**Genuine Genres:**

**An Exploration of Feature Selection for Cluster Analysis Through the Medium of Song Audio Features**



Machine Learning (CS 539) Spring 2023

Professor Ricky Sethi

Author

Dylan Phillips,

B.S. Computer Science, M.S. Data Science

# **Abstract**

Through the development of a new pre-processing step for cluster analysis and the results of test and control models, this project validates the proposed *clusterability* metric to be used for feature selection. This project goes through the several data collection and reduction steps that worked to create a new dataset containing advanced Spotify audio features over 180,000 songs across 1,200 unique genres, as well as the analysis of that dataset which verifies the improvement in models with clusterability-selected features. The clusterability metric is similar to the Calinski Harabasz index (Calinski & Harabasz, 1974) in that it measures intra-variance to inter-variance of groups; however, because it is used in the preprocessing stage, it is a novel application of the concept. The base clustering model of choice was k-nearest neighbors. The test model was fitted with only those features achieving a clusterability score above 1.0, and the control was fitted with all features. The main evaluation metrics used to compare models were the Calinski Harabasz index, silhouette score, the Davies Bouldin index, and normalized mutual information. The outcome of the various tests on each model showed that out of 14 test scenarios with significant differences in performance, the clusterability feature-selected model outperformed the control model in 13 of them. With these results the outcome of this paper is that there is hopefully more research done to either validate this preprocessing step, or to specify in which scenarios it performs best.

***Keywords*:** machine learning, k-nearest neighbors, clustering, dimensionality reduction, Spotify audio features

# 1. Introduction

## 1.1 Motivation

My motivation for this project started with my passion for music, and in particular, my passion for Spotify. I have always enjoyed the musical experience that Spotify provides, and ever since my first computer science class in high school, I have been curious about how they create some of their automated processes. Some specific examples I have been curious about are: Spotify’s shuffle algorithm, song recommendations & automatically curated playlists, and how they determine genres for songs. According to a 2018 article by Paper Magazine, Spotify had over 5,071 unique genres, and that number has likely increased over the past few years. According to a Spotify article containing an interview with their self-proclaimed “Data Alchemist,” Glenn McDonald–– who is the lead creator behind their wacky genres, like technosynth, candy pop, etc.––he categorizes songs using “subjective psychoacoustic attributes” (Johnston, 2018). After understanding some of Spotify’s audio features that they store for every track, it is clear that Spotify works with outside-the-box metrics. Some examples include danceability, instrumentalness, and speechiness. Once I learned about all of these unique attributes, I formed a technical question… Can I train a clustering agent to predict Spotify genre based on audio features?

## 1.2 Aims

Once I formed my technical question, I began getting my hands dirty by working with the data. In Google Colab, I loaded, cleaned and pre-processed the music-genres-dataset––which I describe more thoroughly in Section 2––to prepare for analysis. In my initial analysis, I was curious to test how good the data would be for clustering, so I created a “clusterability” metric. I will go into further detail in the data source section, but the metric was basically a proportion of between-class variance to within-class variance. The assumption was that the features which had good “clusterability” would help the clustering algorithm distinguish genres more than those with bad “clusterability.” Without knowing it, I created a pre-processing version of the Calinski Harabasz Index (Calinski & Harabasz, 1974). This idea is what motivated me to form my research question… Does a clustering model using feature selection based on the *clusterability* metric perform better than a model which uses all features?

If the test model performs better than the control model, then the strategies presented in this project would prove useful as a *simple* preprocessing step for any clustering model.

## 1.3 Background

There are many state-of-the-art methods of feature selection for k-means clustering, but first, we need to understand the general approach to feature selection. In the 2003 article, “An Introduction to Variable and Feature Selection,” Isabelle Guyon and André Elisseeff present a simple, straightforward checklist, displayed in Table 1.

*Table 1: There are many ways to optimize the feature selection process, but this is a simple checklist to optimize the outcome*

| Question | Procedure |
| --- | --- |
| 1. Do you have domain knowledge? | If yes, construct a better set of “ad hoc” features. |
| 2. Are your features commensurate? | If no, consider normalizing them. |
| 3. Do you suspect interdependence of features? | If yes, expand your features by constructing conjunctive features or products of features, as much as your computer resources allow you. |
| 4. Do you need to prune the input variables (e.g., for cost, speed or data understanding reasons)? | If no, construct disjunctive features or weighted sums of features (e.g., by clustering or matrix factorization). |
| 5. Do you need to assess features individually? (e.g., to understand their influence on the system or because their number is so large that you need to do a first filtering)? | If yes, use a variable ranking method; else, do it anyway to get baseline results. |
| 6. Do you need a predictor? | If no, stop |
| 7. Do you suspect your data is “dirty” (has a few meaningless input patterns and/or noisy outputs or wrong class labels)? | If yes, detect the outlier examples using the top ranking variables obtained in step 5 as representation; check and/or discard them. |
| 8. Do you know what to try first? | If no, use a linear predictor. Use a forward selection method with the “probe” method as a stopping criterion or use the l0-norm embedded method. For comparison, following the ranking of step 5, construct a sequence of predictors of same nature using increasing subsets of features. Can you match or improve performance with a smaller subset? if yes, try a non-linear predictor with that subset. |
| 9. Do you have new ideas, time, computational resources, and enough examples? | If yes, compare several feature selection methods, including your new idea, correlation coefficients, backward selection, and embedded methods. Use linear and non-linear predictors. Select the best approach with model selection. |
| 10. Do you want a stable solution (to improve performance and/or understanding? | If yes, sub-sample your data and redo your analysis for several “bootstraps.” |

This step-by-step approach to feature selection generalizes to just about every dimensionality reduction problem. As you will see in Section 3, we use a few of these techniques in our own testing of our *clusterability* predictor. To get a better understanding for our particular problem, however, we need to look at approaches similar to what we are trying to do in the realm of clustering.

