,13366667	9,919666667	41,835	15,112	100,00	0,2210666302	0,2254096019	
4	Birch	32,774	9,697	42,04925	15,47925	100,00	0,2220209191	0,2254096019	
8	SpectralCluster	38,9705	11,343	37,479625	12,207	100,00	0,2224219073	0,2254096019	
2	GaussianMixtur	31,818	9,4915	42,977	15,714	100,00	0,2236919565	0,2254096019	
3	Birch	31,38166667	9,396666667	43,208	16,01333333	100,00	0,2238782779	0,2254096019	MEDIAN
10	Birch	34,0331	9,5483	41,9201	14,4986	100,00	0,2269409259	0,2254096019	MEDIAN
4	MiniBatchKMea	30,42075	9,08975	44,43	16,059	100,00	0,227664019	0,2254096019	
9	MiniBatchKMea	36,252	7,912888889	32,72611111	23,10888889	100,00	0,227790812	0,2254096019	
5	SpectralCluster	37,576	10,928	39,9168	11,5792	100,00	0,2295346447	0,2254096019	
5	GaussianMixtur	27,1278	8,3014	52,2868	12,2842	100,00	0,2578548316	0,2254096019	
5	MiniBatchKMea	27,751	7,9738	52,1392	12,136	100,00	0,2630243576	0,2254096019	
4	SpectralCluster	37,972	7,8835	42,639	11,50575	100,00	0,2635481797	0,2254096019	
2	SpectralCluster	38,4955	8,111	42,5815	10,812	100,00	0,2652406255	0,2254096019	
7	GaussianMixtur	38,37457143	6,874142857	40,36185714	14,38928571	100,00	0,265803625	0,2254096019	
6	SpectralCluster	36,8625	8,028833333	45,9305	9,178333333	100,00	0,2824032048	0,2254096019	
6	MiniBatchKMea	27,86633333	6,599666667	55,23666667	10,2975	100,00	0,3026232666	0,2254096019	Extreme Q3
8	MiniBatchKMea	45,948375	6,25675	38,435875	9,359	100,00	0,3144573241	0,2254096019	Extreme Q1
3	SpectralCluster	40,39966667	5,906666667	43,45866667	10,235	100,00	0,3147672322	0,2254096019	
8	GaussianMixtur	33,293375	5,84425	51,788	9,074375	100,00	0,3306542796	0,2254096019	
7	MiniBatchKMea	42,07428571	5,341428571	42,50357143	10,08085714	100,00	0,3337086679	0,2254096019	
10	MiniBatchKMea	24,033	3,9087	37,0425	35,0157	100,00	0,3530036272	0,2254096019	
6	GaussianMixtur	44,5155	3,307666667	30,47733333	21,6995	100,00	0,4036872913	0,2254096019	
4	GaussianMixtur	8,42425	2,48975	52,001	37,085	100,00	0,5665471605	0,2254096019	Extreme Q4

Figure 4.10: List of clustering algorithms and the percentage of songs in each quadrant from experiment (i). This figure has the same structure as the contents from experiment (ii) in figure 4.7. Numbers highlighted with green are the median of the distributions. Extremes are the highest percentage of songs in a quadrant.

has the highest percentage for quadrant 2. In table 4.17, and corresponding figure 4.14, is an excerpt of the CSV file generated by parse_summary.py for Spectral Clustering with 9 clusters. This algorithm has a good percentage of songs in quadrant 2 for cluster 7, where an extreme example like this can be used to find songs of moods represented by quadrant 2. This algorithm also has a lot of songs in cluster 2 compared to the rest of the clusters. Cluster 2 for this algorithm also has somewhat equally distributed its songs across the quadrants. These two points are important when using the assignment steps we have devised in the previous sections, where algorithms such as this one have an overwhelming number of songs in a cluster compared to other clusters, and the same cluster has no distinct quadrant it distributes

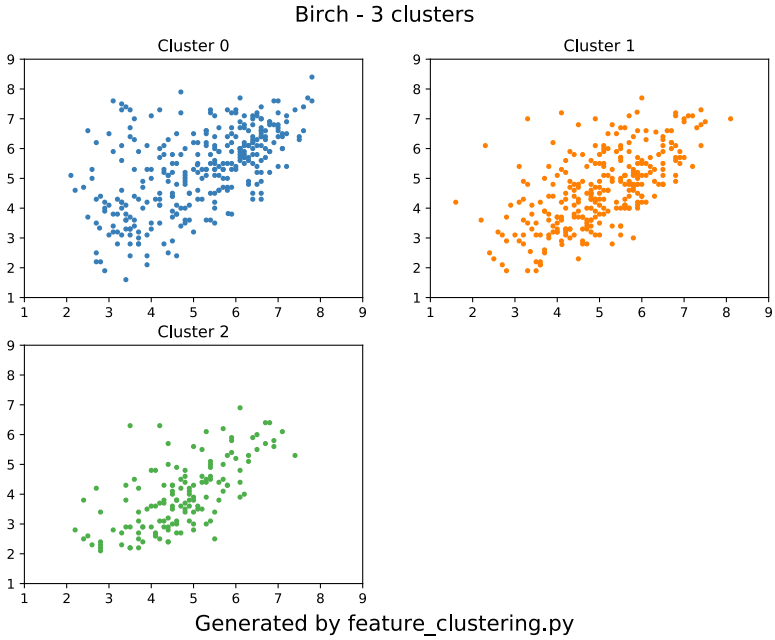


Figure 4.11: Plot of the clustering procedure for feature_clustering.py, using Birch and 3 clusters. X-axis is valence and y-axis is arousal.

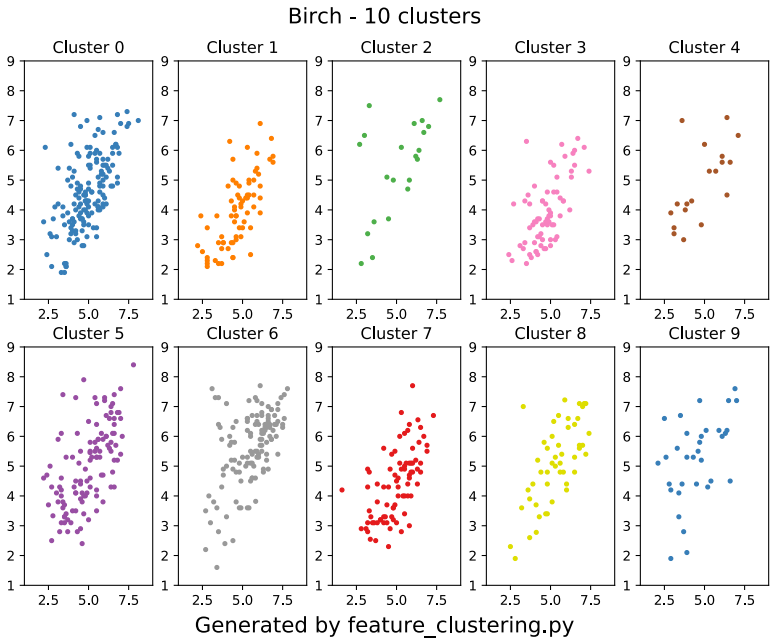


Figure 4.12: Plot of the clustering procedure for feature_clustering.py, using Birch and 10 clusters. X-axis is valence and y-axis is arousal.

its songs into, and should be regarded as an unreliable cluster when clustering new songs. Aside from the unreliable cluster, Spectral Clustering

with 9 clusters can serve as a good algorithm, especially for finding songs with emotions in quadrant 2. In general, extremes are good when a specific emotion represented by the extreme quadrant is desired when clustering new songs.

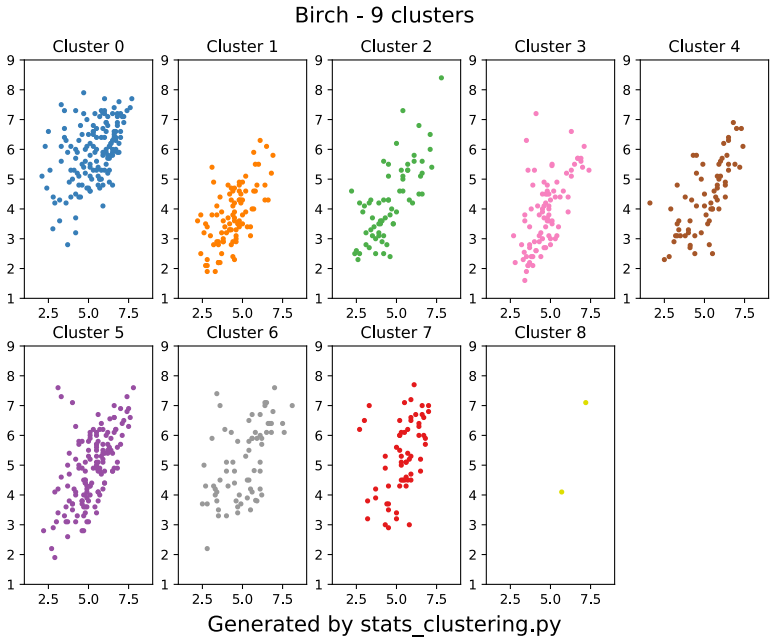


Figure 4.13: Plot of the clustering procedure for stats_clustering.py, using Birch and 9 clusters. X-axis is valence and y-axis is arousal.

4.3.3 Summary of Results

In the previous sections, we have looked at the results from the clustering experiments and what the statistical summary and cluster homogeneity of each clustering algorithm and their parameters are. From these results, we have designed some steps to assign each cluster in a clustering algorithm to a quadrant to use the trained models with new songs. The steps we have designed, which can be viewed in section 4.3.2, are not complete in their way of assigning clusters to quadrants, but they act as a baseline method for doing so. There are some special cases to assigning clusters to quadrants that has been mentioned throughout the previous sections, and we will try to summarize them as follows:

- As we mentioned in section 4.3.2, some songs are spread equally, or have an approximate equal spread, across quadrants that makes the clusters vague in terms of emotion specific characteristics. Clusters that are spread in this manner should be regarded as unreliable and therefore be ignored if songs end up it that specific cluster when clustering new songs.

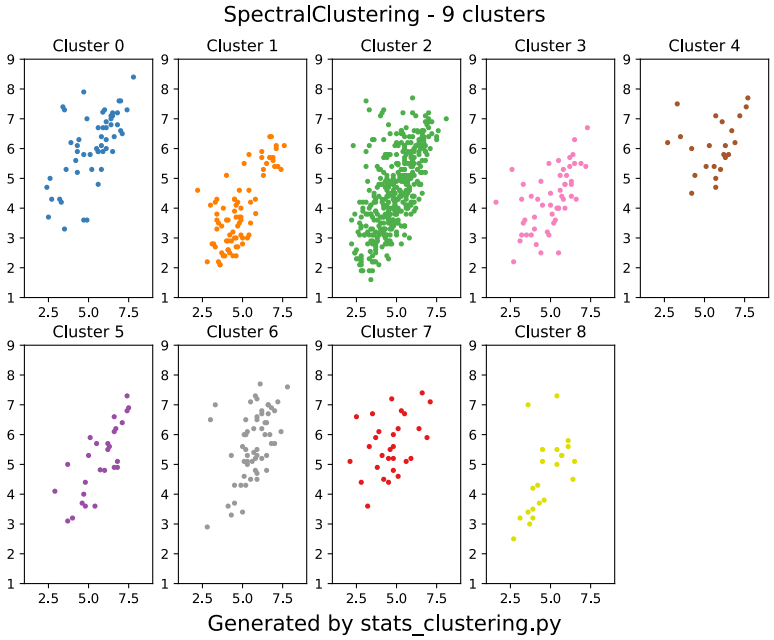


Figure 4.14: Plot of the clustering procedure for stats_clustering.py, using Spectral Clustering and 9 clusters. X-axis is valence and y-axis is arousal.

- We looked at the case for extreme percentage of songs in a quadrant. While this might be useful for clustering new songs to find songs of a certain emotion, which again is represented by a specific quadrant, the extreme percentages may be due to a few number of songs skewing the percentage rating. The clusters that have an underwhelming number of songs in them should also not be considered as a reliable cluster when clustering new songs.

Overall, the experiments have shown good results for clustering songs in terms of the homogeneity of clusters distributing songs across the quadrants. With the assignment steps we have devised in this chapter, the prototype we have designed can be used to cluster new songs. Plots from the results that has not been shown throughout this chapter can be viewed in the appendices. The sorted spreadsheets in figures 4.10 and 4.7 have corresponding plots in appendix B and appendix A, respectively. These appendices can be used to look up desired plots from the spreadsheets.

4.4 Discussion

In this section, we will discuss the results that have emerged from the experiments and how the work that has been done in the thesis can be used. There are some difference in performance from experiment (i) to experiment (i) that we will address in this section. We will discuss around the topic of using the models we have created for clustering new songs. Finally, we will discuss how the results can be optimized for future work

on the prototype we have made in terms of feature selection and clustering algorithm parameters.

4.4.1 Discussion on the results and difference between experiments

In the previous sections, we have used the clustering algorithms that have the median value for the calculated distribution as examples, but we will discuss some other clustering algorithms that have not been included in the previous sections that have plots in appendices A and B. Overall, the clustering procedure from experiment (ii) generated better results than the clustering procedure from experiment (i). If we look at the ten clustering algorithms around the median distribution value from experiment (i), we can see that the distribution of songs are spread out on the valence-arousal dimensions with minimal separation into quadrants. Gaussian Mixture with 3 clusters, Birch with 4 clusters, Gaussian Mixture with 2 clusters, Mini Batch K-means with 4 clusters, Mini Batch K-means with 9 clusters and Gaussian Mixture with 5 clusters are all not separating good into quadrants. Spectral clustering with 8 clusters and Spectral Clustering with 5 clusters are of a different kind, where one cluster has a lot of songs and the other clusters have very few songs. For experiment (ii), we can look at the ten clustering algorithms around the median distribution value. Mini Batch K-means with 6 clusters, Gaussian Mixture with 2 clusters, Birch with 8 clusters and Gaussian Mixture with 5 clusters are good at distinguishing quadrants within clusters. Mini Batch K-means with 4 clusters, Birch with 7 clusters, Birch with 4 clusters and Gaussian Mixture with 3 clusters are somewhat good, but better than the ten clustering algorithms distributed around the median value for experiment (i). The difference in performance between the experiments may be that the statistical summary is a good way of clustering features, or that Principal Component Analysis needs fewer components to account for the variance of the features. Overall, the clustering algorithms from experiment (ii) should be used in further research.

4.4.2 Discussion on the use of trained models

In the previous section, we described ways of assigning clusters to quadrants when clustering new songs with the already trained models. The steps in section 4.3.2 are not exhaustive in their way of describing assignment of clusters to quadrants, but serve as a baseline for how to do the assignments. As described in section 4.3.2, the assignment of clusters to quadrants for the Birch algorithm with 3 clusters, we could see that cluster 0 only covered half of the songs in when it was assigned to two quadrants. If one cluster distributes its songs evenly across the valence-arousal dimensional model, like cluster 0 in table 4.10, it should not be considered as a valid cluster for finding emotions for new songs. As we can also see when finding clusters for quadrants 2 and 4, the percentage of songs in that quadrant within a cluster is very small, and can make

it hard when finding emotions for new songs. This is something that should be taken into consideration when using the steps for assigning clusters to quadrants. The assignment of clusters to quadrants is also a way of finding emotions for new songs, by specifying that a cluster has a quadrant/emotion it belongs to. This would be suitably tested with songs from the papers about sports covered in section 2.1. Finding mood for exercise would be to cluster the songs from the sport papers, and see if the majority of songs was covered by a single cluster, that again is assigned to an emotion.

4.4.3 Discussion on the use of features and clustering algorithms

The experiments that has been done in this thesis was done using pyAudioAnalysis as a means of extracting features from songs. The features that are specified in section 3.1.2 have all been used with both experiments (i) and (ii) for this thesis. The features that are used can contribute in different ways if the clustering had been done with a combination of the various features. As stated in section 3.1.2, MFCCs have uses in genre classification [31], which could be clustered with for example Chroma Vector and Chroma Deviation, that are good at identifying harmonic features [32], to have a good genre clustering approach. Features such as Zero Crossing Rate, Energy, Entropy of Energy and Spectral Spread could be clustered together to see if they are good at identifying low arousal and high arousal songs in terms of the valence-arousal dimensional model. There are many features, but without an exhaustive approach to finding good features for clustering, it can be theorized based on the nature of music and human emotion what features can be clustered together. These types of combinations of features could be something to look for when optimizing the clustering based on the experiments in this chapter.

The clustering algorithms that have been used in the experiments have been limited to Mini Batch K-means, Birch, Gaussian Mixture and Spectral Clustering. As stated in section 3.2, the clustering algorithms that did not have a parameter for specifying a cluster number before cluster were not included in the experiments. The Clustering algorithms that have not been included in the experiments should be tried with different parameters to see if they yield any significant results.

4.5 Summary

In this chapter, we have presented the design of the experiments as well as the results from these experiments. We have shown a variety of results from the clustering procedures that shows the clustering algorithms distributing its clusters across the quadrants. These results can be used effectively to cluster new songs to find the emotional value in terms of what quadrant the clusters represent. We have also presented the distribution of songs in clusters across quadrants, and how we have calculated the distributions to find the best cluster homogeneity to best explain all quadrants for a

given algorithm and number of clusters. We have designed some steps to assign clusters to quadrants for clustering of new songs, which can be viewed in section 4.3.2 with a summary of some exceptions to these steps in the summary of results section 4.3.3. These steps are not limited in the description of how to assign clusters to quadrants, but they provide a baseline method for doing so. Furthermore, we have included some discussion on the topic of using the trained models with new songs, as well as discussion on the use of features extracted from the song clips and clustering algorithms.

