Danmarks Tekniske Universitet

02807 Computational Tools for Data Science - Project

Authors

Jakob Schledermann Winkel - s213555
Frederik Bøttger-Roth - s153277
Nima Jalili - s194825
Julius Olander - s203225

November 27, 2023

Section	s213555	s153277	s194825	s203225
PCA	20%	20%	20%	40%
Data Collection	25%	25%	25%	25%
Clustering	20%	20%	40%	20%
Audio Rec. Sys.	40%	20%	20%	20%
Jaccard Similarity	20%	40%	20%	20%
Result and Conclusion	25%	25%	25%	25%

Contents

1 Introduction **1**

2 Analysis **1**

2.1 Data collection . . . 1

2.2 Selection of genres . . . 1

2.3 PCA . . . 1

2.4 Clustering . . . 3

2.4.1 K-Means . . . 3

2.5 Jaccard Similarity . . . 5

3 Results and Conclusion **6**

4 Appendix **8**

1 Introduction

Recommendation Systems, integral in digital platforms, personalize user experiences by suggesting relevant content and products. Utilizing user data and behavior, they predict preferences, enhancing engagement and satisfaction. These systems are crucial in e-commerce and streaming services, shaping modern consumer habits and driving business success, making them key in tech studies. [2]

This project centers around developing a recommendation system using data sourced from the SpotifyAPI, and the GeniusAPI. The primary objective of this report is to address the challenge of creating an effective recommendation system capable of providing personalized song recommendations based on the content of an input playlist.

2 Analysis

2.1 Data collection

The dataset utilized in this project originates from a collection of playlists obtained through Spotify's API. Each of these playlists comprises the artist's name, album name, and associated genre. Additionally, each individual song is accompanied by a list of audio features, encompassing various audio characteristics [3]. These lists have been combined into one large dataset which contains all the necessary data for the audio component of our recommendation system.

Additionally, to accommodate end-users that are more interested in the lyrical side of music, we are also exploring lyrical data. This involves using the Genius API to fetch lyrics for the same songs. The goal is therefore to develop two distinct recommendation systems based on audio and lyrical information respectively, which can later be integrated into a unified recommendation system. Songs that did not have any lyrics was omitted from the lyrics data set entirely.

2.2 Selection of genres

We have decided to import songs from 11 different genres, each with 100 songs.

The genres are selected somewhat arbitrarily, using our own domain-expertise/experience, with the goal of wanting to cover a wide variety of genres to challenge our model. In example, the genres range from "death metal" and "jazz" and "pop" and "classical". We refer to the notebook for the full list of genres.

2.3 PCA

To reduce the amount of data in the data set Principal Component Analysis (PCA) has been implemented to reduce the dimensionality to a more manageable level. This will prove useful for later clustering of the data, to avoid "the curse of dimensionality". PCA is a dimensionality reduction technique. Its primary purpose is to reduce the dimensionality of a dataset while preserving as much variance or information as possible. PCA accomplishes this by transforming the original features (variables) into a new set of uncorrelated variables, known as principal components. These principal components are linear combinations of the original features and are ordered by their importance in explaining the data's variance. In Figure 1, we have plotted the song data on two components and labeled it by genre. It is clear how the features are captured by even just two components (40% of the variance explained), as we do see some patterns take shape based on the genres.