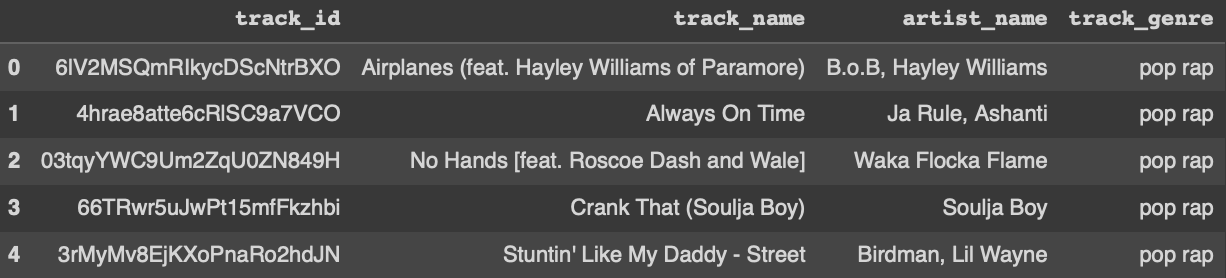
As a response to the plethora of studies on unsupervised feature selection, the 2013 paper, “Deterministic Feature Selection for K-Means Clustering” presents an approach for supervised feature selection. Boutsidis and Magdon-Ismail demonstrate how it is possible to efficiently identify a small set of features that are sufficient for reproducing a given clustering. This is crucial for our problem due to the fact that it is entirely a supervised process..

With an understanding of steps to general feature selection, as well as supervised feature selection for the specific application of clustering, we can get into our methodology.

# 2. Data Source

## 2.1 Raw Data

The music-genres-dataset contains basic information for songs across 1,494 Spotify genres, with each genre containing 200 songs. Each song entry contains artist, song name, position within the list of 200 songs, main genres, sub genres, and tags (Trebichavský, 2017). Figure 1 displays the data loaded into a Python DataFrame.



*Figure 1: music-genres-dataset contains song’s Spotify ID, track name, artist name, and genre*

The interesting thing about this dataset is that genres are not only including your typical musical genres––like pop, country, rock, etc.––but also wacky Spotify genres, which include *focus trance*, *dark techno*, *croatian electronic*, and much more. You can see an organized, two-dimensional map of Spotify’s interesting genres at everynoise.com. Figure 2 displays a small snippet of the map.

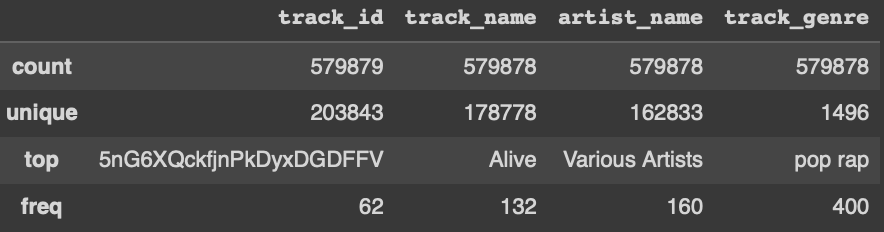


*Figure 2: Spotify’s genres are plentiful, and display relationships to one another*

Clearly, working with a dataset of over 1,400 genres was going to make it difficult to form any kind of accurate predictor. Instead, we can answer another question… if we perform some sort of musical feature extraction, can we find *hidden* groupings of songs––i.e., can we determine a new system for genre classification?

## 2.2 Removing Duplicate Tracks

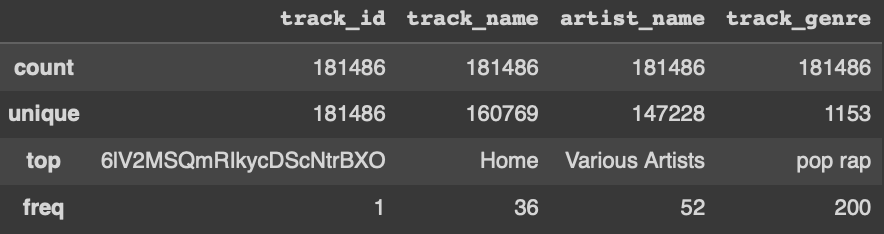
The first problem I ran into with the raw data was the fact that there were duplicate values of *track\_id* (see Figure 3).



*Figure 3: Most of the raw dataset contained duplicate track\_id’s*

The duplicate entries were a problem, because it meant that one song would be associated with multiple genres, and that each genre it was associated with would get trained on the same audio features[[1]](#footnote-0). Because this would upset our clustering performance, I retained only the first song of each duplicate set. After performing that data reduction, I was left with over 200,000 songs, which still covered those same 1,496 genres.

The next step in pre-processing the data was to ensure we had an adequate training size for each genre. Even though my mission with testing the clustering models was to look at clustering metrics not to achieve the highest prediction accuracy, I still wanted to remove under-represented genres to create a healthy dataset. I performed another operation on the set to remove all genres which had less than 100 songs, i.e., the more obscure genres. Figure 4 summarizes our dataset after performing both data reduction steps.



