Exploring the predictive power of musical features via Spotify

Group 12: Ethan Hodys, Vincent Chiang

*Abstract*: Founded in 2006, Spotify’s primary business is providing an audio streaming platform, the "Spotify" platform, that provides DRM-restricted music, videos and podcasts from record labels and media companies (Wikipedia 2008). This paper is an exploratory survey of whether musical features and characteristics can be used predict how a song will be both received and classified by Spotify users. Using different methods for Classification Trees and Clustering in R, we make a case for the feasibility of prediction using a song’s meta data stored by Spotify.

**Contents**

[1 Introduction 2](#_Toc38834544)

[1.1 Histogram Plot Matrix 3](#_Toc38834545)

[1.2 Correlation Matrix 4](#_Toc38834546)

[2 Data Dictionary 5](#_Toc38834547)

[3 Methodology and results 7](#_Toc38834548)

[3.1 Classification Trees 7](#_Toc38834549)

[Histograms 7](#_Toc38834550)

[Analysis of Danceability - RPART 8](#_Toc38834551)

[Analysis of Danceability - CTREE 11](#_Toc38834552)

[Analysis of Popularity - RPART 12](#_Toc38834553)

[Analysis of Popularity - CTREE 13](#_Toc38834554)

[3.2 Clustering 14](#_Toc38834555)

[4 Conclusion 25](#_Toc38834556)

[5 Bibliography 26](#_Toc38834557)

[APPENDIX A – Additional Charts 26](#_Toc38834558)

# Introduction

Music can have a powerful effect upon people emotionally and has played an important role in since the first human society. Thus, being able to classify songs based on their musical features could prove to be a useful tool across many sectors. While this paper will in no way be an exhaustive survey of the topic it will provide an initial exploratory analysis to help guide future research

We have chosen to use the “Spotify Song Attributes” data set on Kaggle uploaded by George McIntire under a CC-BY License. Below is a list of the variables in the dataset with their distributions except for “track id” which is simply a unique identifier for a song (see appendix for details on the dataset).

We also added information on the biological sex of the performers using lists of popular names split between boy in girl. We import the data into MS Sql Server© to add the correct value for the new variable “isMale” which is a simple bit variable of 0/1 for female/male, respectively. From our survey of the data and personal interests we identified 2 response variables of interest to use for our analysis: [1] Danceability and [2] Popularity

## Histogram Plot Matrix

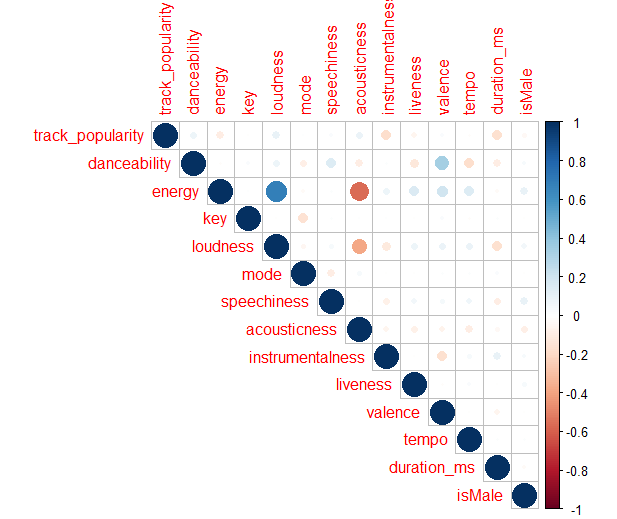
Looking at the matrix of histogram plots of the variables in the dataset shows some interesteing variety. Some distributions look to be close opposits   
of each other like “accouticness” and “loudness” which indicates a negative correlation. While we can see a positive potential correlation between danceability, energy, and loudness.

A close up of a map

Description automatically generated

## Correlation Matrix

It is interesting to not that there is not a lot of correlation withing this dataset. We see only a handful of large circles below and those are evenly split between positive and negative correlations. Unfortunately we don’t see many strong correlations between out variables of interest: danceability and popularity. This indicates that the decision tree algorithms may prove to not generate much useful information.



# 

# Data Dictionary

| **variable** | **class** | **description** |
| --- | --- | --- |
| track\_id | character | Song unique ID |
| track\_name | character | Song Name |
| track\_artist | character | Song Artist |
| track\_popularity | double | Song Popularity (0-100) where higher is better |
| track\_album\_id | character | Album unique ID |
| track\_album\_name | character | Song album name |
| track\_album\_release\_date | character | Date when album released |
| playlist\_name | character | Name of playlist |
| playlist\_id | character | Playlist ID |
| playlist\_genre | character | Playlist genre |
| playlist\_subgenre | character | Playlist subgenre |
| danceability | double | 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 | double | 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 | double | The estimated overall key of the track. 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 | double | 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 typical range between -60 and 0 db. |
| mode | double | 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 | double | 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 | double | 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 | double | 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 | double | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. |
| valence | double | A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive |
| tempo | double | 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 | double | Duration of song in milliseconds |

# Methodology and results

This dataset is a more manageable collection of datapoints than the “FMA” data set archived on the UCI Machine Learning Repository (<http://archive.ics.uci.edu/>). The “FMA” data set is tagged with the ‘Classification’ and “Clustering’ meta tags. Since the “Spotify Song Attributes” is a much smaller subset of the information in “FMA” we split the analysis into the two different algorithms. For the Classification Decision Trees we selected the ‘*rpart*()’ and ‘*ctree*() functions in R. For the Clustering Analysis we chose *K-means*, *Hierarchical*, and *DBSCAN* for the clustering algorithms.

For all of the coding behind out analysis we have used an 80/20 split between the training and testing sets. To form the bins/percentiles we used the dplyr::mutate function first before splitting the data into test/train. Example code snipper from our .rmd file below.

precentile.cut <- .75

data.tree <- mutate(data.tree, dancebin = factor(case\_when(danceability >= precentile.cut ~ "H", TRUE ~ "M")))

index.collection <- sample(nrow(data.tree ),nrow(data.tree )\*0.80)

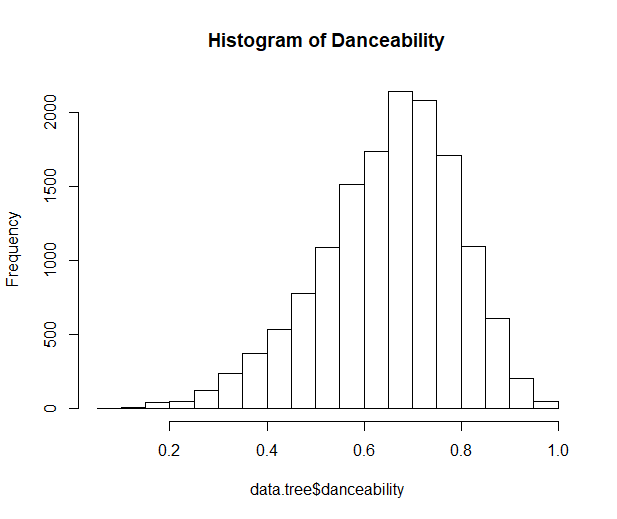
data.tree.train <- data.tree[index.collection,]

