University of Salford MANCHESTER

# Music feature analysis using Supervised and Unsupervised Machine Learning

*Author:*
*Ali Kiyan*

*Assessors:*
*Dr. Mohamad Saraee & Dr. Yuhua Li*

January, 2018

# Music Features analysis

## 1.1 Introduction

Music and more specifically a song has many attributes that can help identifying a song in different manners. Big companies such as Pandora, Last.fm, iTunes and Spotify are using music recommendation systems to enhance their users' experience in a way that they can suggest their user a song that they are likely to enjoy. There are different algorithms that uses musical feature to classify and recommend next song or create different playlist.

In this research, I am going to first analyse songs based on their musical features and then categorise them based on the similarity of their musical features. These groups (clusters) can later be used in conjunction with other methods to enhance the performance of analysis for example for predicting genre of a song or popularity of a song. They can also form more sophisticate playlist of songs that have similar music features. For example, one of the groups (clusters) can be representative of happy songs because it has high tempo and loudness or in a certain rhythm like 4/4.

In the second part of this research I am going to use Supervised Machine algorithm to predict the popularity of songs based on musical features. Lastly I am going to use a hybrid algorithm utilising both type of Supervised and Unsupervised machine learning to enhance my prediction.

I will be using R Programming Language and *SAS Enterprise Miner* as tools for my analysis.

Keywords: Supervised Machine Learning, Unsupervised Machine Learning, Music feature analysis, Echo Nest (Spotify), SAS Enterprise Miner, R,

### 1.1.1 CRISP-DM[1]

CRISP-DM is one of the popular methodologies that provides structured approach to data mining. There are 6 main phases in this methodology as below:

- Business understating
  Determine business objective, Assess situation, Determine data mining goals, produce project plan
- Data Understanding
  Collect initial data, describe the data, explore the data, verify the quality of the data
- Data prepration
  Converting data to tabular format, removing or imputing missing values (Cleaning data), data transformation
- Modelling
  Working on finding patterns and building the model
- Evaluation
  Evaluating meaningful results
- Deployment
  Produce the final report of values

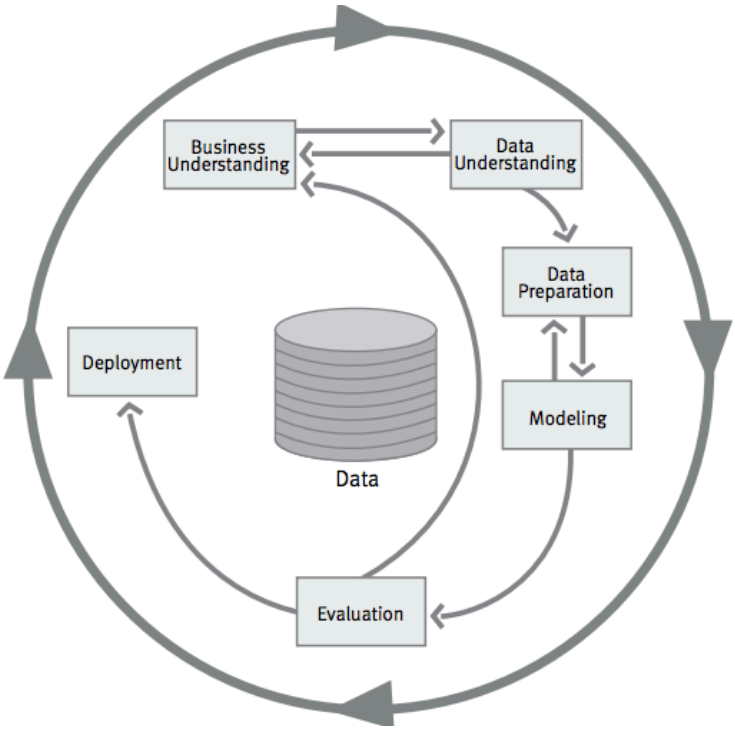



*Figure 1.0 CRISP-DM methodology*

[1] Cross Industry Process for Data Mining

### 1.1.2 SEMMA

SEMMA is a flexible methodology that can be used in conjunction with other methodologies in data mining. Within the SAS Enterprise Miner you can access to tools represent as steps in the SEMMA methodology. These steps are :

- Sample
  Sample the data to a point that convey significant information.
- Explore
  To find anticipated and unanticipated relations in order to get understanding of data.
- Modify
  Model selection process by manipulating variables.
- Model
  Modeling by using analytical tools to predict a desired outcome.
- Assess
  Checking the usefulness and reliability of the model.