*Figure 4: Post-reduction, our dataset still contains 181,486 songs across 1,153 genres*

Now that we cleaned our dataset, it was ready for the next step… adding audio features!

## 2.3 Adding Audio Features

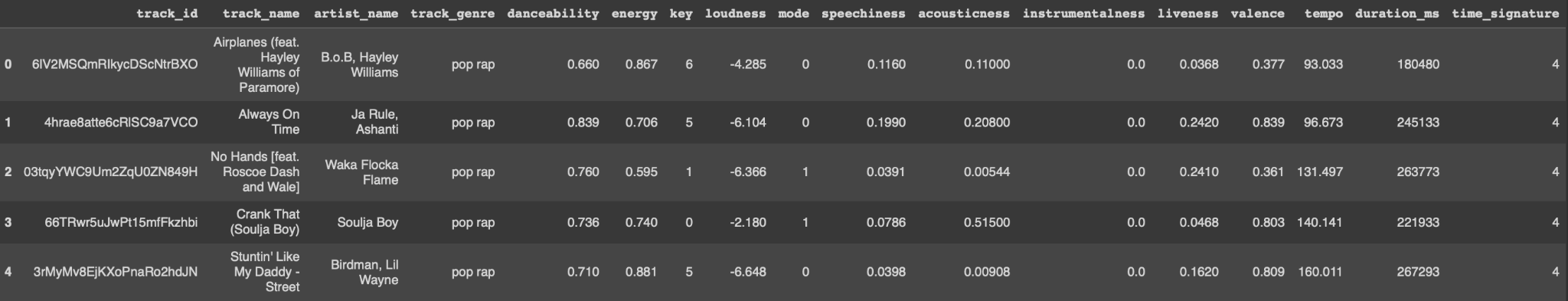
As supported by my background work in Section 1.3, my goal with feature collection was to find simple, yet creative abstractions of musical features. After researching Spotify’s API, I found that the abstraction of musical features I was most curious to test this on was provided by Spotify’s audio features. Table 1 shows a list of all of Spotify’s audio features which can be retrieved through the *get\_audio\_features* API request.

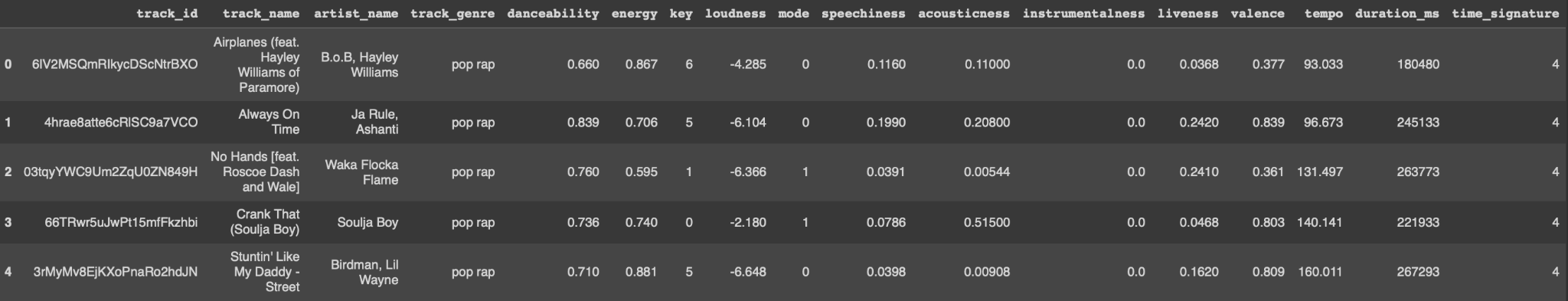
*Table 1: Spotify’s audio features represent a creative, diverse array of song information*

| Feature | Definition |
| --- | --- |
| *danceability* | Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. |
| *energy* | Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. |
| *key* | The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1. |
| *loudness* | The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db. |
| *mode* | Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0. |
| *speechiness* | Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks. |
| *acousticness* | A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic. |
| *instrumentalness* | Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0. |
| *liveness* | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live. |
| *valence* | A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). |
| *tempo* | The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. |
| *duration\_ms* | The duration of the track in milliseconds. |
| *time\_signature* | An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4". |

During my initial discovery, these metrics excited me because they present many dimensions from which to distinctly quantify a song, while also remaining abstract. To support that sentiment, it has been shown that clustering is not straightforward or simple when performed on high-dimensional data with many features, such as music (Guyon et al., 2004). Also, abstraction of data is useful for clustering because it can discard irrelevant data and consolidate into singular features which encompass redundant data (Ismi et al., 2016). I was certainly content with this dataset, as it satisfied my curiosity for Spotify’s interesting *data alchemy* and opened up creative avenues for discovery.

Now, the process to add these audio features to the cleaned dataset was simply to go through the row-by-row, querying audio features by *track\_id* and appending the 13 features to the end of the row. Luckily, connecting to the Spotify API and creating each audio feature request was super easy with the Spotipy library (*Paul Lamere Revision*, 2014), but not so fortunately was the fact that it took about 1.5 minutes for every 1000 songs. After about 4.5 hours, the dataset was complete[[2]](#footnote-1). A preview of the final audio features file is displayed in Figure 5.