Number of clusters	Clustering algorithm	Cluster number	Q1	Q2	Q3	Q4	Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock	Songs in cluster
data	...	...	...	...	...	...	...	...	...	...	...	...	...	...	*data*

Table 4.3: Structure of the output CSV file from parse_summary.py. The Q's are the respective quadrants.

176							
Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock
27	6	15	25	10	31	29	33
15.341	3.409	8.523	14.205	5.682	17.614	16.477	18.75
287							
Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock
41	30	37	52	36	50	24	17
14.286	10.453	12.892	18.118	12.544	17.422	8.362	5.923
141							
Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock
6	54	23	10	10	14	15	9
4.255	38.298	16.312	7.092	7.092	9.929	10.638	6.383
134							
Blues	Classical	Country	Electronic	Folk	Jazz	Pop	Rock
26	26	30	5	15	10	10	12
19.403	19.403	22.388	3.731	11.194	7.463	7.463	8.955

Table 4.4: CSV file containing the summary of genres for Birch clustering algorithm with 4 clusters. The structure of this file is according to table 4.2.

Birch — 5 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	25.926	5.761	52.675	15.638
1	49.77	19.355	19.816	11.06
2	51.562	7.812	20.312	20.312
3	19.608	5.229	62.745	12.418
4	25.373	5.97	40.299	28.358

Table 4.5: Birch clustering Algorithm with 5 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by stats_clustering.py.

Mini Batch K-means — 5 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	63.115	7.377	13.934	15.574
1	15.819	6.215	57.627	20.339
2	25.749	5.389	49.701	19.162
3	51.724	22.989	18.391	6.897
4	12.5	3.846	70.192	13.462

Table 4.6: Mini Batch K-means clustering Algorithm with 5 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by stats_clustering.py.

Birch — 5 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0			52.675	
1	49.77			
2	51.562			
3			62.745	
4			40.299	

Table 4.7: Birch clustering Algorithm with 5 clusters. Assignment of clusters to quadrants after the first two steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 5 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0			52.675	
1	~~49.77~~	19.355		
2	51.562			
3			62.745	
4			~~40.299~~	28.358

Table 4.8: Birch clustering Algorithm with 5 clusters. Assignment of clusters to quadrants after the last steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Mini Batch K-means — 5 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	63.115			
1				20.339
2			49.701	
3		22.989		
4			70.192	

Table 4.9: Mini Batch K-means clustering Algorithm with 5 clusters. Assignment of clusters to quadrants following the steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 3 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	38.71	10.645	37.097	13.548
1	33.449	10.453	41.463	14.634
2	21.986	7.092	51.064	19.858

Table 4.10: Birch clustering Algorithm with 3 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by feature_clustering.py.

Birch — 10 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	31.41	11.538	43.59	13.462
1	22.535	11.268	43.662	22.535
2	52.381	9.524	23.81	14.286
3	21.429	2.857	58.571	17.143
4	31.579	5.263	57.895	5.263
5	31.304	11.304	40.87	16.522
6	45.902	9.836	31.967	12.295
7	37.805	8.537	36.585	17.073
8	32.653	10.204	42.857	14.286
9	33.333	15.152	39.394	12.121

Table 4.11: Birch clustering Algorithm with 10 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by feature_clustering.py.

Birch — 3 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	38.71			
1			41.463	
2			51.064	

Table 4.12: Birch clustering Algorithm with 3 clusters. Assignment of clusters to quadrants after the first steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 3 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0	38.71	10.645		
1			41.463	
2			51.064	19.858

Table 4.13: Birch clustering Algorithm with 3 clusters. Assignment of clusters to quadrants after the last steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 10 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0			43.59	
1				22.535
2	52.381			
3			58.571	
4			57.895	
5			40.87	
6	45.902			
7	37.805			
8			42.857	
9		15.152		

Table 4.14: Birch clustering Algorithm with 10 clusters. Assignment of clusters to quadrants following the steps in 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 10 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4
0		11.538		
1				22.535
2	52.381			
3				17.143
4			57.895	
5			40.87	
6	45.902			
7	37.805			
8			42.857	
9		15.152		

Table 4.15: Birch clustering Algorithm with 10 clusters. Assignment of clusters to quadrants using the technique described in section 4.3.2. The percentage of songs for the given clusters in a quadrant has been used as placeholders.

Birch — 9 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4	···	Songs in cluster
0	55.102	23.129	13.605	8.163	···	147
1	8.333	1.042	77.083	13.542	···	96
2	23.881	5.97	59.701	10.448	···	67
3	16.279	4.651	65.116	13.953	···	86
4	25.373	5.97	40.299	28.358	···	67
5	37.415	8.844	36.735	17.007	···	147
6	38.571	11.429	32.857	17.143	···	70
7	51.613	8.065	20.968	19.355	···	62
8	50.0	0.0	0.0	50.0	···	2

Table 4.16: Birch clustering Algorithm with 9 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by stats_clustering.py.

Spectral Clustering — 9 clusters

Cluster	Quadrant 1	Quadrant 2	Quadrant 3	Quadrant 4	···	Songs in cluster
0	59.259	22.222	16.667	1.852	···	54
1	24.051	1.266	65.823	8.861	···	79
2	26.131	8.04	48.241	17.588	···	398
3	22.642	3.774	41.509	32.075	···	53
4	65.217	21.739	4.348	8.696	···	23
5	48.148	3.704	29.63	18.519	···	27
6	67.797	6.78	11.864	13.559	···	59
7	31.034	44.828	20.69	3.448	···	29
8	31.818	13.636	45.455	9.091	···	22

Table 4.17: Spectral Clustering Algorithm with 9 clusters. Excerpt from CSV file created by using parse_summary.py on the data created by stats_clustering.py.

Chapter 5

Conclusion

In this chapter, we will answer the problem statement in section 1.2 based on the results from the previous chapter. We will also provide some information about future works for the work that has been conducted in this thesis.

5.1 Summary and Main Contributions

Based on the problem statement in section 1.2, we have studied the related works on the topic of mood in music to start working towards answering these problems. The background for mood in music provided some good directions towards using the valence-arousal dimensional model for emotion categorization, where the MediaEval dataset [50] has been used as the starting point for developing a solution to the problem statement described in section 1.2. The approach to finding correlations between music and mood has been aided by machine learning, where we have reviewed clustering algorithms to use with the dataset. The dataset has been pre-processed and used with the clustering algorithms in two ways, with the intent to explore algorithms for finding optimal clusters in terms of explaining the correlation between music and mood. Features have been extracted from 45-second clips with Principal Component analysis before clustering, and features have been extracted from clips of whole songs and summarized statistically before clustering. In the methodology chapter, we have specified metrics to determine the quality of the clusters from the different clustering algorithms. The main metric we have focused this thesis around is if the clusters of a clustering algorithm can represent each quadrant of the valence-arousal dimensional model (see figure 3.1), where a cluster would have the majority of its songs in a quadrant. Furthermore, we have presented the results from the different clustering procedures, as well as devised a way of assigning clusters to quadrants.

Based on our results, we answer the stated research questions:

1. *Can songs, based on extracted features, be grouped into clusters that correlate with their annotated emotion, and what is the quality of the clusters in terms of cluster separation and statistical summary?*

The clustering algorithms that perform best in terms of grouping songs into clusters that correlate with their annotated emotion, seem to mostly divide their clusters into quadrants 1 and 3. An overview of the quadrants can be seen in figure 3.1. Quadrant 1 is represented by high arousal and positive valence, where quadrant 3 are represented by low arousal and negative valence. Quadrants 2 and 4 are the quadrants that are the hardest to make clusters within, as the dataset is already ordered so that quadrants 1 and 3 have the most songs in them. Quadrant 2 is represented by high arousal and negative valence, where quadrant 4 is represented by low arousal and positive valence. Even though the clustering algorithms orders the clusters towards quadrants 1 and 3, the assigning of clusters to quadrants makes use of quadrants 2 and 4 for clustering new songs with the trained models. The clustering procedures from experiment (ii) shows good cluster separation between quadrants, and are the best of the two experiments. The clustering algorithms with median distribution values, as well as the algorithms surrounding the median distribution value, are the ones that perform the best. These algorithms can be seen in figure 4.7 with plots in appendix B.

2. *How can we evaluate the cluster homogeneity?*

   To evaluate the homogeneity of clusters, we have calculated the distribution of song in clusters across the quadrants. This calculation takes the percentage of songs in a quadrant, after the percentage of songs in a quadrant across all clusters has been averaged, and evaluates the distribution across quadrants on the form $Distribution = 1/Q1 + 1/Q2 + 1/Q3 + 1/Q4$, where the Q's represent the percentage of songs in the respective quadrants. When this formula is applied to the clustering algorithms, we get an ordering of the algorithms where the most homogeneous clusters are around the median, and the algorithms with extreme percentages of songs in a quadrant are on the top and the bottom of the list. This formula allows us to pick clustering algorithms in terms of what distribution we want to use when clustering new songs. The exceptions to using this method, as mention in the summary of results section 4.3.3, is when the clusters have a low number of songs in them, and therefore skew the percentage for songs in a quadrant within a cluster to be represented as extreme.

3. *Is it possible to make a reliable model that can be applied to find songs for certain moods?*

   The work we have done in the thesis is readily available to use to find songs for certain moods. The clustering procedures from experiment (ii), performed by stats_clustering.py, shows the best performance of the two experiments. The clustering algorithms around the median of the distribution value in table 4.7 are the best models to use when clustering new songs. The easiest moods to find songs for are the ones represented by quadrants 1 (excited, happy, pleased) and 3 (sad,

bored, sleepy) as they have the most reliable clustering overall. Songs that end up in the clusters that represents these quadrants can be used in for example exercise and sports, if it is a cluster that represents quadrant 1, and can be for example soothing, if it is a cluster that represents quadrant 3.

The work we have done in the thesis yields a lot of good results, and it is an interesting approach for finding correlation between music and mood. The clustering procedures from experiment (ii), generated by stats_clustering.py, shows the best results. The clustering algorithms Mini Batch K-means with 5 clusters and Birch with 5 clusters are the configurations with the median distribution values, and they are good starting points for clustering new songs. Validating the clustering by using songs from papers reviewed in the music and exercise section2.2 is an important step in the future work. This should be a good starting point as the clustering algorithms perform best for separating clusters into quadrant 1 and 3. Quadrant 1 is high arousal and positive valence, which should be explored if it correlates with the songs in the reviewed papers on music and exercise in section 2.2. Furthermore, there is possibility for improving the clustering in terms of using carefully selected features and clustering parameters.

5.2 Future Work

In the discussion provided in section 4.4, we described ways the prototype we have designed can be used in future work. The songs in the papers on music and exercise, reviewed in section 2.2, is a good way to determine if music used in exercise is of a certain mood. Features from these songs can be extracted with pyAudioAnalysis the same way we have done for the songs in the MediaEval dataset. Furthermore, the discussion on the use of features and clustering algorithms mentions some points regarding how the clustering could become more efficient in distinguishing emotions in terms of clusters in quadrants. Using a combination of different features with the clustering algorithms may yield better results if the features are selected carefully according to the applicability of these features in prior research.

Bibliography

[1] James J Annesi. Effects of music, television, and a combination entertainment system on distraction, exercise adherence, and physical output in adults. *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement*, 33(3):193, 2001.

[2] Martin J Barwood, Neil JV Weston, Richard Thelwell, and Jennifer Page. A motivational music and video intervention improves high-intensity exercise performance. *Journal of sports science & medicine*, 8(3):435, 2009.

[3] Javier Béjar Alonso. K-means vs mini batch k-means: A comparison. 2013.

[4] Kerstin Bischoff, Claudiu S Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. Music mood and theme classification-a hybrid approach. In *ISMIR*, pages 657–662, 2009.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[6] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[7] Imogen N Clark, Felicity A Baker, Casey L Peiris, Georgie Shoebridge, and Nicholas F Taylor. The brunel music rating inventory-2 is a reliable and valid instrument for older cardiac rehabilitation patients selecting music for exercise. *Psychology of Music*, 44(2):249–262, 2016.

[8] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):603–619, 2002.

[9] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, January 1989.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[11] Judy Edworthy and Hannah Waring. The effects of music tempo and loudness level on treadmill exercise. *Ergonomics*, 49(15):1597–1610, 2006.

[12] Dave Elliott, Sam Carr, and Duncan Orme. The effect of motivational music on sub-maximal exercise. *European Journal of Sport Science*, 5(2):97–106, 2005.

[13] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[14] Jerome H Friedman. Data mining and statistics: What's the connection? *Computing Science and Statistics*, 29(1):3–9, 1998.

[15] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*, volume 1, pages 191–194, 2005.

[16] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12), 2015.

[17] Fabien Gouyon, François Pachet, Olivier Delerue, et al. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, page 26, 2000.

[18] Charles J Hardy and W Jack Rejeski. Not what, but how one feels: the measurement of affect during exercise. *Journal of sport and exercise psychology*, 11(3):304–317, 1989.

[19] Xiao Hu and J Stephen Downie. Exploring mood metadata: Relationships with genre, artist and usage metadata. In *ISMIR*, pages 67–72, 2007.

[20] Jasmin C Hutchinson, Todd Sherman, Lyndsey Davis, Dusty Cawthon, Nathan B Reeder, Gershon Tenenbaum, et al. The influence of asynchronous motivational music on a supramaximal exercise bout. *International Journal of Sport Psychology*, 42(2):135–148, 2011.

[21] Costas I Karageorghis. The scientific application of music in sport and exercise. *Sport and exercise psychology*, 109:138, 2008.

[22] Costas I. Karageorghis, Leighton Jones, David-Lee Priest, Rose I. Akers, Adam Clarke, Jennifer M. Perry, Benjamin T. Reddick, Daniel T. Bishop, and Harry B.T. Lim. Revisiting the relationship between exercise heart rate and music tempo preference. *Research Quarterly for Exercise and Sport*, 82(2):274–284, 2011. PMID: 21699107.

[23] Costas I Karageorghis, Denis A Mouzourides, David-Lee Priest, Tariq A Sasso, Daley J Morrish, and Carolyn L Walley. Psychophysical and ergogenic effects of synchronous music during treadmill walking. *Journal of sport and exercise psychology*, 31(1):18–36, 2009.

[24] Costas I Karageorghis and David-Lee Priest. Music in the exercise domain: a review and synthesis (part i). *International review of sport and exercise psychology*, 5(1):44–66, 2012.

[25] Costas I Karageorghis and David-Lee Priest. Music in the exercise domain: a review and synthesis (part ii). *International Review of Sport and Exercise Psychology*, 5(1):67–84, 2012.

[26] Costas I Karageorghis, David-Lee Priest, Peter C Terry, Nikos LD Chatzisarantis, and Andrew M Lane. Redesign and initial validation of an instrument to assess the motivational qualities of music in exercise: The brunel music rating inventory-2. *Journal of sports sciences*, 24(8):899–909, 2006.

[27] Costas I Karageorghis, DavidLee Priest, LS Williams, RM Hirani, KM Lannon, and BJ Bates. Ergogenic and psychological effects of synchronous music during circuit-type exercise. *Psychology of Sport and Exercise*, 11(6):551–559, 2010.

[28] Costas I Karageorghis, Peter C Terry, and Andrew M Lane. Development and initial validation of an instrument to assess the motivational qualities of music in exercise and sport: The brunel music rating inventory. *Journal of sports sciences*, 17(9):713–724, 1999.

[29] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *Proc. ISMIR*, volume 86, pages 937–952, 2010.

[30] Alexander Lerch. *An introduction to audio content analysis: Applications in signal processing and music informatics*. Wiley-IEEE Press, 2012.

[31] Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.

[32] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[33] Sebastian Napiorkowski. Music mood recognition: State of the art review. *MUS-15 July*, 10, 2015.

[34] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. *CUIDADO IST Project Report*, 54(0):1–25, 2004.

[35] Selen Razon, Itay Basevitch, William Land, Brooke Thompson, and Gershon Tenenbaum. Perception of exertion and attention allocation as a function of visual and auditory conditions. *Psychology of Sport and Exercise*, 10(6):636–643, 2009.

[36] W Jack Rejeski. Perceived exertion: an active or passive process? *Journal of Sport and Exercise Psychology*, 7(4):371–378, 1985.

[37] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

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

[39] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543–565, 2012.

[40] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, July 1959.

[41] Eric Scheirer and Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *1997 IEEE international conference on acoustics, speech, and signal processing*, volume 2, pages 1331–1334. IEEE, 1997.

[42] Stefan Schneider, Christopher D Askew, Thomas Abel, and Heiko K Strüder. Exercise, music, and the brain: is there a central pattern generator? *Journal of Sports Sciences*, 28(12):1337–1343, 2010.

[43] Emery Schubert, Joe Wolfe, and Alex Tarnopolsky. Spectral centroid and timbre in complex, multiple instrumental textures. In *Proceedings of the international conference on music perception and cognition, North Western University, Illinois*, pages 112–116. sn, 2004.

[44] Susan Elens Schwartz, Bo Fernhall, and Sharon A Plowman. Effects of music on exercise performance. *Journal of Cardiopulmonary Rehabilitation and Prevention*, 10(9):312–316, 1990.

[45] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[46] Naama Shaulov and Dubi Lufi. Music and light during indoor cycling. *Perceptual and motor skills*, 108(2):597–607, 2009.

[47] DS Shete, SB Patil, and S Patil. Zero crossing rate and energy of the speech signal of devanagari script. *IOSR-JVSP*, 4(1):1–5, 2014.

[48] Stuart D Simpson and Costas I Karageorghis. The effects of synchronous music on 400-m sprint performance. *Journal of sports sciences*, 24(10):1095–1102, 2006.

[49] Robert R Sokal. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.*, 38:1409–1438, 1958.