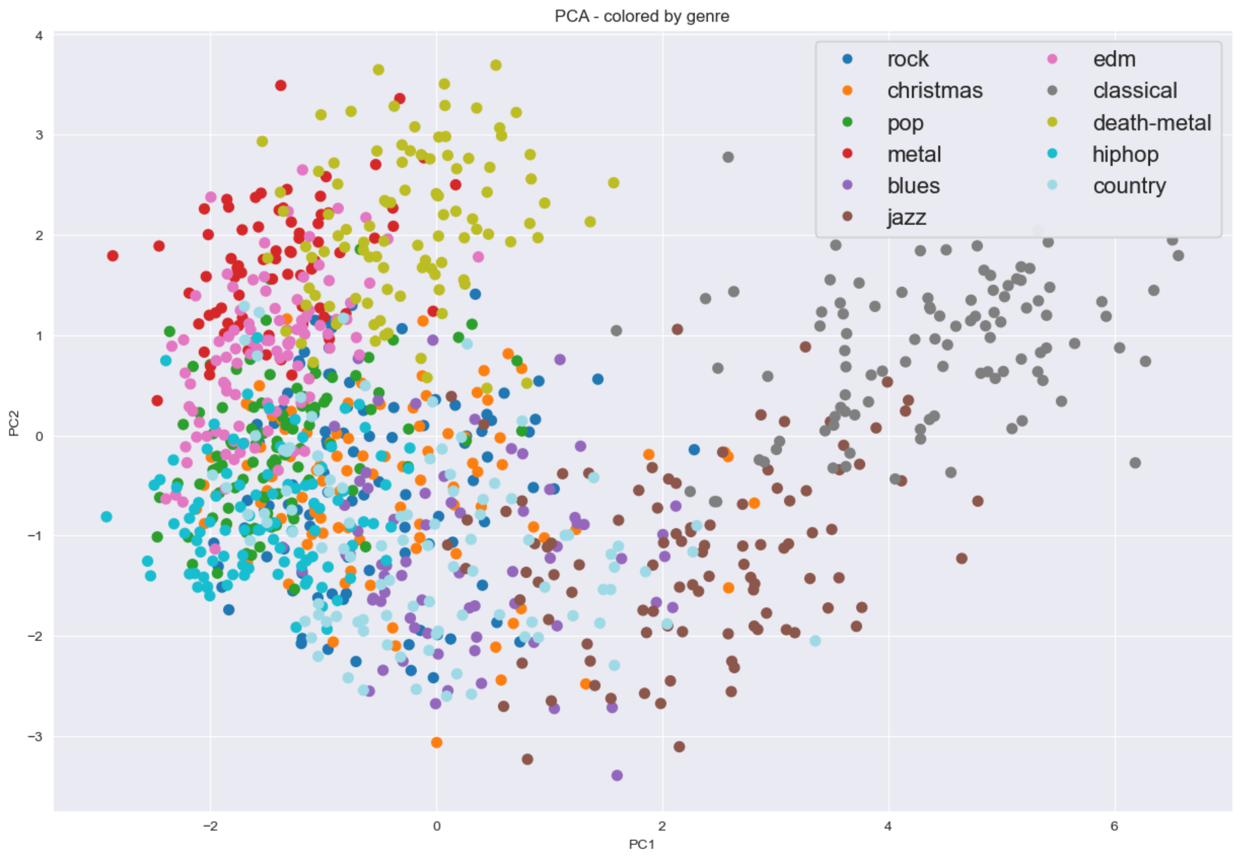

Figure 1: Song data plotted on two principal components and labeled by genre

The PCA showed that two principal components can explain 41.53% of the variance, four components explain 59.32% of the variance, and at around 8 principal components we begin to see the curve taper off, and we get little extra variance explained with additional components. This is visualized in Figure 2.