•••

•••

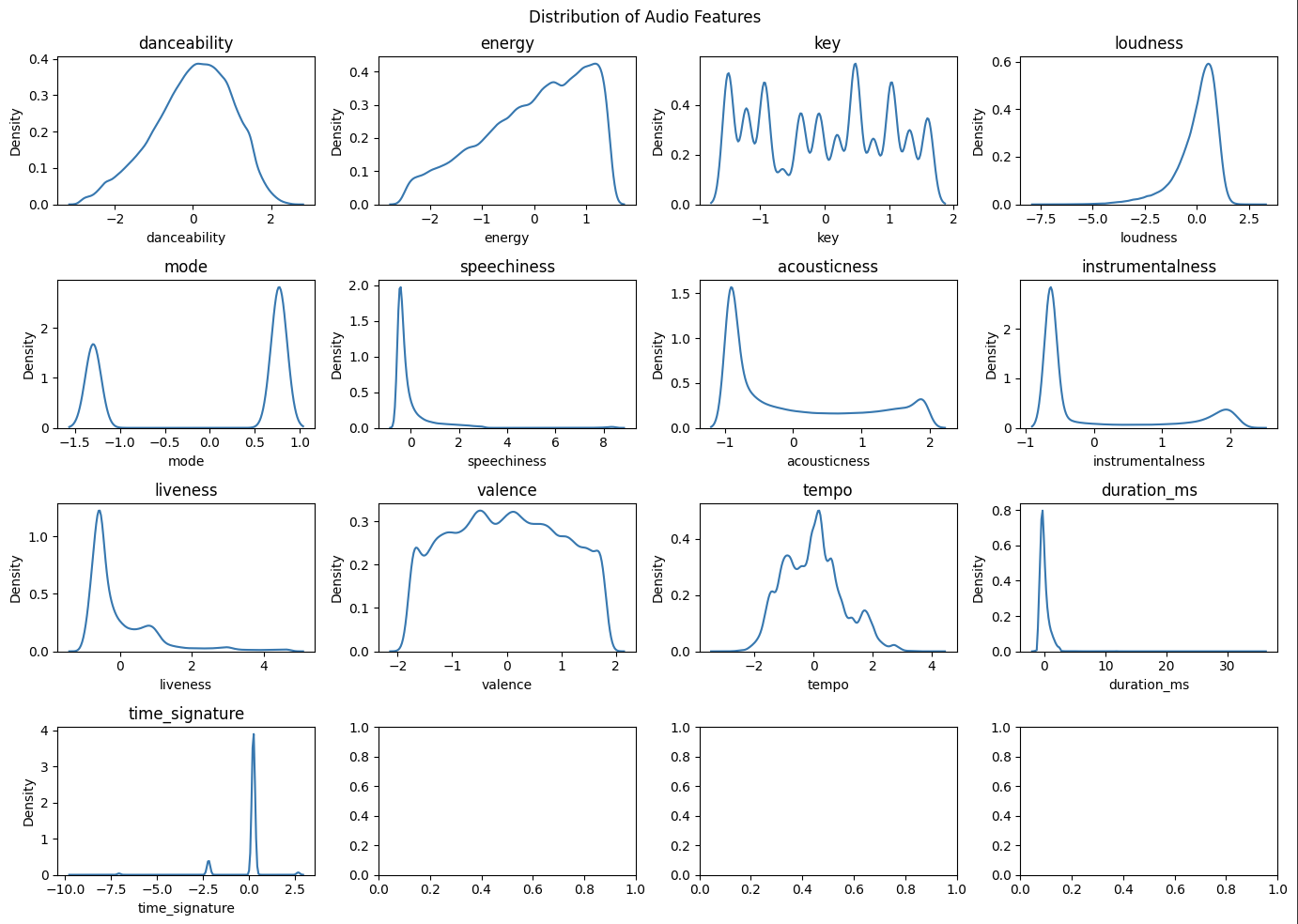
*Figure 5: Audio features dataset has 13 cool features for every song!*

Our final, cleaned dataset gives us Spotify’s full assortment of unique audio features for each of the 180,000+ songs in our dataset. This abstract assortment of musical features will carry us through the rest of our exploration into clustering methodologies, but first…let’s do some exploratory analysis of our data.

## 2.4 Exploratory Data Analysis

My exploratory analysis of the audio features data is the phase which made me change the entire focus of the project, after some imaginative leaps guided by my curiosity. After the initial menial tests regarding data completeness and form (see Figures 6 & 7), my analysis went into visual representations for each of the audio features distributions. An assorted graph of each feature’s kernel density estimation (KDE) plot can be seen in Figure 8. The KDE plot follows the shape of a histogram plot, but uses distortions to smooth out the graph (Wascom, 2022).