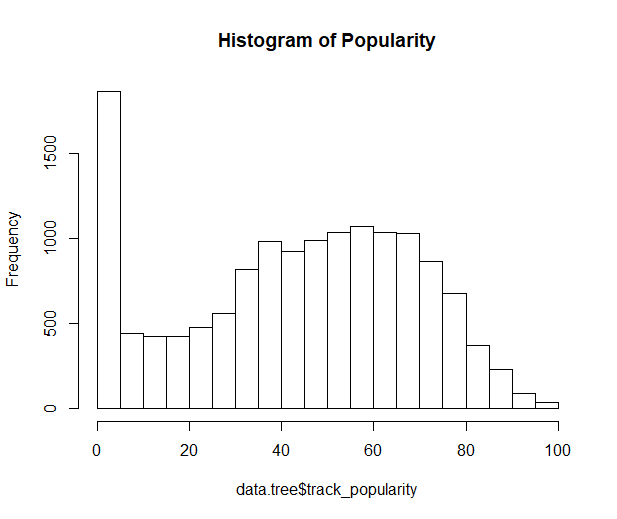
data.tree.test <- data.tree[-index.collection,]

## Classification Trees

Before writing the code in R to being generating models and tree graphs, we analyzed the distribution of both response variables of personal interest to us. This gave us an rough idea of where the breaks were in the data and a possible range for each variables cutoff value that could potentially return meaningful results. For “Danceability” we began with range of .75 - .99 to iterate over. For “Popularity” we began with .7 - .99.

### Histograms





Two different classification tree packages (and functions) were used for the analysis: [1] the “rpart” library with the *rpart()* function and [2] the “party” library with the *ctree()* function.

Conditional Inference trees, also referred to as *unbiased recursive partitioning*, is a non-parametric class of decision trees that uses a statistical theory (selection by permutation-based significance tests) in order to select variables. This is an alternative approach to selection by maximization of an information measure such as the *Gini coefficient* or *Information Gain*. The benefit being the removal of potential bias inherently present in CART or similar decision trees.

The function ctree() is used to create conditional inference trees. (A language, not a letter: Learning Statistics in R n.d.) The main components of this function are formula and data. Other components include subset, weights, controls, xtrafo, ytrafo, and scores.

* arguments
  + formula: refers to the the decision model we are using to make predicitions. Similarly to ANOVA and regression models in R, the formula will take the shape of outcome~factor1+factor2+...factor(n): where the outcome is the variable we are trying to predict, and each of the factors are the bases for the decision nodes.
  + data: tells the function which dataset to pull the variables listed in the model from.
  + subset: is an optional add on which specifies a subset of observations to be used in the fitting process. Should be used if you don’t want to fit the model to the entire dataset.
  + weights: is an optional vector that provides weighted values that can be used in the model fitting process. Can only consist of non-negative integers.

### Analysis of Danceability - RPART

Below is a subset of the information generated and reviewed as part of the analysis. We have selected cuttoff percentages of 70, 75, and 80 just for illustration purposes. Interestingly, a tree with only a root node was generated for anything < 70 or > 79. The greatest model accuracy was found to be at 70% but with approximately 25% of the variance not explained by the model.

|  |  |
| --- | --- |
| Model | dancebin ~  energy + valence + key + loudness + acousticness + instrumentalness + liveness + tempo |
| Confusion  Matrix | tree.pred H M  H 192 187  M 517 1975 |
| Accuracy % | 0.7547893 |
| Pruned Tree | n=9167 (2316 observations deleted due to missingness)  node), split, n, loss, yval, (yprob)  \* denotes terminal node  1) root 9167 2380 M (0.2596269 0.7403731)  2) valence>=0.6125 3250 1258 M (0.3870769 0.6129231)  4) tempo< 128.0345 2325 1048 M (0.4507527 0.5492473)  8) energy< 0.7885 1446 689 H (0.5235131 0.4764869)  16) tempo>=89.9755 1207 520 H (0.5691798 0.4308202) \*  17) tempo< 89.9755 239 70 M (0.2928870 0.7071130) \*  9) energy>=0.7885 879 291 M (0.3310580 0.6689420) \*  5) tempo>=128.0345 925 210 M (0.2270270 0.7729730) \*  3) valence< 0.6125 5917 1122 M (0.1896231 0.8103769) \* |
| Popular  cutoff % | 70 |

|  |  |
| --- | --- |
| Model | dancebin ~  energy + valence + key + loudness + acousticness + instrumentalness + liveness + tempo |
| Confusion  Matrix | tree.pred H M  H 322 411  M 869 1269 |
| Accuracy % | 0.5541623 |
| Pruned Tree | n=9155 (2328 observations deleted due to missingness)  node), split, n, loss, yval, (yprob)  \* denotes terminal node  1) root 9155 3658 M (0.3995631 0.6004369)  2) valence>=0.5965 3474 1519 H (0.5627519 0.4372481)  4) tempo< 128.0345 2489 939 H (0.6227401 0.3772599)  8) tempo>=89.9535 2151 738 H (0.6569038 0.3430962) \*  9) tempo< 89.9535 338 137 M (0.4053254 0.5946746) \*  5) tempo>=128.0345 985 405 M (0.4111675 0.5888325) \*  3) valence< 0.5965 5681 1703 M (0.2997712 0.7002288) \* |
| Popular  cutoff % | 75 |