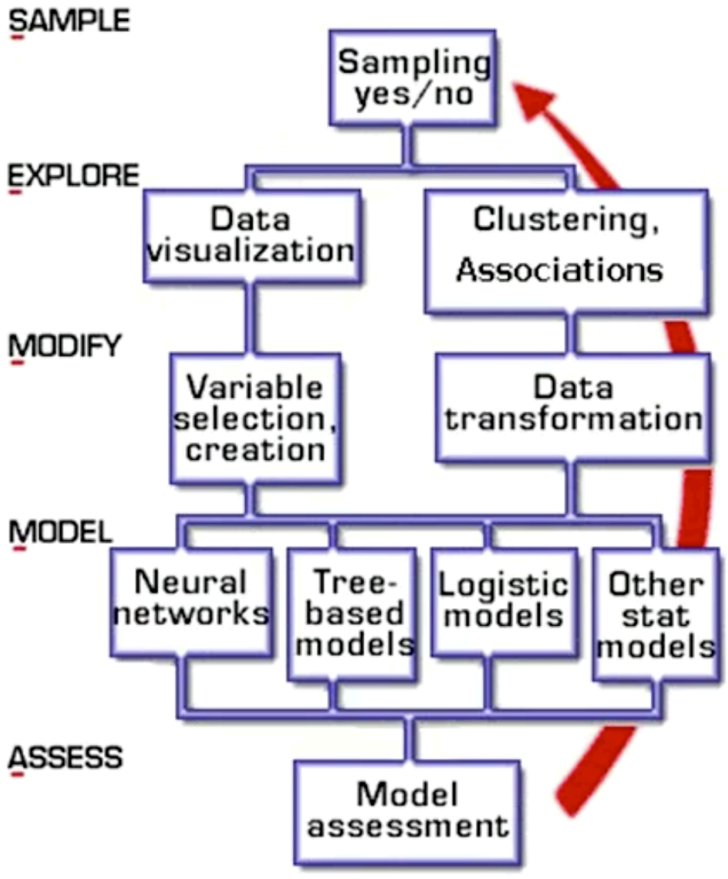



*Figure 1.1 SEMMA methodology process*

I followed a combination of these two methodologies in my data mining analysis. To have a better understanding of data and exploring it (SEMMA), first I will use clustering and then to predict an outcome I will use classification (Decision tree).

## 1.2 Search Strategies and Explanation of dataset

One of the largest dataset available for analysis of musical data is Million Song dataset which initially was a collaborative project between The Echo Nest (acquired by Spotify) and LabROSA,a laboratory working towards intelligent machine listening. This dataset also contains information from the following datasets:

Table 1-1 Million Songs dataset origin

| Complementary dataset | Area |
|---|---|
| SecondHandSongs dataset | Covered songs information |
| musiXmatch dataset | Lyrics |
| last.fm dataset | Music tags |
| Taste Profile subset | Users |
| thisismyjam-to-MSD mapping | Users |
| tagtraum genre annotations | Genre Tags |
| Top MAGD dataset | Genre Tags |

The initial size of 1 Million Songs dataset is 300GB, Therefore, I have used CORGIS[2] Dataset Project which provided me with the same dataset in a more maintainable size for machine learning. The initial dataset before normalisation consists of 10001 observations with 35 different attributes about each track.

### 1.2.1 Fields associated with each track in the dataset

In the following, some attributes that needs clarifications, related to each track have been concisely described.

[2] The **C**ollection of **R**eally **G**reat, **I**nteresting, **S**ituated Datasets