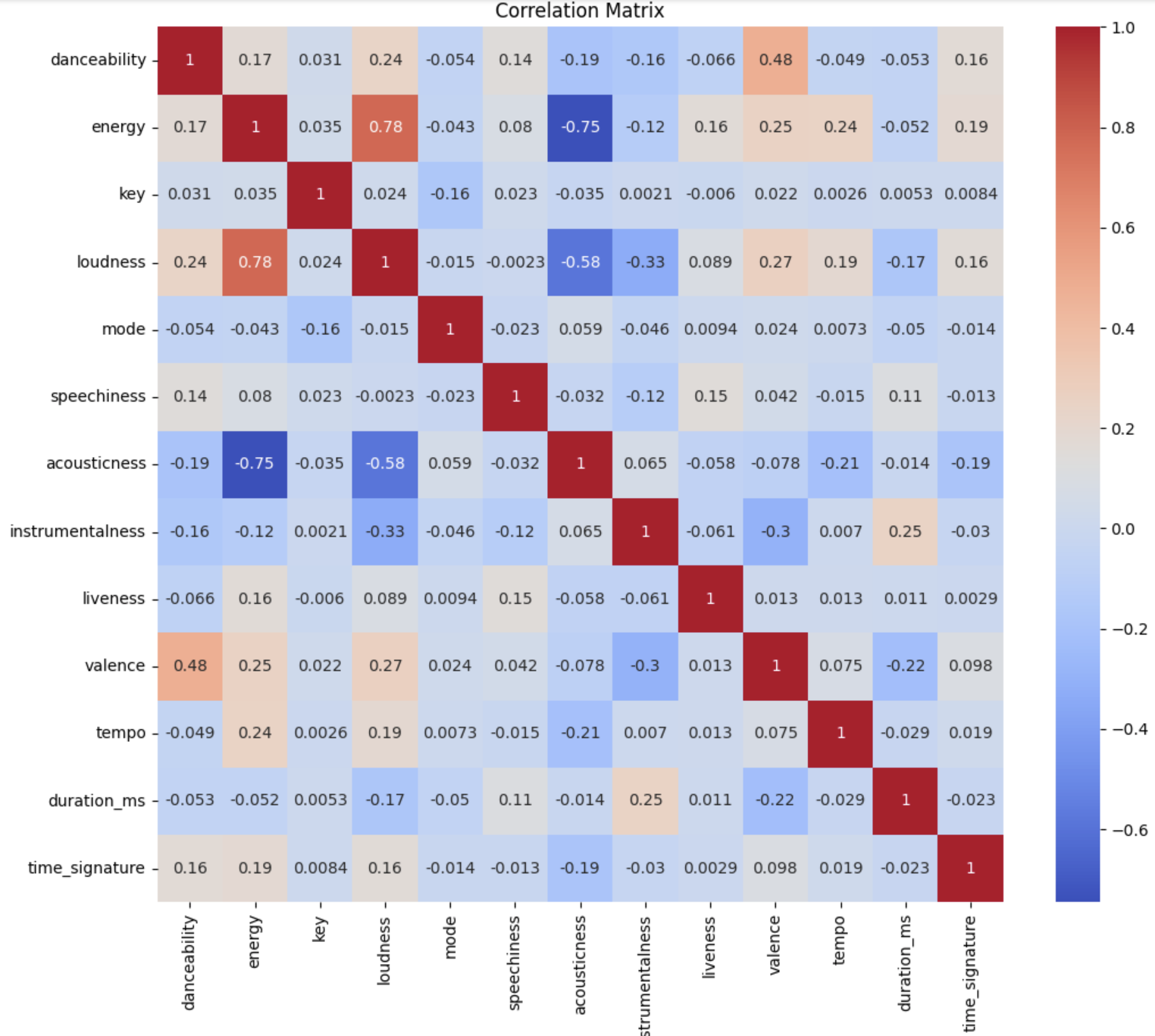
| *Figure 6: No null values in the dataset* | *Figure 7: All features are represented properly in the dataset* |
| --- | --- |



*Figure 8: Audio features display a large variety of distributions*

Within the assortment of KDE plots for audio features, we can see that there is a lot of variation between the shapes of each graph. There are bi-modal & many-modal, left-skewed & right-skewed, bell curves, and plain old weird shapes. The fact that our dataset has a large variety of features will add further dimensions of separation between clusters[[3]](#footnote-2).

Now that we have assessed the shapes of our features, we can look into correlations between them. Figure 9 displays a correlation heatmap.



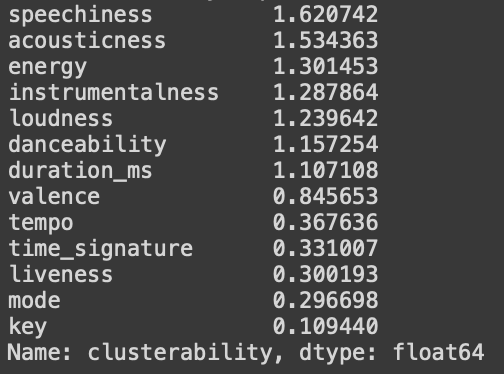
*Figure 9: Data displays strong positive and negative correlations between audio features*

Some of the strongest correlations are trivial. When considering loudness & energy, they should be positively linked; the same goes for valence (happiness rating) & danceability. Some less obvious correlations are between acousticness & energy, as well as acousticness & loudness. These are two relationships I did not consider beforehand, but it makes sense that acoustic songs––i.e., songs produced without electric instruments––are quieter and therefore, less energetic. The main point of analysis for this data, however, is that there is a fair amount of multicollinearity between audio features. It is well known that multicollinearity dampens any significance of outcomes in regressions, but it can also cause unstable and inconsistent results in cluster analysis (Kaufman & Rousseeuw, 2009). We can keep that in mind as we weigh our final results.

The next section for our data analysis is grouping and aggregating the data by genre. During this analysis, I was curious to see if I could predict which genres would be most helpful to our cluster analysis. I initially did not intend to create a metric, but stumbled into it nonetheless.