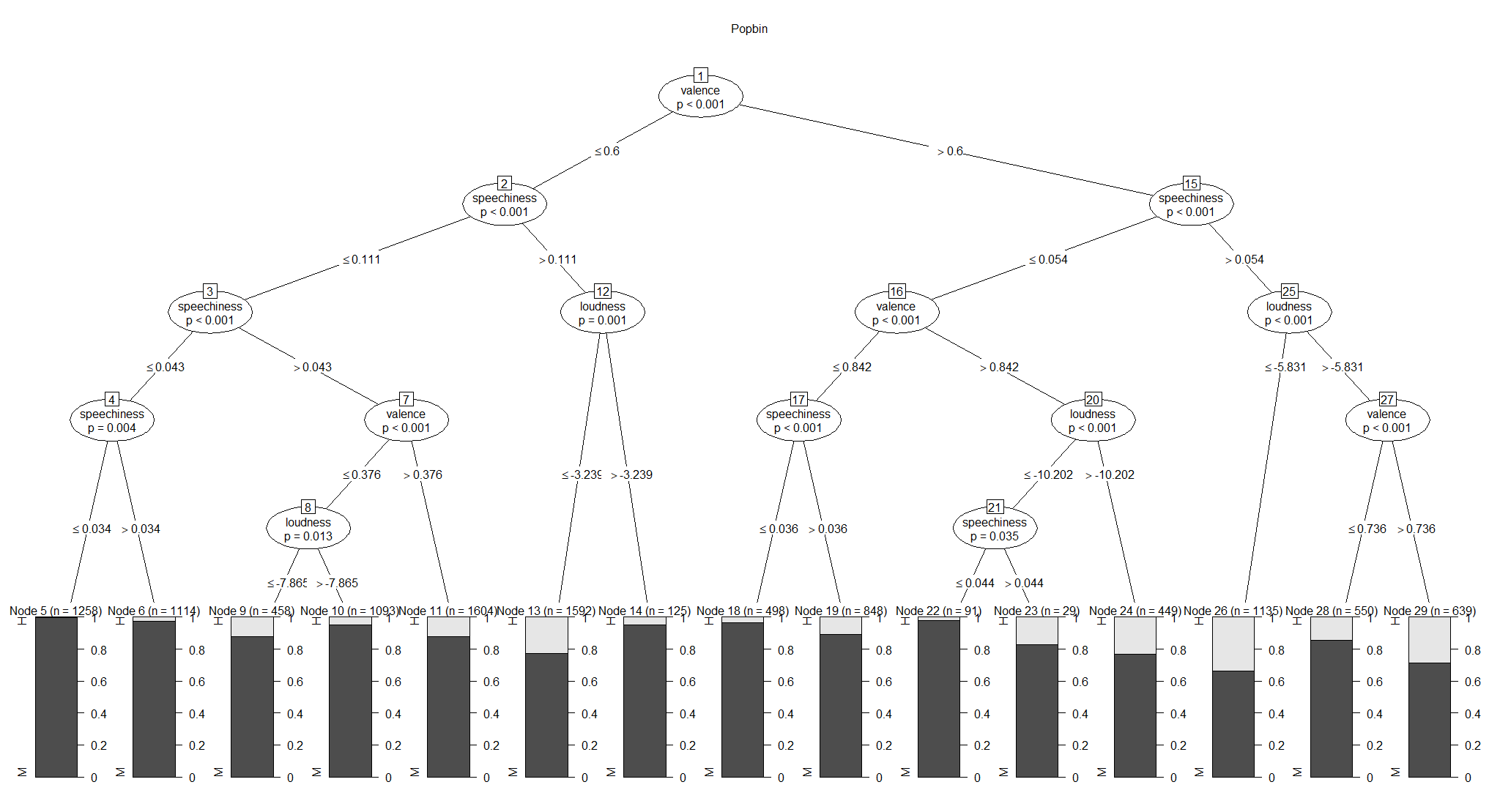
|  |  |
| --- | --- |
| Model | dancebin ~  energy + valence + key + loudness + acousticness + instrumentalness + liveness + tempo |
| Confusion  Matrix | tree.pred H M  H 0 0  M 379 2492 |
| Accuracy % | 0.8679902 |
| Pruned Tree | root 9167 1287 M (0.1403949 0.8596051) \* |
| Popular  cutoff % | 80 |

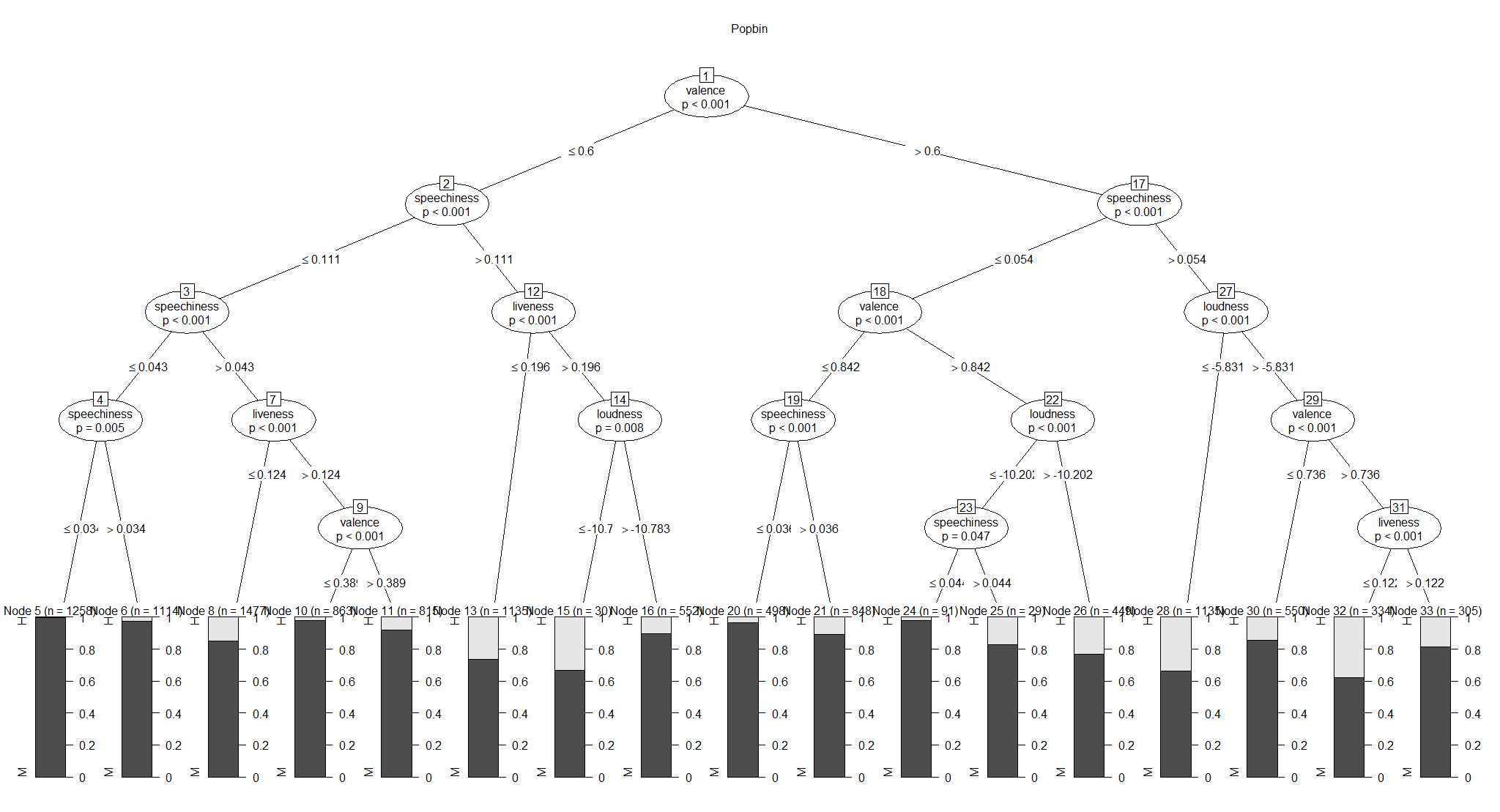
### Analysis of Danceability - CTREE

Two output graphs are shown below for the *ctree()* analysis for “Danceability”. The first is for a 75% cutoff for defining a highly danceable song and the second for a cutoff of 80%. These plots are more powerful that those generated with *rpart* as the decision paths lead to a probability distribution of the classification categories. ‘H’ is defined as “Highly Danceable” and as mentioned above are either a value in the top 25th percentile and for the second graph the top 20th percentile.