Table 1-2 Audio features of the Million Songs Dataset

| Data set Attribute | Description |
|---|---|
| Artist.Hotttnesss | The popularity of the singer |
| artist.id | Artist Echo Nest ID |
| Artist_mbtags | Music Tags from the musicbrainz.org |
| Artist_mbtags_count | Number of tags from musicbrainz.org |
| bars_confidence | confidence value (between 0 and 1) associated with each bar by The Echo Nest |
| bars_start | start time of each bar according to The Echo Nest |
| Beats_confidence | confidence value (between 0 and 1) associated with each beat by The Echo Nest |
| Beats_start | start time of each beat according to The Echo Nest |
| End_of_fade_in | time of the end of the fade in, at the beginning of the song, according to The Echo Nest |
| Key | estimation of the key the song is in by The Echo Nest |
| Key_confidence | confidence of the key estimation |
| Mode | estimation of the mode the song is in by The Echo Nest |
| Mode_confidence | confidence of the mode estimation |
| Release.id | the ID of the release (album) |
| Release.Name | the Name of the release (album) |
| Similar | a list of 100 artists (their Echo Nest ID) |
| Song.hotttnesss | according to The Echo Nest, when downloaded (in December 2010), this song had a 'hotttnesss' of 0.8 (on a scale of 0 and 1) |
| song.id | The Echo Nest song ID |
| Start_of_fade_out | start time of the fade out, in seconds, at the end of the song, according to The Echo Nest |
| Tatums_confidence | confidence value (between 0 and 1) associated with each tatum by The Echo Nest |
| Tatums_Start | start time of each tatum according to The Echo Nest |
| Tempo | tempo in BPM according to The Echo Nest |
| Terms | Genre |
| Terms_freq | frequency of the terms from The Echo Nest (number between 0 and 1) |

| Data set Attribute | Description |
|---|---|
| Time_Signature | time signature of the song according to The Echo Nest, ( usual number of beats per bar ) |
| Time_Signature_Confidence | confidence of the time signature estimation |

Extra information about the attributes:

Tatum: List of Tatum markers, in seconds. Tatums represent the lowest regular pulse train that a listener intuitively infers from the timing of the perceived musical events (segments).

Tempo: beats per minutes determines the overall speed of the song.

### 1.2.2 Problem description

A song consists of a variety of attributes and I am interested in grouping songs and finding patterns amongst its features using unsupervised machine learning methods. These methods enable me to find uncovered groups and relations between different attributes of the song. These groups (clusters) will have multidimensional similarities such as tempo, loudness, etc. If I combine these musical features with popularity of the song and popularity of the artist, I will get even more interesting result. In this case I can have groups indicating the popularity of the song as well as musical characteristics. If the inter-groups are separated clearly it means that these songs are similar together in terms of their musical attributes and popularity. (performance metric). This can later be used in many cases. For example, the songs in the same clusters are more likely to be pleasant and familiar to their listener because they have similar musical attributes like loudness, or they can indicate certain moods like happiness with certain level of popularity. If I manage to find non-overlapping clusters with unsupervised learning algorithm, it means songs can be grouped by their musical features and popularity and therefore there is a logical relation between them.

At the second part of my analysis I am going to use classification algorithm to to build a model that can predict the popularity of song. Since the popularity of a song is depending on many different musical and non-musical factors, I am not expecting perfectly accurate predicting model but I will be trying different method to maximise my prediction accuracy. One of the other application of this analysis can be understanding the importance of attributes determining the popularity of songs.

There are multiple reasons why I have chosen this dataset. Firstly, with 35 attributes and 10001 music observation, it provide me with sufficient data about the music features, therefore, I can have a more proper understanding of the subject. Secondly, This is real world comprehensive dataset is widely used in big companies like Spotify, Soundhound and Last.fm. I can find patterns and categories that may have not considered before. I am also interested in finding patterns in songs and building a model that can accurately predict popularity of songs.
Lastly, I have spent a considerable amount of time to find a primary dataset that is interesting enough to discover the patterns, large enough to be viewed form different perspectives, and has not been outdated and music data sets is a great match. I experimented different combination of attributes and methods to get the best result.

KEYWORD : Music recommendation systems, EchoNest, Spotify, Feature selection, Clustering, Supervised Machine Learning Algorithm, Classification Tree, Hybrid unsupervised and supervised algorithms

### 1.2.3 Data preparation

## Feature selection

I have limited my dataset from 35 to 12 attribute to only attributes which are musically significant.

There are pros and cons to having less amount of feature such as :

**Less number of features:**

- Easy to interpret
- Less likely to overfit
- Low prediction accuracy

**More number of feature**

- Difficult to interpret
- More likely to overfit
- High prediction Accuracy
- Interpretability
- Insight