I first wanted to see within-genre standard deviation across all features to test their “closeness of cluster” capability. Figure 10 displays the results from that query. Then, I naturally decided to see between-genre standard deviation across all features to test their “separation of clusters” capability, which is displayed in Figure 11. Finally, my mind went to the natural step of combining these features into a single clusterability metric, which put value on separation of clusters as it related to closeness of cluster for each audio feature. This final series displayed in Figure 12 shows the *clusterability score* for each audio feature, ranked highest to lowest.

| *Figure 10: Between-genre deviation* | *Figure 11: Within-genre deviation* |
| --- | --- |



*Figure 12: Clusterability score, the ratio of between-to-within deviation*

It was at this point I had an *aha!* moment. I stumbled onto something that seemed quite revolutionary to me personally, when it turns out it is very similar to the existing metric for cluster analysis, the Calinski Harabasz index (Calinski & Harabasz, 1974). What I did discover though, was that I could take this project in a completely new direction. I initially intended on tuning other clustering parameters within the model to optimize results for musical genre determination, but now, I had a fresh take. I decided to run with it…and onto the methodology I went!

# 3. Methodology

## 3.1 Dimensionality Reduction

After discovering the clusterability metric, my goal was now to either validate or disregard the metric as an efficient pre-processing step to cluster analysis. With that goal in mind, the next natural step for my project was to create a test model and a control model, and compare their clustering scores.

To split up the audio features, I made a cutoff at the 1.0 clusterability score[[4]](#footnote-3). Table 2 displays how the audio features were split into two models.

*Table 2: 7 of the 13 features are selected for the feature-reduced model*

| Feature-Reduced Model | Clusterability | Control model |
| --- | --- | --- |
| speechiness | 1.62 | speechiness |
| acousticness | 1.53 | acousticness |
| energy | 1.30 | energy |
| instrumentalness | 1.29 | instrumentalness |
| loudness | 1.24 | loudness |
| danceability | 1.16 | danceability |
| duration\_ms | 1.11 | duration\_ms |
|  | 0.85 | valence |
|  | 0.37 | tempo |
|  | 0.33 | time\_signature |
|  | 0.30 | liveness |
|  | 0.30 | mode |
|  | 0.11 | key |

Once we split up the features into two groups––the feature-reduced model and control model––it was simply a matter of training and testing each on multiple input parameters. On to the training!

## 3.2 Training the Model

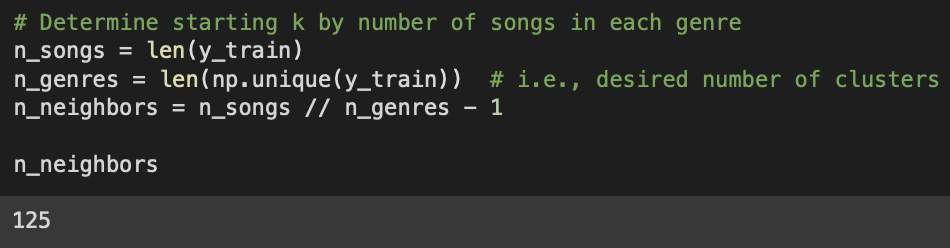
### 3.2.1 Train-Test Split

Once I collected both groups of data, the first step was to split each group into training and testing sets. I devoted 20% of the data in each group to testing––which is a standard amount––for a couple reasons. First, I wanted to be able to get some sense of our model’s prediction accuracy, because I was curious how it would perform at distinguishing between over 1,200 genres with our abstracted data. Second, because of our original data source and our preprocessing data filtering techniques, we still had over 180,000 lines of data to work with. For those reasons, I was not concerned that devoting 20% of the data to test accuracy would be a burden on the cluster analysis, even though it was tangential to my overall goal.

### 3.2.2 Initializing the k-NN Models

For the clustering models, I decided to use the k-nearest neighbors model. The basic description for k-nearest neighbors is a clustering model which does not store internal weights, but instead stores training data points, and calculates for every one it’s *k* nearest neighbors to classify the entire set into clusters, where *n* is the total number of datapoints. Every test data point is then assigned to the cluster which has the most representation out of its *k* nearest neighbors (*scikit-learn developers*, 2023). There are many methods for music genre prediction that are far superior to a nearest neighbors classifier (Prockup et al., 2015), (Sturm, 2014), but within our project scope of clustering, k-nearest neighbors is a perfectly standard foundation for comparison.

With the data now split into train and test sets, we had to decide the main hyperparameter for our models: *k*. To determine *k*, I simply calculated the quotient of the number of songs to the number of genres represented in the training set. This calculation and result is displayed in Figure 13.



*Figure 13: Each cluster (i.e., genre) should be expected to have 125 nearest neighbors*

With this base number for our *k*, we could build the models, which was very easy thanks to the scikit-learn library. Now onto testing our models!

## 3.3 Testing the Models

To test the models, we used a variety of *k* values for both the control and feature-selected model. We used the base value of 125, but also high and low outlying values to see their effects. The metrics we tested for were the baseline metrics for model prediction accuracy––e.g., basic accuracy, precision, recall, and f1 score––but also clustering metrics, which are listed and described in Table 3.