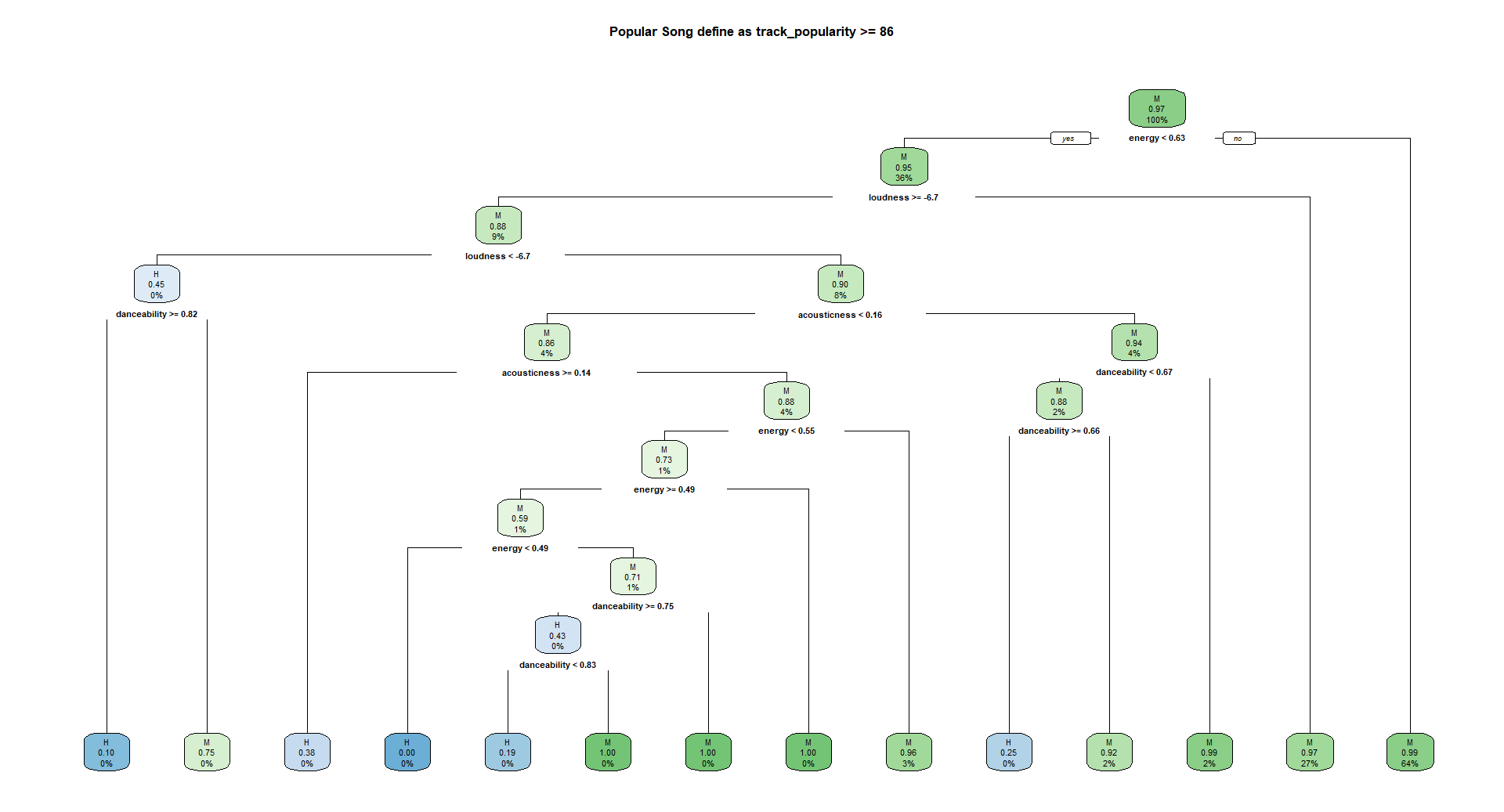
We can see from the first graph that the left-most bin indicates a path for a song that is predicted to have 0% danceability. In contrast, the 3rd distribution from the right shows an approximate 35% chance of such a song being classified as “Highly Danceable” by our metric/feature definition. Going one step further, we have an approximate 35% chance of producing a “Highly Danceable” song if we give it the characteristics of [a] variance > 0.6, [b] speechiness > .54, [c] loudness <= -5.832

Looking at the second graph for the top 20th percentile, we see that we have found a path for the highest percentage of producing a “Highly Danceable” song by making sure the *liveness* of the song is also <= .736