Quality data is essential for having a quality data mining result. More than 90% of work in data mining is dedicated to data preparation, therefore it plays a vital role.
There are various data preprocessing techniques. *Data cleaning* which remove noise and correct inconsistency, *Data integration* which merges data from multiple sources into a coherent data source (data-warehouse) *Data Reduction* can reduce the size for example with elimination of some attributes, aggregating certain features. *Data transformation* consists of scaling and normalisation where data scaled between a specified range.
As mentioned earlier, 1 Million data set is a combination of multiple other sources (Table 1-2) therefore different sources merged together forming Music data set. Data reduction has been used to reduce the number of unrelated attributes. For example in my analysis I eliminated some of musical attributes that cannot be decisive in a selection of a track such as all confidence attributes because they are derived attributes and redundant. Since I had a lot of different attributes that are in the different range such as millisecond in fade out and tempo which is a decimal, I have to use data transformation to have different attributes in a same scale. There are different option such as using scaling and normalisation. Normalisation rescale the values based on desired range (mostly between -1,1). Standardisation rescales the values based on the number of standard deviation from each value's distance from the mean of the values.
Sometimes some observations are meaningless. In music dataset I have found zero values for tempo and loudness of the song which are meaningless and therefore, I removed them.
Lastly NA or missing observations are not very descriptive, therefore, they need to be handled.

**Why it is important to deal with missing data?**
Biased estimates
Incorrect standard errors
Incorrect inferences/results

There are two types of missing data Unit nonresponse and Item non response
**Unit nonresponse :**
When the data for entire unit is missing
**Item nonresponse :**
When the data for individual items are missing

**Dealing with missing data**

**Complete case analysis (list wise deletion)**

Restricts the analysis to individuals with observed data

**Imputation**

**Single imputation**

Taking educated guess

**Mean imputation**

Using the mean value of attribute to fill the missing data.

**Regression prediction**
Predict missing data within categories defined by other variables
For example we can regress age as a function of state and gender

**Regression prediction plus error (hot Deck)**
Like the regression prediction but it adds some noises
It randomly chooses rows from the selected variables and use their value to the missing data(the range of imputations will definitely falls between the the range of our observed data).

In my case I have used genre attribute (terms) and its mean value for each genre to imputate missing data in popularity (song.hotttnesss) attribute. After cleaning the dataset I have 6608 rows without any missing values.

| artist.hotttnesss | bars_start | beats_start | duration | end_of_fade_in | loudness | song.hotttnesss | start_of_fade_out | tatums_start | tempo | terms | time_signature |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.40199754 | 0.58521 | 0.58521 | 218.93179 | 0.247 | -11.197 | 0.6021200 | 218.932 | 0.28519 | 92.198 | hip hop | 4 |
| 0.41749964 | 0.71054 | 0.20627 | 148.03546 | 0.148 | -9.843 | *NA* | 137.915 | 0.20627 | 121.274 | blue-eyed soul | 4 |
| 0.34342838 | 0.73152 | 0.73152 | 177.47546 | 0.282 | -9.689 | *NA* | 172.304 | 0.42132 | 100.070 | salsa | 1 |
| 0.45423116 | 1.30621 | 0.81002 | 233.40363 | 0.000 | -9.013 | *NA* | 217.124 | 0.56254 | 119.293 | pop rock | 4 |
| 0.40172369 | 1.06368 | 0.13576 | 209.60608 | 0.066 | -4.501 | 0.6045007 | 198.699 | 0.13576 | 129.738 | pop punk | 4 |

*Figure 1.1 Music Dataset*

Both SAS and R can be used for data preparation.


*Figure 1.2 Data preparation is SAS*

### 1.2.4 Dependent and independent variables

Since I am using K-Means clustering algorithm, I am interested in groups of attributes and in the end I am looking to see group of songs that have similar musical features and have similar popularity.
In the second part of my research, Classification tree, I will convert song popularity to three range. High, low and Unknown. I will use this attribute as my Dependant variable (class variable).

- High:

Songs that at least have 50% or more in song.hotttnesss attribute.

- Low:

Songs that have less than 50% in song.hotttnesss attribute.

- Unknown songs:

Songs that 0 in song.hotttnesss attribute.

Independent attributes :

- "artist.hotttnesss"
- "bars_start"
- "beats_start"
- "duration"
- "end_of_fade_in"
- "loudness"
- "start_of_fade_out"
- "tatums_start"
- "tempo"
- "time_signature"
- "popularity"

- In my last experiment I will use a hybrid approach where I use cluster number of each song in training set and use it as independent variable to form my classification tree.