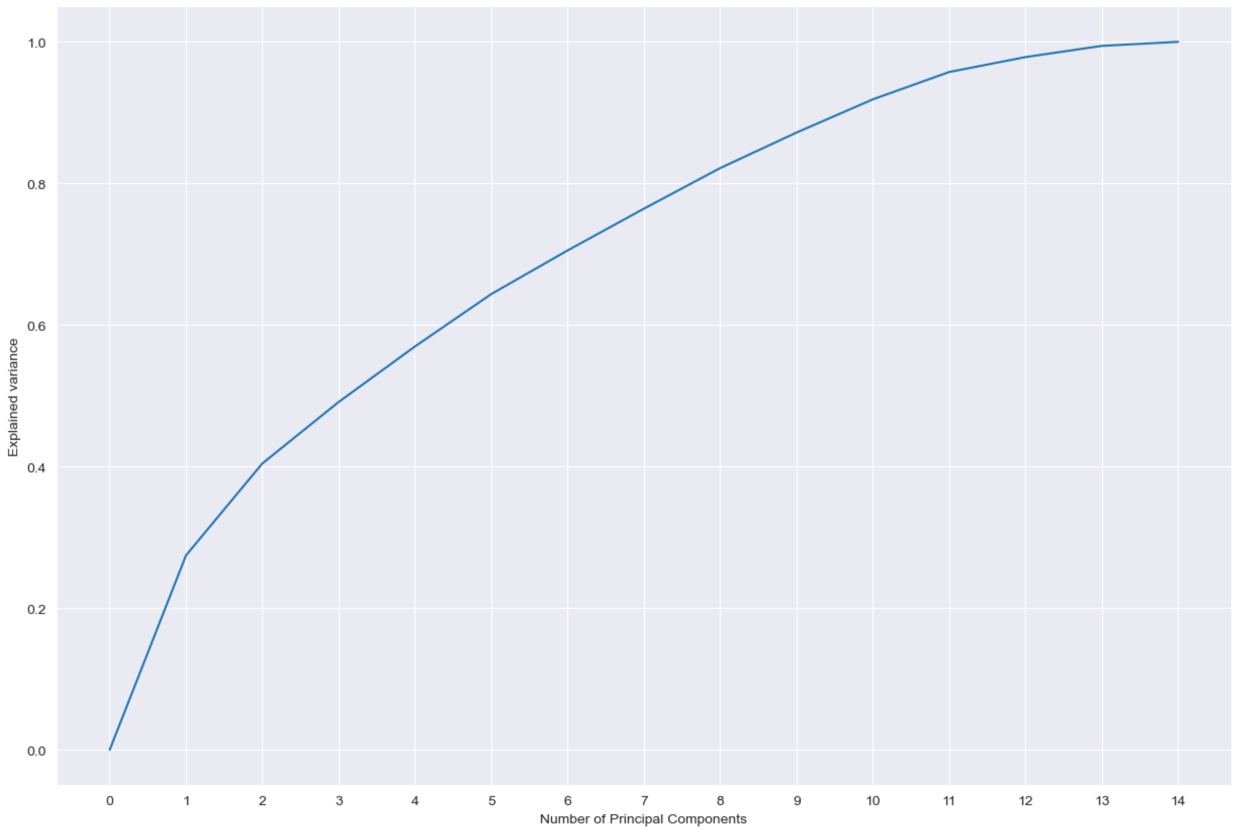

Figure 2: The explained variance by number of Principal Components

2.4 Clustering

When building a recommender system, we have to group songs together based on their similarity of features. To do this, we will use clustering.

2.4.1 K-Means

K-means clustering aims to partition n observations into k clusters, where each observation is located in the cluster with the nearest mean, refered to as the cluster centroid.

It can be problematic clustering using all features due to the "curse of dimensionality", where the performance of the clustering algorithms can be impacted negatively due to the high dimension size of 14 features [1]. As K-means relies on Euclidean distance to form clusters, in high dimensional spaces, the distances can become less meaningful as all points tend to be equidistant from one another. A similar issue will occur with the hierarchical clustering and DB SCAN will also will be negatively affected. Thus, the algorithms ability to define clear clusters is limited.

To accommodate this can cluster using the principal components from before. This way we avoid including too many dimensions, as we have projected all features into lesser dimensions. As we have 14 features, we will cluster on 8 principal components. As we recall from the PCA plot, this was enough to explain 82% of the variance in our data. From conducting research on the amount of components to select, we believe this is sufficient to provide meaningful recommendations, while still avoiding "the curse of dimensionality" [?]. If the solution is scaled to Spotify's entire musical library, using less dimensions will also be less computationally demanding. The plot in Figure 3 provides some additional context as to why 8 principal components were selected.