*Table 3: Four highly advanced measures for cluster performance*

| Metric | Description | Equation |
| --- | --- | --- |
| Calinski Harabasz Index | A clustering performance metric that measures the ratio of between-cluster variance to within-cluster variance. Higher values indicate better-defined and more separated clusters (Calinski & Harabasz, 1974) | Where B is the between-cluster sum of squares and W is the within-cluster sum of squares |
| Silhouette Score | A clustering performance metric that measures how well each data point fits its assigned cluster relative to other clusters. Higher values indicate well-separated clusters with distinct data points (Rousseeuw, 1974). | Where ai is the average distance between a data point i and all other points in its cluster  and bi is the average distance between a data point i and all other points in the nearest cluster that i is not a member of |
| Davies Bouldin Index | A clustering performance metric that measures the average similarity between each cluster and its most similar cluster, relative to the average dissimilarity between the cluster and its least similar cluster. Lower values indicate more compact and well-separated clusters (Davies & Bouldin, 1979). | and  Where di is the average distance between data point i and all other points in its cluster and d(cij) is the distance between the centroids of clusters i and j. The Davies Bouldin Index is the average of the maximum Rij values across all pairs of clusters. |
| Normalized Mutual Information | A clustering performance metric that measures the mutual information between the true labels and the predicted labels of the clustering algorithm, normalized by the entropy of the true labels. Higher values indicate better agreement between the true and predicted labels (Strehl & Ghosh, 2002), (Vinh et al., 2010). | Where Y is the vector of true labels, C is the vector of predicted labels, IYC is the mutual information between Y and C, HY is the entropy of Y, and HC is the entropy of C |

With these four advanced measures for clustering performance, we will have plenty of significance to our tests. Now for the results!

# 4. Analysis

## 4.1 Results

With the understanding of these four metrics, let’s take a look at Table 4, which displays the final results of our control and high-clusterability feature-reduced models after various tests. It is important to note that I decided to include the test results from *before* fixing the problem of duplicate *track\_id*’s, because I believe the effects of the clusterability strategy, even with that duplicate data, shows evidence of improvement over the control.

*Table 4: In most settings, the test model outperforms the control!*

| Model  (ctl / rdc)  *k*-neighbors | | Accuracy | Precision | Recall | F1 Score | Calinski Harabasz Index | Silhouette Score | Davies Bouldin Index | Normalized Mutual Information Score |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Over-Saturated Data Model | | | | | | | | | |
| k=3 | ctl | 0.11057 | 0.09402 | 0.11057 | 0.09849 | 62.7623 | -0.6160 | 3115.80 | 0.52988 |
| rdc | 0.11158 | 0.09520 | 0.11158 | 0.09958 | 65.7527 | -0.6142 | 4276.18 | 0.5303 |
| Note: % increase from control to test: | | | | | | +4.76% | +0.29% | +37.2% | +0.08% |
| k=319 | ctl | 0.00765 | 0.00717 | 0.00765 | 0.00588 | 466.191 | -0.8547 | 893.614 | 0.37139 |
| rdc | 0.00772 | 0.00629 | 0.00772 | 0.00580 | 490.223 | -0.8401 | 1350.40 | 0.37094 |
| Note: % increase from control to test: | | | | | | +5.15% | +1.71% | +51.1% | -0.12% |
| Clean Data Model | | | | | | | | | |
| k=3 | ctl | 0.00465 | 0.00438 | 0.00445 | 0.00387 | 35.0210 | -0.9469 | 2171.27 | 0.46371 |
| rdc | 0.00432 | 0.00427 | 0.00415 | 0.00374 | 37.2627 | -0.9509 | 3409.03 | 0.46433 |
| Note: % increase from control to test: | | | | | | +6.40% | -0.42% | +57.0% | +0.13% |
| k=125 | ctl | 0.00523 | 0.00276 | 0.00488 | 0.00315 | 191.069 | -0.8494 | 884.782 | 0.44139 |
| rdc | 0.00524 | 0.00280 | 0.00489 | 0.00316 | 196.768 | -0.8350 | 1157.80 | 0.44113 |
| Note: % increase from control to test: | | | | | | +2.98% | +1.70% | +30.9% | -0.06% |
| k=319 | ctl | 0.04659 | 0.03500 | 0.04251 | 0.02956 | 61.6071 | -0.2324 | 3.39469 | 0.47705 |
| rdc | 0.04540 | 0.03161 | 0.04135 | 0.02799 | 215.892 | -0.2237 | 3.17726 | 0.47705 |
| Note: % increase from control to test: | | | | | | +250% | +3.74% | -6.40% | +0.00% |
| Note: Highlighted values represent combinations where the high-*clusterability* feature-selected model outperformed the control model | | | | | | | | | |

## 4.2 Discussion

Aside from the fact that we managed to create an extremely poor genre predictor, the differences in cluster metrics is amazing! In 15 out of 20 total test scenarios, the models which selected high-clusterability features had more efficient clustering than their control model! It is important to notice the areas where it did not outperform; we will focus on the clean data model sections. For *k=3*, clusterability features produced a worse silhouette score. This is particularly interesting when looking down the column to results from higher *k* values, where we see that silhouette score improves not only overall between control and reduced models, but the difference *between* reduced and control also improves with each increase in cluster size. From *k=3*, there is a 0.422% decrease in silhouette score from control to reduced, then for *k=125*, there is a 1.70% increase, and for *k=319*, there is a 3.74% increase from control to reduced model. This suggests two things; first, this model has better tightness within and distinction between clusters at higher *k*-values; and second, the clusterability-feature selection creates tighter and more distinct clusters with larger *k*-values. This pattern is interesting within this dataset, but I would suggest future research look into generalizing that concept.

The same pattern can be observed in the Calinski Harabasz Index, with a slight hiccup at the *k=125* range. With 6.40% (*k=3*), 2.98% (*k=125*), and an astounding 250% increase in reduced over control with the *k=319* models. When looking at the Davies Bouldin index, we see the inverse effect, with 57.0%, 30.9%, and -6.40% for *k=3*, *125*, and *319*, respectively. Finally, for the normalized mutual information score, we see results with no significant difference between control and clusterability selected features.