## 1.3 Implementation of Clustering in R (K-means Algorithm)

### 1.3.1 Packages

**Unsupervised Learning (Clustering):**

I have used “cluster” package because it has hierarchical clustering built-in functions for complete linkage (hclust function) and average linkage (avhclust with method option set to average)

Although Time complexity of Hierarchical clustering is $O(N^3)$ (in case of using priority trees in can be reduced to $O(N^2 \log N)$ therefore Hierarchical clustering is not considered the best choice for datasets with high numbers of data point. (Anand Rajaraman, 2014)

I have used “stringr" package for string manipulations in my dataset. This package has a lot of string manipulation functions that can be very useful such as regex function for finding a particular string pattern, and string detection (str_detect) for finding sub string within all observations.

I have also used “dplyr” package. This powerful package provides 5 main verbs. ‘Filter’, ‘Select’, ‘Arrange’, ‘Mutate’, ‘Summarise’ and ‘group_by’
It has different joins (inner joins, left join, …) functionality as well as windows functionality for ranking.

The way these functions can be used is for example for filtering you should provide your data frame, then the condition that you want to be met. You can have multiple conditions separating by commas. As an out put it returns a dataset.

“Chaining” and “pipelining” can be done by then operator “%>%”. You can pipe results or dataset to the other operation with the help of this operator. I have used this operator in conjunction with filtering to target my genre column in my dataset. Then I have used the str_detect function in ”stringr” package to search for a particular package.

I have psych package for basic data entry and descriptive analyses. This package contains functions for personality, psychometric, and psychological research but I have used this package for finding correlation in my music dataset.

I have used ggplot2 for some of my plotting.it is based on the grammar of graphics, which tries to take the good parts of base and lattice graphics. It provides a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

### Classification Tree (Decision Tree):

I have used two packages to use decision tree on my dataset. First I used "PARTY" package (Hothorn, Hornik, and Zeileis 2006) for producing decision tree and later pruning I used "RPART[3]" to check the performance of each package on my data set.
"PARTY" package is using CART[4] algorithm. It aims at providing a recursive partitioning for building tree-based regression and classification models.
At the core of the package is ctree, an implementation of conditional inference trees, which offers regression trees that are applicable to different types of regression problem such as nominal, ordinal, numeric, etc. The advantage of CART algorithms is that they handle outliers properly. One of the disadvantage of this algorithm is that it produce binary trees which can split data into two subgroup.

The RPART program builds classification or regression models using CART algorithm. It has two stage procedure. First the best variable for splitting the data ,

One of the disadvantages of decision trees is overfitting where the training model has high level of accuracy because it is fits to the data very well but cannot have high level of accuracy in testing set. There is a general notion that wide full tree are often overfit the data, therefore, pruned tree which has less yet more confident nodes, are desirable. In this research, I have pruned RPART tree using complexity parameter to minimise overfitting error and maximising overall accuracy.

Lastly, I used RPART.PLOT package to enhance the plotting in my rpart trees.

## 1.3.2 Application of Clustering in R with K-means Algorithm

I want to exploit k-means unsupervised algorithm to group different songs based on their musical attributes and their popularity. I am considering 11 different attributes to form clusters. The songs that fall into the same cluster will have similar musical characteristics (more pleasant to their listeners) and have similar popularity rate. K-means clusters can be viewed in relation to different attributes of the song and one of the key point is the notion of determining optimal number of k or number of clusters.
In case of some attributes due to have well-separated clusters where the members of the cluster are near to their centroid, I have to change the number of clusters. (Performance metric)
One of the application of this analysis would be creating an smart playlist. Depending on the type of the playlist, it can have group of songs. These songs have similar musical features and

[3] Recursive Partitioning

[4] Classification And Regression Trees

popularity rate. For example, tempo, loudness and popularity can be taken to account and different clusters can represent different playlist such as slow music for night which has low tempo and less loudness. In case of 3 well-separated clusters, they can be representative of 3 type of fast, medium , slow speed playlist, where can have similar rate of popularity.

### 1.3.3 Explanation of the experimental procedure

Since this is unsupervised clustering algorithm, the objective is to form group within existing dataset, therefore training and cross validation are not considered in this types of unsupervised algorithms.

After bringing up dataset to R environment, I have selected 12 musically significant attributes of each songs along with the song popularity and the singing artist popularity.