[50] Mohammad Soleymani, Micheal N Caro, Erik M Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, pages 1–6. ACM, 2013.

[51] Gershon Tenenbaum and Robert C Eklund. *Handbook of sport psychology*. Wiley Online Library, 2007.

[52] Robert E Thayer. *The biopsychology of mood and arousal*. Oxford University Press, 1990.

[53] Miller Tiev, Swank Ann Manire, John Robertson Robert, and Wheeler Barbara. Effect of music and dialogue on perception of exertion, enjoyment, and metabolic responses during exercise. *International Journal of Fitness*, 6(2), 2010.

[54] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[55] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[56] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[57] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[58] Bin Wu, Erheng Zhong, Andrew Horner, and Qiang Yang. Music emotion recognition by multi-label multi-layer multi-instance multi-view learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 117–126. ACM, 2014.

[59] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Pacific-Rim Conference on Multimedia*, pages 566–574. Springer, 2004.

[60] Rui Xu and Donald C Wunsch. Survey of clustering algorithms. 2005.

[61] Yi-Hsuan Yang and Homer H Chen. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):40, 2012.

[62] Yi-Hsuan Yang, Chia-Chu Liu, and Homer H Chen. Music emotion classification: A fuzzy approach. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 81–84. ACM, 2006.

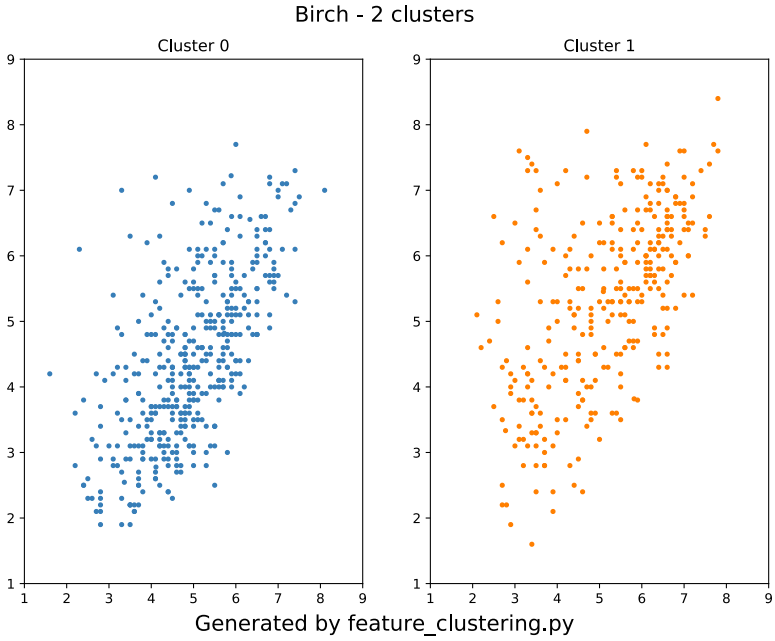
[63] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2011.

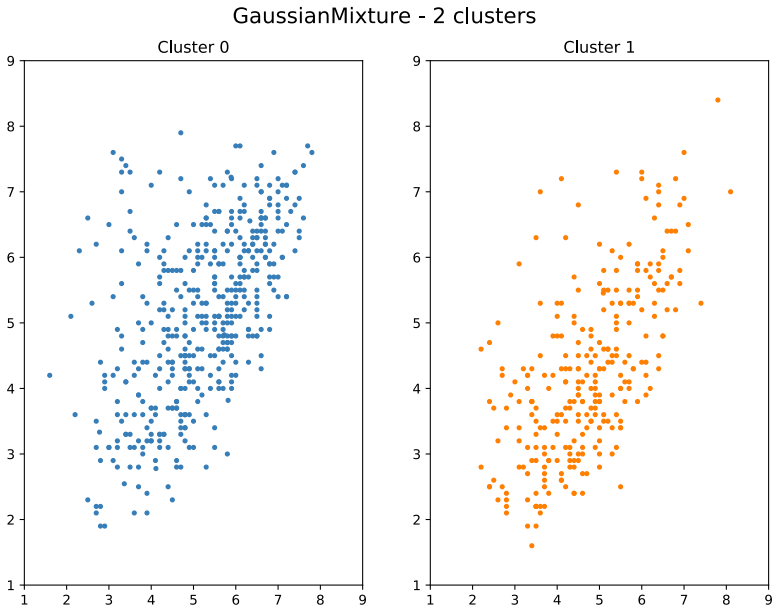
[64] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

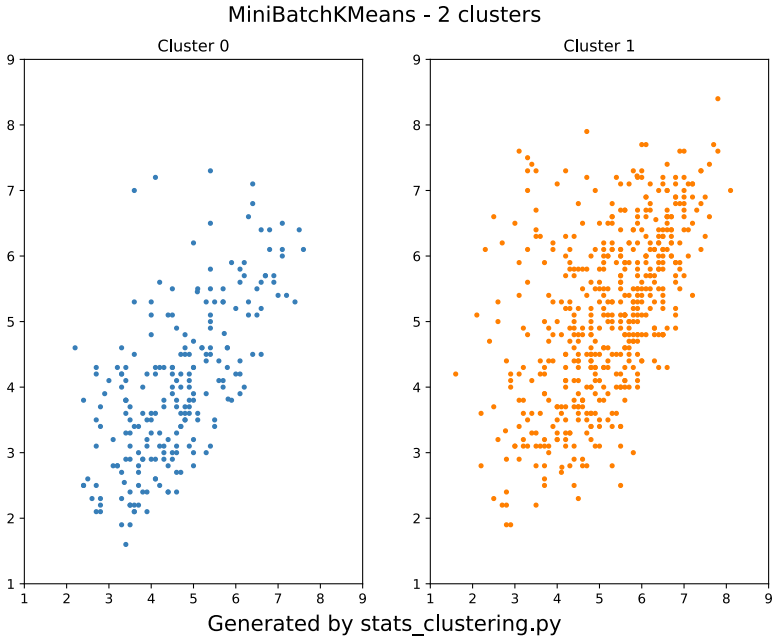
Appendices

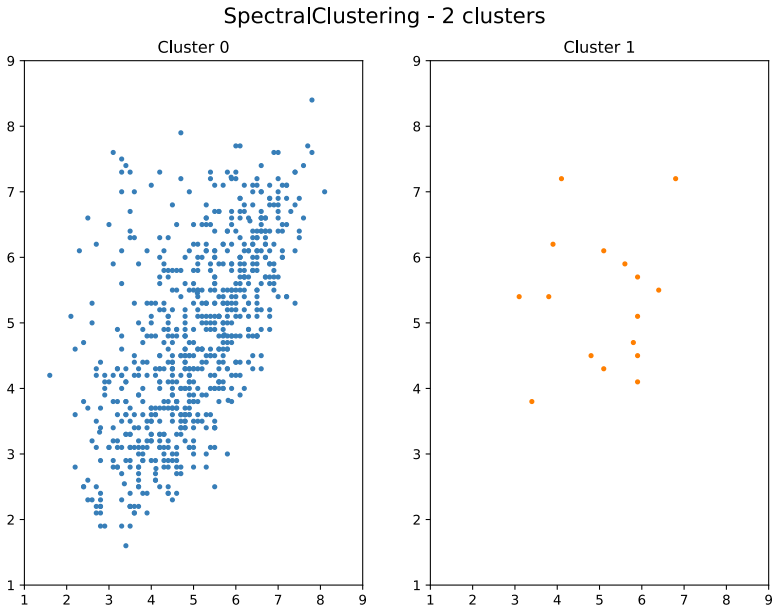
Appendix A

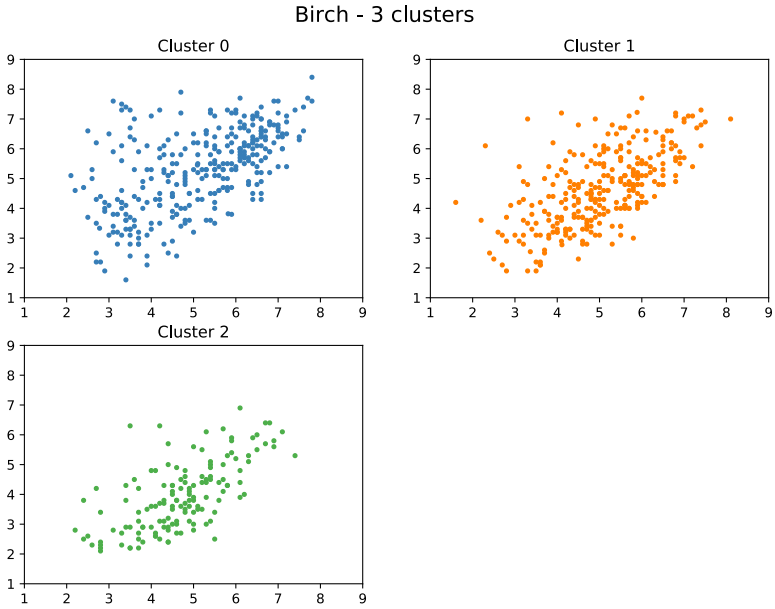
Plots from feature_clustering.py

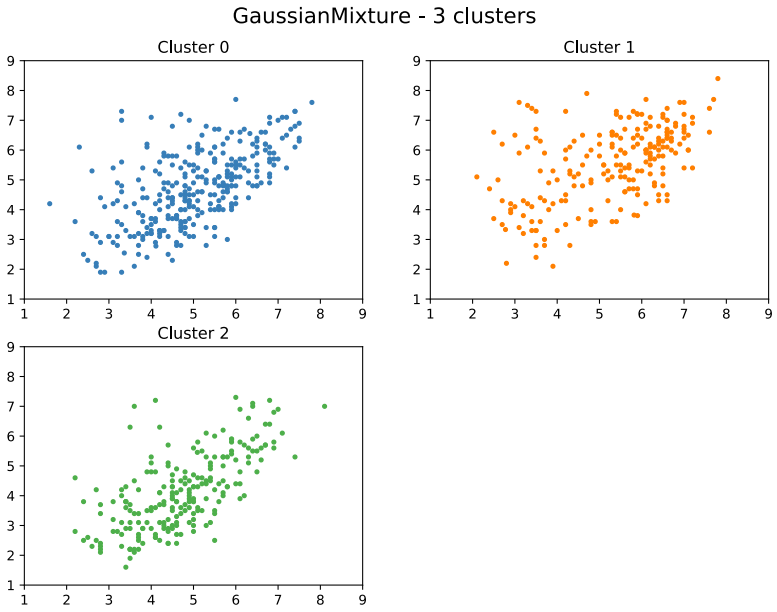



GaussianMixture - 2 clusters
Cluster 0
Cluster 1
Generated by feature_clustering.py

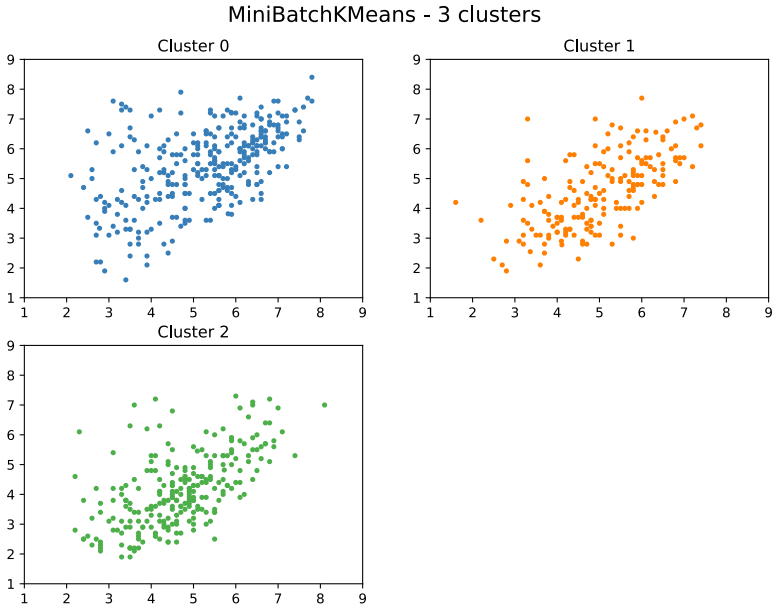

MiniBatchKMeans - 2 clusters
Cluster 0
Cluster 1
Generated by stats_clustering.py


SpectralClustering - 2 clusters
Cluster 0
Cluster 1
Generated by feature_clustering.py

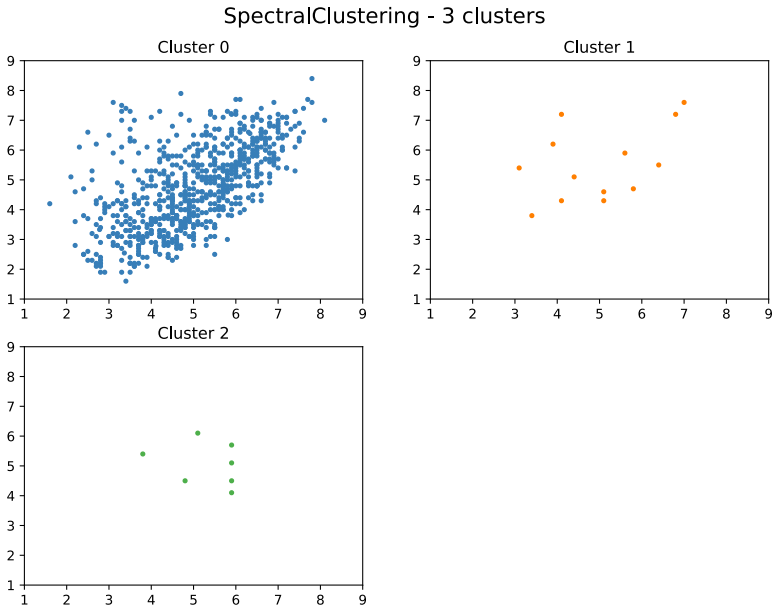

Birch - 3 clusters
Cluster 0
Cluster 1
Cluster 2
Generated by feature_clustering.py



