To what extent is cluster similarity affected by feature engineering techniques and to what degree do these techniques affect the performance of a clustered recommender system for Spotify?

Student details
Name: Tessa Roes
Student number: 1280113

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee
Supervisor: A. Hendrickson
Second reader: M. Dias Da Silva-van Riel

Tilburg University
School of Humanities & Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
January 14, 2022

Word count: 8,763

Table of contents

Preface

This study digs deeper into feature engineering techniques applied on a Spotify recommender system. You will probably deal with recommender systems in your daily life. Hopefully, this thesis will give more insight into the complexity of these systems.

These past years at Tilburg University have been an amazing experience. I enjoyed meeting so many new people and I am really happy that I could make friends for life. On top of this, I had the opportunity to study in Texas for five months, which is something I had always wanted to do. Despite the fact that these last 1.5 years have been strange, I am very grateful with the student life I have had.

I would like to thank my supervisor, Andrew Hendrickson, for his time and motivational speeches. You gave me a lot of new ideas and you always helped me out when I was stuck on a problem. Also, I am very thankful for the other students in my thesis group. I feel like we really supported each other throughout the process.

Lastly, I want to thank my family and friends for believing in me. Besides, I would not have spent this many hours in the library if my friends had not been there, so thank you for the constant support!

Abstract

Recommender systems are extensively used to recommend songs to users. In this research, a hybrid two-stage recommender system for Spotify is constructed. The dataset contains users, songs, song features and play counts. To generate recommendations, clusters of similar songs are generated. Within the cluster, similar songs are found. A user's rating for a certain song is predicted by the rating of similar songs. Subsequently, songs with a high prediction value are recommended (Ahuja, Solanki & Nayyar, 2019).

This thesis researches to what extent feature engineering techniques affect cluster similarity, as well as the performance of recommendation systems. To do this, a number of feature engineering techniques are used before clustering. The feature engineering methods that are discussed are feature selection, dimensionality reduction and missing data imputation.

This study concludes that feature selection and dimensionality reduction improve cluster similarity, as well as model performance. In contrast, missing data imputation leads to lower cluster similarity. The effect of data imputation on performance cannot be determined with certainty, since the test sets are dissimilar. However, data imputation is still preferred since it improves the naïve baseline to a greater extent than the degree to which the standard model outperformed its naïve baseline. Further, the method correctly deals with missing data. This results in higher robustness among results.

Keywords: recommender system, content-based, item-based collaborative filtering, hybrid, clustering, feature engineering, feature selection, dimensionality reduction, missing data imputation

Data source/Code/Ethics

The dataset that is used in this thesis consists of two separate datasets, which will be merged. The first dataset concerns a playlist by Ansari (2020), which is obtained from Kaggle. It is stated in the license that the data may be shared and adapted, as long as attributions are made. However, the data may not be used for commercial use.

The second dataset is a subset of the Million Song Dataset (Bertin-Mahieux, Ellis, Whitman & Lamere, 2011), which is obtained from Kaggle (Banerjee, 2018). It is mentioned that the data may be distributed and modified. Additionally, it is allowed to use the data for commercial purposes.

1. Introduction

1.1 Context

Spotify is the largest streaming service in the world (Pérez-Marcos & Batista, 2017). It has over 75 million active users, most of whom are constantly searching for new content. Recommendation systems strongly encourage the discovery of content. A recommendation system (RS) is a method that searches through a large volume of information to provide customers with personalized content (Isinkaye, Folajimi & Ojokoh, 2015). Most recommender systems suggest songs based on user feedback or content features. However, researchers argue that these approaches could be combined by generating clusters based on content (Jin & Han, 2020). Subsequently, recommended songs will be found within the cluster. This approach is expected to decrease the computation time and improve model accuracy (Ahuja et al., 2019).

Feature engineering methods are expected to improve the similarity of songs within clusters (Panda & Misra, 2021). As a result, the accuracy of recommendations is expected to increase. Feature engineering techniques are methods such as feature selection, dimensionality reduction and missing data imputation. The similarity of clusters that are constructed with and without feature engineering techniques will be discussed. Additionally, it will be discussed to what extent feature engineering techniques affect performance.

1.2 Relevance

This research contributes to the literature in multiple ways. From a business perspective, this topic is relevant for Spotify's IT department and shareholders. IT employees could build recommendation systems using the most relevant feature engineering techniques. Feature engineering techniques are expected to increase the quality of recommendations. High-quality recommendations will lead to relevant user content. The company benefits from relevant content, since this will result in higher customer satisfaction and greater user engagement (Rubtsov, Kamenshchikov, Valyaev, Leksin & Ignatov, 2018). Increased customer satisfaction and engagement will lead to customers being loyal to the music platform, which is necessary for customer retention (Millecamp, Htun, Jim & Verbert, 2018). Ultimately, retaining customers will result in more profit, which is the main goal for shareholders (Lozic, Vojcovic & Milkovic, 2020).

From a scientific perspective, this study explores the nature of clusters among feature engineering techniques. This has not been discussed in earlier literature. Further, this research provides an evaluation of the effect of cluster similarity on the performance of a recommendation system, which is also lacking in the literature. In this study, performance is measured by the accuracy of the results. Results on cluster similarity and model performance could be used for future research.

1.3 Research questions

In this thesis, the following research question is answered: *To what extent is cluster similarity affected by feature engineering techniques and to what degree do these techniques affect the performance of a clustered recommender system for Spotify?* The research question is answered by a two-stage analysis. First, song clusters are generated based on song features. The following sub-questions follow from the first stage of the research question:

1) *To what extent does feature selection affect cluster similarity?* With feature selection, a subset of most informative features is selected.

2) *To what extent does dimensionality reduction affect cluster similarity?* Dimensionality reduction transforms features into a lower number of new features, while aiming to retain predictive power.

3) *To what extent does missing data imputation affect cluster similarity?* A missing data imputation model will be compared to the standard model, in which missing data is discarded. Secondly, the performance of the recommender system will be evaluated within each cluster. The second stage of the research question leads to the following sub-questions:

4) *To what extent does feature selection affect the performance of the recommender system?*

5) *To what extent does dimensionality reduction affect the performance of the recommender system?*

6) *To what extent does missing data imputation affect the performance of the recommender system?*

1.4 Findings

This study concludes that feature selection and dimensionality reduction improve cluster similarity. Further, these techniques result in higher accuracy. In contrast, the imputed model leads to aggravated similarity. This may be due to the fact that the model was unable to impute plausible values. Further, the imputation model seems to perform better than the standard model. However, this cannot be concluded with certainty, since these models operate with different datasets.

1.5 Structure

This thesis is structured in the following manner. In the second chapter, related work is discussed. The methodology and experimental setup are thoroughly explained in chapter three. This chapter also includes a description of the hyperparameter tuning process, which means that the optimal settings for model parameters are found. Hyperparameter values and model results are demonstrated in chapter four. Chapter five provides context on the results. Lastly, chapter six presents a general conclusion.

2. Background

2.1 Recommender systems

In the past decades, many individuals and companies have attempted to improve recommender systems. Generally speaking, a recommender system (RS) is a system that suggests relevant items to users (Zhou, Xu, Li, Josang & Cox, 2011). A recommender system is considered as a tool that could be used for dealing with large amounts of items (Zhou et al., 2011). These items are the products that are recommended, such as songs or movies. Three recommender methods could be utilized; the traditional content-based method, a collaborative filtering method or a hybrid method, which combines the latter (Aggarwal, 2016).

A content-based (CB) recommender system recommends items based on music that is similar to songs that the user has already listened to (Aggarwal, 2016). Thus, songs are solely recommended based on their content, instead of their popularity (Chemeque-Rabel, 2020). However, a so-called 'cold start' problem occurs, because recommendations are only be reliable when sufficient information on preferences is available. Further, many possibly interesting items are not revealed, since the method does not tend towards community (Thi Do, Nguyen & Van Nguyen, 2010).

The collaborative filtering-based (CF) approach does take the community into consideration. CF can be item-based or user-based (Bhatnagar, 2017; Wei, Ye, Zhang, Huang & Zhu, 2012; Yadav, Shukla, Tripathi & Maurya, 2021). The item-based approach predicts the rating per user per song by the average rating of similar songs. This approach is generally preferred, due to its ability to quickly react to changes in ratings (Kużelewska, 2020). In contrast, the user-based approach determines the predicted rating per user per song by calculating the average rating of similar users (Xie et al., 2012). An important drawback of the CF approach is the 'scalability problem', which implies that the entire database must be searched to compute similarities among users. Hence, computation time increases linearly when the size of the dataset increases (Sánchez-Moreno, Gil González, Muñoz Vicente, López Batista & Moreno García, 2016).

A hybrid method combines the CB and CF approach. As a result, the disadvantages of individual methods are diminished. A commonly used hybrid model clusters songs based on content and generates recommendations based on collaborative filtering (Li et al., 2017). This model improves computation time as well as accuracy, since the community is regarded. However, hybrid models are complex, which may diminish interpretability (Thuan & Puntheeranurak, 2014).

Researchers agree that every method has advantages and disadvantages. CB approaches suffer from a cold-start problem. Also, the method does not regard preferences of the community. CF models imply scalability problems. The hybrid method is generally preferred since it combines the best of both approaches.

2.2 Reoccurring issues

A major issue in the field of recommender systems is the scalability problem. Recommender systems generally suffer from highly dimensional data, which slows down the recommendation process. According to Jin and Han (2020), this problem could be solved by creating clusters of similar items. Similar songs will then be found within their cluster. This method reduces the running cost of high dimensional data. Clustering also improves generalization, which subsequently leads to higher accuracy (Ahuja et al., 2019; Kim, Kim, Park, Lee & Lee, 2007; Liao & Lee, 2016; Sarwar, Karypis, Konstan, & Reidl, 2001). In contrast, other researchers argue that clustering decreases the quality of recommendations, since their nearest neighbors may not be in the same cluster (Aggarwall, 2016; Kużelewska, 2020). Thus, some predictive power may be lost. Several researchers solves this issue by using a two-stage hybrid recommender system with fuzzy clusters (Puntheeranurak & Tsuji, 2007). Fuzzy clustering implies that each item belongs for a certain extent to each cluster. Therefore, the problem argued by Aggarwall (2016) and Kużelewska (2020) is no longer valid.

Another issue concerns the implicit feedback problem. To determine similarity among users or items, explicit user-feedback is required (Sánchez-Moreno et al., 2016). However, when information on actual preferences is lacking, preferences must be estimated (Najafabadi, Mahrin, Chuprat & Sarkan, 2017; Pacula, 2010). Music recommender systems often use the play count of songs, which indicates how many times a person listened to a song (Lee & Lee, 2015). Transforming the variable is helpful since play counts are difficult to compare. Namely, a larger nominal play count does not necessarily imply a larger preference (Hu, Koren & Volinsky, 2008). For instance, *a* might listen to his playlist on repeat, while *b* listens to music occasionally. Additionally, play counts are positively skewed, since most songs are infrequently played. This could affect the quality of recommendations (Pacula, 2010). The representativeness of preferences could be solved by converting implicit ratings to relative play counts. The relative play count indicates how much a person likes a specific song, compared to other songs. This allows for comparing preferences among different people (Jawaheer, Szomszor & Kostkova, 2010). Skewness could be diminished by converting an implicit rating to a binary variable. In this case, 1 symbolizes a positive score (liking the song) and 0 represents a negative score (not liking the song). A disadvantage of converting implicit feedback to two classes is that information the amount of appreciation is lost (Pacula, 2010).

2.3 Feature engineering

Clustering songs based on song features improves efficiency and accuracy. However, time and generalization issues could be further reduced by applying feature engineering methods on data before clustering. Two main feature engineering methods are feature selection and dimensionality reduction (Panda & Misra, 2021). Researchers argue that feature engineering techniques decrease

computation costs and improve the quality of recommendations (Ahuja et al., 2019). However, research on the effect of feature engineering techniques on cluster similarity, as well as the accuracy of clustered recommender systems, is lacking. A special feature engineering technique is missing data imputation. This method is addressed separately, since applying data imputation techniques on all models could be too computationally expensive.

Feature selection uses a subset of most informative features. This is expected to improve model performance. (Ndung'u, Kamau & Mariga, 2021). The method is used because several features are usually correlated, meaning that it is inefficient to include all features in the analysis (Ndung'u et al., 2021). Further, the method reduces noise in the data. This is expected to improve generalization, which will improve performance. The most widely used feature selection methods are the filter and the wrapper strategy (Panda & Misra, 2021). In the filter method, features are ranked by importance based on a chosen statistic (Afoudi, Lazaar & Al Achhab, 2019). In contrast, the wrapper strategy measures all possible combinations of features and returns the optimal set of values (Panda & Misra, 2021). The wrapper strategy is more accurate than the filter strategy (Ndung'u et al., 2021). However, the wrapper method is more computationally expensive, since it regards dependencies between features (Panda & Misra, 2021). The choice of method therefore depends on whether the possibly higher accuracy is actually worth its higher computational cost. Researchers have not yet determined the best feature selection strategy for clustered recommender systems.