Filter

| | artist.hotttnesss | bars_start | beats_start | duration | end_of_fade_in | loudness | song.hotttnesss | start_of_fade_out | tatums_start | tempo | terms | time_signature |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.40199754 | 0.58521 | 0.58521 | 218.93179 | 0.247 | -11.197 | 0.6021200 | 218.932 | 0.28519 | 92.198 | hip hop | 4 |
| 2 | 0.41749964 | 0.71054 | 0.20627 | 148.03546 | 0.148 | -9.843 | *NA* | 137.915 | 0.20627 | 121.274 | blue-eyed soul | 4 |

*Figure 1.3 musically significant attributes for analysis*

Next step is preparing the dataset for further analysis. I start with checking columns for NA values and I have found out song hotttnesss attribute has missing values. For Imputation, I tried different methods.
At first, I created five different data frame based on e main genres, namely, Rock, Jazz, Pop, Classic, Country.(I used dplyr package to chain my dataset and filter out observations containing one of the 5 mentioned genre). Then I calculated the Mean value for each Main genre. I found this method of imputation (based on group of another related attribute) the most accurate in my dataset rather than removing all missing value observations or imputing the mean of an song.hotttnesss attribute without grouping it.
Later, After experimenting my clusters, I realised the best way for minimising overlapping values, is by not considering the pop genre into observations. My speculation is that Pop is generic genre with loose boundaries and the popularity of the song and artist in this genre is not highly dependent of the song musical attribute, therefore, Pop songs characteristics hinders analysis based on the musical attributes in this genre and can analyse separately . As a result I avoid imputing data for missing values in this genre.

```r
#using stringr and dplyr retrun a separate dataset if the track has one of the 5 main genre type in its terms column
#Rock dataset
contains_rock <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "rock"))
nrow(contains_rock)
#Jazz dataset
contains_Jazz <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "jazz"))
nrow(contains_Jazz)
#Pop dataset
#contains_Pop <- music_cleaned %>%
#filter(str_detect(music_cleaned$terms, "pop"))
#nrow(contains_Pop)
#classic dataset
contains_Classic <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "classic"))
nrow(contains_Classic)
#country dataset
contains_Country <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "country"))
nrow(contains_Country)
```

*Figure 1.4 Making dataset based on Main Genre in R Programming Language*

```r
Rock_Song_hotness_mean <- mean(contains_rock$song.hotttnesss, na.rm = TRUE)


#3.impute the mean to every NA row
length <- nrow(music_cleaned)


#for Rock Tracks
#for(i in 1:length)
{
  if(is.na(music_cleaned$song.hotttnesss[i]))
  {
    if(grepl("rock", music_cleaned$terms[i]))
      {
         music_cleaned$song.hotttnesss[i] <- Rock_Song_hotness_mean
      }
  }
}
#remaining NA rows
sum(is.na(music_cleaned))
```

*Figure 1.5 Imputing mean value for Rock songs in the missing observation*

Then I checked for zero values in the columns where having zero values is meaningless. I removed zeros from “tempo” and “loudness”.

I get the pair of clusters between different attributes to have better understanding about database.

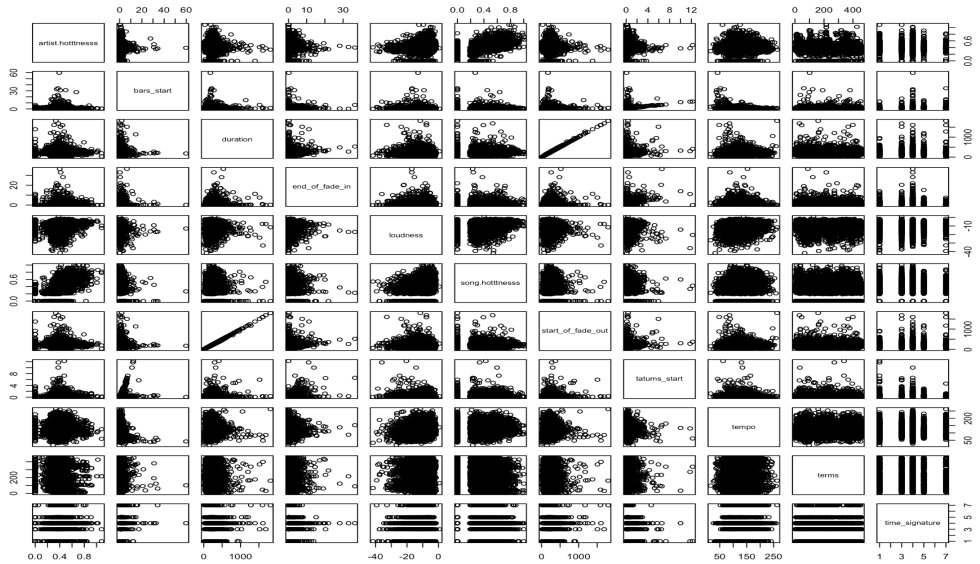



*Figure 1.6 Music dataset cluster pairs*

To have a better understanding of the data I have used "psych" library to find correlation between musical features.