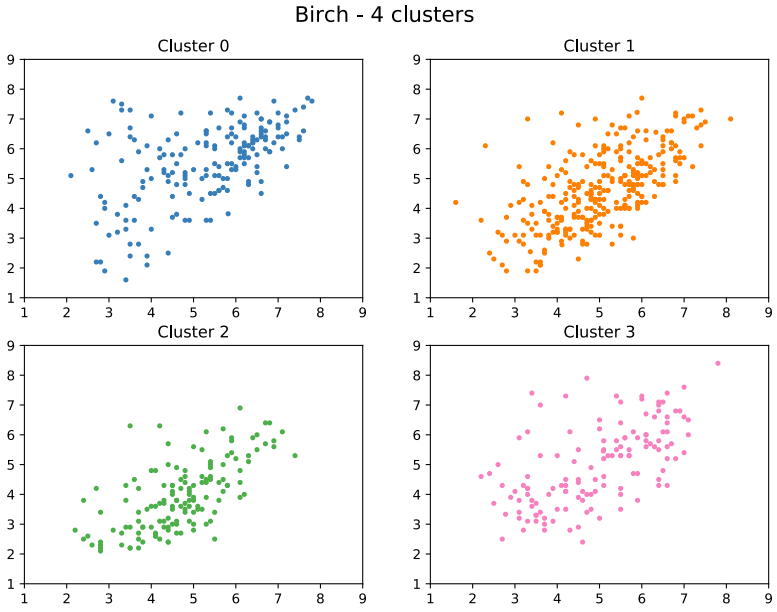
Generated by feature_clustering.py



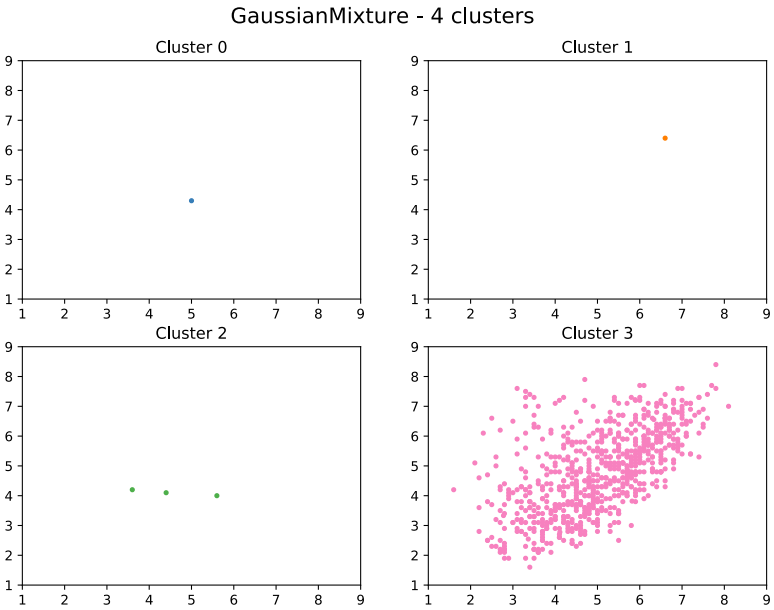
Generated by feature_clustering.py

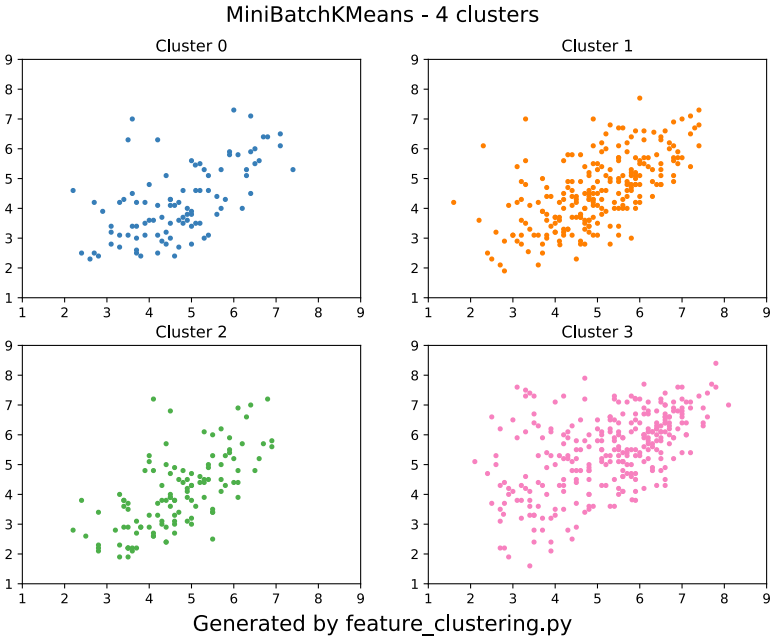


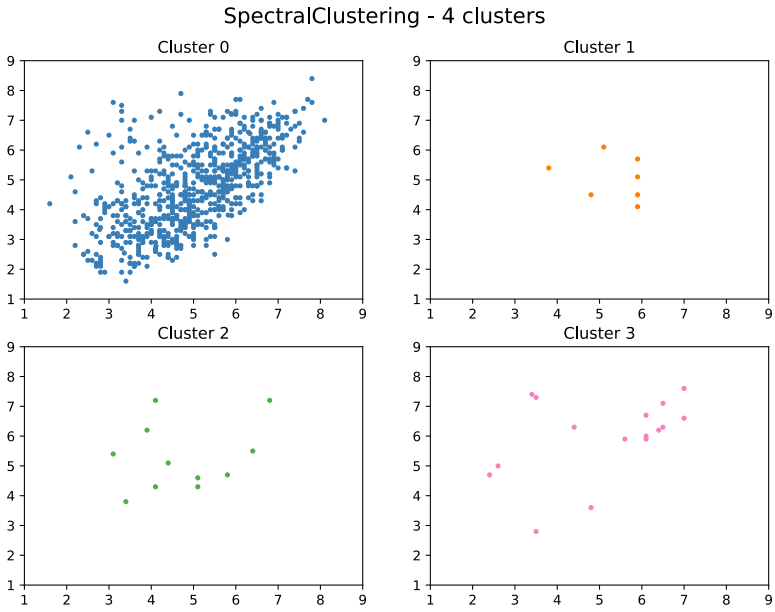
Generated by feature_clustering.py



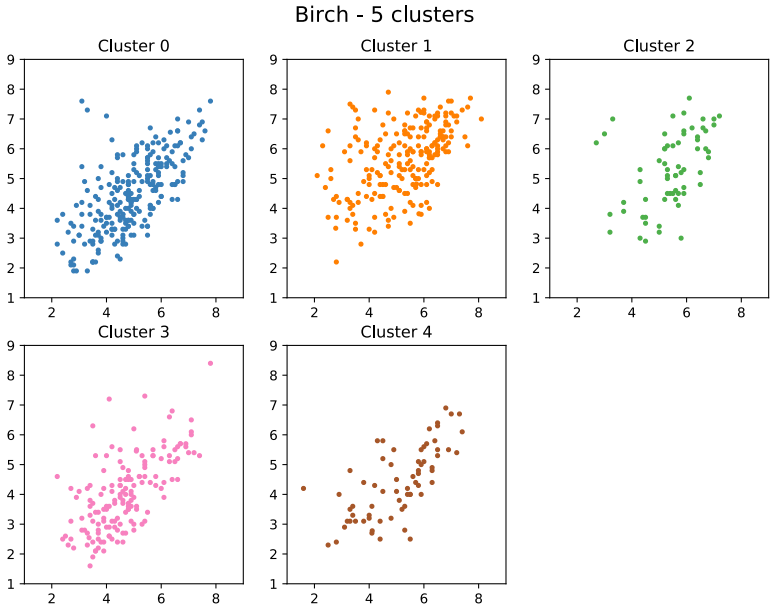
Generated by feature_clustering.py


GaussianMixture - 4 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Generated by feature_clustering.py

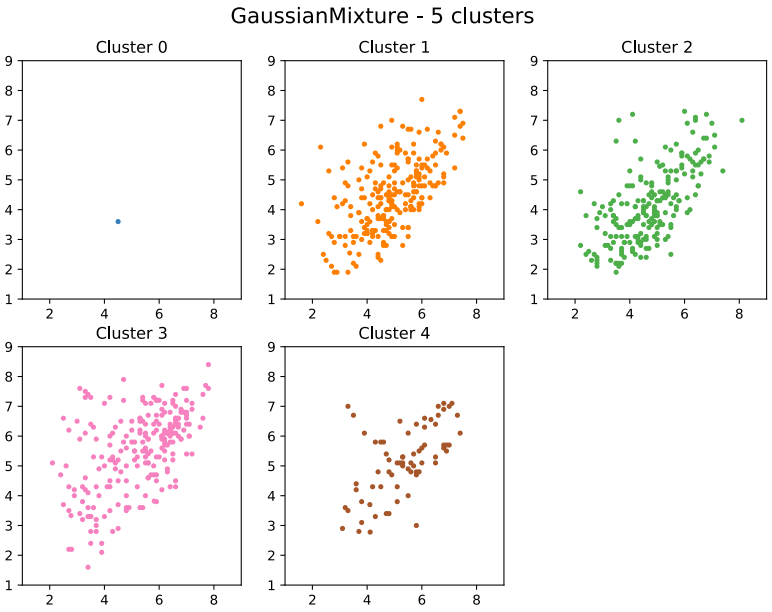

MiniBatchKMeans - 4 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Generated by feature_clustering.py