Dimensionality reduction aims to decrease the number of features in the data, without losing predictive power (Lü et al., 2012). This reduces the computational cost of the model (Panda & Misra, 2021). Further, the model is expected to improve generalization. Dimensionality reduction methods can be linear or nonlinear. Linear methods are easier to use and easier to interpret than nonlinear methods. Also, they are less computationally expensive. Nguyen and Holmes (2019) argue that nonlinear methods generally perform better. In contrast, Van der Maaten, Postma and Van den Heerik (2009) argue that nonlinear methods are often not capable of outperforming linear methods. Examples of linear dimensionality reduction techniques are Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). PCA is generally preferred due to its simplicity and its ability to create new and uncorrelated variables (Langensiepen, Cripps & Cant, 2018; Van der Maaten et al., 2009). Furthermore, the method performs well with continuous data (Nguyen & Holmes, 2019). In contrast, commonly used nonlinear dimensionality reduction techniques are Isomaps and Kernel PCA (KPCA) (Nguyen & Holmes, 2019).

Datasets generally include missing data (Panda & Misra, 2021), which can be solved by data imputation methods. Missing data imputation is important because it ensures that all observations are preserved (Panda & Misra, 2021). This leads to better generalization. A large disadvantage of missing data imputation is that imputed datasets are large, which results in computationally expensive models

(Panda & Misra, 2021). To address this issue, missing data could be handled by deleting all incomplete observations. This approach is called listwise deletion. Since less observations complicate generalization, the predictive power of the model is expected to decreases. Researchers argue that listwise deletion should only be used when data is missing at random (MAR), which means that all data points are equally likely to be absent. In contrast, listwise deletion may lead to selection bias when data is missing not at random (MNAR), which means that a specific situation causes data to be missing. MNAR data must be treated with caution, since no method can fully account for this type of missingness. Nevertheless, it is argued by James, Witten, Hastie and Tibshirani (2021) that multiple imputation is preferred. Multiple imputation imputes the average of a number of plausible values. This method is expected to lead to the most robust results. However, listwise deletion may still be beneficial for large MNAR datasets, since listwise deletion largely decreases the computation cost (Panda & Misra, 2021). It is relevant to detect to what extent data multiple imputation and listwise deletion affect performance. Namely, if the robustness and accuracy of the results is not compromised, the listwise deletion method is preferred.

3. Methodology and experimental setup

This chapter contains a thorough explanation of the methodology that is used for this study. Section 3.1 elaborates on the choice for a two-stage hybrid model. Section 3.2 describes which dataset this study uses. In section 3.3, pipelines for the standard model and feature engineering models are demonstrated. Section 3.4 discusses the process of hyperparameter tuning for each model separately. Algorithms and packages are discussed in section 3.5.

3.1 Hybrid recommender system

Researchers argue that content-based (CB), collaborative filtering-based (CF) and hybrid recommender systems have both advantages and disadvantages. Nevertheless, hybrid systems are generally preferred, because of their ability to combine content features with ratings. In this study, a two-stage hybrid model is constructed. First, songs are clustered based on their content. Therefore, clusters contain songs with similar song features. Subsequently, a CF item-based approach is used within each cluster. The model recommends songs that are similar to other songs which the user rated positively (Cintia Ganesha Putri, Leu & Seda, 2020). This standard recommender system is compared to models that use feature engineering methods before clustering songs. In this manner, it can be evaluated to what extent feature engineering techniques affect cluster similarity and model performance. The feature engineering techniques that are used in this study are feature selection, dimensionality reduction and missing data imputation. Normally, missing data imputation is applied on all models, since this is expected to increase the robustness of the results. However, the imputed dataset increases computation time to an extent that it would be impossible for a private computer to process the data in a feasible amount of time. For this reason, missing data imputation is performed as a separate model. It is concluded whether missing data imputation is necessary for the robustness of results, as well as retaining accuracy.

3.2 Dataset description

To answer the research question, two datasets are combined. The first dataset is named the ‘Spotify Audio Features Hit Predictor Dataset’, which is obtained from Kaggle (Ansari, 2020). The total set contains 35,860 unique song titles. Ansari (2020) has combined song titles, which originate from his own playlist, with audio features, which are extracted from Spotify’s Web API. This so-called ‘song dataset’ consists of 35,860 rows and 19 columns. Each row indicates one song and each column contains information about that song. The second dataset is a subset of data that was published for the Spotify Recommender Challenge (Banerjee, 2018; Bertin-Mahieux et al., 2011). Each row contains a user ID, a song ID and their corresponding title, album, artist, play count and release year.

This so-called ‘user dataset’ contains 2,086,946 rows and 7 columns. The user dataset includes 76,353 unique users and 10,000 unique songs. The data in the user dataset is combined with song features originating from the song dataset. This approach is chosen because implicit feedback is only available for songs in the user dataset. This feedback is necessary for generating item-based recommendations.

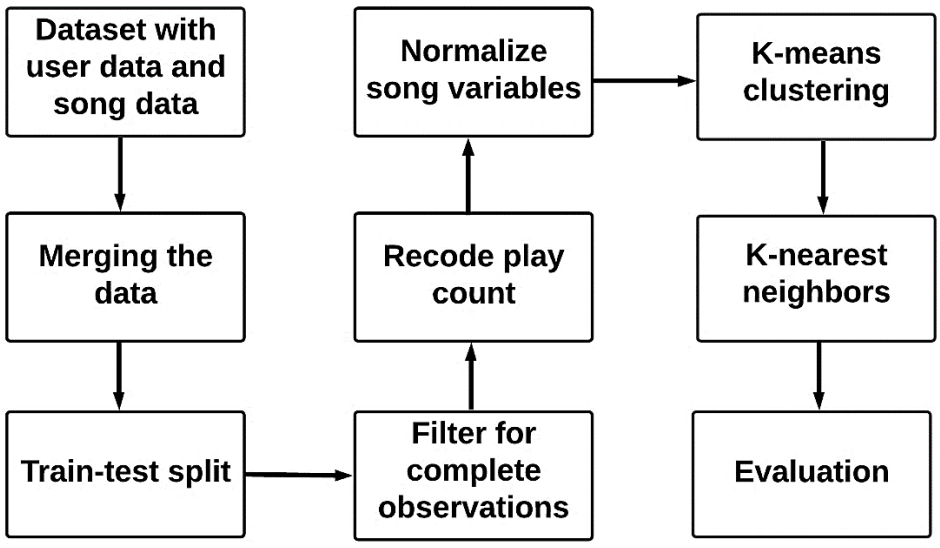
3.3 Pipelines

The pipelines for feature engineering models slightly differ from the pipeline of the standard model. Therefore, model pipelines are discussed separately. Section 3.3.1 thoroughly explains the pipeline for the standard model. Section 3.3.2 describes the methodology for the feature selection model. The dimensionality reduction model is described in section 3.3.3. Finally, section 3.3.4 compares the methodology for the data imputation model.

3.3.1 Standard model

Figure 1 displays the pipeline for the standard model. This model performs as a baseline, since feature engineering models are compared to this model. However, the standard model is no naïve model which simply predicts the majority class. The model is constructed in such manner that both item features and ratings are used to predict whether or not a person will like a song.

Figure 1 – Pipeline standard model



The process of building a recommender system starts with two individual datasets. The user dataset is combined with data from the song dataset, because the merged dataset allows for combining a CB and CF approach. After merging the data, a train-test-split is created. For this purpose, data from 25 per cent of the users is exported as test set. These users are randomly selected. The remaining of the data serves as train set. Subsequently, a train-validation-split is performed on the train set.

Therefore, data from 33 per cent of the users is exported as validation set. Thus, the complete set of data consists of a train set, a validation set and a test set. The validation set and the train set are also exported. Further modifications on the data, such as selecting features for feature selection, are executed on each set for each model separately. The train set is used for training the model. Further, the performance on the train set is compared to performance on the test set, which allows for checking the validity of the results. Namely, if the performance on the train set is substantially higher than the performance on the test set, it is likely that the model is overfitted. In contrast, a higher test performance implies selection bias. The validation set is used for model selection and hyperparameter tuning. Furthermore, the test set is only used to compare results of different models. Using splits allows for obtaining unbiased results.

The song dataset and the user dataset do not contain any missing values. However, missing data on song features emerge when songs in the user dataset are not in the song dataset. One way to deal with this is by listwise deletion, which implies that all observations that contain missing values are deleted. The dataset that remains is called the 'complete cases dataset'. Although the merged dataset contains 10,000 unique songs and 76,353 unique users, listwise deletion leads to a combined dataset of merely 2,449 unique songs and 71,196 unique users. For the standard model, the dataset with listwise deletion is used. This approach is chosen because using the imputed dataset for each model is too computationally expensive to perform in a reasonable amount of time. However, it must be noted that missing data in the complete cases dataset is missing not at random (MNAR). Namely, data is missing when songs are not in the song dataset. Thus, preferences of Ansari (2020) cause the data to be missing.

After filtering for complete observations, the numeric variable *play_count* is recoded into a relative feedback variable (*rel_rating*). Recoding is performed because nominal ratings fail to represent preferences. Jawaheer et al. (2010) argued that relative feedback allows for comparing ratings among people. The column names, descriptions and feature types of all variables can be found in figure 13 in appendix I (Ansari, 2020).

Additionally, variables that describe characteristics of songs are normalized from 0 to 1. This is necessary because clustering song features requires these features to have equal distributions. Most variables already range from 0 to 1, however, the variables *sections*, *key*, *loudness*, *duration_ms* and *tempo* work with different ranges. Figure 14 in appendix I illustrates the distribution of song features after normalizing.

The first step of the hybrid analysis is to cluster songs by the K-means clustering algorithm. Clustering is performed since this reduces the computation cost of the analysis. Namely, only part of the dataset needs to be searched (Ahuja et al., 2019). Additionally, the quality of recommendations is expected to improve, since clustering simplifies generalization (Kim et al., 2007). The K-means

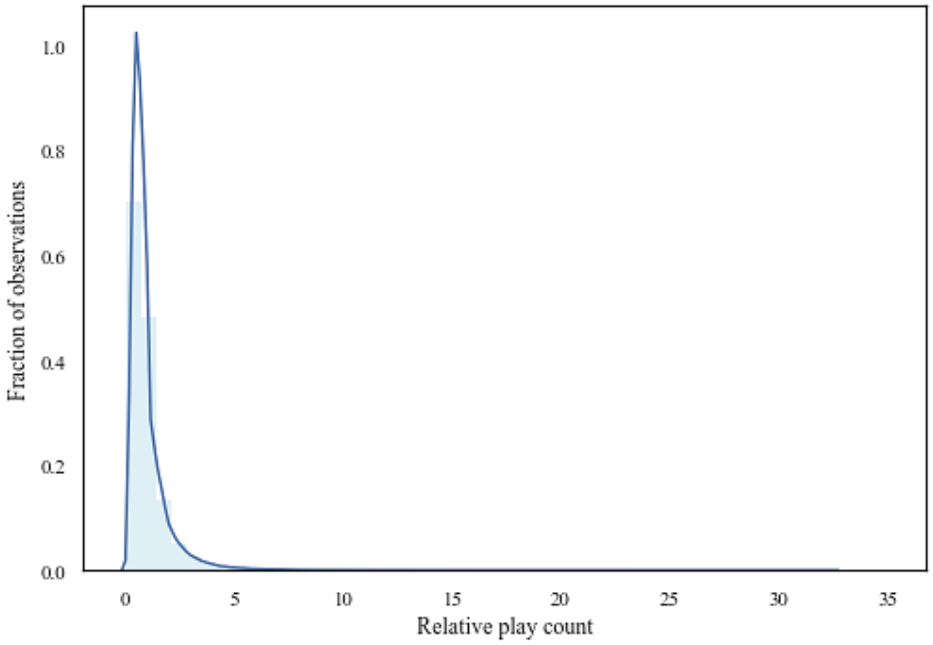
clustering algorithm is used because it is an efficient and interpretable clustering method, which performs well with large datasets (Ju & Xu, 2013). Train data is used to group songs by similarities across song description features (Sánchez-Moreno et al., 2016). The K-means clustering algorithm selects *k* random centroids. Subsequently, the remaining data points are assigned to the cluster that is associated with the closest centroid. Cosine similarities are computed to determine which centroid is closest. This metric is chosen because cosine similarities generally lead to the highest accuracy, while also being efficient and simple (Sánchez-Moreno et al., 2016). The cosine similarity can be computed by the following equation, where x is the play count of song 1 and y is the play count of song 2 (Jin & Han, 2020):

$$\cos(x, y) = \frac{\Sigma(x \cdot y)}{\Sigma|x| \cdot \Sigma|y|} \quad (1)$$

After assigning new data points, the average data point of the cluster becomes the new centroid. This process is repeated until clusters no longer change (Ju & Xu, 2013). Test and validation data is assigned to the closest train data centroid. In this manner, the test and validation set are not used for training the model. Another possibility would be to split the dataset into train, validation and test data after clustering all data. However, test and validation data would then be used for training, which would diminish the purpose of splitting the data.

The second step of the hybrid analysis is to generate a prediction of *rel_rating* per song per user. The model uses the K-nearest neighbors algorithm to find *k* similar songs within each cluster. The cosine similarity determines what songs are similar, since this metric is efficient, simple and is expected to lead to the highest accuracy (Sánchez-Moreno et al., 2016). However, figure 2 indicates that relative ratings are very skewed.

Figure 2 – Distribution of rel_rating



Pacula (2010) argued that skewness could be solved by converting the continuous prediction value to a binary variable. This will increase the accuracy of recommendations. Thus, the continuous outcome of the predicted *rel_rating* is converted to 0 if it is expected that the person will not like the song, or 1 if the person is expected to like the song. The continuous value 1.0 is chosen as boundary, meaning that someone is expected to like the song when *rel_rating* is 1.0 or higher. This boundary is chosen because a *rel_rating* of 1.0 indicates the average preference. It is intuitive that a *rel_rating* of 1.0 or higher means that someone likes the song.