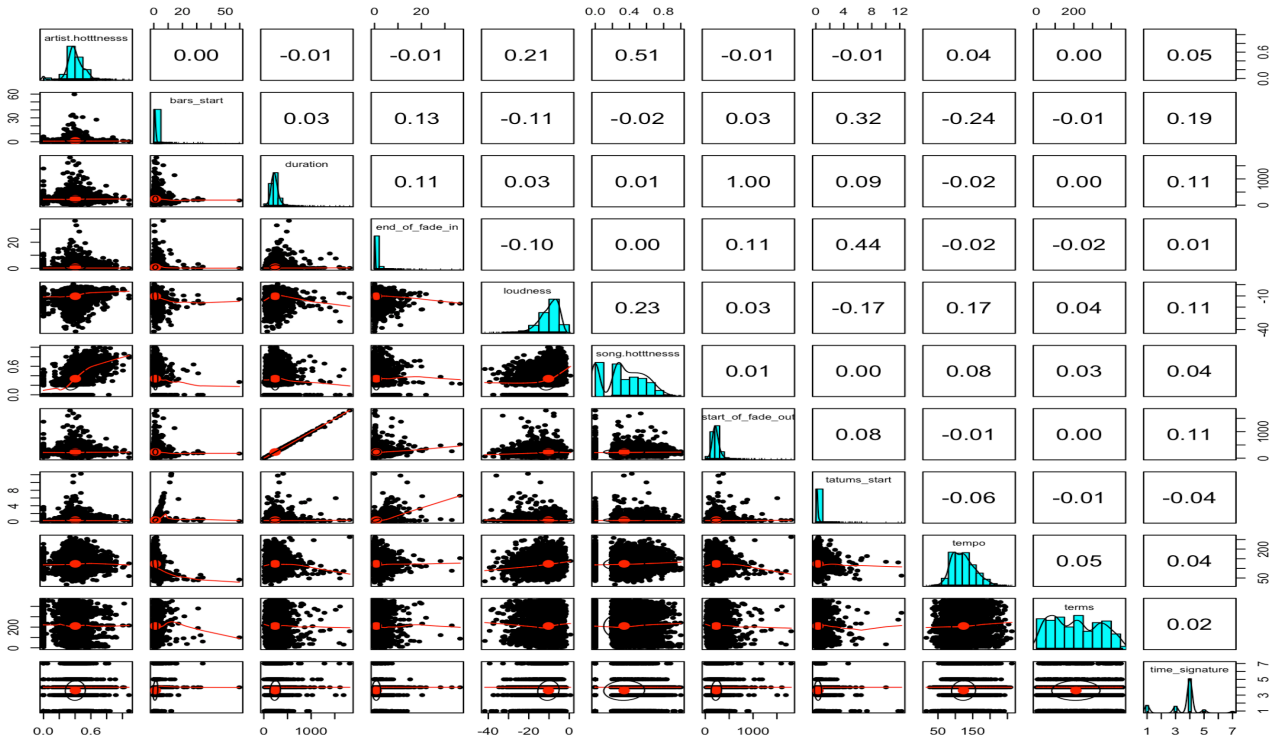



*Figure 1.7 Correlations in Musical attributes*

The correlation between attributes can be viewed by the numbers and graphs and their centres. There are some patterns as below :

- Popularity of the artist is  correlated with loudness, time signature, and tempo of the song.
- Duration of a song is corrected with start to fadeout attributes and after that end of fade in and time signature can be determinative factors.

In my analysis, song popularity is important attribute and it can be seen that three attributes, namely, tempo, loudness of the song and popularity of the singer are highly correlated in popularity of a song.