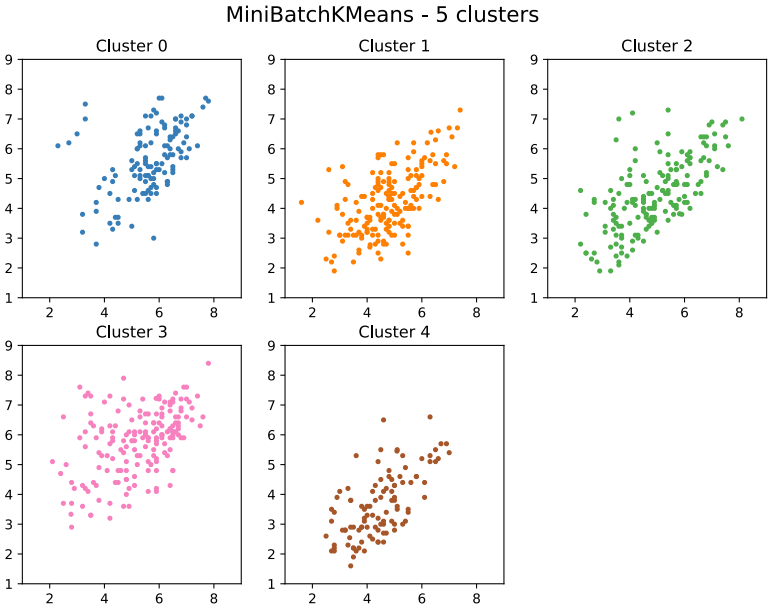
Generated by feature_clustering.py



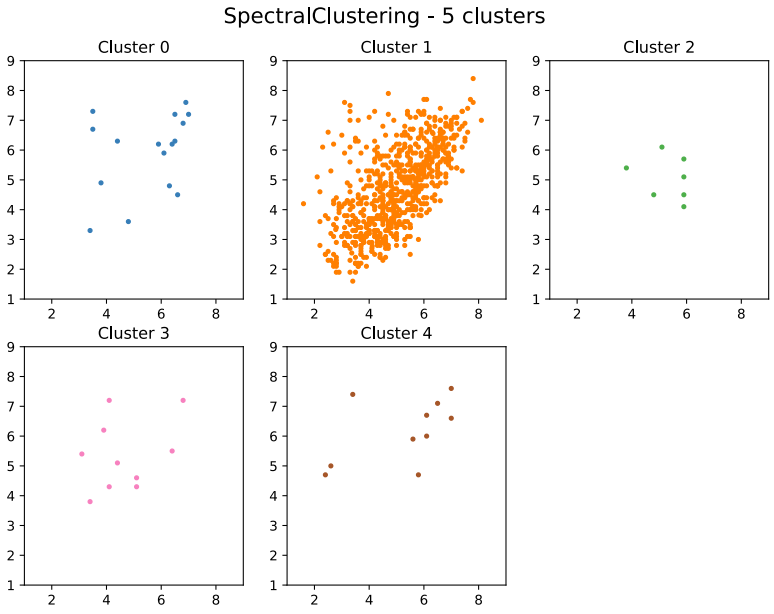
Generated by stats_clustering.py



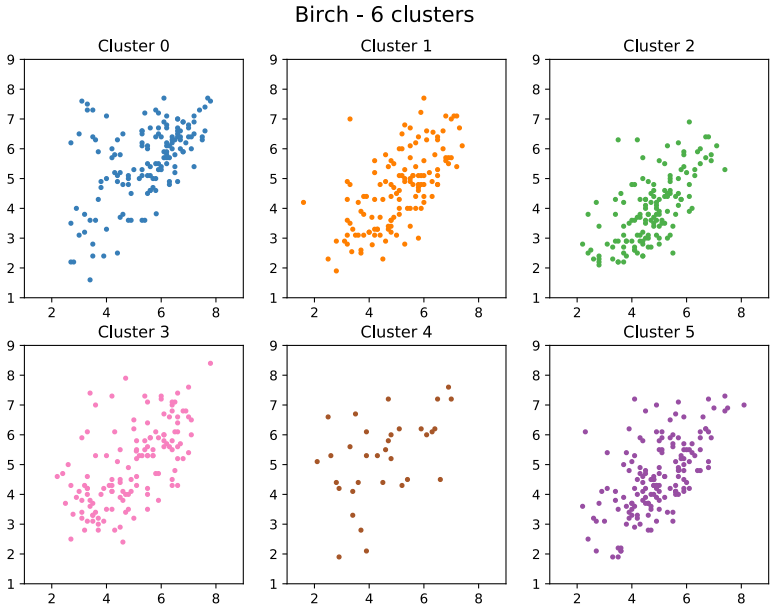
Generated by feature_clustering.py



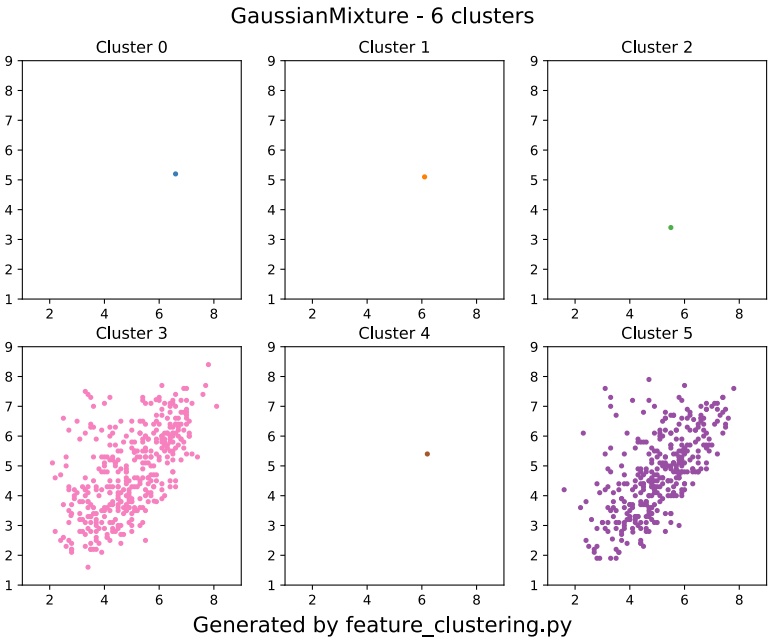
Generated by stats_clustering.py

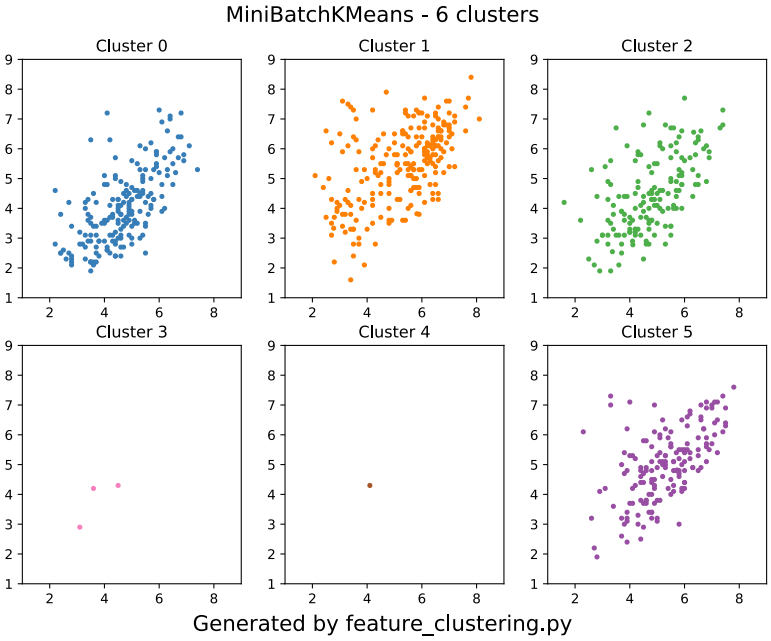


Generated by feature_clustering.py

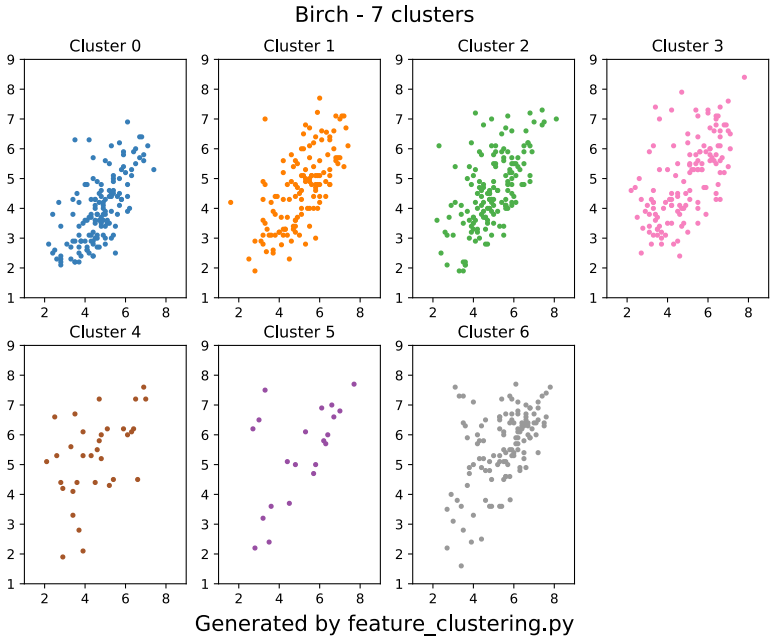


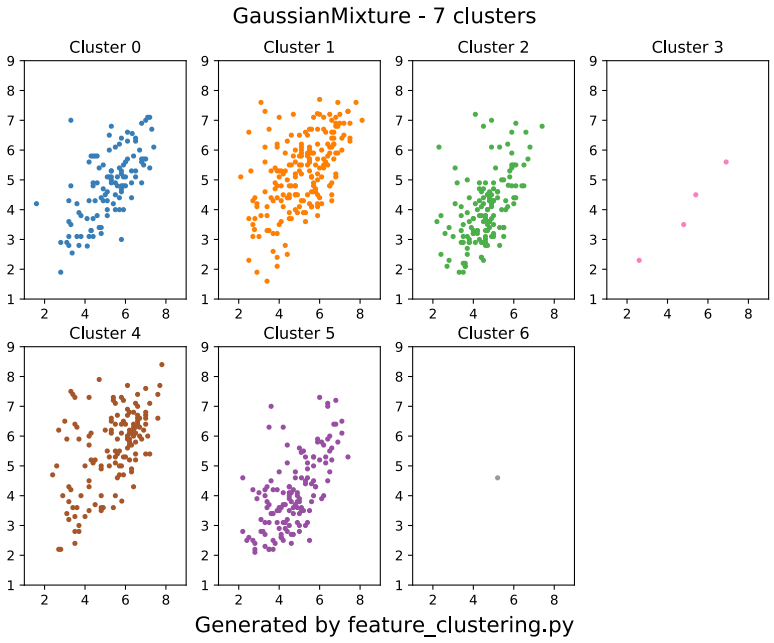
Generated by feature_clustering.py

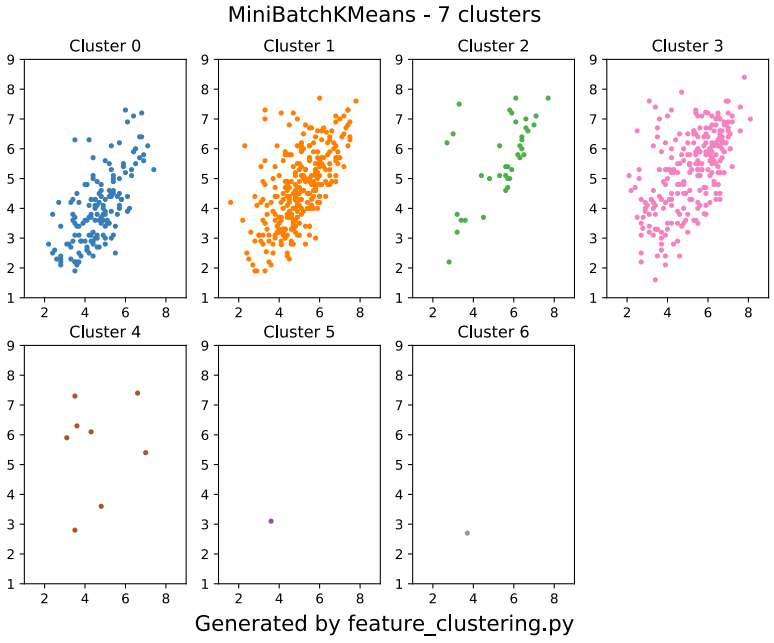

GaussianMixture - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

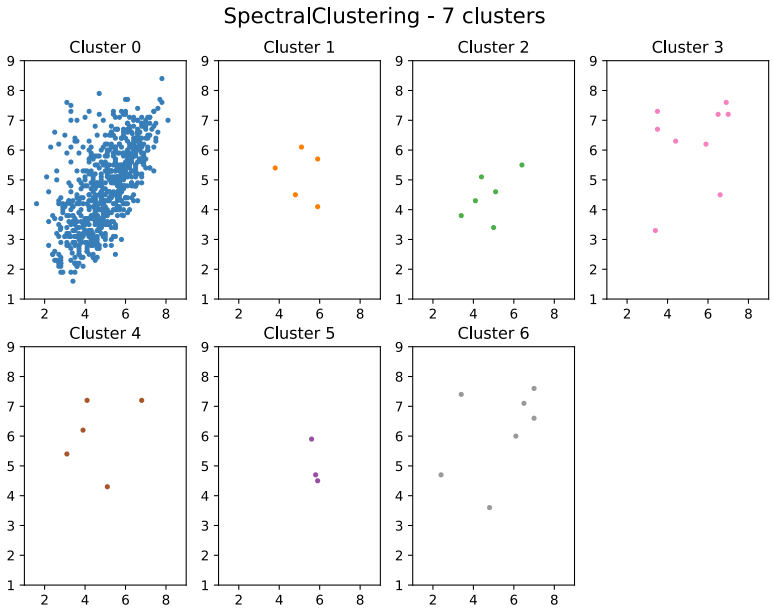

MiniBatchKMeans - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py


SpectralClustering - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

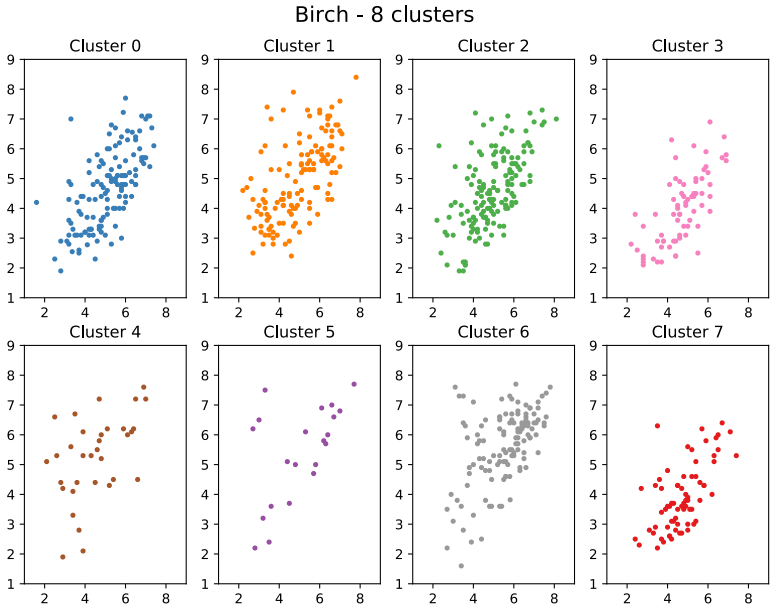

Birch - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py


GaussianMixture - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py

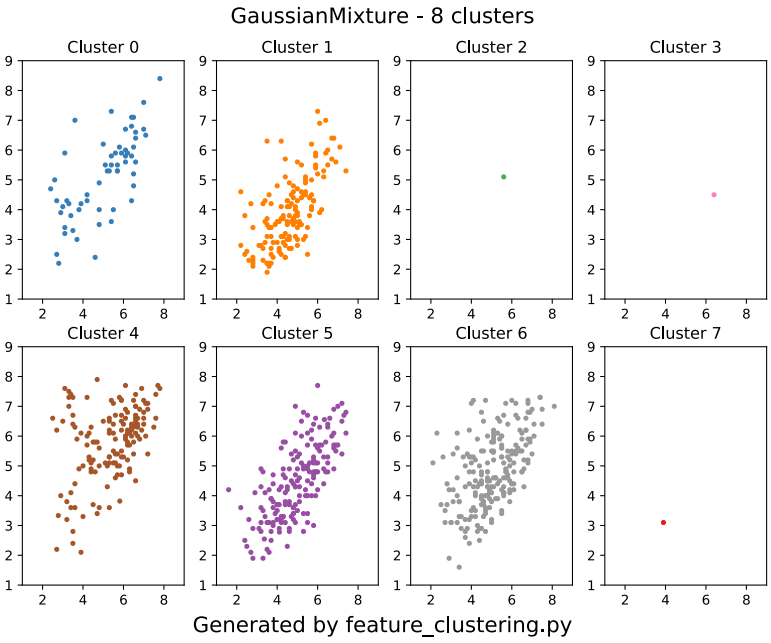

MiniBatchKMeans - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py

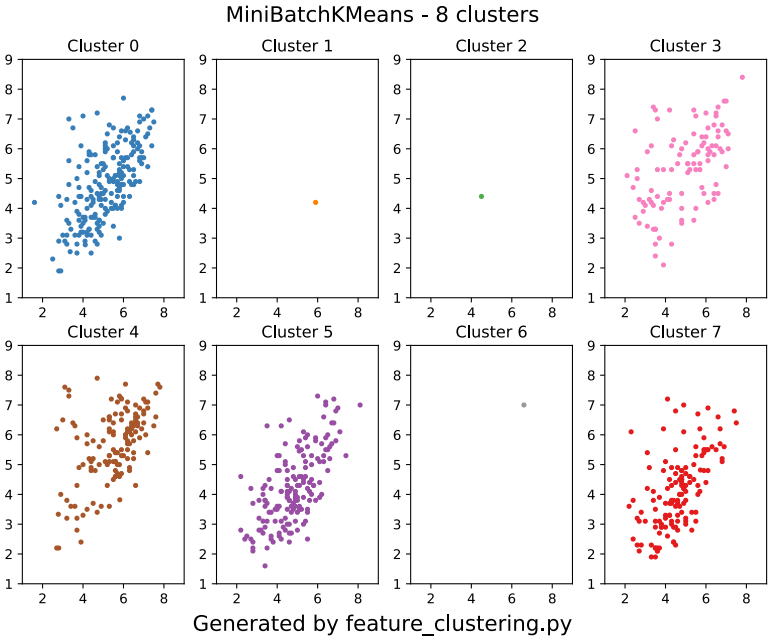


Generated by feature_clustering.py

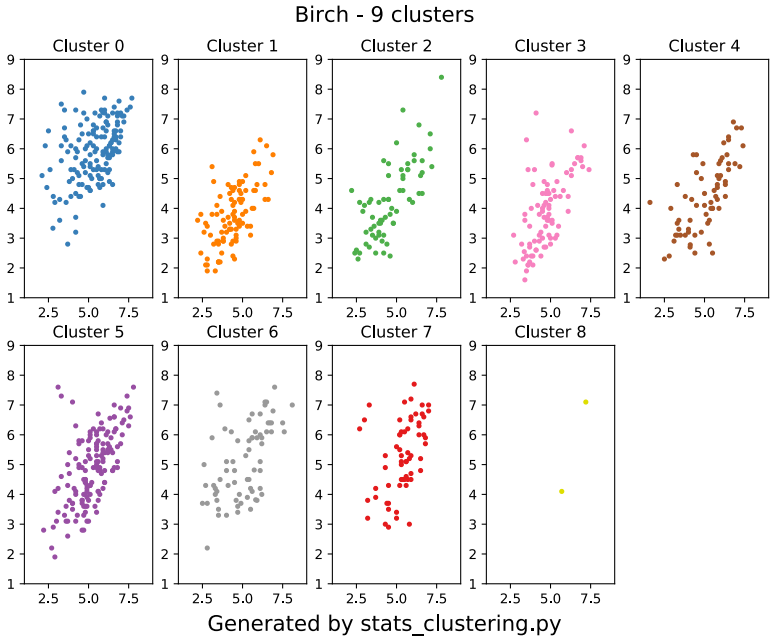


Generated by feature_clustering.py

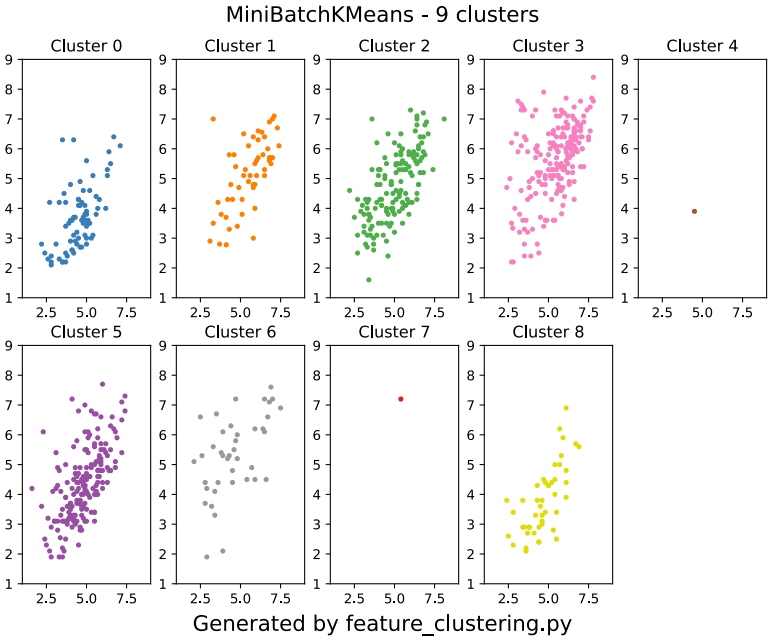

GaussianMixture - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

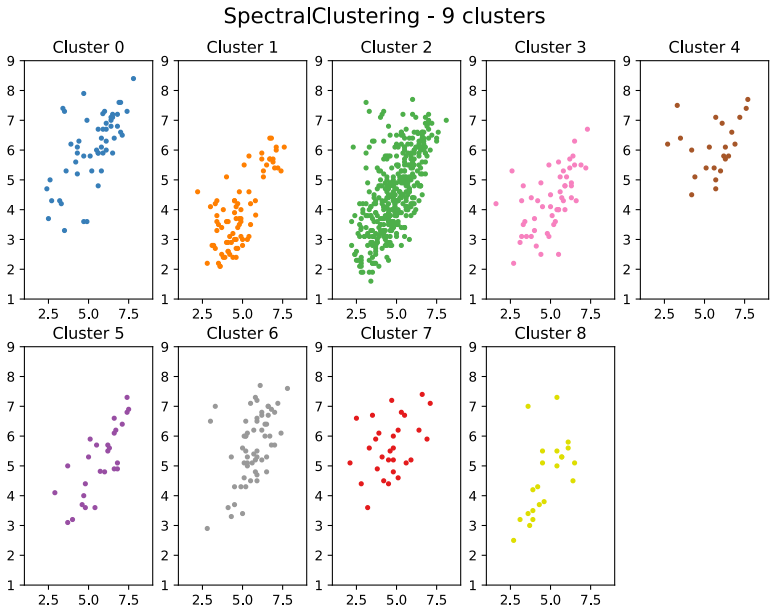

MiniBatchKMeans - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py


SpectralClustering - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

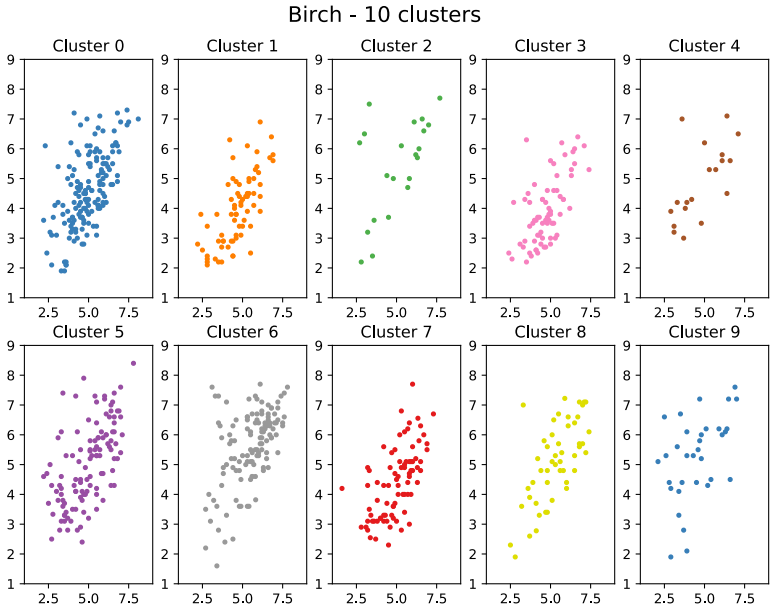

Birch - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by stats_clustering.py


GaussianMixture - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by feature_clustering.py

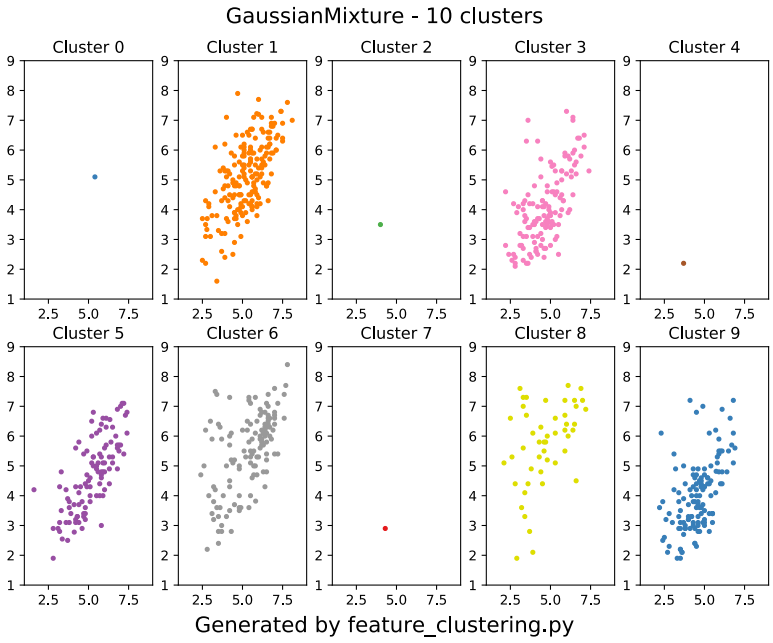

MiniBatchKMeans - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by feature_clustering.py