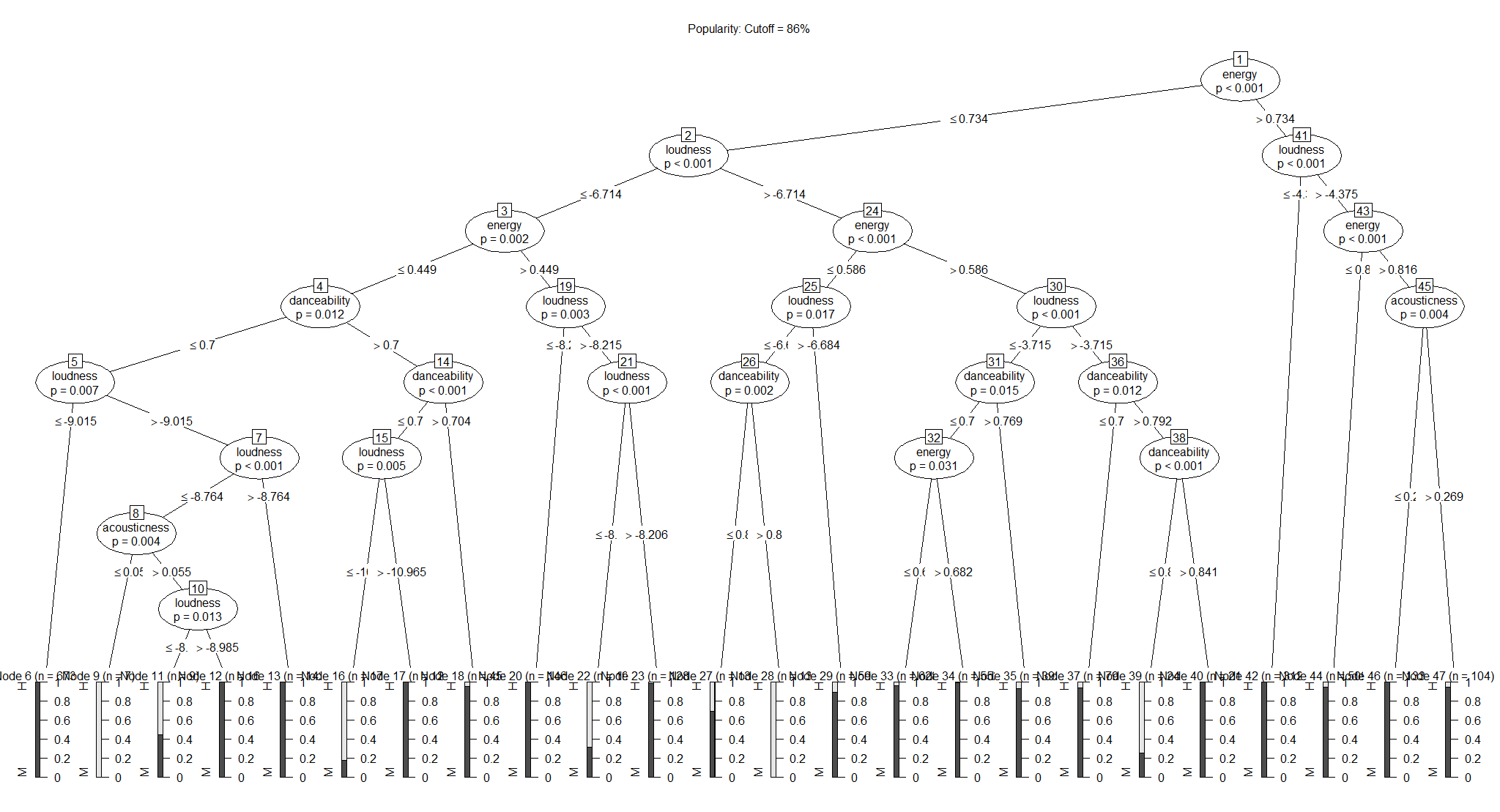




### Analysis of Popularity - RPART

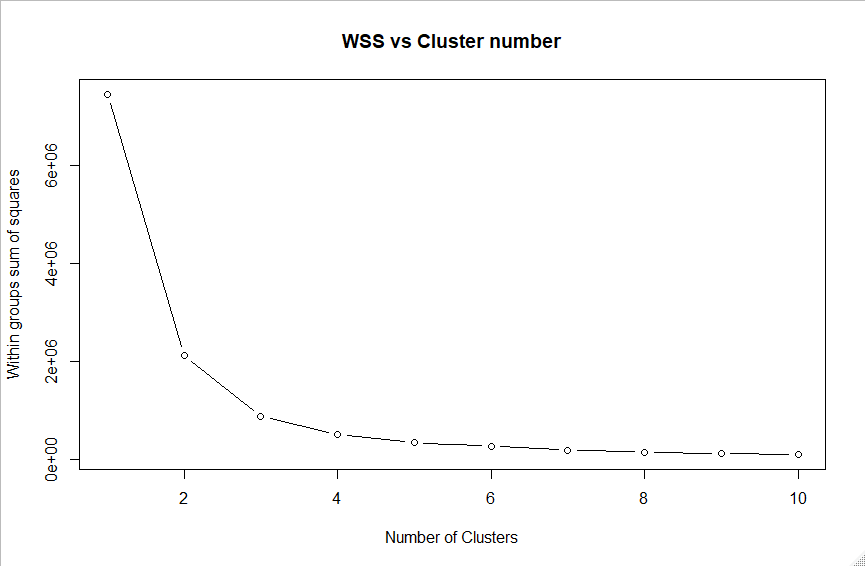


### Analysis of Popularity - CTREE

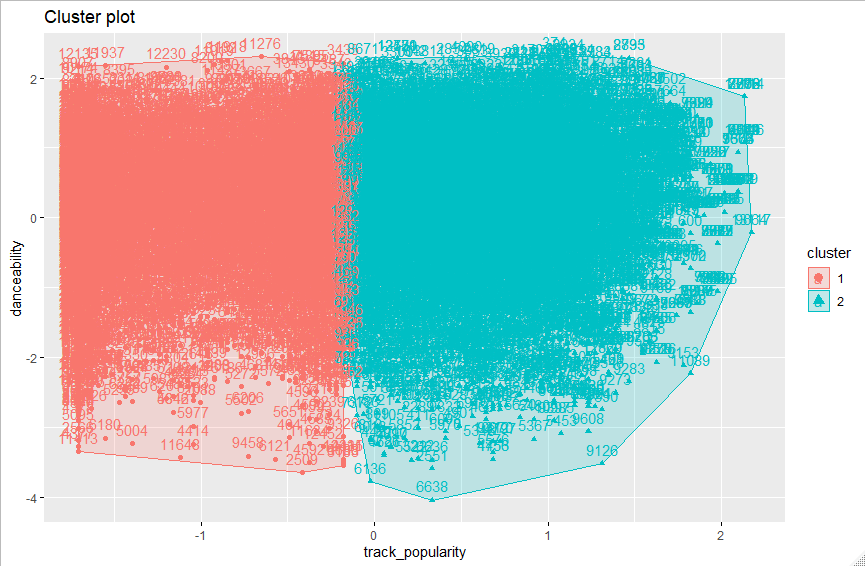


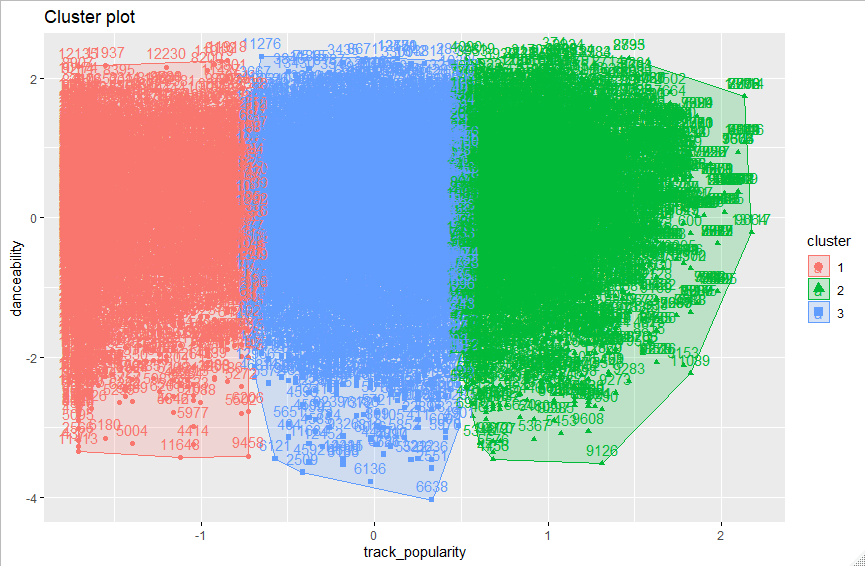
## Clustering

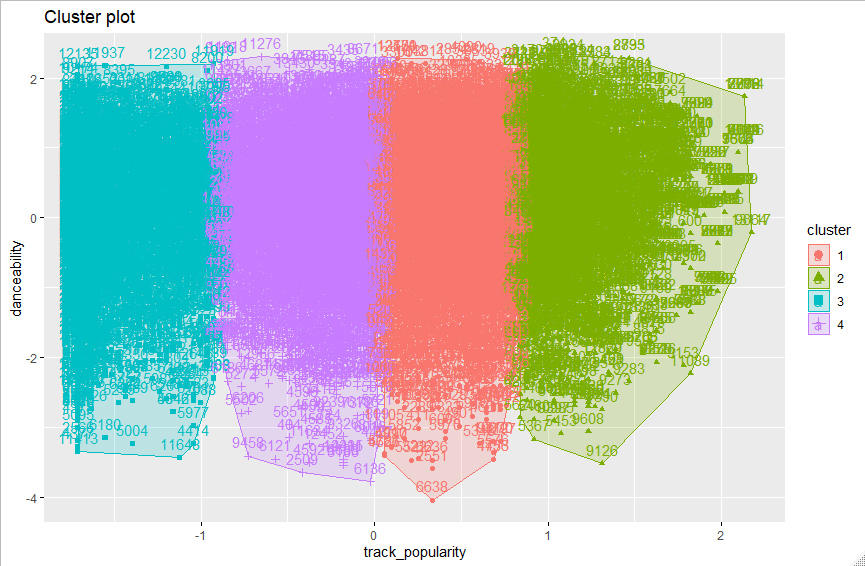
For Clustering Analysis, 3 different methods were used: K means, Hierarchical, and DBscan. For K-means clustering, a random 80% of the data was used for clustering. After that, a subset