The goodness of clusters is evaluated by cluster similarity. To evaluate cluster similarity, the Davies Bouldin (DB) score is calculated. DB measures cluster compactness and divides this value by a value representing the separation among clusters. A lower score indicates low dispersion within clusters and a high distance between clusters (Baarsch & Celebi, 2012). Thus, a low score indicates high cluster similarity. The DB index is calculated by formula 2 (Davies & Bouldin, 1979):

$$DB = \left\{\frac{1}{T_i}\sum_{j=1}^{T_i}|X_j - A_j|^2\right\}^{\frac{1}{2}} \quad (2)$$

In this formula, i indicates the cluster that is regarded, T_i represents the number of observations in cluster i, the jth observation in cluster i is denoted by X_j and A_j represents the centroid of cluster i. This similarity score is determined for the test set, which allows for comparing cluster similarity among different models. As opposed to the KNN algorithm, the train and test score will not be compared. This is due to the fact that K-means clustering is an unsupervised algorithm. It cannot be checked which clusters are most optimal, since the ground truth is unknown.

The accuracy of recommendations is evaluated by the test accuracy. The final accuracy scores are computed by calculating the weighted average accuracy of each cluster. Weights represent the size of the cluster, compared to the entire dataset. Thus, larger clusters are better represented than smaller clusters. Accuracy is calculated by formula 3:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (3)$$

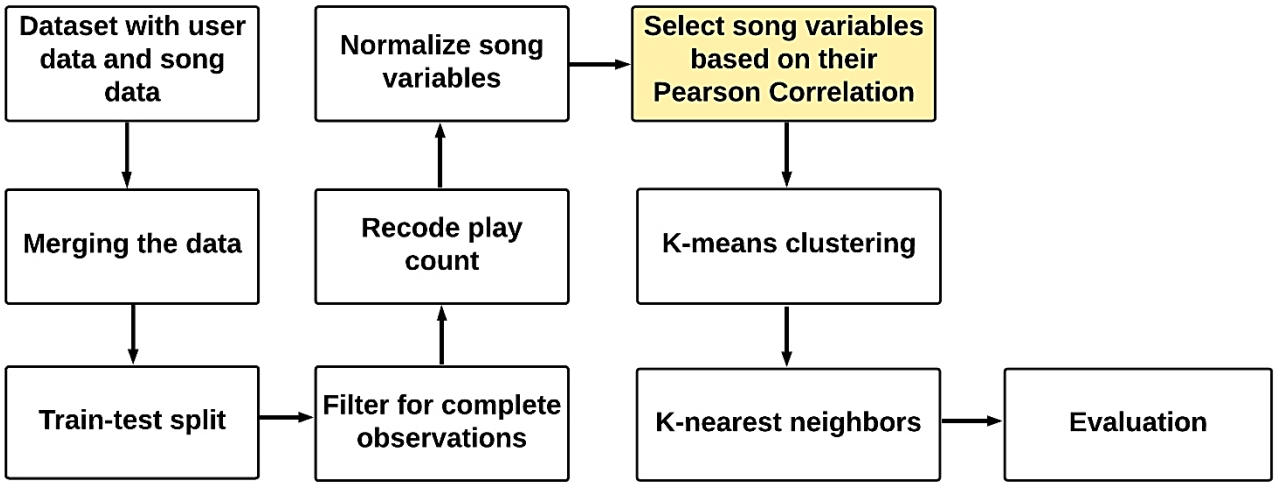
In this formula, TP indicates the number of true positives, which means that both the predicted and actual class is 1. True negatives (TN) indicate that the predicted and actual class is 0. False positives (FP) emerge if class 1 is predicted while the actual class is 0. In contrast, false negatives (FN) indicate that class 0 is predicted, while the actual class is 1. The choice for the accuracy score lies in the fact that a high fraction of correctly identified cases is desired. All classes are equally important, since missing recommendations is not worse than recommending irrelevant songs (Alabdulrahman, Viktor & Paquet, 2018). Generally, the F1 score is used for unbalanced classes, which is the case in this study. However, the imbalance is not unusually large. Further, the accuracy score is expected to be more informative than the F1 score, since the F1 score concentrates on false negatives and false positives. These numbers are not the focus of attention in this study. As a result, the choice for the accuracy score can be justified.

3.3.2 Feature selection models

The feature selection model is constructed in the same manner as the standard model. However, a number of features are selected, meaning that only part of the dataset is introduced to the model. With feature selection, the most informative features of the train set are extracted. Researchers argue that removing the least important features improves clustering, which will improve the quality and efficiency of the model (Ramezani, Moradi & Tab, 2013). However, Aggarwall (2016) and Kużelewska argue that strong clusters deteriorate quality, since part of the predictive power will be lost (2020). Nevertheless, since most researchers argue that stronger clusters improve predictions, it is hypothesized that feature selection increases accuracy. This study uses the filter strategy, since it is aimed to find whether the feature selection model outperforms the standard model. Therefore, it is not necessary to use the computationally expensive wrapper strategy.

Figure 3 displays the pipeline of the feature selection model.

Figure 3 – Pipeline feature selection model

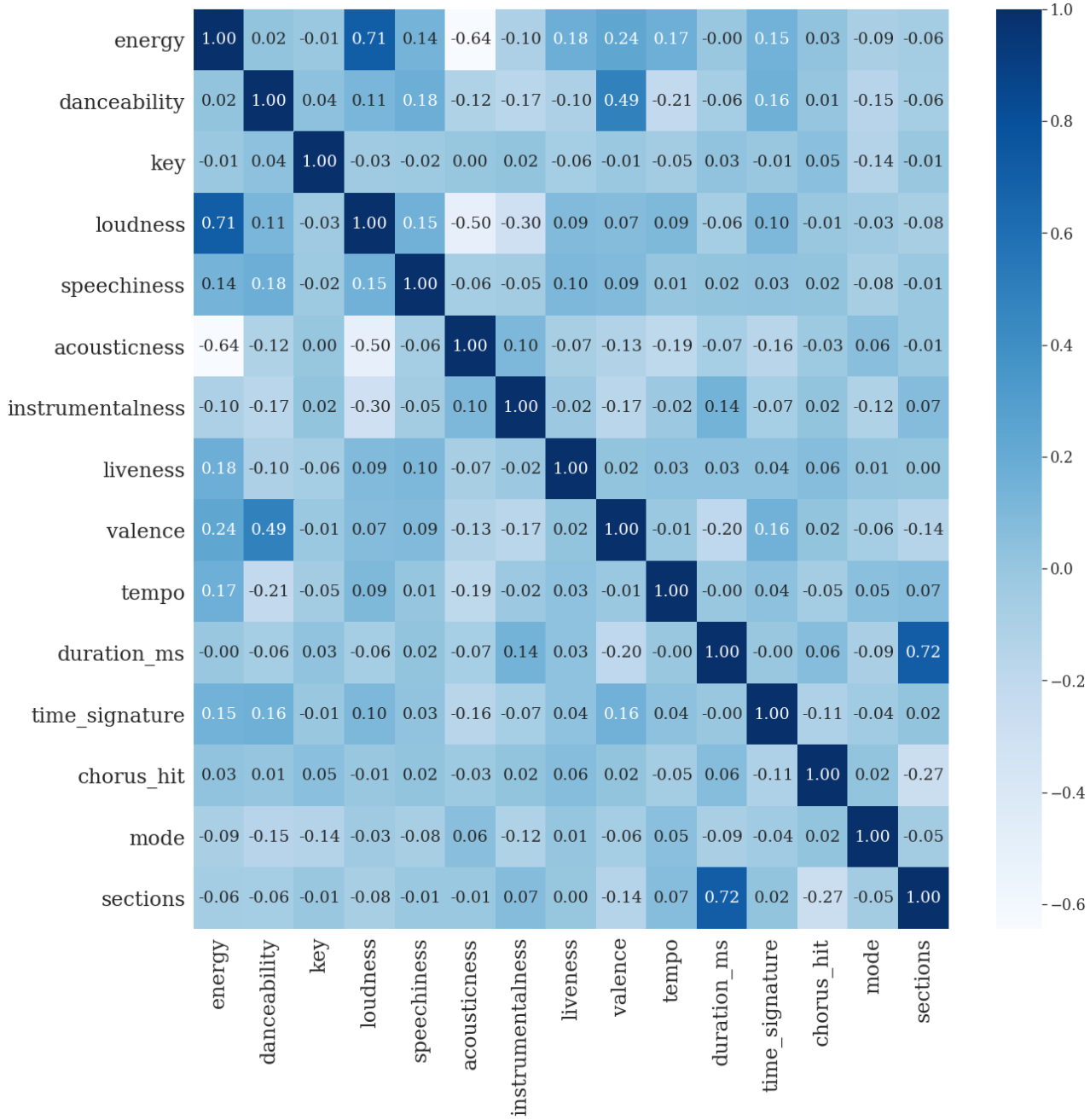


The methodology extends the standard model by filtering the dataset for the most informative variables. This step, which is marked in yellow, is performed before clustering the data. The Pearson Correlation is used to rank variables based on their importance. Namely, the Pearson Correlation is generally used for continuous features that aim to predict a continuous target value (Ndung'u et al., 2021; Panda & Misra, 2021). The linear dependency between *rel_rating* and each song feature is calculated (Chandrashekar & Sahin, 2014). The *n* variables with the highest absolute linear dependency are selected. In addition, the variables *song_id*, *user_id* and *rel_rating* are included, since these are essential for recommending songs.

3.3.3 Dimensionality reduction model

Figure 4 demonstrates that all song variables are correlated to some extent. For instance, energy and loudness are 71 per cent correlated. It is therefore ineffective to include all features in the analysis. For this reason, dimensionality reduction is applied to transform song features into a lower number of dimensions (Panda & Misra, 2021). Using this technique generally leads to better performance (Panda & Misra, 2021). It is therefore hypothesized that the dimensionality reduction model produces a higher accuracy than the standard model. Further, the model increases efficiency.

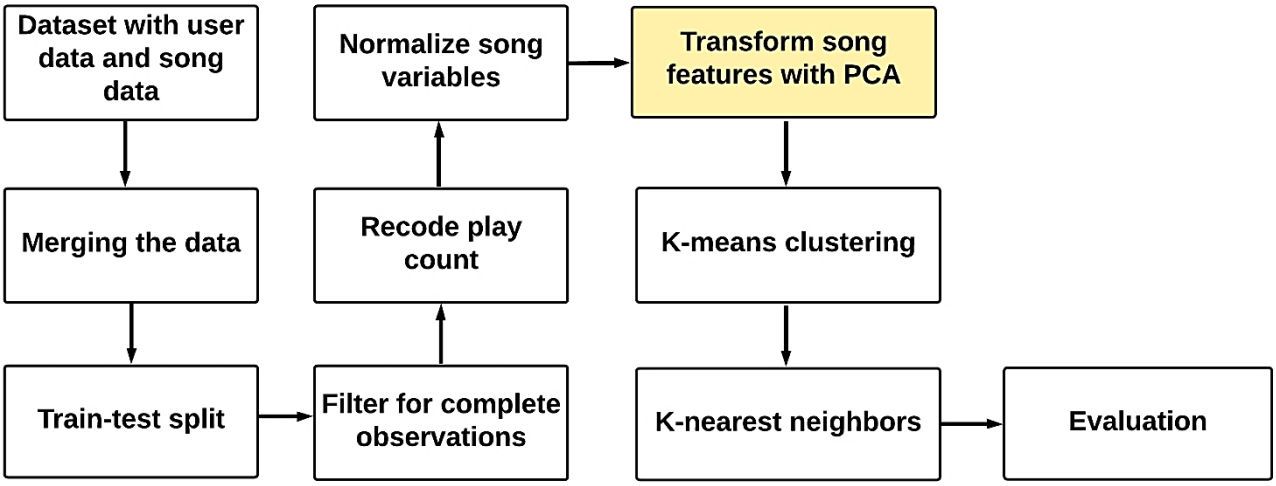
Figure 4 – Correlation matrix



A linear dimensionality reduction technique is used, since complex nonlinear methods are often not capable of outperforming linear methods (Van der Maaten, Postma and Van den Heerik, 2009). Principal Component Analysis (PCA) is chosen, because PCA is a simple method which creates new and uncorrelated variables (Langensiepen et al., 2018). Additionally, PCA is able to deal with continuous data, which is necessary for this study (Nguyen & Holmes, 2019). Namely, even though predictions are converted to a binary scale, the KNN algorithm predicts continuous values.

The pipeline for this model can be found in figure 5.