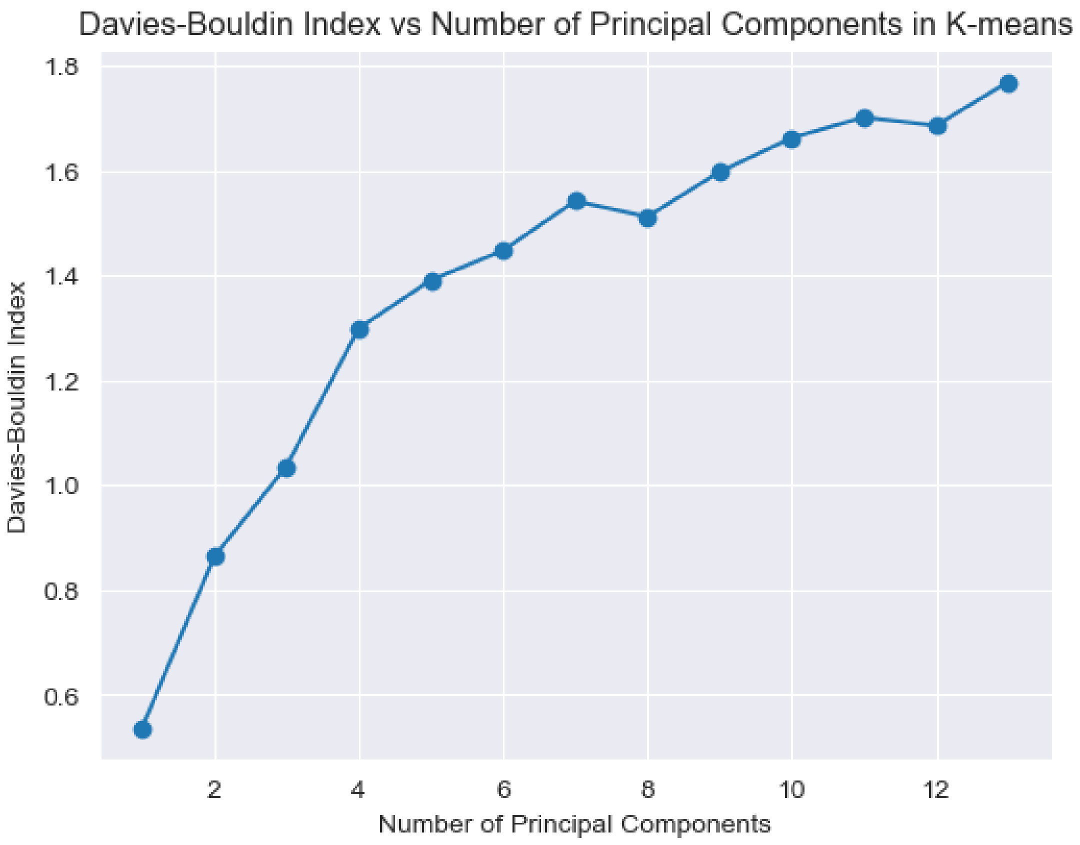

Figure 3: Number of PC's vs David-Bouldin Index

As we can see in the plot, the Davies-Bouldin index increases as we include more components in the K-means algorithm. However, as we look at different dimensions and therefore different configurations of our data, each set of principal components will represent a different view of our data. This will therefore affect the Davies-Bouldin score, as clusters become less compact and clearly separated. Now, it would be "easy" to get a low Davies-Bouldin using only two components, but we would not be explaining much of the variance that additional components provide. Thus, this is where we must select a trade-off. As seen in the plot, the score increases up two 7 components, but then we see a slight dip in Davies-Bouldin at 8 components. This further supports our decision, that clustering on 8 components is a reasonable choice, given the context of our data.

We considered three clustering methods; K-Means, DBSCAN and Hierarchical clustering. To evaluate them, the Davies-Bouldin index was used as an evaluation metric. It evaluates the quality of clusters by measuring the seperation between them. Therefore, a lower Davies-Bouldin index indicates a better clustering as it suggests that the clusters are well separated and distinct from each other. The results showed that K-Means was superior.

We will use K-Means to measure the distance from one song (input) to the nearest other data point (output/recommendation), and thus recommend the song with the smallest Euclidean distance. In Figure 4 the result of the K-Means clustering using 8 principal components is projected onto 2d-space. K-means seem to have identified very similar "clusters" as we saw with the labeled jazz and classical clusters in Figure 1. This is a good indicator that the clustering has managed to divide the data in a way where the features are represented in a meaningful way. This further supports our decision of selecting K-means, rather than by solely judging from the Davies-Bouldin index.