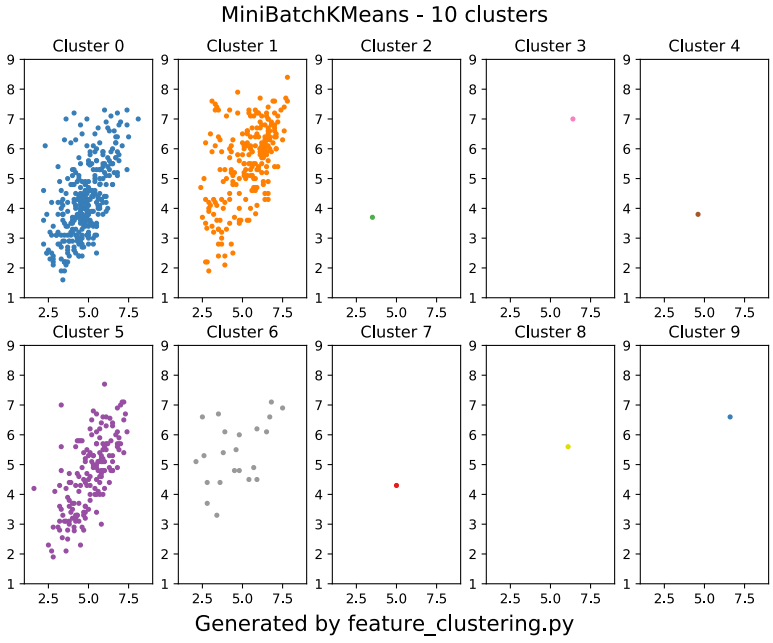


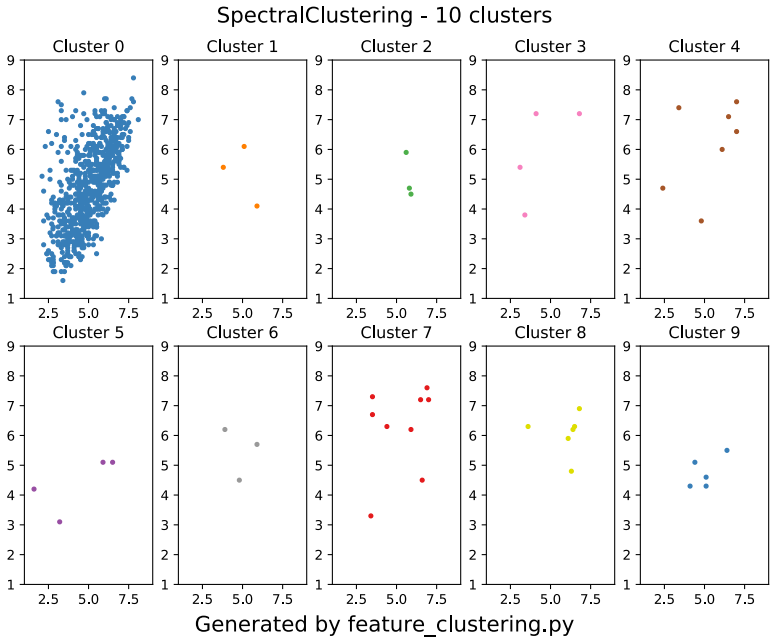
Generated by stats_clustering.py



Generated by feature_clustering.py

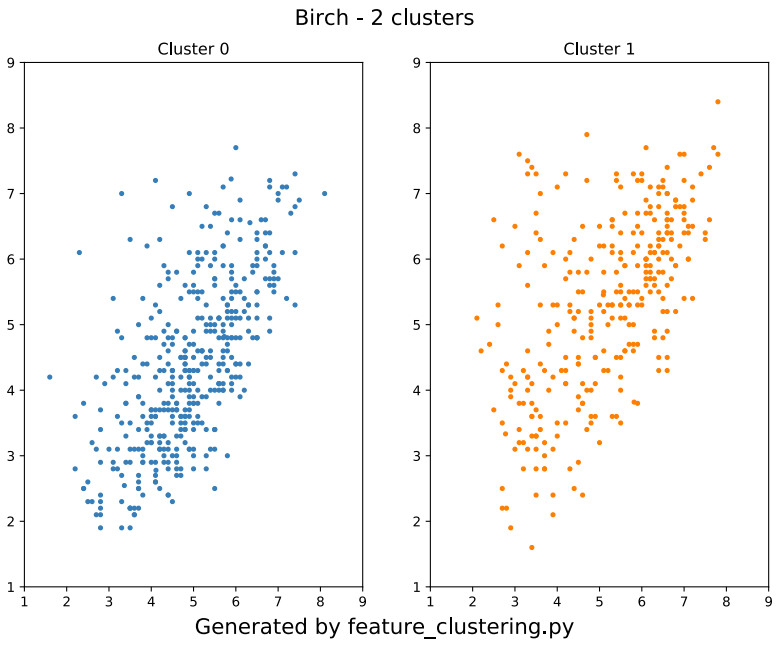

GaussianMixture - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py

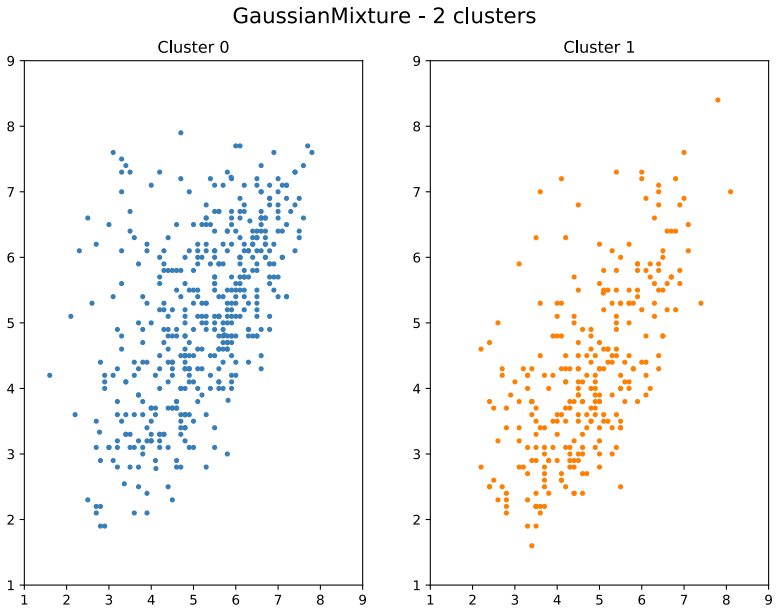

MiniBatchKMeans - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py

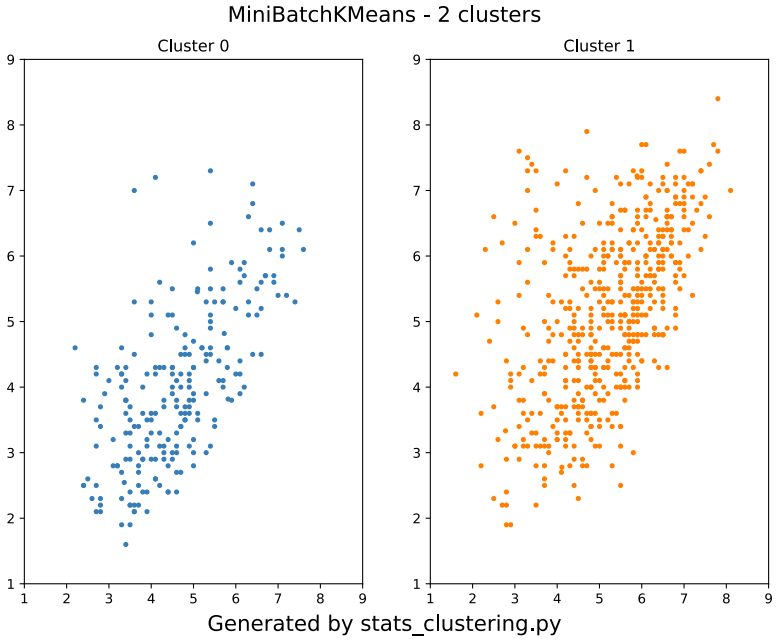

SpectralClustering - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py

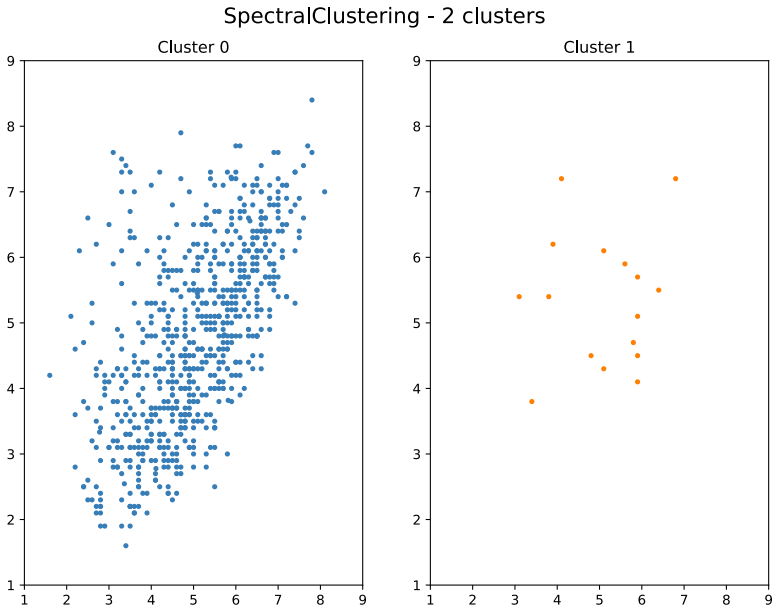
Appendix B

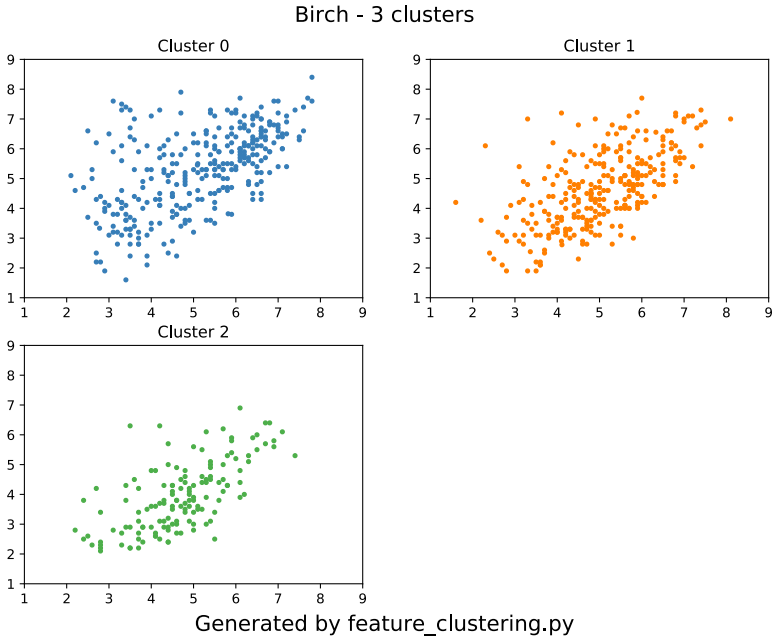
Plots from stats_clustering.py

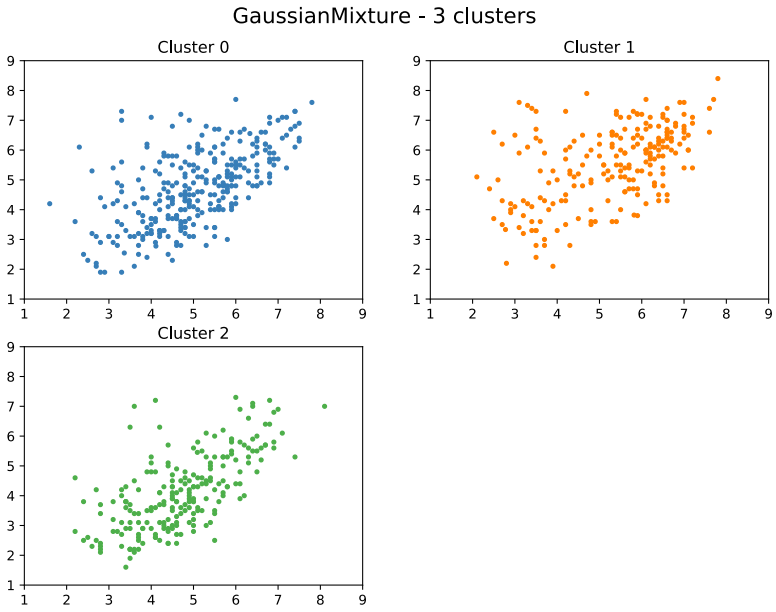



GaussianMixture - 2 clusters
Cluster 0
Cluster 1
Generated by feature_clustering.py

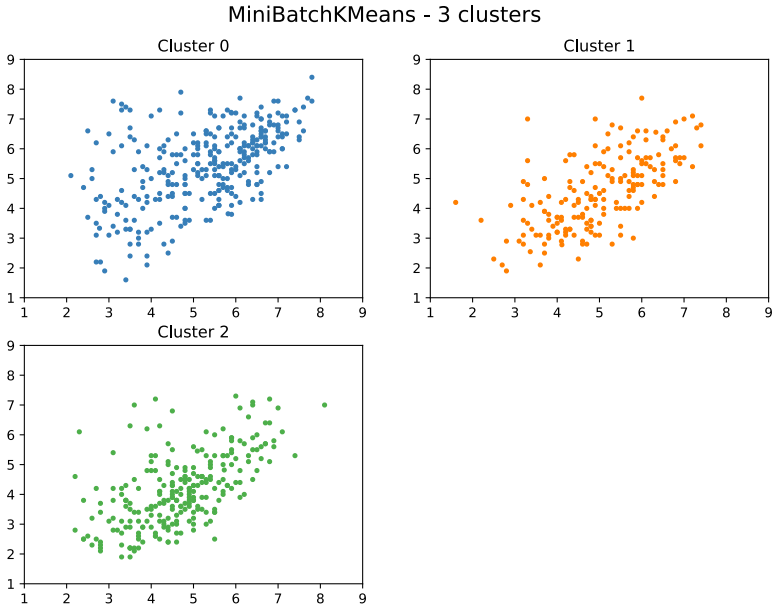

MiniBatchKMeans - 2 clusters
Cluster 0
Cluster 1
Generated by stats_clustering.py


SpectralClustering - 2 clusters
Cluster 0
Cluster 1
Generated by feature_clustering.py

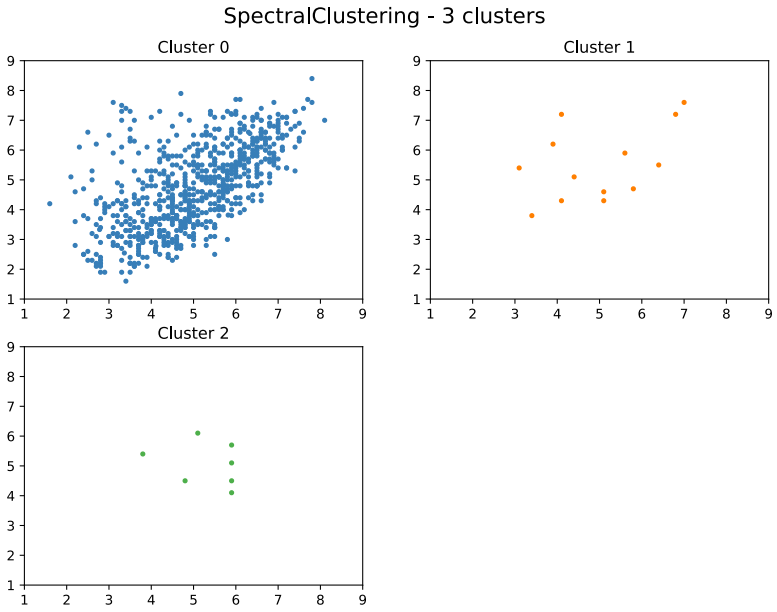

Birch - 3 clusters
Cluster 0
Cluster 1
Cluster 2
Generated by feature_clustering.py