Figure 5 – Pipeline PCA model



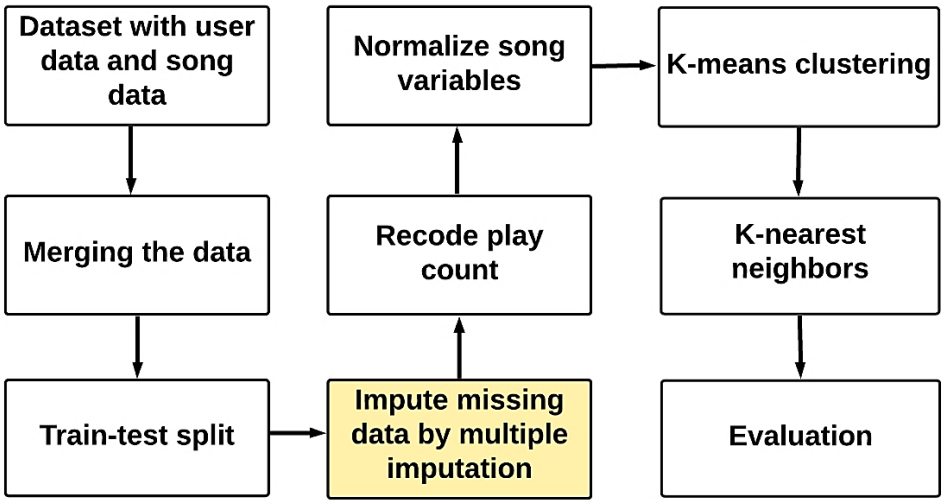
The pipeline for the PCA model is similar to the pipeline for the standard model. However, the PCA model transforms song features before generating song clusters. This step is marked in yellow. Song features are transformed into linearly uncorrelated sets (dimensions) of features (Panda & Misra, 2021). Features are ranked based on the amount of explained variance. It is assumed that each additional dimension explains less variance than the dimension before. PCA aims to explain the highest amount of variance with the lowest number of PCA.

3.3.4 Missing data imputations models

Finally, the performance of a data imputation model is evaluated. This is done by comparing a model that uses the imputed dataset with the standard model, which uses the complete cases dataset. Missing data imputation is expected to improve cluster similarity and improve generalization, since more data can be analyzed. Better generalization will increase performance (Kuhn & Johnson, 2019). Additionally, data imputation is expected to improve the validity of the results. However, data imputation is computationally expensive. For this reason, it will be evaluated whether using data imputation is necessary.

Figure 6 demonstrates the pipeline for the imputed model.

Figure 6 – Pipeline multiple imputation model



This method is similar to the standard model. As opposed to listwise deletion, missing data is imputed (James et al., 2021). This step is marked in yellow. Therefore, the dataset that is used as input for the recommender system has changed. Multiple imputation is used to impute data, because this method is preferred for data that is missing not at random (James et al., 2021). Multiple imputation imputes the mean of several plausible values. In contrast to other models, it must be noted that the performance of the data imputation model cannot directly be compared to the standard model. This is due to the fact that the test set is different. Namely, the imputed dataset contains more observations, since different ways of missing value treatment is performed.

3.4 Hyperparameter tuning

Hyperparameter tuning is the process where values of parameters are set before training the model (Ghawi & Pfeffer, 2019). Finding the optimal values of hyperparameters benefits model accuracy. Optimal values are selected by validation data. Validation accuracy is used because this value is impartial, as opposed to the train accuracy. Namely, the validation set is not used for training the model. Hyperparameters are mutually dependent, meaning that the order of hyperparameter tuning needs to be determined. For simplicity, the order of the methodology will be pursued. Thus, model-specific hyperparameters will be tuned in section 3.4.1. Model-specific hyperparameters are parameters such as the number of dimensions in PCA and the number of features in feature selection. Subsequently, the optimal *k* in K-means clustering will be determined in section 3.4.2. Finally, paragraph 3.4.3 discusses the number of *k* in KNN.

3.4.1 Model-specific hyperparameters

The model-specific hyperparameters that are tuned in this study are the number of most informative features in feature selection (*n_features*) and the number of dimensions in PCA. Since the optimal number of features is unknown, the order of best-performing features for each number of features ranging from one to fifteen will be determined by the Pearson Correlation. This range is chosen because the dataset contains fifteen song features. The validation accuracy will be calculated for each number of features. Subsequently, the number of features that produces the highest validation accuracy score will be selected.

For PCA, the number of dimensions (*n_components*) must be specified. The choice depends on the amount of variance that is explained by each additional component. The dimension after which the variance drops substantially is chosen. This point can be determined by analyzing a graph which displays dimensions in combination with the accompanying amount of explained variance. If no clear turning point is visible, the optimal number of dimensions will be the number of dimensions that leads to the highest validation accuracy score.

3.4.2 K in K-means clustering

The *k* in K-means clustering is chosen by the silhouette score and the elbow method. These methods are commonly used (Shi et al., 2021). For the elbow method, the sum of squared distances (SSD) of each number of clusters is plotted. The SSD indicates the distance of each data point to their closest centroid. The optimal number of *k* will be determined by choosing the 'elbow' of the plot. This method relies on the idea that the first few clusters introduce a lot of variance. However, at some point, introducing a larger number of clusters no longer provides additional information (James et al., 2021). Shi et al. (2021) argued that the elbow method leads to the optimal number of clusters. However, this point is ambiguous, since the actual 'elbow' may be disagreed upon. The sum of squared distances (SSD) is calculated by equation 4 (Pedregosa et al., 2011):

$$SSD = \sum_{j=0}^{n} \min \left(|x_j - \mu_j|^2 \right) \tag{4}$$

In this formula, x_j indicates the datapoint that is concerned. Additionally, μ_j indicates the average data point.

If the elbow method does not suffice, silhouette coefficients are used. Silhouette coefficients represent cluster similarity. When the number is close to 1, items are similar to other items in their cluster and different from items in other clusters. When the number is close to -1, clusters are arbitrary. The number of clusters associated with the highest score is chosen. Equation 5 can be used to calculate the silhouette score (Cintia Ganesha Putri et al., 2020):

$$S_j = \frac{a(j)-b(j)}{\max\{a(j),b(j)\}} \quad (5)$$

In this formula, $a(j)$ is the average dissimilarity of data point *j* with other data within its cluster. $b(j)$ is the minimum average dissimilarity of data point *j* with any other cluster (Cintia Ganesha Putri et al., 2020).

If the silhouette analysis does not lead to an obvious conclusion, the validation accuracy affiliated with each number of clusters is calculated. The number of clusters that is associated with the highest validation accuracy is selected.

3.4.3 K in K-nearest neighbors

The value of *k* in KNN is chosen by looping over all integer values between *k*=2 and *k*=20. For each value of *k*, the validation accuracy is calculated. The number of neighbors that leads to the highest validation accuracy is selected. Wazirali (2020) argued that an odd number of *k* is recommended, since this prevents ending up with a tie. Nevertheless, prediction outcomes are continuous variables, meaning that no tie could occur. As a result, it is irrelevant whether *k* is odd or even.

3.5 Algorithms and packages

To answer the research question, an analysis is performed in the Jupyter Notebook environment in Anaconda (Anaconda Software Distribution, 2020). Python (Van Rossum & Drake, 1995) is used for programming, since this language is commonly used for recommender systems (Ahuja et al., 2019). Data preprocessing is performed using *Pandas* (McKinney, 2010) and Numpy (Harris et al., 2020). K-means clustering is performed by the *scikit learn* tool (Pedregosa et al., 2011). Additionally, this package is used for silhouette analysis, PCA and feature selection. K-nearest neighbors is performed by the *surprise* library (Hug, 2020). Furthermore, the *MICE* package is used for multiple imputation (Van Buuren & Groothuis-Oudshoorn, 2011). Further, several visualizations are designed in R (R Core Team, 2016), since customizing visualizations is easier with the *ggplot* package (Fahad & Yahya, 2018) than with Python packages. Other visualizations are created in Python. These visualizations are designed using the *Matplotlib* (Hunter, 2007) or *Seaborn* (Waskom et al., 2017) package.

4. Results

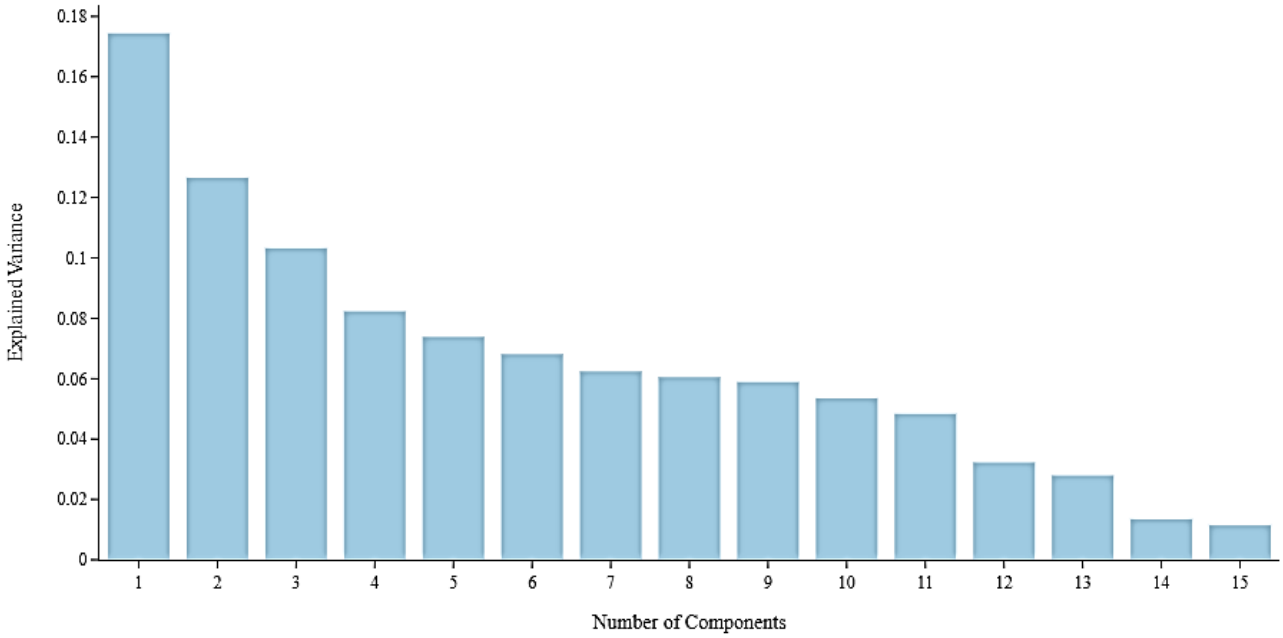
Hyperparameter values and model results are disclosed in this chapter. Section 4.1 discloses the values of hyperparameters. When these are determined, the models can be run. Results on the similarity of items within clusters are examined in section 4.2. Further, section 4.3 demonstrates results on model performance.

4.1 Hyperparameters

It is described in paragraph 3.4 that model-specific hyperparameter values are determined first. Figure 15 in appendix I suggests that the optimal number of features for the feature selection method is twelve. It is determined by absolute Pearson Correlation scores that the features *loudness*, *speechiness*, *acousticness*, *duration_ms*, *instrumentalness*, *danceability*, *energy*, *chorus_hit*, *mode*, *valence*, *sections* and *key* are most informative. These are included in the model. Additionally, the features *song_id*, *user_id* and *rel_rating* are included, since these are crucial for generating recommendations.

The optimal number of dimensions in PCA is determined by a drop of variance, compared to an earlier dimension. Figure 7 indicates that no substantial drop could be detected. For this reason, the validation accuracy of each dimension is calculated. Figure 16 in appendix I indicates that PCA3 is optimal, since this leads to the highest validation accuracy. Similarly, the features *song_id*, *user_id* and *rel_rating* are included in the analysis.

Figure 7 – Explained variance for PCA



After determining the values of model-specific hyperparameters, the number of *k* in K-means clustering are chosen. This study uses the elbow plot and the silhouette score to tune this hyperparameter. Scores are illustrated in figure 17 in appendix I. In these graphs, the black lines indicate the elbow plot, while the silhouette scores are presented by the blue line. For the visualizations, the SSD scores are normalized. This is because the ranges from the SSD scores and silhouette scores differ. Normalizing allows visualizing both plots in one graph.

The elbow plot suggests that k=6 should be chosen for the feature selection model. For each other model, the elbow method does not produce clear results. Thus, silhouette scores will be regarded. The silhouette plot for the standard model peaks at k=4. However, the silhouette plot for the dimensionality reduction model is unambiguous. However, figure 18 in appendix I indicates that k=7 produces the highest silhouette score. Hence, this value will be chosen. For the imputation method, calculating SSD scores and silhouettes scores is too computationally expensive. Thus, the validation accuracy of all values of *k*, ranging from 2 to 8, will be calculated. This range is chosen because the optimal number of clusters for the other sub-questions falls within this range. Figure 18 in appendix I suggests that k=8 should be chosen for the imputation model, since this value leads to the highest validation accuracy.

Lastly, the number of *k* in KNN is selected. To do this, the validation accuracy for each number of *k* between 2 and 20 is calculated. The *k* that is accompanied with the highest validation accuracy is selected. The *k*'s that are used for this analysis are reported in figure 8. This figure also provides a summary for the other hyperparameter values that are chosen for this research.

Figure 8 – Summary of hyperparameter values

Model	**Model-specific hyperparameters**	***k* in K-means clustering**	***k* in KNN**
Standard model	-	4	18
Feature selection model	12 features	6	14
Dimensionality reduction model	3 dimensions	7	14
Missing data imputation	-	8	16

4.2 Results on cluster similarity

Sub-question 1, 2 and 3 study the effect of feature engineering techniques on cluster similarity. Cluster similarity is determined by the Davies Bouldin score of test clusters. A lower Davies Bouldin score indicates high similarity. The scores are demonstrated in figure 9.

Figure 9 - Similarity of clusters

Model	Davies Bouldin score
Standard model	2.701
Feature selection model	2.217
Dimensionality reduction model	1.460
Missing data imputation model	2.749

Sub-question 1 concludes that feature selection improves cluster similarity. An equivalent conclusion can be drawn for sub-question 2, which regards the cluster similarity for the dimensionality reduction model. In contrast, missing data imputation leads to lower cluster similarity, although the difference is relatively small. Cluster visualizations for each model can be found in figure 18 in appendix I.