Overall, while there are shifting relationships among metrics with increasing *k*-values, I believe the more significant analysis is that 13 out of 14 of the significant (i.e., > 1%) differences between models favored the clusterability-selected model over the control model.

# Conclusions and Further Study

After performing analysis across several base set-ups, the results within this musical audio features dataset show that the preprocessing step of feature selection based on within-cluster variance and between-cluster variance––A.K.A. *clusterability*––created a model that performed better according to advanced clustering metrics. Specifically, this model performed best when the *k*-value, i.e., number of nearest neighbors in the clustering model was selected appropriately to the dataset’s outcome variable of genre (performed best at *k=125*).

I would suggest that further studies carry out these same procedures on various datasets to determine the validity of such a preprocessing step, and determine which scenarios it works best on.

# References

Boutsidis, C., Drineas, P., & Mahoney, M. W. (2009). Unsupervised feature selection for the k-means clustering problem. *Advances in neural information processing systems*, *22*. https://proceedings.neurips.cc/paper/2009/hash/c51ce410c124a10e0db5e4b97fc2af39-Abstract.html

Boutsidis, C., & Magdon-Ismail, M. (2013). Deterministic feature selection for k-means clustering. *IEEE Transactions on Information Theory*, *59*(9), 6099-6110. https://ieeexplore.ieee.org/abstract/document/6488848/

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, *3*(1), 1-27. https://www.tandfonline.com/doi/abs/10.1080/03610927408827101

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, (2), 224-227. https://ieeexplore.ieee.org/abstract/document/4766909

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157-1182. https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf?ref=driverlayer.com/web

Guyon, I., Gunn, S., Ben-Hur, A., & Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Advances in neural information processing systems*, *17*. https://proceedings.neurips.cc/paper\_files/paper/2004/hash/5e751896e527c862bf67251a474b3819-Abstract.html

Ismi, D. P., Panchoo, S., & Murinto, M. (2016). K-means clustering based filter feature selection on high dimensional data. *International Journal of Advances in Intelligent Informatics*, *2*(1), 38-45. http://www.ijain.org/index.php/IJAIN/article/view/54

Johnston, M. (2018). How Spotify discovers the genres of tomorrow. Spotify for Artists, *Stories*. https://artists.spotify.com/blog/how-spotify-discovers-the-genres-of-tomorrow

Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis, p 68. John Wiley & Sons.

Paul Lamere Revision. (2014). Welcome to Spotipy! *spotipy 2.0 documentation*. https://spotipy.readthedocs.io/en/2.22.1/

Prockup, M., Ehmann, A. F., Gouyon, F., Schmidt, E. M., Celma, O., & Kim, Y. E. (2015, October). Modeling Genre with the Music Genome Project: Comparing Human-Labeled Attributes and Audio Features. In *ISMIR* (pp. 31-37). https://archives.ismir.net/ismir2015/paper/000276.pdf

Rodgers, K. (2018). Since when was ‘escape room’ a genre? PAPER magazine, *break the internet*. https://www.papermag.com/spotify-wrapped-music-genres-escape-room-2649122474.html#rebelltitem21

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, 53-65. https://www.sciencedirect.com/science/article/pii/0377042787901257

scikit-learn developers. (2023). 1.6 Nearest neighbors, *1.6.2 Nearest neighbors classification*. https://scikit-learn.org/stable/modules/neighbors.html#classification

Spotify. (n.d.). Get Track’s Audio Features. Web API Reference, *Spotify for Developers*. https://developer.spotify.com/documentation/web-api/reference/get-audio-features

Sturm, B. L. (2014). The state of the art ten years after a state of the art: Future research in music information retrieval. *Journal of new music research*, *43*(2), 147-172. https://www.tandfonline.com/doi/full/10.1080/09298215.2014.894533

Strehl, A., & Ghosh, J. (2002). Cluster ensembles – a knowledge reuse framework for combining multiple partitions. Journal of machine learning research, 3(Dec), 583-617. https://www.jmlr.org/papers/volume3/strehl02a/strehl02a.pdf

Trebichavský, R. (2017). music-genres-dataset. Github Repository. https://github.com/trebi/music-genres-dataset

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. Journal of Machine Learning Research, 11(Oct), 2837-2854.

Wascom, M. (2022). seaborn.kdeplot. *seaborn 0.12.2 documentation*. https://seaborn.pydata.org/generated/seaborn.kdeplot.html

1. I only noticed the duplicate *track\_id* problem after performing the entire methodology; to see comparative model results between over-saturated data and clean data, check out Table 4 [↑](#footnote-ref-0)
2. It took 4.5 hours to add audio features to the *cleaned* dataset. When I added audio features to the original dataset of 500,000+ songs, it took 12.5 hours! I was able to run this overnight…but still! [↑](#footnote-ref-1)
3. I could not find any viable sources on the effect that feature distribution variety has on models. I am curious whether this is founded or not; perhaps this paper can contribute to that finding as well. [↑](#footnote-ref-2)
4. The clusterability score cut-off of 1.0 might be an arbitrary cutoff point. I am not sure what should be the ideal ratio of between-within cluster variance; however, it worked out nicely due to the proportion of passing to non-passing features, so I ran with it. [↑](#footnote-ref-3)