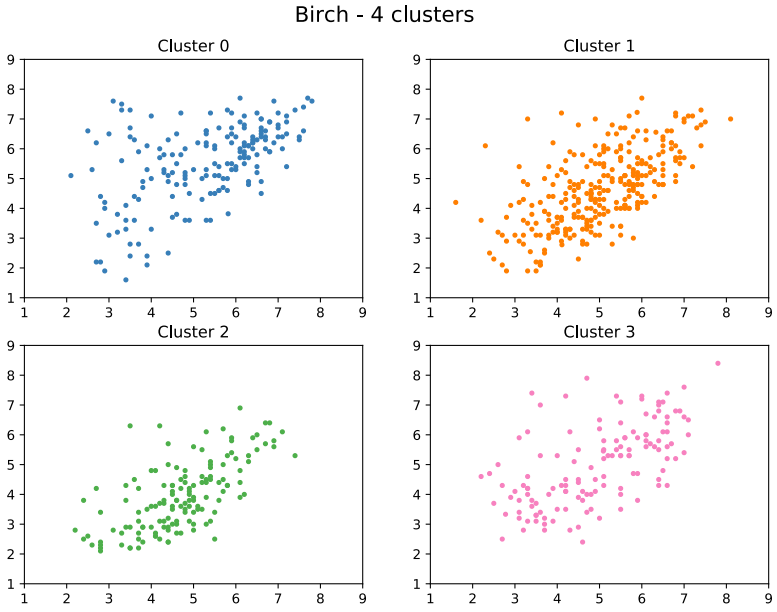
Generated by feature_clustering.py



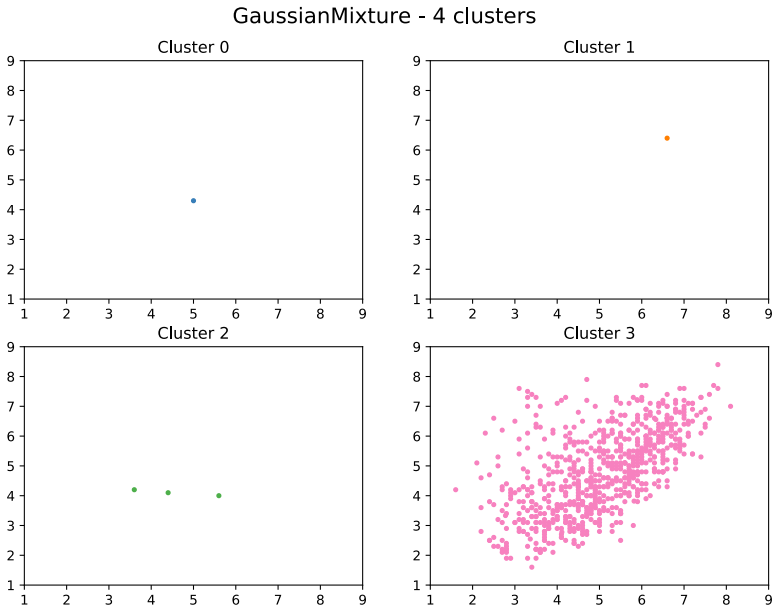
Generated by feature_clustering.py

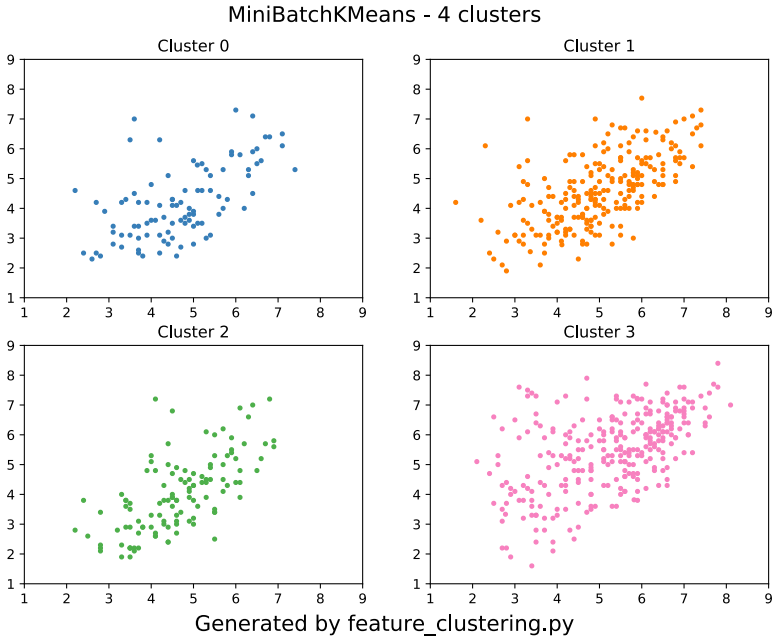


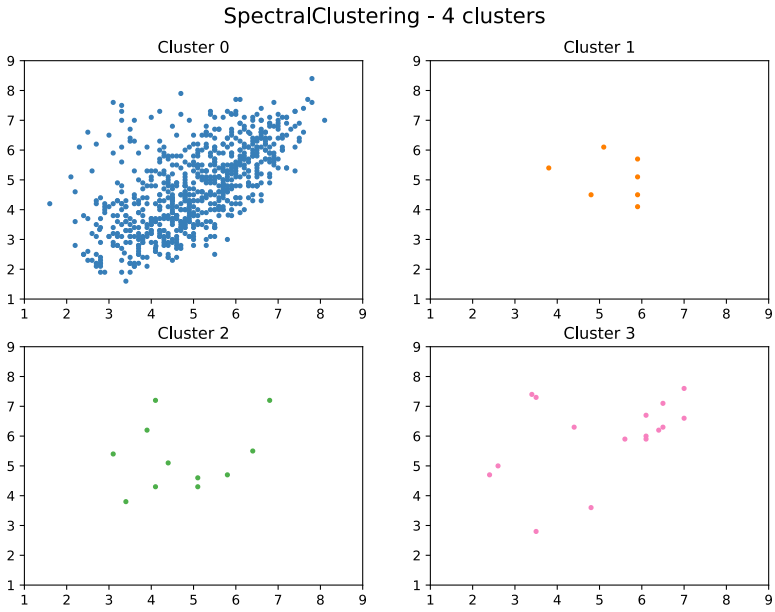
Generated by feature_clustering.py



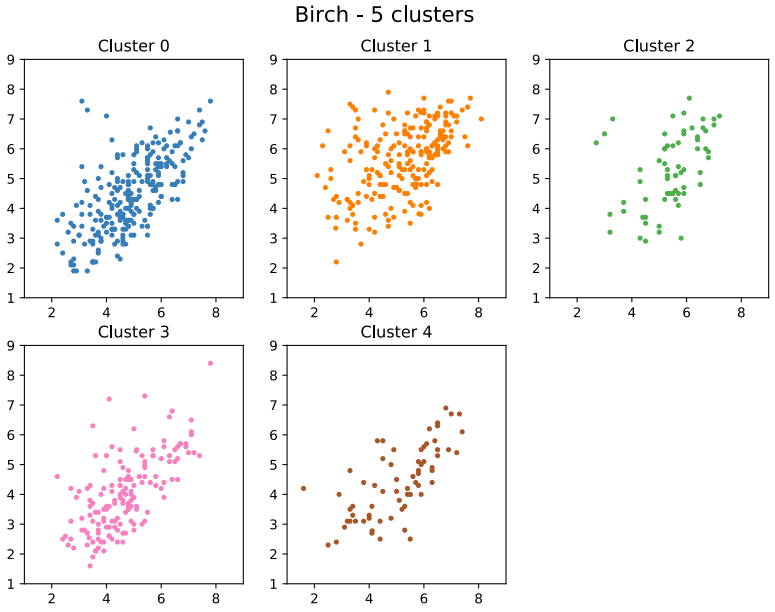
Generated by feature_clustering.py


GaussianMixture - 4 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Generated by feature_clustering.py

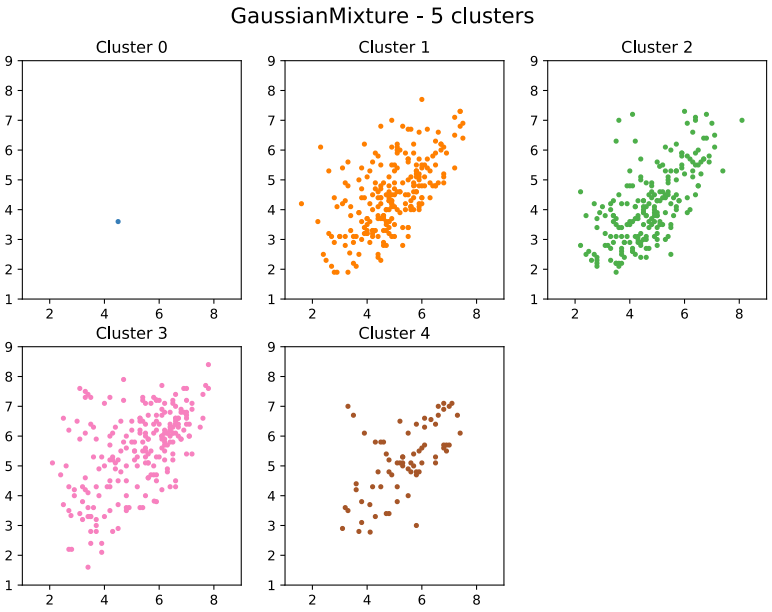

MiniBatchKMeans - 4 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Generated by feature_clustering.py



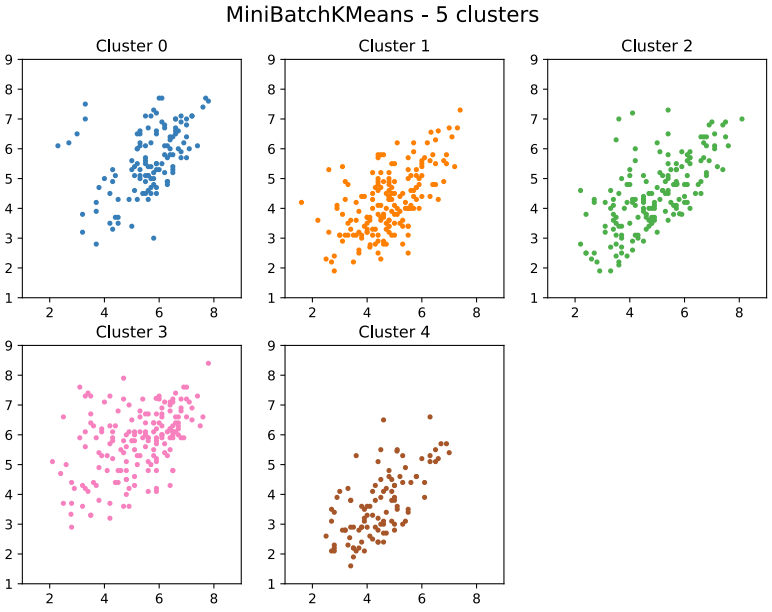
Generated by feature_clustering.py



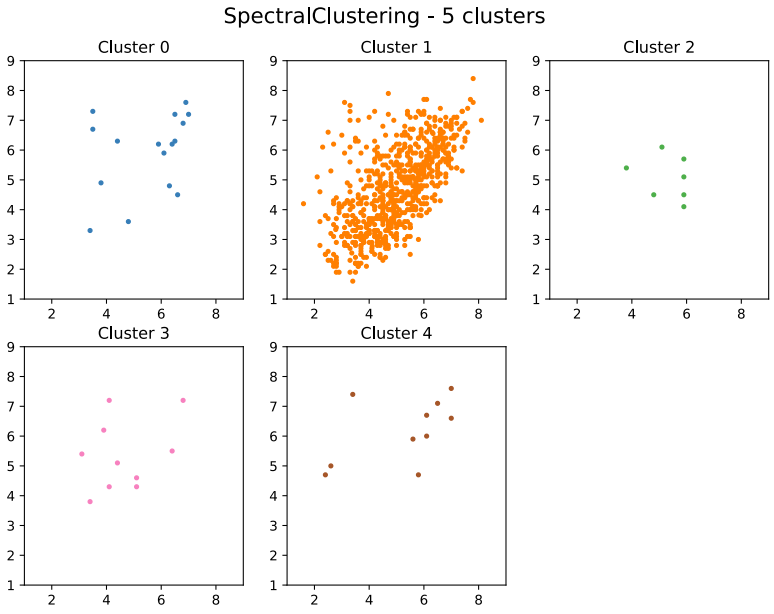
Generated by stats_clustering.py

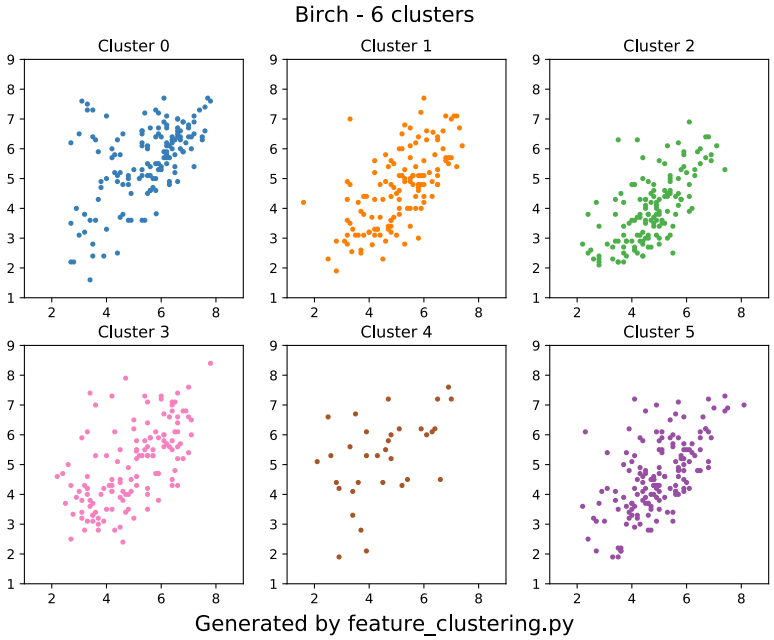


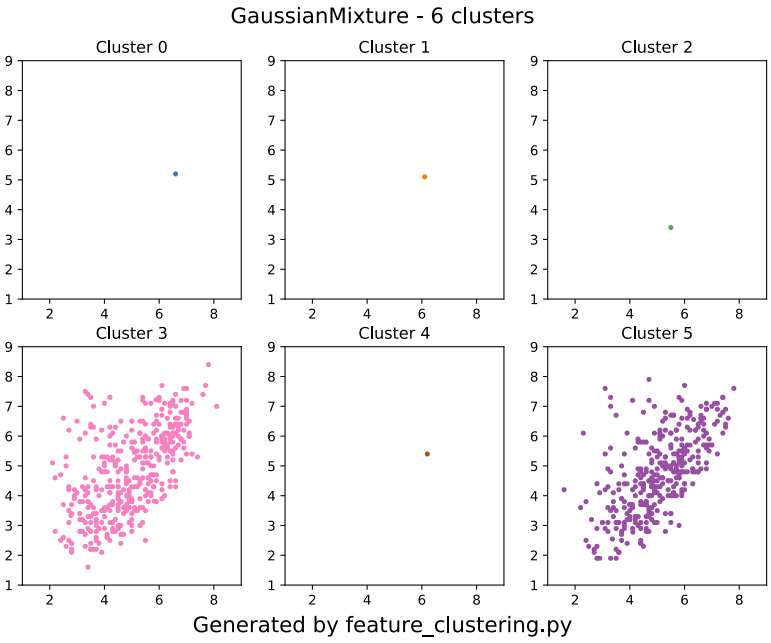
Generated by feature_clustering.py

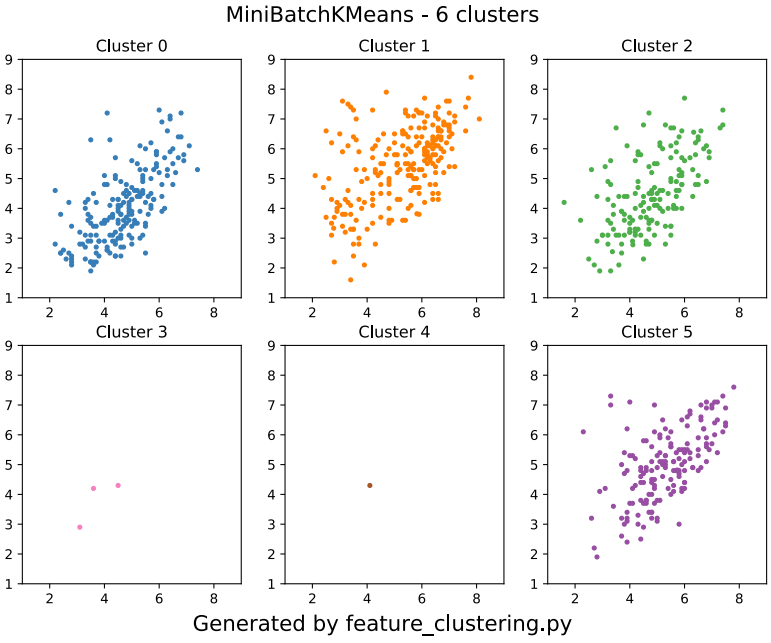


Generated by stats_clustering.py

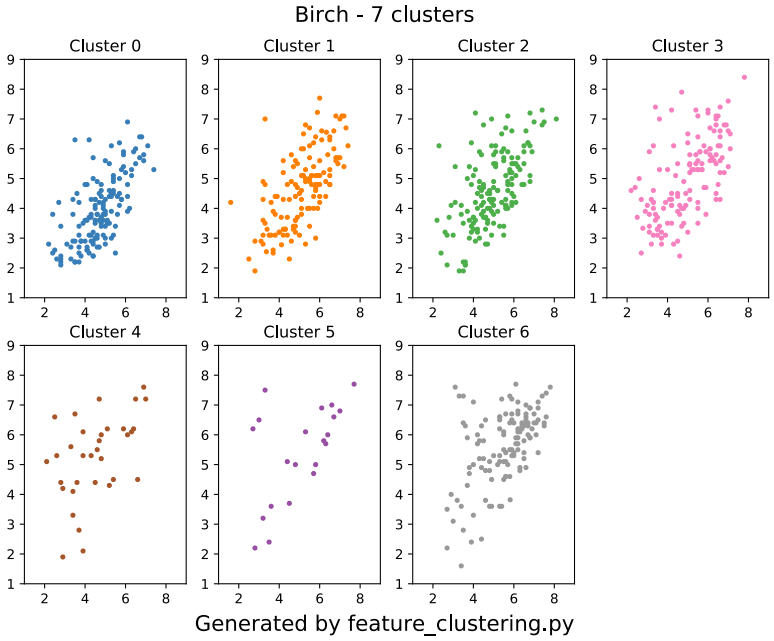

SpectralClustering - 5 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Generated by feature_clustering.py