4.3 Results on recommendation accuracy

The performance of recommender systems that use feature engineering techniques is studied in sub-question 4, 5 and 6. Performance is measures by the accuracy of recommendations. Before interpreting the accuracy scores, confusion matrices are discussed. These provide more information on the effect of feature engineering techniques. Figure 10 illustrates the confusion matrix for each of the models. The matrices reveal the number of true and false positives and true and false negatives. Additionally, the accuracy, precision, recall and F1 scores are demonstrated. This increases interpretability of the results. Further, scores are calculated by the weighted average per cluster. This means that larger clusters are better represented than smaller clusters. For the sake of completeness, the number of true and false positives and negatives per cluster can be found in figure 20 in appendix I. The numbers and shares originate from test data, since test data is used to evaluate and compare models.

Figure 10 - Confusion matrices

Figure 10.1 - Standard model

	Predicted 0	**Predicted 1**	**Accuracy: 0.749**
Actual 0	90,227	23,882	Precision: 0.572 Recall: 0.653
Actual 1	17,015	31,967	F1: 0.610

Figure 10.2 - Feature selection model

	Predicted 0	**Predicted 1**	Accuracy: 0.770
Actual 0	92,586	21,526	Precision: 0.605 Recall: 0.674
Actual 1	15,979	32,999	F1: 0.638

Figure 10.3 - Dimensionality reduction model

	Predicted 0	**Predicted 1**	Accuracy: 0.772
Actual 0	93,937	20,175	Precision: 0.613 Recall: 0.652
Actual 1	17,044	31,937	F1: 0.632

Figure 10.4 - Missing data imputation model

	Predicted 0	**Predicted 1**	Accuracy: 0.677
Actual 0	251,822	57,805	Precision: 0.627 Recall: 0.472
Actual 1	108,263	96,968	F1: 0.539

In confusion matrices, higher performance scores indicate better performance. Figure 10 implies that each feature engineering technique results in higher precision, which means that the models are better capable of predicting the positive class. The feature selection model leads to a higher recall score. This score quantifies the share of positive predictions that is actually positive. In contrast, the missing data imputation model leads to a lower recall score. Dimensionality reduction does not increase nor decrease recall. The F1 score is the weighted average of recall and precision. The F1 scores for feature selection and dimensionality reduction are better than the F1 score of the standard model. In contrast, the F1 score for data imputation is aggravated.

However, it must be regarded that the distribution of the complete cases dataset and the imputed dataset differs. Specifically, the majority class of the complete cases dataset is 70.0 per cent. This dataset is used for the standard model, the feature selection model and the dimensionality reduction model. Thus, if the model simply predicts the majority class for all observations, it will already reach an accuracy of 0.7. In contrast, the majority class of the imputed dataset is only 60.1 per cent. This difference should be taken into account when interpreting the results. For the classification scores in figure 10, it is important to note that obtaining high scores is more difficult for the imputed model. Thus, it cannot be concluded that the imputed model performs worse than the standard model.

This thesis evaluates quality by the accuracy score. Since the accuracy scores of the feature selection and dimensionality reduction model are compared to the standard model, these accuracies are displayed in figure 11.

Figure 11 - Accuracy scores feature selection and dimensionality reduction model

Model	**Train accuracy**	**Test accuracy**
Standard model	0.749	0.749
Feature selection model	0.769	0.770
Dimensionality reduction model	0.774	0.772

First of all, figure 11 demonstrates little differences between the train and test accuracy for each of the models. This outcome is important, since large differences between train and test evaluation metrics may compromise the validity of the results. To answer sub-question 4 and 5, it is concluded that feature selection and dimensionality reduction improve performance, compared to the standard model. This can be established since the test accuracy of these models is higher than the test accuracy of the standard model.

Figure 12 demonstrates accuracy scores for the imputation model.

Figure 12 – Accuracy scores imputation model

Model	Train accuracy	Test accuracy
Standard model	0.749	0.749
Missing data imputation model	0.675	0.677

These scores are displayed in a separate figure, since the accuracy of the imputed model cannot be directly compared to the standard model. Figure 12 implies that the model does not overfit nor underfit, since little differences between the train and test accuracy are detectable. Regarding sub-question 6, figure 12 demonstrates that the accuracy of the missing data imputation model is lower than the accuracy of the standard model. However, the naïve baseline that is affiliated with the imputed model is lower than the naïve baseline of the standard model. This is due to the fact that the dataset for the standard model is filtered for all complete observations, while the test set for the imputation model imputed missing values. Therefore, it cannot be concluded that the missing data imputation model performs worse than the standard model. In fact, the test accuracy of the imputed model outperforms the corresponding naïve baseline with a larger proportion than the test accuracy of the standard model, compared to its corresponding naïve baseline. It is therefore suggested that the missing data imputation model performs better, although this cannot be argued with certainty.

5. Discussion

5.1 Findings

In this study, a standard two-stage hybrid recommender system was compared to similar models which used feature engineering techniques before clustering. This study is conducted because earlier literature had not yet analyzed the effect of feature engineering techniques on cluster similarity. Moreover, the effect of cluster similarity on the quality of recommendations had not yet been studied.

It was hypothesized that feature selection, dimensionality reduction and missing data imputation would result in higher cluster similarity. The hypotheses for feature selection and dimensionality reduction models are confirmed, since their Davies Bouldin (DB) statistic is lower than the DB index for the standard model. However, the DB index for the missing data imputation model is higher than the standard model. This indicates that multiple imputation aggravates similarity. Supposedly, multiple imputation is unable to impute appropriate values for song features. As a result, the model fails to generate clusters with high similarity. This conclusion is in line with research performed by James et al. (2021) and Panda and Misra (2021), who argued that no method is able to fully account for missing values. However, according to researchers, more data also simplifies creating clusters since more observations can be tested. Thus, the rejection of the hypothesis implies that using more data for better generalization (Panda & Misra, 2021) is less valuable for clustering than imputing plausible values.

Further, it was expected that each feature engineering method would outperform the standard model. These hypotheses are confirmed for the feature selection and dimensionality reduction model. This supports findings by Jin and Han (2020), Ahuja et al. (2019), Liao and Lee (2016) and Kim et al. (2007), who argued that higher cluster similarity improves the quality of recommendations. Furthermore, this conclusion contrasts matters addressed by Aggarwall (2016) and Kużelewska (2020), who argued that better defined clusters may lead to worse performing models. In contrast, the data imputation model leads to lower accuracy than the standard model. However, the test set of the imputation model is larger than the test set of the standard model, since the standard model deleted all troubled observations. Therefore, comparing models is complicated. The data imputation model outperformed its naïve baseline by a larger share than the degree to which the standard model outperformed its naïve baseline. Thus, it is expected that missing data imputation improves performance. However, more research on data imputation is necessary to be certain. This suggestion is in line with Panda and Misra (2021), who argued that multiple imputation preserves accuracy. However, this contrasts research by Jin and Han (2020), since lower cluster similarity actually decreases accuracy in this case. Nevertheless, data imputation is recommended because it leads to

more robust results (James et al, 2021; Panda & Misra, 2021). Namely, the multiple imputation model is expected to better represent society.

5.2 Limitations

In this study, a number of issues should be considered. The main limitation is that the test set which is corresponding with the standard model is smaller than the test set corresponding with the imputation model. As a result, the models are confronted with different majority classes. This study aimed to check whether multiple imputation actually leads to better results than listwise deletion, since multiple imputation leads to higher computation costs. However, differences in the test set complicate the evaluation. Nevertheless, this seems like an insurmountable issue, since the test set naturally needs to be different. If the test sets were equal, both test sets would have to be filtered for all observations. However, the difference between these models is the input. As a result, no differences would occur. Another option would be that both sets preserve missing values. However, clustering cannot be performed when song features are missing. Hence, results could only be drawn for an item-based CF model, instead of a hybrid two-stage model.

Another related limitation concerns the fact that missing values in the complete cases dataset are MNAR. MNAR data is a problem because it does not truly represent society. Therefore, results on the impact of feature selection and dimensionality reduction may be biased. In contrast, the data imputation model largely solved this issue by using an imputation method that is expected to be best for MNAR data.

Furthermore, the absence of explicit user feedback requires guessing actual preferences. For this purpose, the relative rating is calculated. This method is expected to better represent preferences than *play_count.* Nevertheless, preferences still need to be estimated.

Additionally, the boundaries chosen for category 1 (liking the song) and category 0 (not liking the song) affect performance. Namely, it is argued in this research that a relative play count of at least 1.0 indicates liking the song. However, other people may say that a song is only liked when the relative play count is substantially higher than 1.0.

Lastly, hyperparameter must be tuned in succession. Therefore, the order of the methodology is maintained. For the feature engineering models, the hyperparameter values of the standard model were used for *k* in K-means clustering and *k* in KNN. These assumptions are made because model-specific hyperparameters such as *n_features* and *n_components* must be specified before determining the number of clusters and the number of neighbors. However, the optimal values of the number of clusters and the number of neighbors were not yet known, meaning that other values could have been optimal.

5.3 Future work

For future work, it is recommended to follow up on missing data handling methods. Even though it is expected that missing data imputation leads to higher accuracy and more robust results, the results cannot confirm this with certainty. More research on the accuracy and robustness of MNAR data is therefore suggested.

6. Conclusion

In this study, a two-stage recommender system was built. In the first stage, songs were clustered based on song features. This study aimed to analyze whether feature engineering techniques affect cluster similarity. In the second stage, recommendations were generated based on implicit ratings. This stage was evaluated by measuring the effect of feature engineering techniques on the performance of the recommendation system. The feature engineering techniques that were examined are feature selection, dimensionality reduction and missing data imputation.

It was found that feature selection and dimensionality reduction result in higher cluster similarity. Further, these models improve the quality of recommendations, compared to the standard model. The missing data imputation model leads to aggravated clusters. This model also produces a lower accuracy for recommendations. However, the imputed model outperformed the naïve baseline by a larger share than the performance of the standard model, compared to its naïve baseline. As a result, it is suggested that the imputed model performs better. Further, the imputed model is preferred because of its higher robustness. However, it is recommended for future research to follow up on missing data imputation methods, since no evident conclusions can be drawn.

References

Afoudi, Y., Lazaar, M., & Al Achhab, M. (2019). Impact of Feature selection on content-based recommendation system. *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*. https://doi.org/10.1109/wits.2019.8723706

Aggarwal, C. (2016). *Recommender Systems*. Springer Publishing. https://doi.org/10.1007/978-3-319-29659-3

Ahuja, R., Solanki, A., & Nayyar, A. (2019). Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. Published. https://doi.org/10.1109/confluence.2019.8776969

Alabdulrahman, R., Viktor, H., & Paquet, E. (2018). Beyond k-NN: Combining Cluster Analysis and Classification for Recommender Systems. *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. https://doi.org/10.5220/0006931200820091

Anaconda Software Distribution. (2020). *Anaconda Documentation*. Anaconda Inc. Retrieved from https://docs.anaconda.com/

Ansari, F. (2020). *The Spotify Hit Predictor Dataset (1960–2019)*. Kaggle. https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset

Baarsch, J., & Celebi, M.E. (2012). Investigation of Internal Validity Measures for K-Means Clustering.

Banerjee, A. (2018, February 22). *Million Song Data Set Subset*. Kaggle. https://www.kaggle.com/anuragbanerjee/million-song-data-set-subset

Bertin-Mahieux, T., Ellis, D., Whitman, B., & Lamere, P. (2011). The Million Song Dataset. *Proceedings of the 12th International Conference on Music Information Retrieval*. Published.

Bhatnagar, V. (Red.). (2017). Collaborative Filtering Using Data Mining and Analysis. *Advances in Data Mining and Database Management*. Published. https://doi.org/10.4018/978-1-5225-0489-4

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

Chemeque Rabel, M. (2020). Content-based music recommendation system : A comparison of supervised Machine Learning models and music features (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-288534

Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., & He, X. (2020). Bias and Debias in Recommender System: A Survey and Future Directions. *Association for Computing Machinery*, *1*(1). https://doi.org/10.1145/1122445.1122456

Cintia Ganesha Putri, D., Leu, J. S., & Seda, P. (2020). Design of an Unsupervised Machine Learning-Based Movie Recommender System. *Symmetry*, *12*(2), 185. https://doi.org/10.3390/sym12020185

Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227. https://doi.org/10.1109/tpami.1979.4766909

Fahad, S. A., & Yahya, A. E. (2018). Big Data Visualization: Allotting by R and Python with GUI Tools. *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*. Published. https://doi.org/10.1109/icscee.2018.8538413

Ghawi, R., & Pfeffer, J. (2019). Efficient Hyperparameter Tuning with Grid Search for Text Categorization using kNN Approach with BM25 Similarity. *Open Computer Science*, *9*(1), 160–180. https://doi.org/10.1515/comp-2019-0011

Harris, C. R., Millman, K. J., Van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining*. https://doi.org/10.1109/icdm.2008.22

Hug, N. (2020). Surprise: A Python library for recommender systems. *Journal of Open Source Software*, *5*(52), 2174. https://doi.org/10.21105/joss.02174

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/mcse.2007.55

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1_1

Jawaheer, G., Szomszor, M., & Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems - HetRec '10*. https://doi.org/10.1145/1869446.1869453

Jin, Y., & Han, C. (2020). A music recommendation algorithm based on clustering and latent factor model. *MATEC Web of Conferences, 309*, 03009. https://doi.org/10.1051/matecconf/202030903009

Ju, C., & Xu, C. (2013). A New Collaborative Recommendation Approach Based on Users Clustering Using Artificial Bee Colony Algorithm. *The Scientific World Journal, 2013*, 1–9. https://doi.org/10.1155/2013/869658

Kim, D., Kim, K. S., Park, K. H., Lee, J. H., & Lee, K. M. (2007). A music recommendation system with a dynamic k-means clustering algorithm. *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. https://doi.org/10.1109/icmla.2007.97

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection*. Taylor & Francis. https://doi.org/10.1080/00031305.2020.1790217

Kużelewska, U. (2020). Effect of Dataset Size on Efficiency of Collaborative Filtering Recommender Systems with Multi-clustering as a Neighbourhood Identification Strategy. *Lecture Notes in Computer Science*, 342–354. https://doi.org/10.1007/978-3-030-50420-5_25

Langensiepen, C., Cripps, A., & Cant, R. (2018). Using PCA and K-Means to Predict Likeable Songs from Playlist Information. *2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim)*. https://doi.org/10.1109/uksim.2018.00017

Lee, K., & Lee, K. (2015). Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. *Expert Systems with Applications*, *42*(10), 4851–4858. https://doi.org/10.1016/j.eswa.2014.07.024

Li, X., Xing, J., Wang, H., Zheng, L., Jia, S., & Wang, Q. (2017). A Hybrid Recommendation Method Based on Feature for Offline Book Personalization. *Journal of Computers*.