of data was generated using track popularity and danceability. A plot relating the Within Group Sum of Squares to the number of clusters was generated to determine the optimal number of clusters. This plot is shown below:

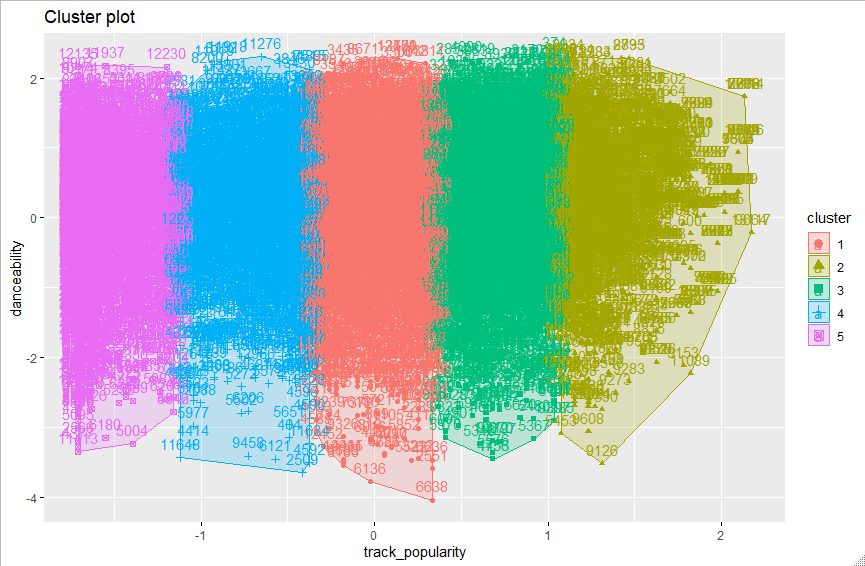


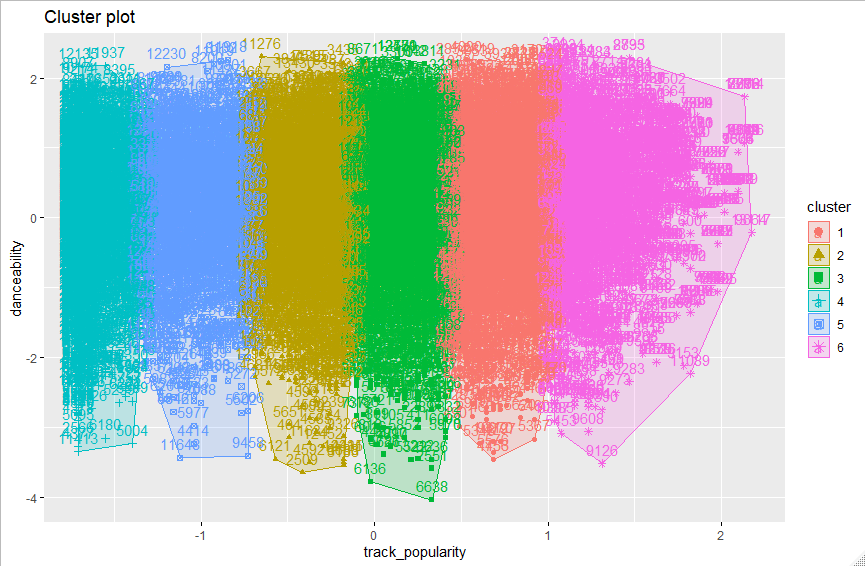
It is shown that between 2 to 6 clusters is an optimal amount. As such, cluster plots were made using the random 80% of initial data for 2,3,4,5 and 6 clusters, shown below.