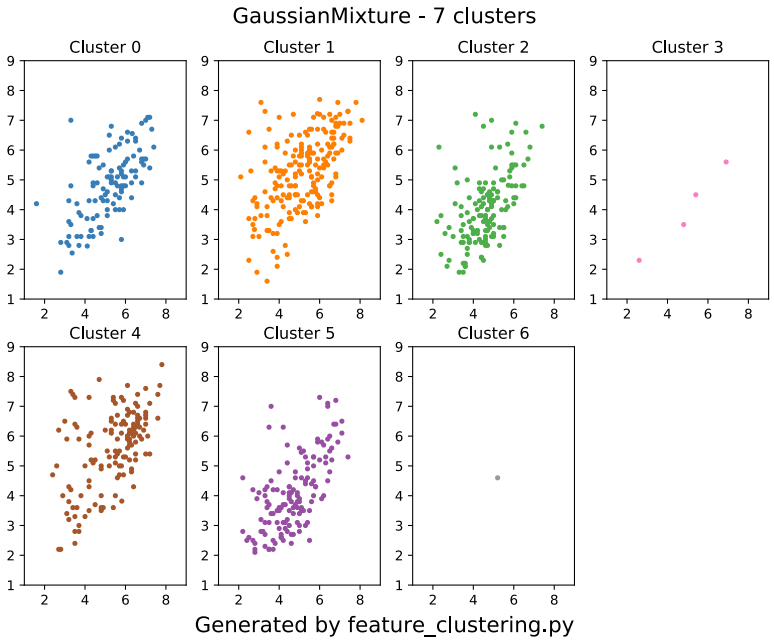

Birch - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

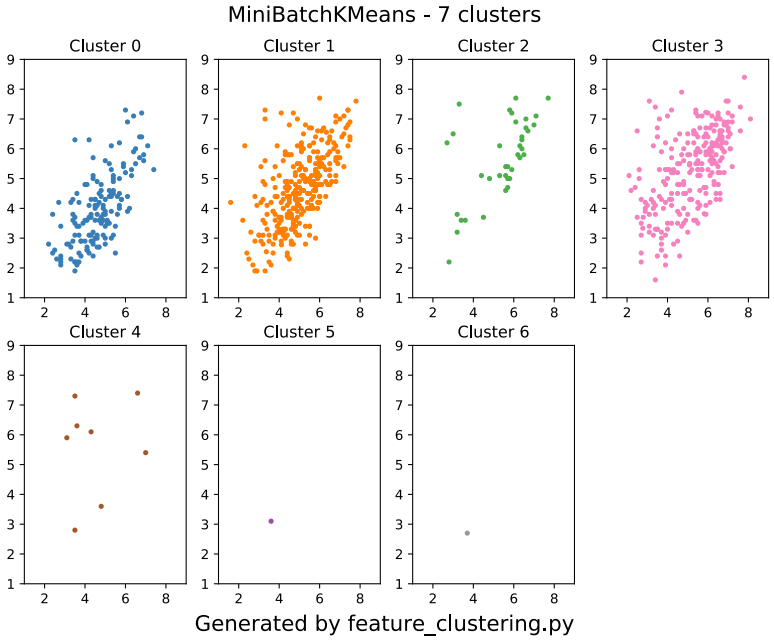

GaussianMixture - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

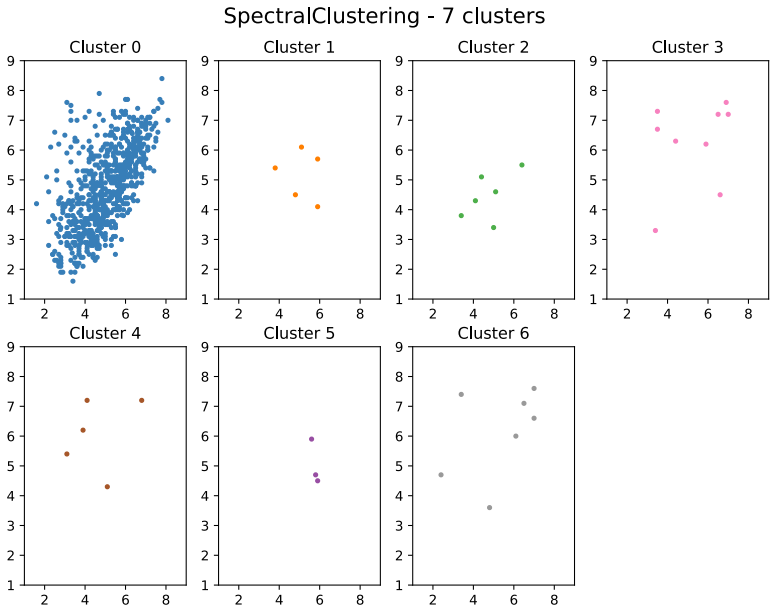

MiniBatchKMeans - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

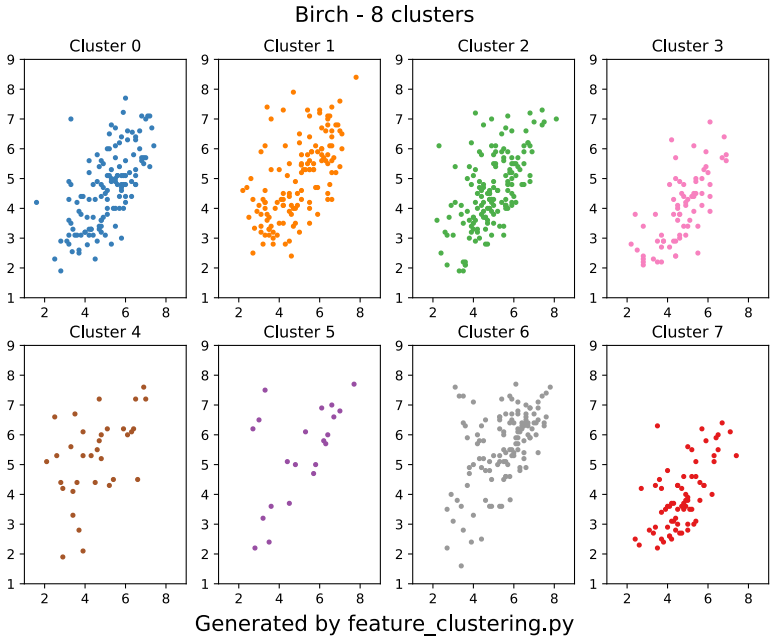

SpectralClustering - 6 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Generated by feature_clustering.py

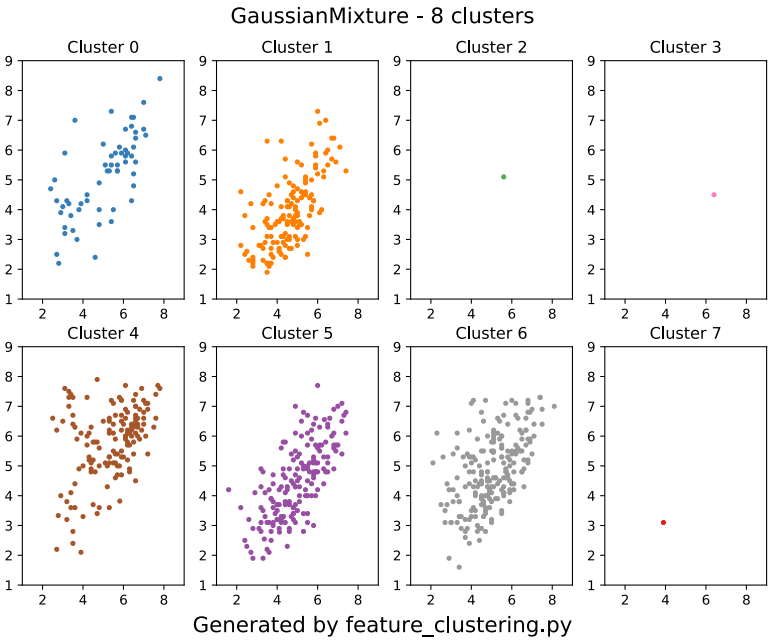

Birch - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py

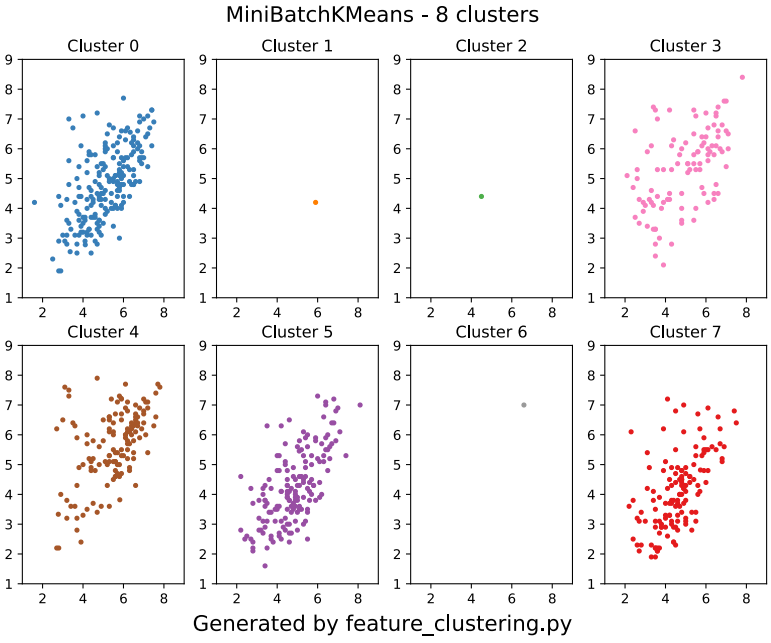

GaussianMixture - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py


MiniBatchKMeans - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py

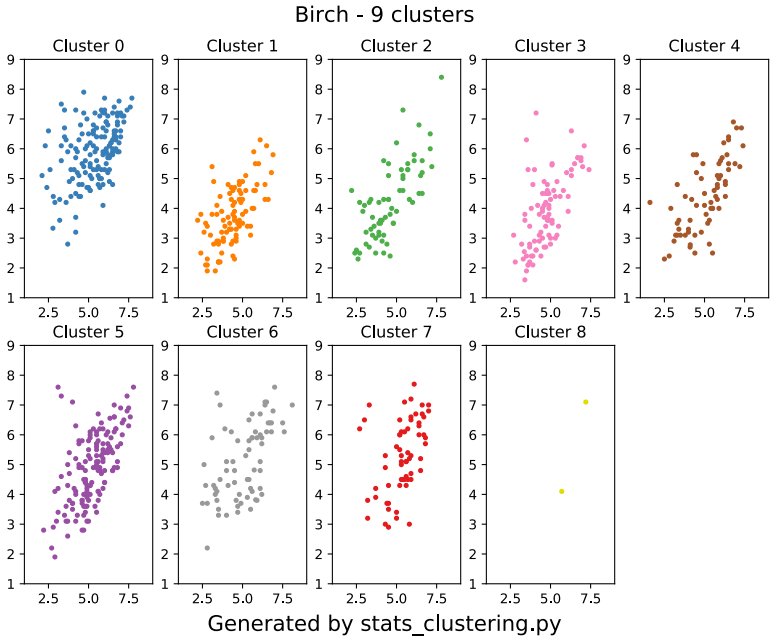

SpectralClustering - 7 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Generated by feature_clustering.py


Birch - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

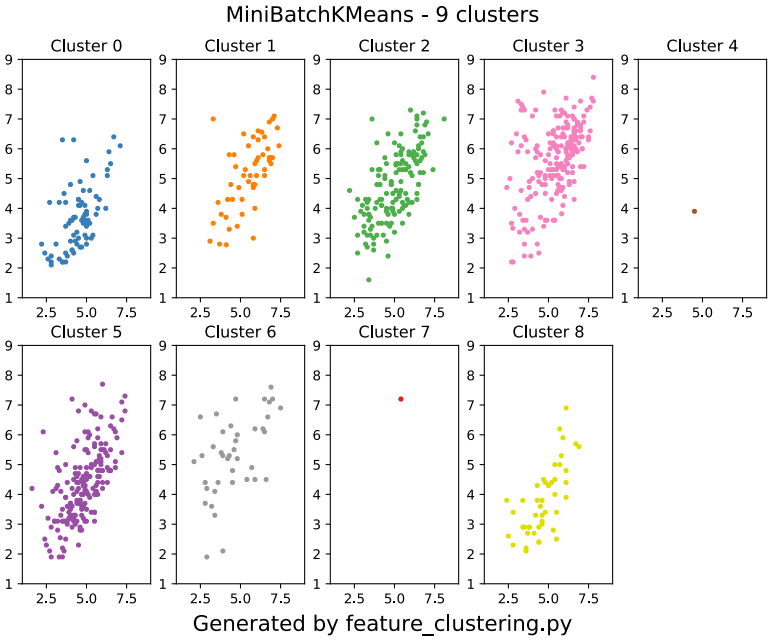

GaussianMixture - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

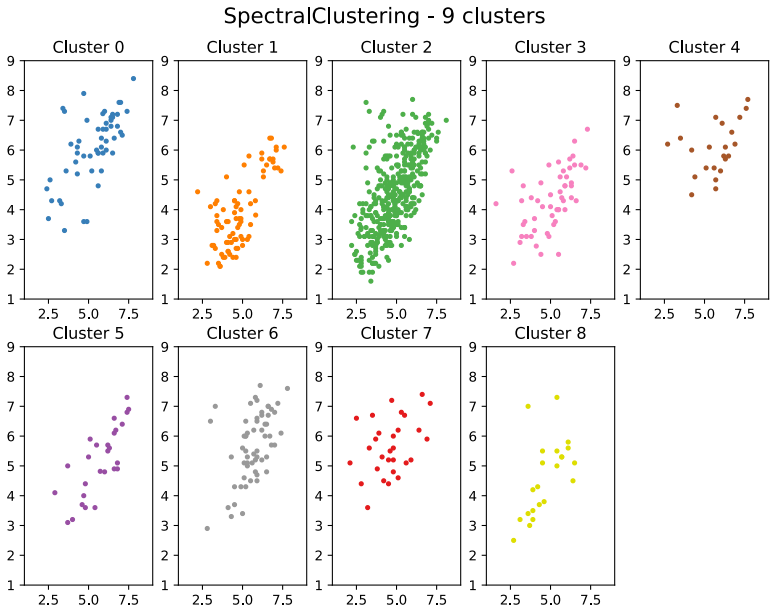

MiniBatchKMeans - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

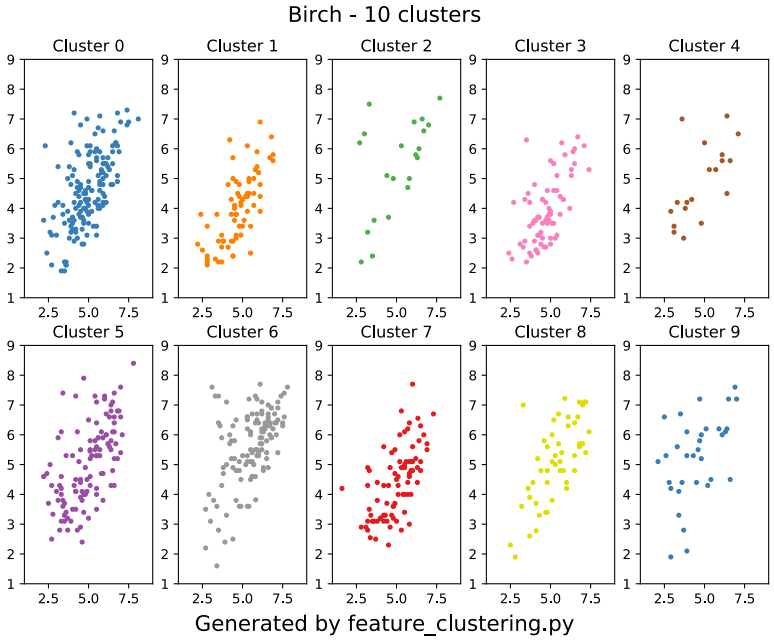

SpectralClustering - 8 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Generated by feature_clustering.py

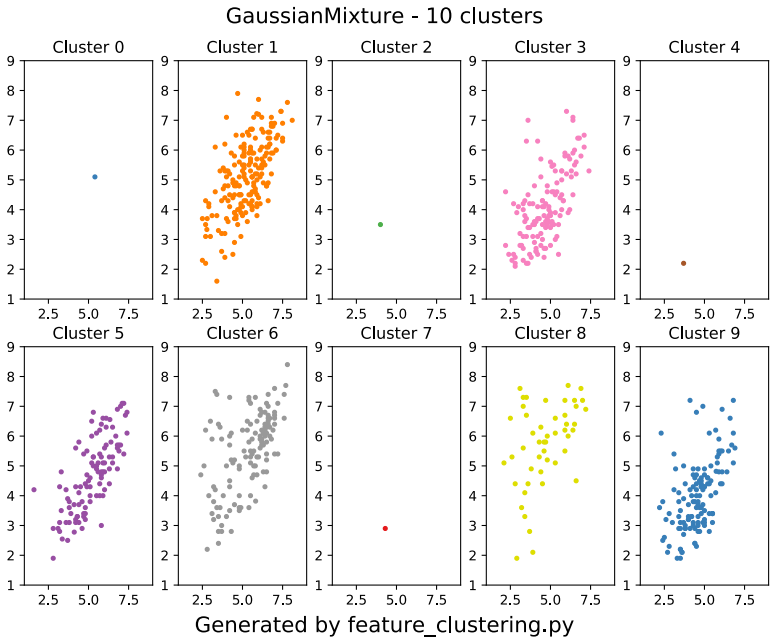

Birch - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by stats_clustering.py

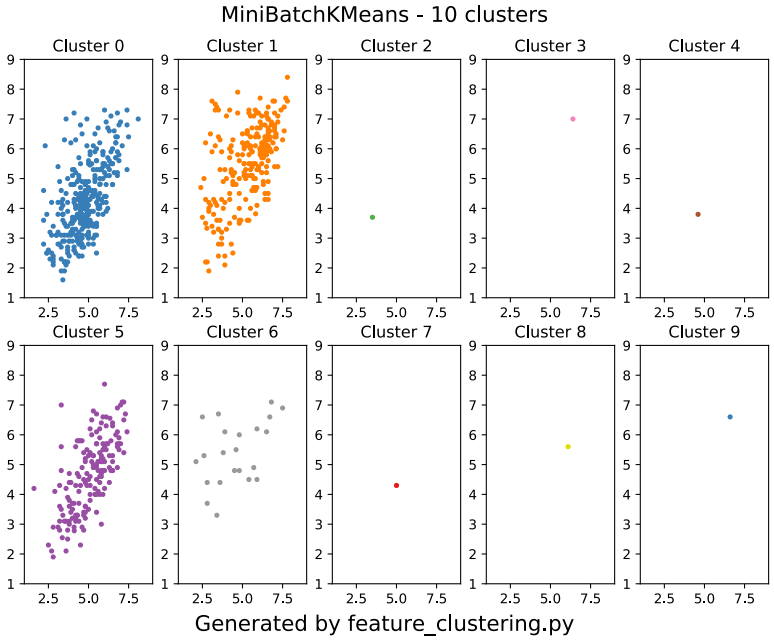

GaussianMixture - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by feature_clustering.py

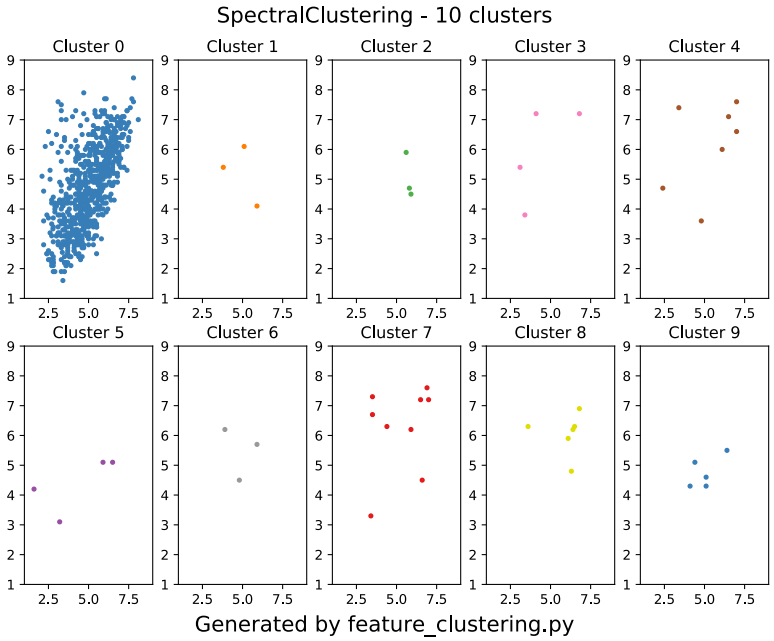

MiniBatchKMeans - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by feature_clustering.py


SpectralClustering - 9 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Generated by stats_clustering.py


Birch - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py


GaussianMixture - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py


MiniBatchKMeans - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py


SpectralClustering - 10 clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Generated by feature_clustering.py