Liao, C. L., & Lee, S. J. (2016). A clustering based approach to improving the efficiency of collaborative filtering recommendation. *Electronic Commerce Research and Applications*, *18*, 1–9. https://doi.org/10.1016/j.elerap.2016.05.001

Lozic, J., Vojkovic, G., & Milkovic, I. M. (2020). "Financial" Aspects of Spotify Streaming Model. *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. https://doi.org/10.23919/mipro48935.2020.9245185

Lü, L., Medo, M., Yeung, C. H., Zhang, Y. C., Zhang, Z. K., & Zhou, T. (2012). Recommender systems. *Physics Reports*, *519*(1), 1–49. https://doi.org/10.1016/j.physrep.2012.02.006

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*. Published. https://doi.org/10.25080/majora-92bf1922-00a

Millecamp, M., Htun, N. N., Jin, Y., & Verbert, K. (2018). Controlling Spotify Recommendations. *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. Published. https://doi.org/10.1145/3209219.3209223

Music audience understanding. (2014). *The Echo Nest*. Published. http://musicalidentity.echonest.com/post/58057887326/engagement-white-paper

Najafabadi, M. K., Mahrin, M. N., Chuprat, S., & Sarkan, H. M. (2017). Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. *Computers in Human Behavior*, *67*, 113–128. https://doi.org/10.1016/j.chb.2016.11.010

Ndung'u, R.N., Kamau, G., & Mariga, G. (2021). Using Feature Selection Methods to Discover Common Users' Preferences for Online Recommender Systems.

Nguyen, L. H., & Holmes, S. (2019). Ten quick tips for effective dimensionality reduction. *PLOS Computational Biology*, *15*(6), e1006907. https://doi.org/10.1371/journal.pcbi.1006907

Pacula, M. (2010). A Matrix Factorization Algorithm for Music Recommendation using Implicit User Feedback.

Panda, M., & Misra, H. (2021). *Handbook of Research on Automated Feature Engineering and Advanced Applications in Data Science (Advances in Data Mining and Database Management)* (1ste ed.). IGI Global. https://doi.org/10.4018/978-1-7998-6659-6

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & others (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825--2830.

Pérez-Marcos, J., & López Batista, V. (2017). Recommender System Based on Collaborative Filtering for Spotify's Users. *Advances in Intelligent Systems and Computing*, 214–220. https://doi.org/10.1007/978-3-319-61578-3_22

Puntheeranurak, S., & Tsuji, H. (2007). A Multi-clustering Hybrid Recommender System. 7th IEEE International Conference on Computer and Information Technology (CIT 2007). https://doi.org/10.1109/cit.2007.54

R Core Team. (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. Retrieved from https://www.R-project.org/

Ramezani, M., Moradi, P., & Tab, F. A. (2013). Improve performance of collaborative filtering systems using backward feature selection. *The 5th Conference on Information and Knowledge Technology*. Published. https://doi.org/10.1109/ikt.2013.6620069

Rubtsov, V., Kamenshchikov, M., Valyaev, I., Leksin, V., & Ignatov, D. I. (2018). A hybrid two-stage recommender system for automatic playlist continuation. *Proceedings of the ACM Recommender Systems Challenge 2018 on - RecSys Challenge '18*. Published. https://doi.org/10.1145/3267471.3267488

Sánchez-Moreno, D., Gil González, A. B., Muñoz Vicente, M. D., López Batista, V. F., & Moreno García, M. N. (2016). A collaborative filtering method for music recommendation using

playing coefficients for artists and users. *Expert Systems with Applications*, *66*, 234–244. https://doi.org/10.1016/j.eswa.2016.09.019

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the tenth international conference on World Wide Web - WWW '01*. Published. https://doi.org/10.1145/371920.372071

Shi, C., Wei, B., Wei, S., Wang, W., Liu, H., & Liu, J. (2021). A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2021*(1). https://doi.org/10.1186/s13638-021-01910-w

Thi Do, M., Nguyen, L., & Van Nguyen, D. (2010). Model-based approach for Collaborative Filtering. *The 6th International Conference on Information Technology for Education*. Published.

Thuan, T. T., & Puntheeranurak, S. (2014). Hybrid recommender system with review helpfulness features. *TENCON 2014 - 2014 IEEE Region 10 Conference*. Published. https://doi.org/10.1109/tencon.2014.7022397

Van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. In *Journal of Statistical Software* (Vol. 45, Issue 3, pp. 1–67). https://www.jstatsoft.org/v45/i03/

Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Waskom, M., Botvinnik, O., O'Kane, D, Hobson, P., Lukauskas, S., Gemperline, D., … Qalieh, A. (2017). *mwaskom/seaborn: v0.8.1 (September 2017)*. Zenodo. https://doi.org/10.5281/zenodo.883859

Wazirali, R. (2020). An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation. *Arabian Journal for Science and Engineering*, *45*(12), 10859–10873. https://doi.org/10.1007/s13369-020-04907-7

Wei, S., Ye, N., Zhang, S., Huang, X., & Zhu, J. (2012). Collaborative Filtering Recommendation Algorithm Based on Item Clustering and Global Similarity. *2012 Fifth International Conference on Business Intelligence and Financial Engineering*. Published. https://doi.org/10.1109/bife.2012.23

Xie, J., Leishman, S., Tian, L., Lisuk, D., Koo, S., & Blume, M. (2012). Feature Engineering in User's Music Preference Prediction. *KDD Cup*.

Yadav, V., Shukla, R., Tripathi, A., & Maurya, A. (2021). A New Approach for Movie Recommender System using K-means Clustering and PCA. *Journal of Scientific & Industrial Research*, *80*, 159–165.

Zhou, X., Xu, Y., Li, Y., Josang, A., & Cox, C. (2011). The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, *37*(2), 119–132. https://doi.org/10.1007/s10462-011-9222-1

Appendices

Appendix I

Figure 13 – Dataset description

Variable	Description	Data Type
Title	Name of the song	Object
Artist	Artist of the song	Object
Uri	Resource identifier for this song	Object
Danceability	Danceability describes how suitable the song is for dancing, where 1 is most suitable and 0 is least suitable	Float
Energy	Activity and intensity are measured, where 1 is most energetic and 0 is least energetic	Float
Key	Estimated average key of the song, so the higher the key, the higher the overall note of the song. If no key is detected, key = -1	Integer
Loudness	The overall loudness of the song in decibel (dB)	Float
Mode	Modality (scale of the melodic content) of the song, where major = 1 and minor = 0	Integer
Speechiness	This variable indicates the presence of spoken words in a song. The higher the value, the more spoken words	Float
Acousticness	Confidence measure that describer whether the song is acoustic, where 1 represents the highest confidentiality of the track being acoustic	Float
Instrumentalness	This variable predicts whether a song contains no vocals, so a value close to 1 indicates that the track is very instrumental	Float
Liveness	Confidence measure of the song being performed live, so a value close to 1 means that the song was probably recorded when performed live	Float
Valence	Valence represents the positivity of a song, a value of 1 means that the song is very positive	Float
Tempo	Tempo of the song in beats per minute (BPM)	Float
Duration_ms	Duration of the song in milliseconds	Integer

Time_signature	A high time signature indicates a high number of beats per bar (or other measure)	Integer
Chorus_hit	Estimate of the moment that the chorus would start for the track. This value is represented by milliseconds	Float
Sections	Number of sections in the song	Integer
User_id	ID of the user	Object
Song_id	ID of the song	Object
Play_count	The number of times the user has played the song	Integer
Album	Album in which the song was released	Object
Year	Release year of the song	Integer
Rel_rating	Relative rating of the song, where 1.00 means that the person thinks the song is average. When rel_rating > 1.00 it is concluded that the person likes the song. Also, when rel_rating < 1.00 the person will not like the song. The relative rating is calculated by the play count divided by that person's mean play count.	Float

Source: Ansari (2020)

Figure 14 – Distribution of song variables

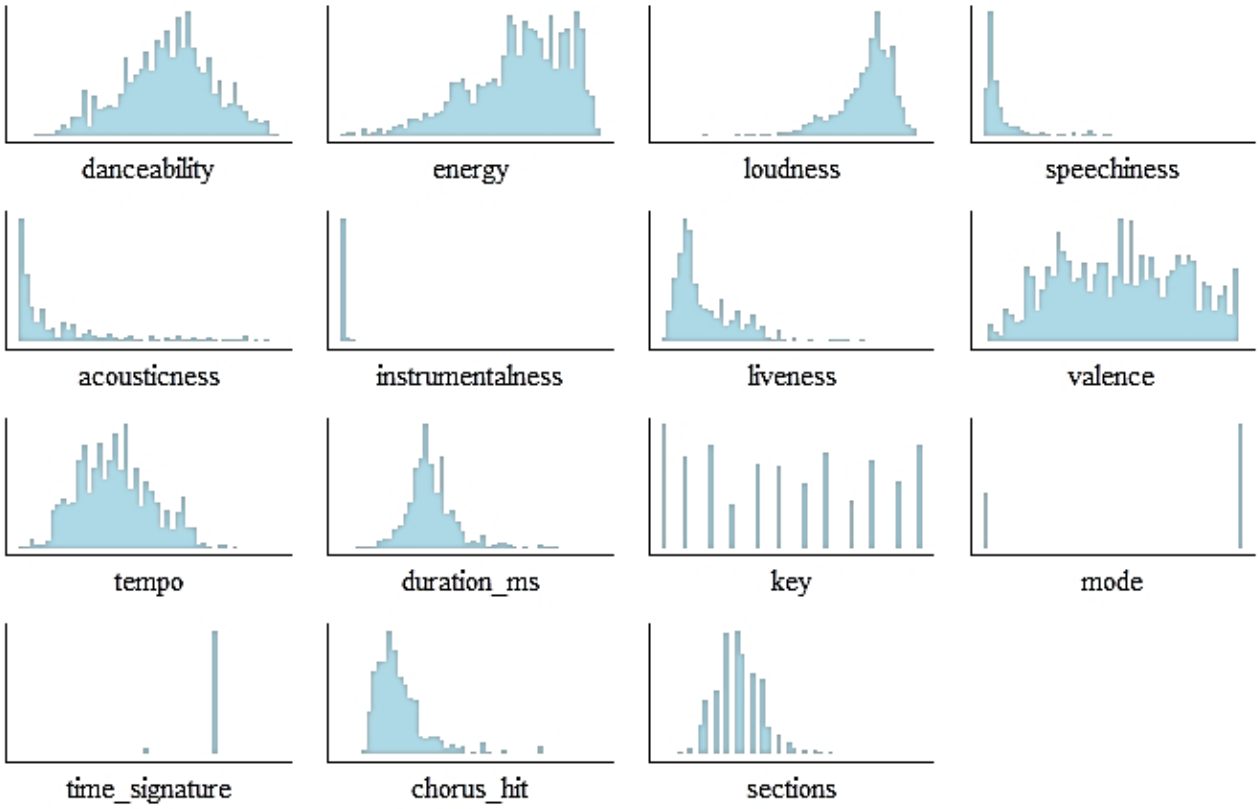


Figure 15 – Number of features with feature selection

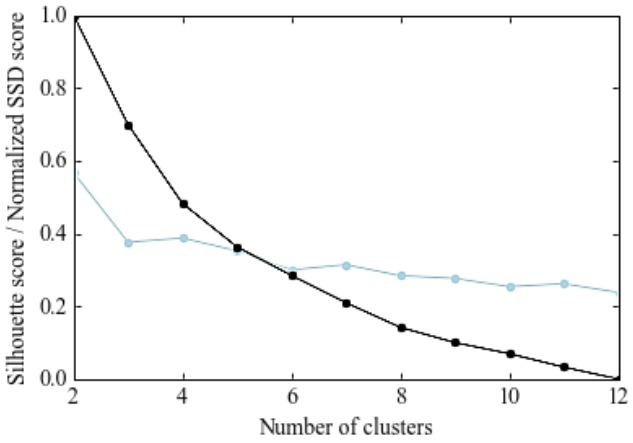
Number of features	Feature to add	Validation accuracy
1	Loudness	0.591
2	Speechiness	0.602
3	Acousticness	0.601
4	Duration_ms	0.613
5	Instrumentalness	0.601
6	Danceability	0.601
7	Energy	0.604
8	Chorus_hit	0.604
9	Mode	0.608
10	Valence	0.606
11	Sections	0.608
12	Key	0.623
13	Liveness	0.605
14	Time_signature	0.619
15	Tempo	0.621