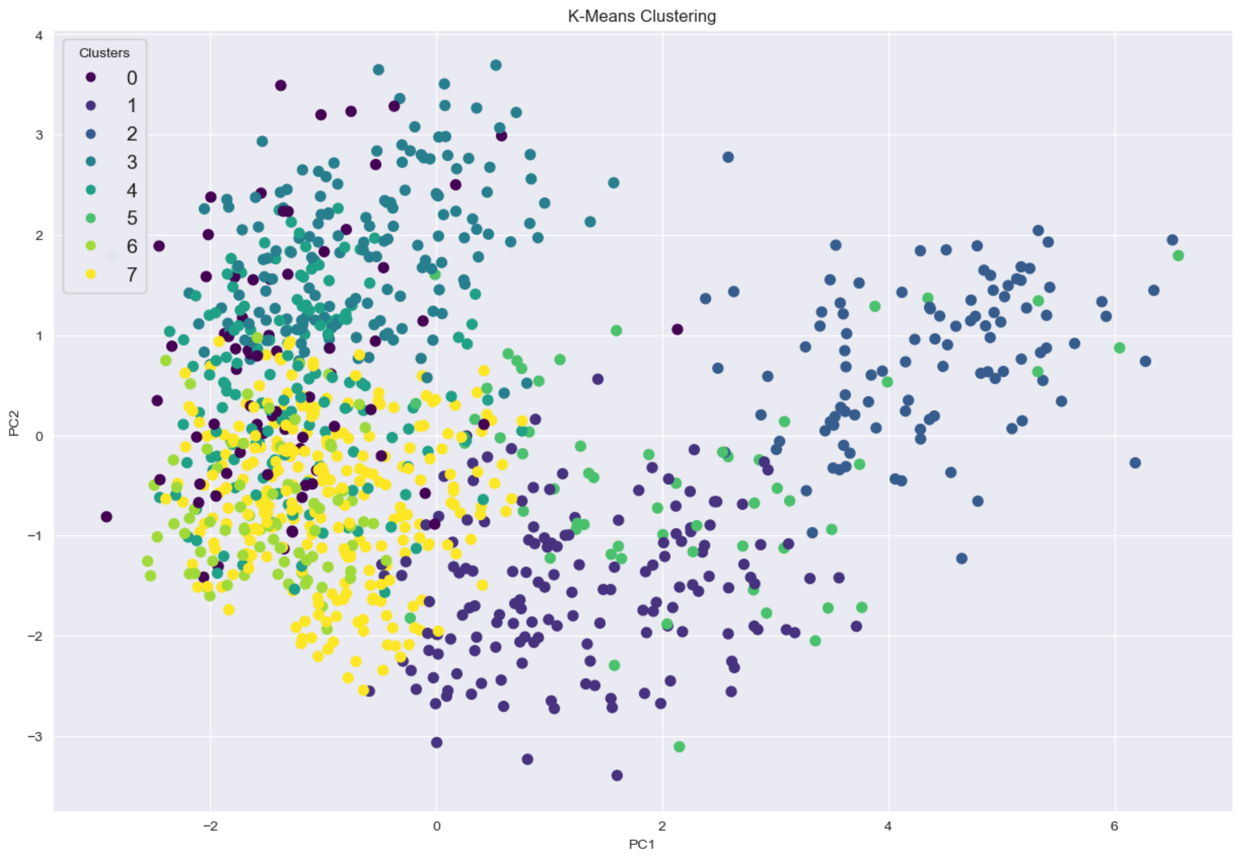

Figure 4: The K-Means clustering projected onto two principal components

2.5 Jaccard Similarity

The second part of our recommendation system utilizes Jaccard Similarity (JS) [4] using shingles and minhashing. While the above recommends based on audio features that each track contains, we use JS to calculate the similarity between two documents, in this case song lyrics. To achieve this, the lyrics are converted into sets of shingles and minhashed to smaller values, called signatures. JS takes these lists of signatures and gets the intersection of both. The amount of intersection between the two lists, i.e. the two lyric texts, correlates to the percentage of similarity between the two lyrics.