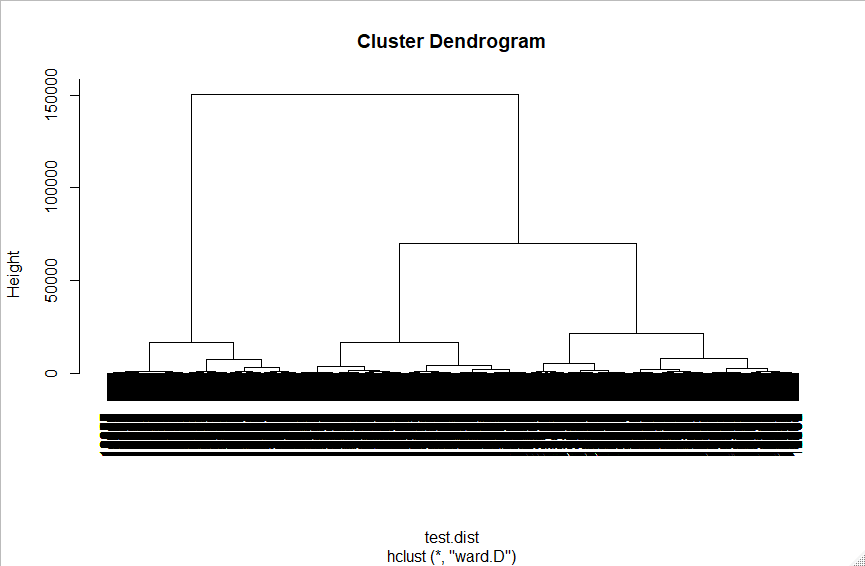




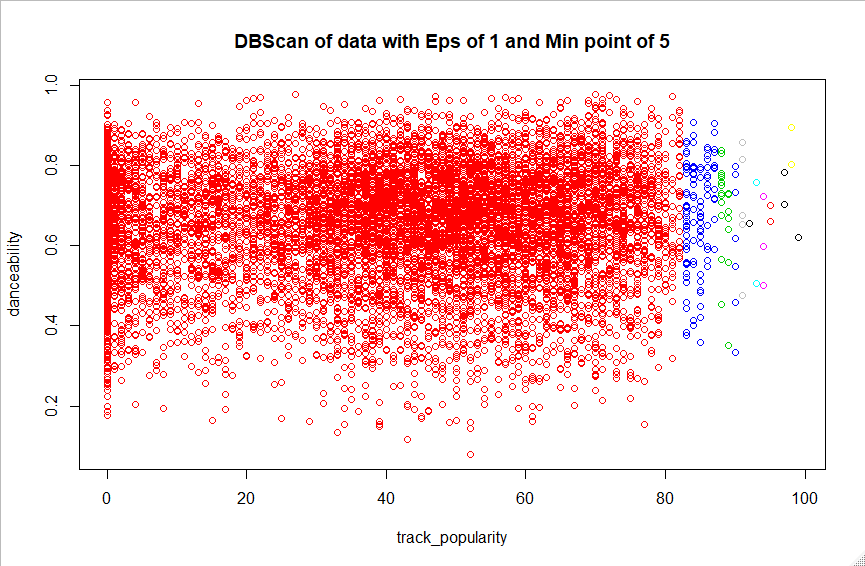
The main thing to take away from the progression of the cluster plots is that the initial 2 cluster plot didn’t have an issue of overlap but was merely too large. Increasing the number of clusters divided the larger clusters into subclusters which means the clustering is more discerning.

Interpreting the k-means clustering plots, it can be shown that the major dimension that affects clustering is the popularity of the track with danceability being fairly widely distributed regardless of track popularity; this implies that danceability has no impact on the clustering of data.

Hiearchical clustering was also done using the Ward method and the plot of the clustering is below:



Finally, DB Scan or Density-Based Spatial Clustering of Applications with Noise was used on the test data. DB Scan works to form clusters based on how many points are within proximity to core points of a cluster. For the following plot, the minimum number of points required is 5 with an epsilon neighborhood size of 1

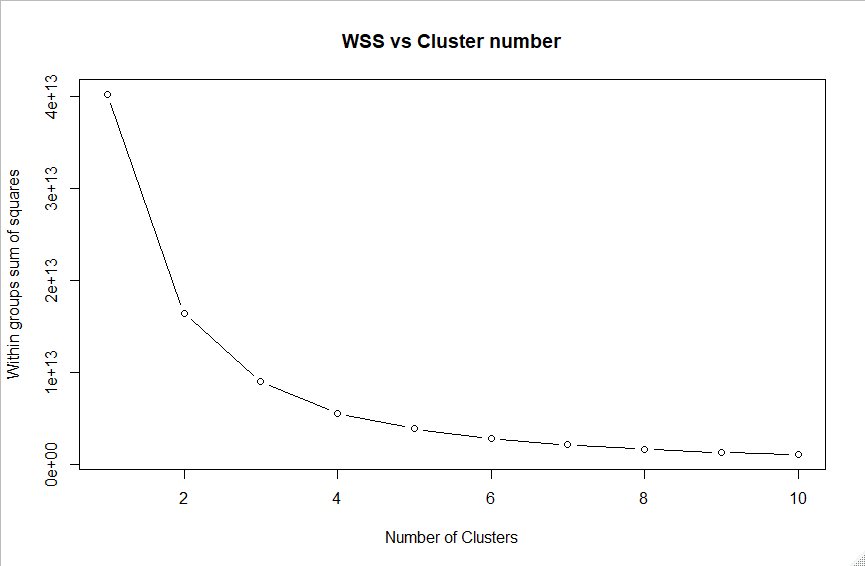


Attempts were made to resolve the issue of the large red cluster on the left. However, increasing the Eps from 1 to 2 only made the cluster grow and increasing the min point only made more of the right portion of the plot considered noise.

Interpretation wise, we can conclude that there is a vast majority of data within a broad range track popularity with less tracks above around 80% popularity and even less above around 85% popularity. In addition, as seen in the K-means clustering plots, Danceability didn’t affect the clustering at all as all clusters within the plot have a broad range of danceability.

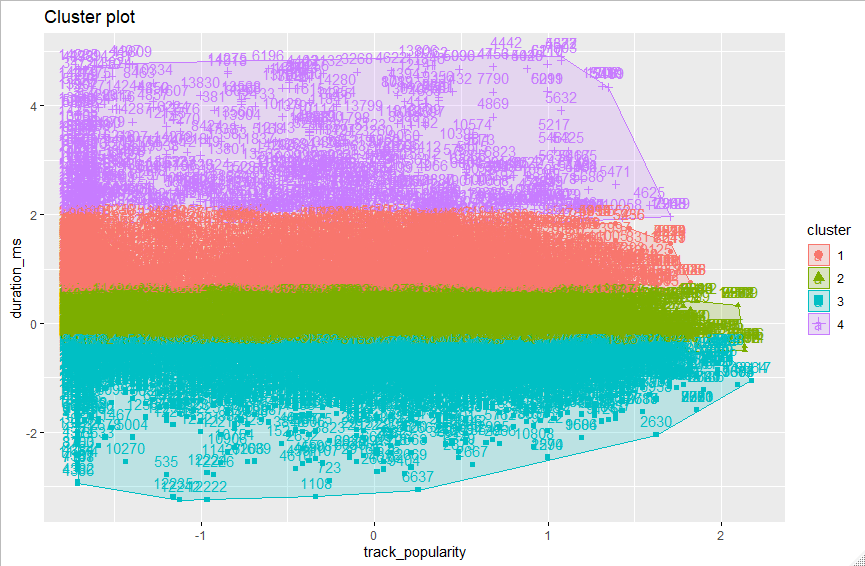
Seeing as danceability doesn’t seem to have a large impact on the clustering of data, the clustering analysis was redone with duration instead.