Figure 16 – Number of dimensions with PCA

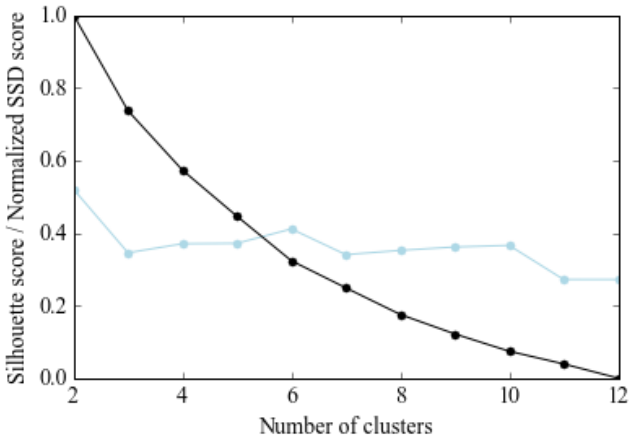
PCA	Variance explained (cumulative)	Validation accuracy
1	0.175	0.610
2	0.301	0.610
3	0.405	0.637
4	0.487	0.634
5	0.561	0.599
6	0.629	0.602
7	0.692	0.604
8	0.753	0.580
9	0.812	0.605
10	0.866	0.609
11	0.914	0.604
12	0.947	0.601
13	0.975	0.621
14	0.988	0.612
15	1.000	0.621

Figure 17 – Silhouette analyses

17.1 Standard model



17.2 Feature selection model



17.3 Dimensionality reduction model

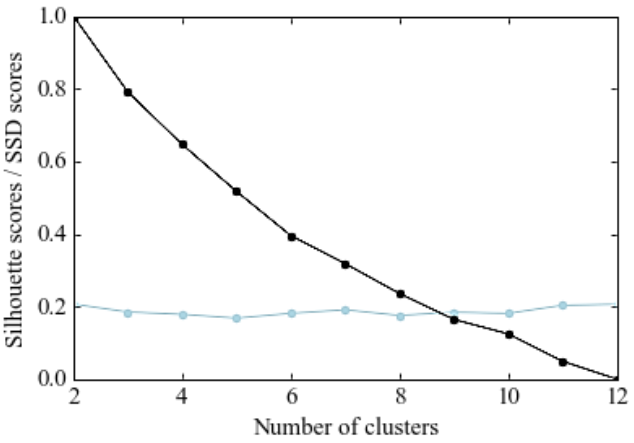


Figure 18 – Determine the number of clusters

18.1 Standard model

Number of clusters	Silhouette score	Sum of squared distances
2	0.567	214518.168
3	0.376	184249.085
4	0.387	162437.609
5	0.352	150404.899
6	0.300	142501.372
7	0.314	134948.680
8	0.284	128129.867
9	0.277	123918.663
10	0.254	120843.830
11	0.262	117124.111
12	0.238	113830.312

18.2 Feature selection model

Number of clusters	Silhouette score	Sum of squared distances
2	0.520	31846.544
3	0.345	26944.295
4	0.370	23900.675
5	0.371	21540.262
6	0.411	19257.441
7	0.340	17869.232
8	0.352	16503.908
9	0.361	15506.707
10	0.366	14620.249
11	0.272	13969.130
12	0.272	13240.642

18.3 Dimensionality reduction model

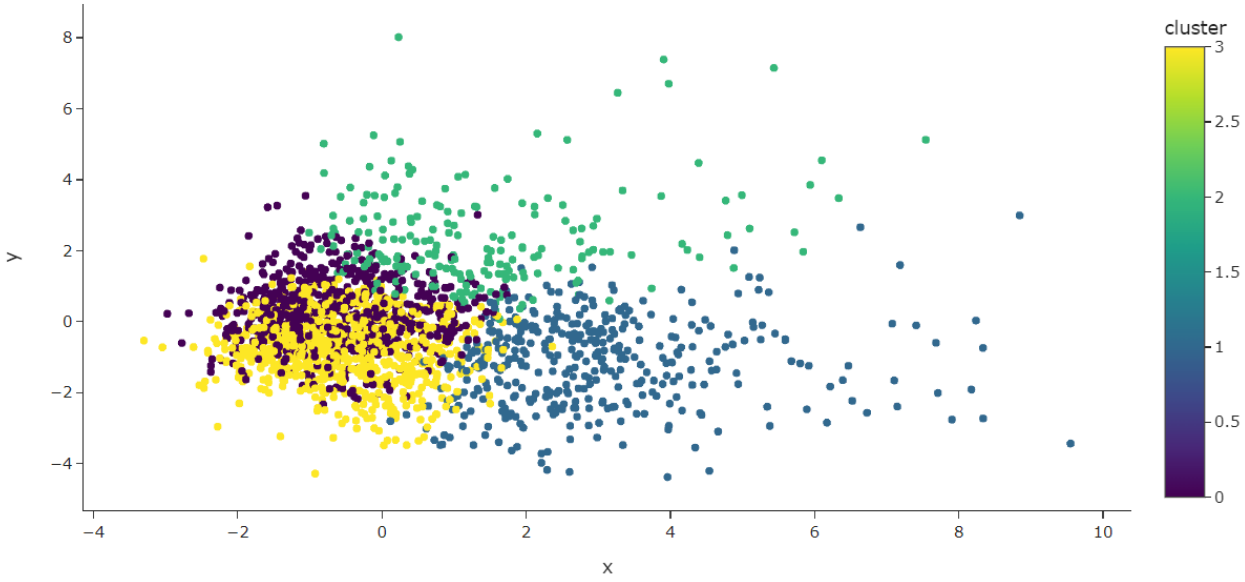
Number of clusters	Silhouette score	Sum of squared distances
2	0.207	5529558.704
3	0.185	5055565.479
4	0.179	4726832.509
5	0.168	4433059.393
6	0.181	4156209.622
7	0.191	3980815.822
8	0.176	3793614.586
9	0.184	3631590.543
10	0.181	3541232.722
11	0.203	3369847.577
12	0.206	3257968.397

18.4 Data imputation model

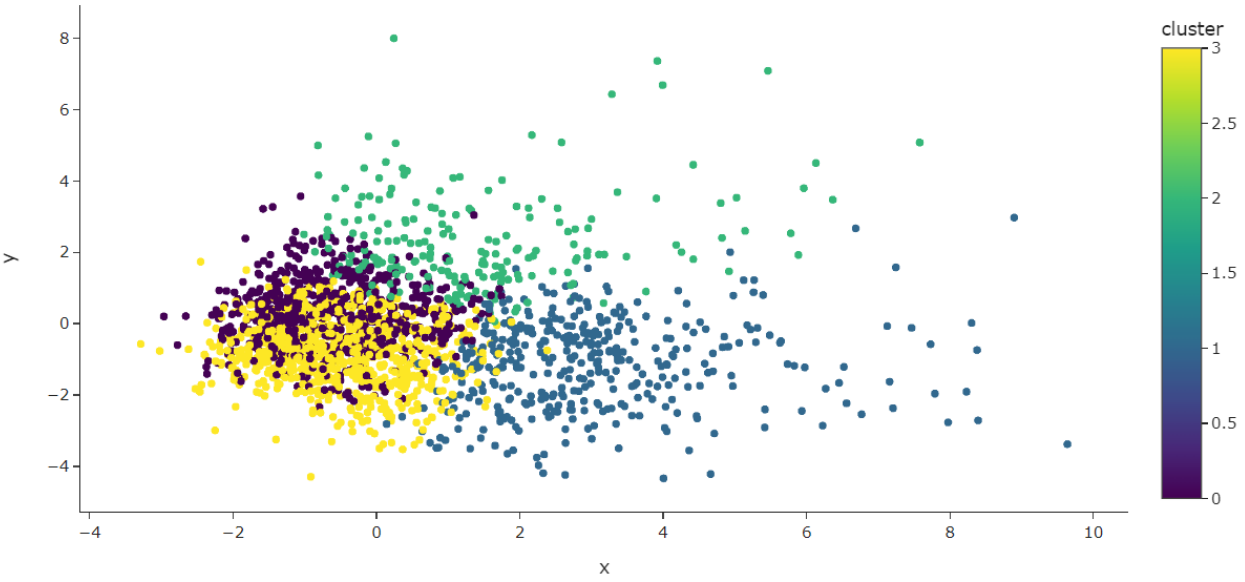
Number of clusters	Validation accuracy
2	0.656
3	0.665
4	0.664
5	0.671
6	0.547
7	0.656
8	0.675