Computing the hashes is a resource intensive process. In order to get the smallest hash, hashing must be done multiple times, k times, for each shingle list. Therefor a faster way of computing signatures is implemented, based on 1

Algorithm 1 Computing Signatures - Many At Once

```
1: Input: All Shingles set $U$,
2: Output: Signature matrix, SigMatrix(i,S) where i is the entry of the signature of S
3: Initialization:
4:   Initialize SigMatrix(i,S)
5: for each shingle $\in$ U do
6:     Compute hash(s), k times
7:     for each set S do
8:         if s is contained in S then
9:             for i $\in$ {i..k} do
10:                SigMatrix(i,S) = min{hash(s), SigMAtrix(i,S)}
11:            end for
12:        end if
13:    end for
14: end for
```

The computed signature matrix is saved to a file, in order to avoid minhash computing for every recommendation. Having pre-computed signature matrices reduce the overhead of recommendation dramatically.

The implementation in this project experimented with multiple shingling methods. When it comes to shingling two approaches can be taken, word- or character based. Word based used a q value of 2 and a k value of 20000, while character based used a q value of 5 and a k value of 2000. The value of q is selected based on the use case. Smaller documents require smaller q values. A larger k value gives more nuance between signatures, resulting in more precise comparisons. More precise results is often yielded by character-based shingling. However, this also results in less thematic recommendations. Word based shingling, while giving worse evaluations, results in better song recommendations that fit the theme of the songs. In the end, word based shingling is the better choice due to the importance of thematic similarity, instead of calculated text similarity.

3 Results and Conclusion

The final product of this project is a recommendation system that recommends songs based on both audio features and lyrics. A user inputs a playlist, in this case a sample of our data, and four song recommendations are given in total. Two audio recommendations are made per song in the playlist, by assigning the input songs two a cluster, then measuring the Euclidean distance from each song in the input playlist and selecting the two nearest neighbours. Additionally, two lyrical recommendations are given using the Jaccard method, if the song contains any lyrics. A sample output can be seen in the appendix.

The resulting audio recommendations are mostly deemed satisfactory through qualitative judgement. We generally see similar genres recommended, however different genres recommended frequently as well. This accomplishes the initial goal of urging users to explore different genres.

When it comes to lyrical recommendations, The resulting evaluation scores are generally very low compared to other applications of Jaccard. This might stem from how each artists lyricism differs, and how lyrics often include alternative spellings and explicit written elongating of words. Despite this, the recommendations do often result in similar songs from the same genre as the input.

In addition to this report, we urge you to read the attached Jupyter Notebook that has a lot of interesting additional theory, analysis and results.

4 Appendix

- Recommendations for Song *DarkSide*:
  1. Song: *Welcome To The Jungle*, Distance: 2.207
  2. Song: *Lake Bodom*, Distance: 2.265
- Recommendations for Song *Last Christmas - Remastered*:
  1. Song: *Mack the Knife*, Distance: 1.143
  2. Song: *(What This World Needs Is) A Few More Rednecks*, Distance: 1.167

Figure 5: Audio Features Recommendations

Example output for lyrics

- Recommendations for Song *DarkSide*:
  1. Song: *Breaking the Law*, Eval: 0.023
  2. Song: *I Believe*, Eval: 0.022
- Recommendations for Song *Last Christmas - Remastered*:
  1. Song: *Santa Tell Me*, Eval: 0.0193
  2. Song: *Should I Wait*, Eval: 0.0193

Figure 6: Lyrics Recommendations

Example output for final combined recommendation

- Recommendations for Song *The Summoning*:

By features:

1. Song: *Jaded* by Spiritbox
2. Song: *Nobody* by Avenged Sevenfold

By lyrics:

1. Song: *Paint It, Black* by The Rolling Stones
2. Song: *I Believe* by Jonas Brothers

Figure 7: Combined Recommendations

References

[1] Curse of dimensionality. https://en.wikipedia.org/wiki/Curse_of_dimensionality.

[2] Recommendation System. https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/.

[3] Spotify API Documentation. https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features.

[4] Jure Leskovec, Anand Rajaraman, and Jeff Ullman. *Mining of Massive Datasets*. Cambridge University Press, Cambridge, UK, 2014.