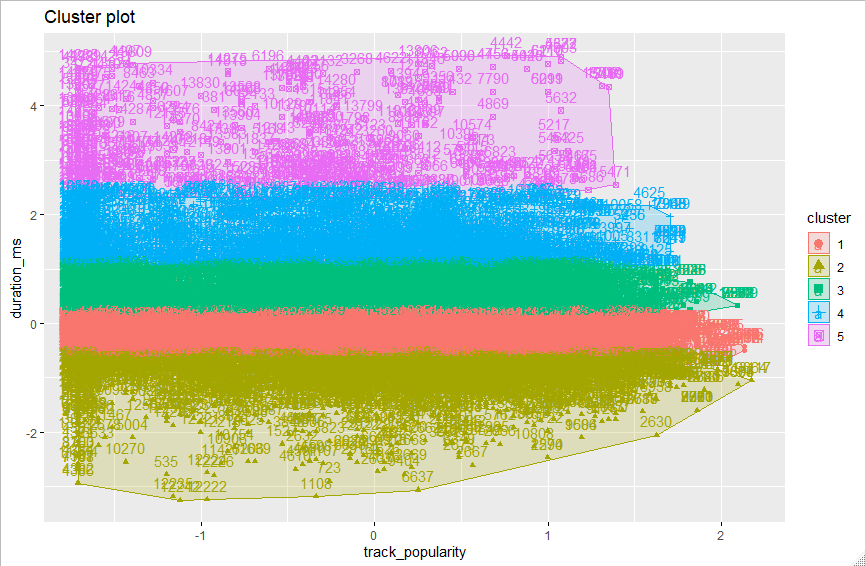
**K-Means:**

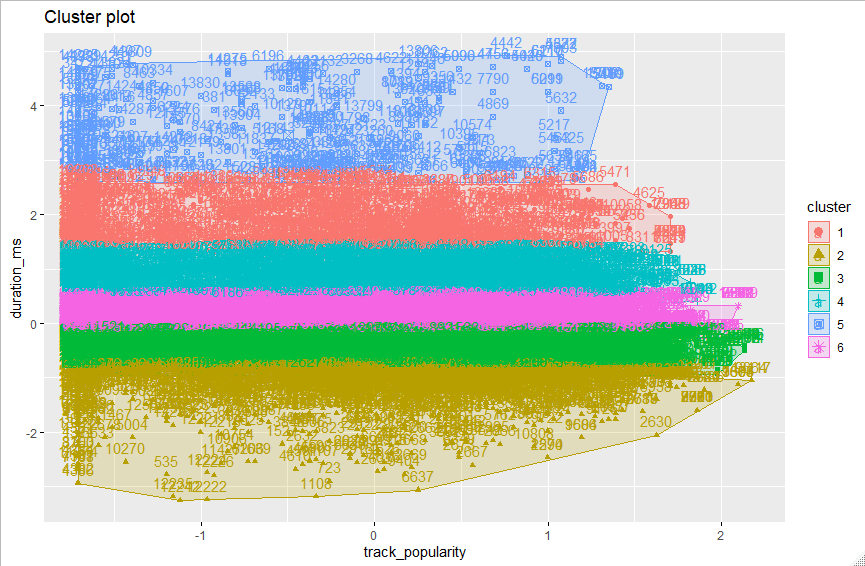




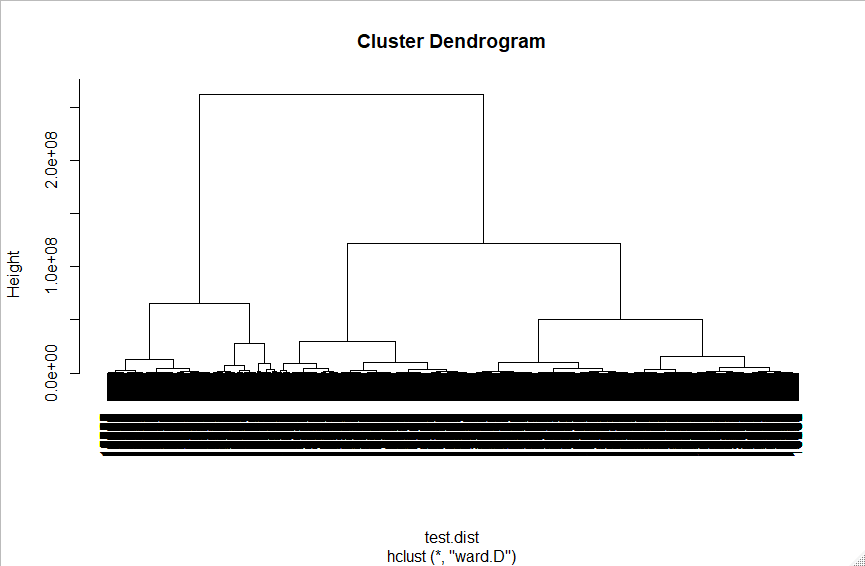




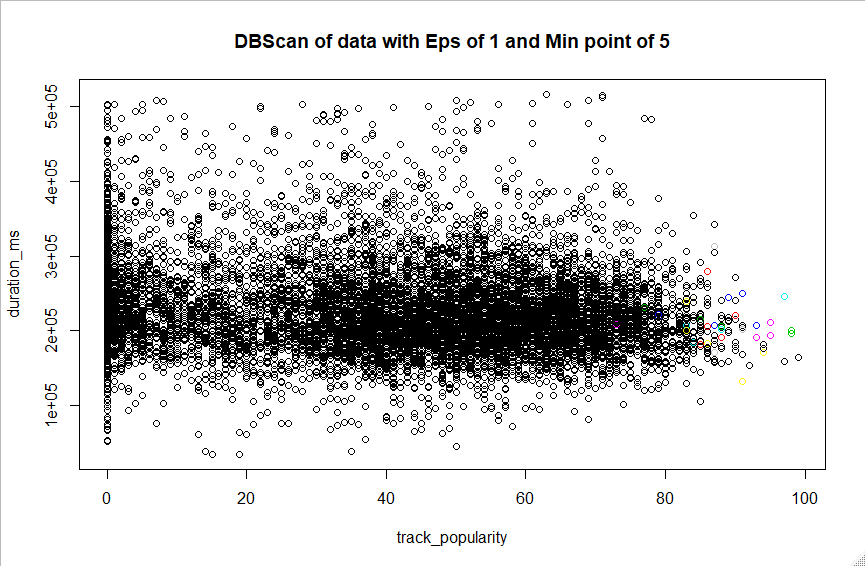




**Hierarchical Clustering:**



**DBScan:**



As seen in the K means clustering, the inverse issue of track popularity vs danceability came up. Now the major clustering factor is the duration in milliseconds rather than the track popularity What’s more curious is the DBscan plot; the plot like the popularity vs danceability plot has a large cluster taking up the majority of the plot. However, the ones that don’t fit within the major cluster aren’t arranged in a clean fashion like with the popularity vs danceability plot. In the previous DBScan plot, there were clean cut off points in popularity that determined the minor clusters. This isn’t the case with popularity vs duration.

# Conclusion

Unfortunately, this was not a very interesting dataset with regards to the output of the decision tree algorithms. It did should the relative power of the ctree() function in R both for reducing bias as well as producing a more detailed tree for review.

We believe though that the original research goal is still valid and in future work plane to use the “FMA” dataset on the UCI website. This is a music dataset with over 700 variables. This is a large undertaking and would require significant processing power which we did not have available to us before the due date of the report.

# Bibliography

[Various. 2008.](Various. 2008. Wikipedia. Dec. Accessed 04 23, 2020. https://en.wikipedia.org/wiki/Spotify.) *[Wikipedia.](Various. 2008. Wikipedia. Dec. Accessed 04 23, 2020. https://en.wikipedia.org/wiki/Spotify.)* [Dec. Accessed 04 23, 2020. https://en.wikipedia.org/wiki/Spotify.](Various. 2008. Wikipedia. Dec. Accessed 04 23, 2020. https://en.wikipedia.org/wiki/Spotify.)

<https://rpubs.com/coleeagland/decisiontreesislr831>

<file:///C:/Users/Ethan%20Hodys/Documents/MastersWork/DataMining/Data-Mining-R-master/5.%20Tree%20models/5_Tree.html#regression-tree-boston-housing-data>

<https://ademos.people.uic.edu/Chapter24.html#22_the_function:_ctree()>

# APPENDIX A – Additional Charts