Figure 19 – Clusters

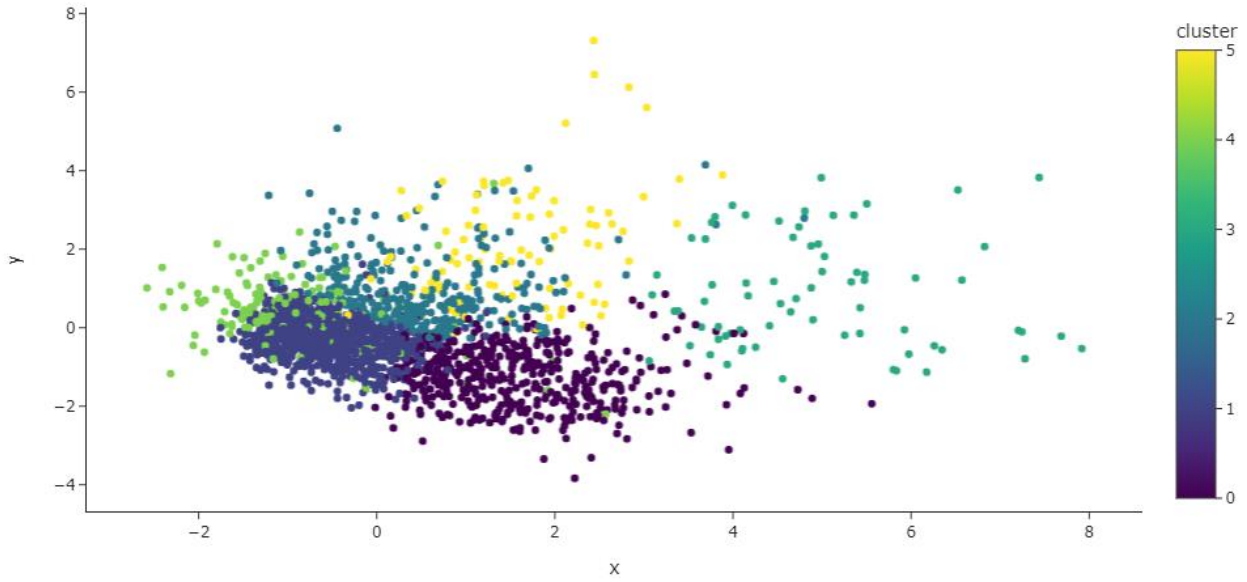
19.1 Standard model, train clusters



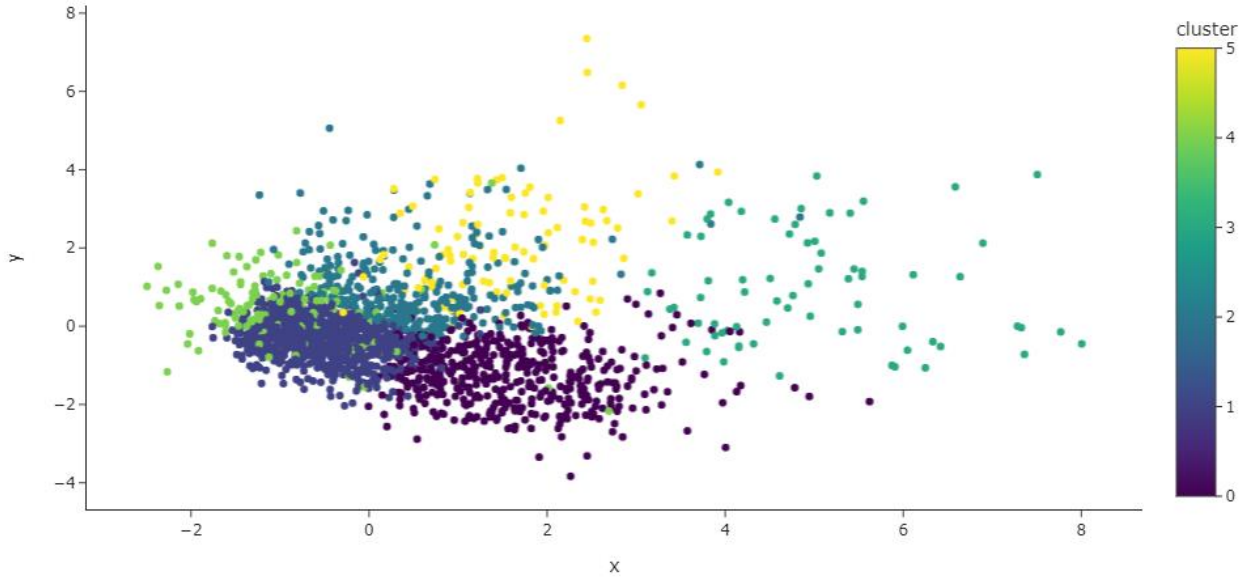
19.2 Standard model, test clusters



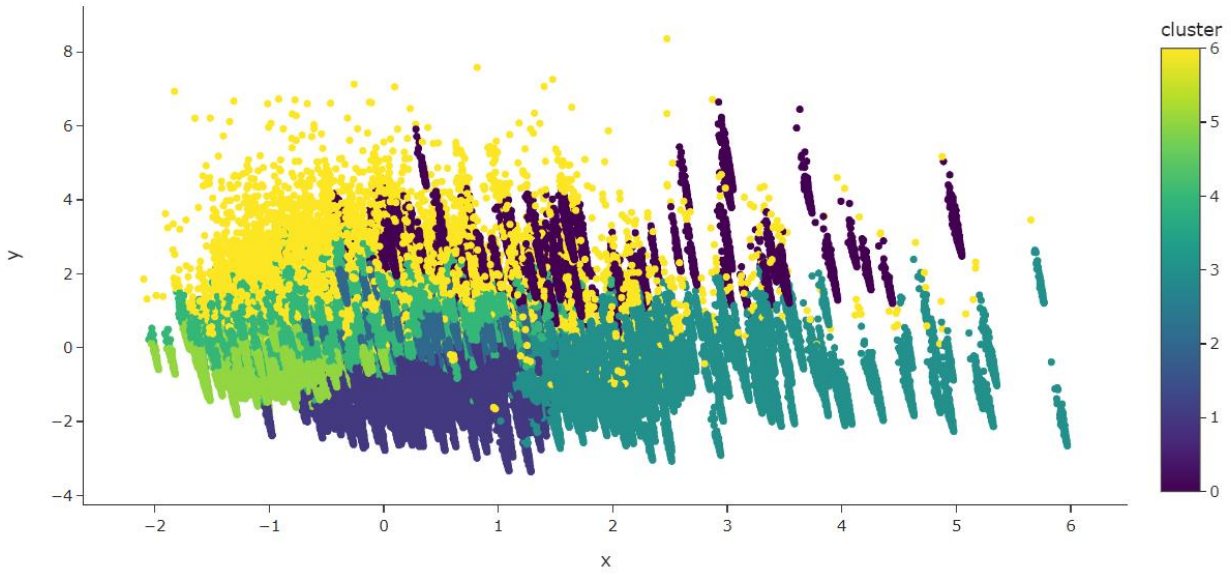
19.3 Feature selection model, train clusters



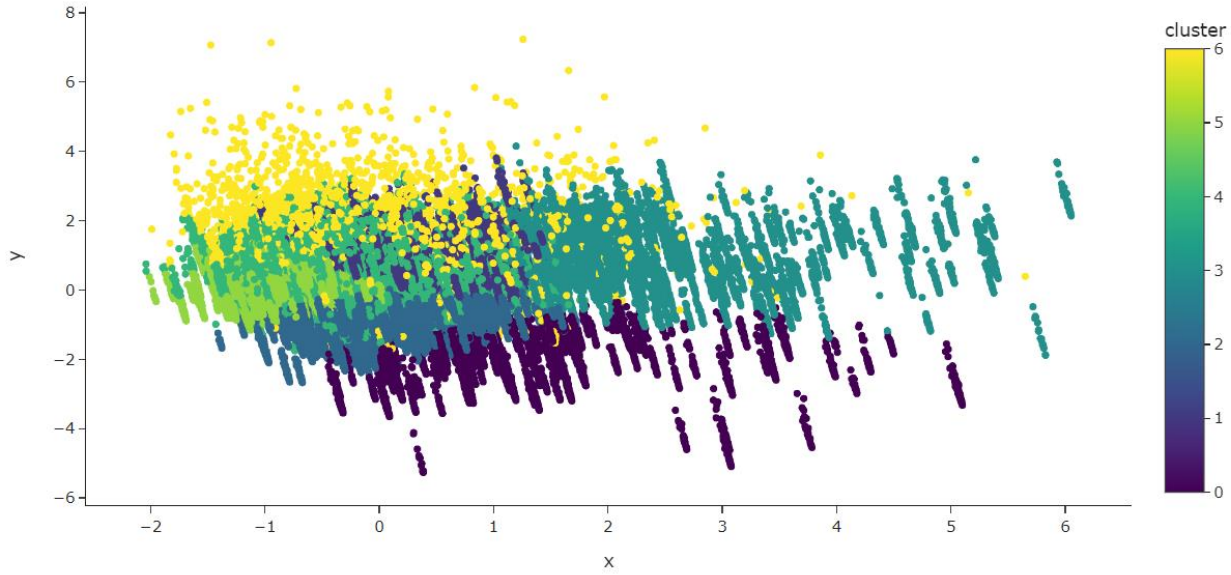
19.4 Feature selection model,, test clusters



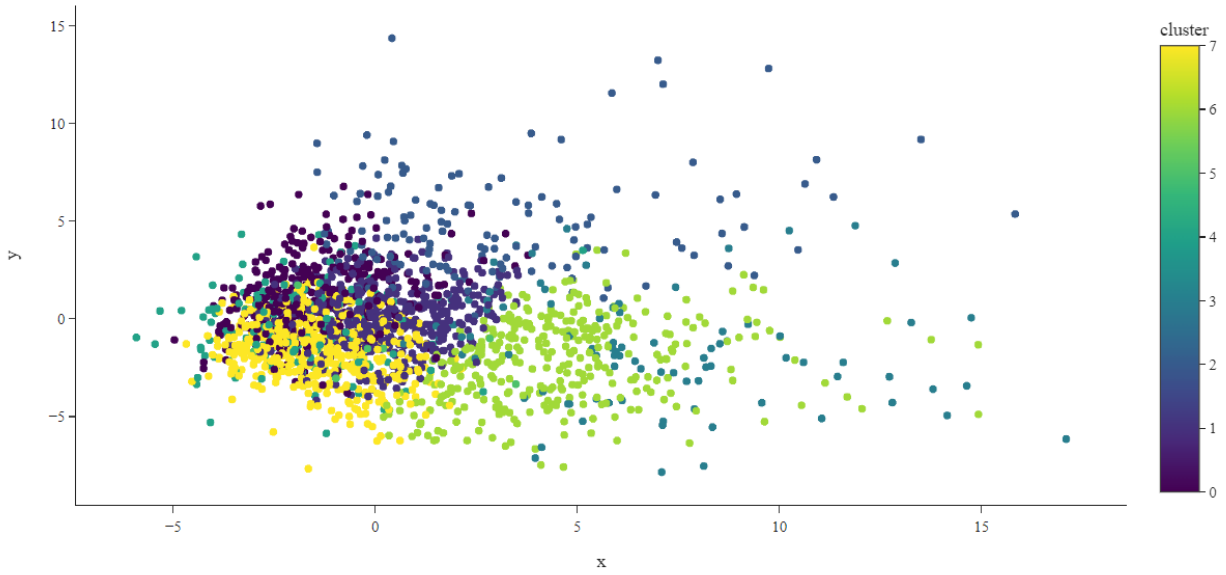
19.5 Dimensionality reduction model, train clusters



19.6 Dimensionality reduction model, test clusters



19.7 Data imputation model, train clusters



19.8 Data imputation model, test clusters

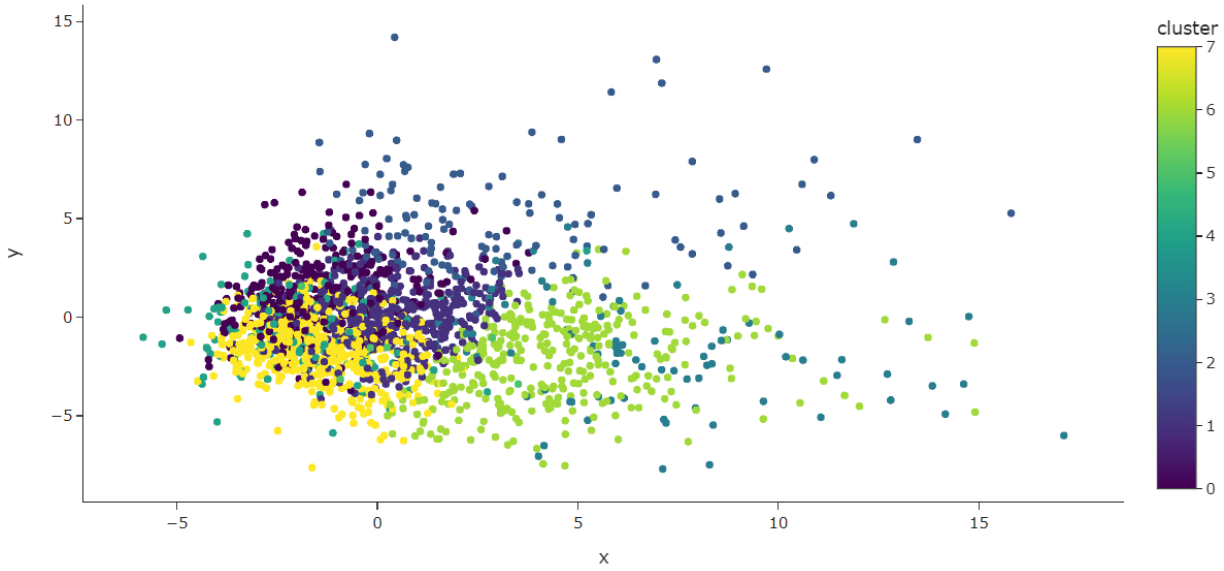


Figure 20 – Confusion matrix per model per cluster

20.1 – Standard model

20.1.1 Cluster 1

	Predicted 0	**Predicted 1**
Actual 0	33,826	10,285
Actual 1	6,678	12,743

20.1.2 Cluster 2

	Predicted 0	**Predicted 1**
Actual 0	11,556	2,132
Actual 1	1,965	3,960

20.1.3 Cluster 3

	Predicted 0	**Predicted 1**
Actual 0	9,927	1,622
Actual 1	1,565	3,534

20.1.4 Cluster 4

	Predicted 0	**Predicted 1**
Actual 0	34,918	9,843
Actual 1	6,807	11,730

20.2 – Feature selection model

20.2.1 Cluster 1

	Predicted 0	**Predicted 1**
Actual 0	15,374	2,861
Actual 1	2,435	5,482

20.2.2 Cluster 2

	Predicted 0	**Predicted 1**
Actual 0	19,717	4,043
Actual 1	3,300	6,497

20.2.3 Cluster 3

	Predicted 0	**Predicted 1**
Actual 0	7,595	946
Actual 1	1,122	2,369

20.2.4 Cluster 4

	Predicted 0	**Predicted 1**
Actual 0	12,887	2,471
Actual 1	2,232	4,452

20.2.5 Cluster 5

	Predicted 0	**Predicted 1**
Actual 0	4,316	1,349
Actual 1	362	2,202

20.2.6 Cluster 6

	Predicted 0	**Predicted 1**
Actual 0	32,697	9,856
Actual 1	6,528	11,997

20.3 – Dimensionality reduction model

20.2.1 Cluster 1

	Predicted 0	**Predicted 1**
Actual 0	13,501	2,558
Actual 1	2,314	4,312

20.3.2 Cluster 2

	Predicted 0	**Predicted 1**
Actual 0	6,340	1,775
Actual 1	1,706	1,125

20.3.3 Cluster 3

	Predicted 0	**Predicted 1**
Actual 0	26,734	6,118
Actual 1	4,905	8,163

20.3.4 Cluster 4

	Predicted 0	**Predicted 1**
Actual 0	12,280	1,750
Actual 1	1,783	3,713

20.3.5 Cluster 5

	Predicted 0	**Predicted 1**
Actual 0	6,595	789
Actual 1	996	2,172

20.3.6 Cluster 6

	Predicted 0	**Predicted 1**
Actual 0	27,429	6,280
Actual 1	5,021	8,530

20.3.7 Cluster 7

	Predicted 0	**Predicted 1**
Actual 0	1,056	905
Actual 1	319	3,922

20.3 – Data imputation model

20.2.1 Cluster 1

	Predicted 0	**Predicted 1**
Actual 0	19,574	4,666
Actual 1	3,391	7,339

20.3.2 Cluster 2

	Predicted 0	**Predicted 1**
Actual 0	179,461	93,389
Actual 1	46,274	71,295

20.3.3 Cluster 3

	Predicted 0	**Predicted 1**
Actual 0	5,330	1,623
Actual 1	475	2,672

20.3.4 Cluster 4

	Predicted 0	**Predicted 1**
Actual 0	3,492	220
Actual 1	450	1,082

20.3.5 Cluster 5

	Predicted 0	**Predicted 1**
Actual 0	7,415	860
Actual 1	1,018	2,325

20.3.6 Cluster 6

	Predicted 0	**Predicted 1**
Actual 0	1,467	17
Actual 1	66	556

20.3.7 Cluster 7

	Predicted 0	**Predicted 1**
Actual 0	10,394	1,794
Actual 1	1,722	3,604

20.3.8 Cluster 8

	Predicted 0	**Predicted 1**
Actual 0	24,689	5,694
Actual 1	4,409	8,095

Appendix II

Code can be found on https://github.com/tessaroes/SpotifyRecommenderSystem.