I used “ggplot2” package to enhance my visualisations. I was interested in exploring characteristics of songs based on their genres and to focus my observation I used 4 main genres that I had created earlier. (I did not include pop genre as I found in essence, the popularity measure in this type depends on many factors and finding a distinguishable pattern is often need further analysis)

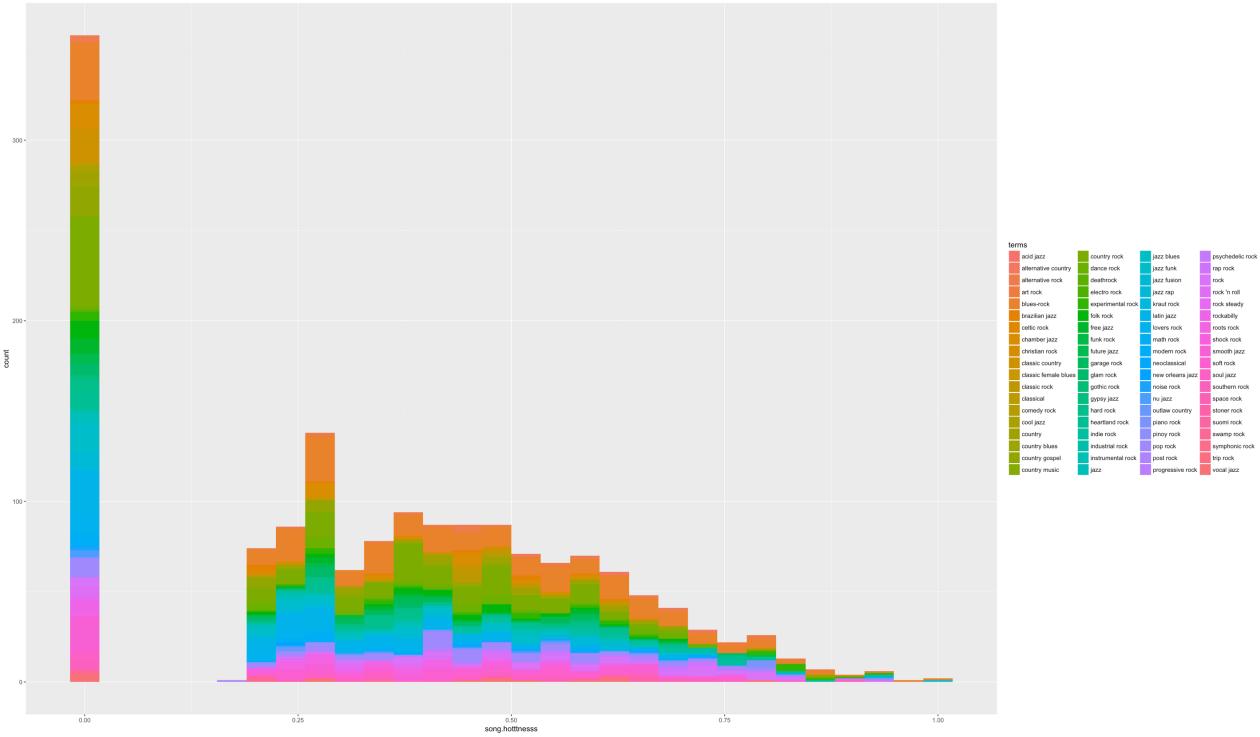



*Figure 1.8 Song popularity histogram indicating popularity of the songs and their genre*

Then I used scatter plot to determine the relations between loudness, tempo, popularity of the song  and their genre. My observation was as below :

- Most of the popular songs are loud
- Their tempo is most likely to falls between 80 to 180 BPM

Due to high number of result I could not able to have a perfect analysis on different genre, therefore, I make my observation more precise and checked for each 4 main genre.
I realised 2 genre of “jazz” and “rock” have more popular songs and pop-rock and jazz rap in these genre has more popular songs.


*Figure 1.9 Music genre analysis*

After my analysis on different genres I remove this column from the dataset because it is a nominal attribute and it has 460 distinct values. It will cause overlapping in clusters since genre inherently is an overlapped value between different songs.
As it can be seen from cleaned data, different attributes have different unites and scales. For example the unit of start_of_fade_out is in seconds while tempo is in BPM or beat per

minute. To tackle this issue I have used two methods. Firstly, I used Scaling which is subtract of each value divided by mean. Secondly, I used Normalising by defining a range (In this case -1 and 1 but normally it is between 0 and 1) for each attribute.
After experimenting with both of mentioned approaches, and comparing their k-means clustering results, I reached to the conclusion that normalising is better for the dataset because the ending clusters are less overlapped with Normalising method.
K-means is an R built in function which allows K-means clustering algorithm. One of the most important issue in K-means clustering is deciding the number of cluster because we are looking for well-separated clusters that representing songs with similar musical attributes and overlapping between clusters can be problematic.

## Post-Processing

There are ways to find the optimal number of clusters. 1.Average silhouette method 2.Gap statistic method 3.Elbow method

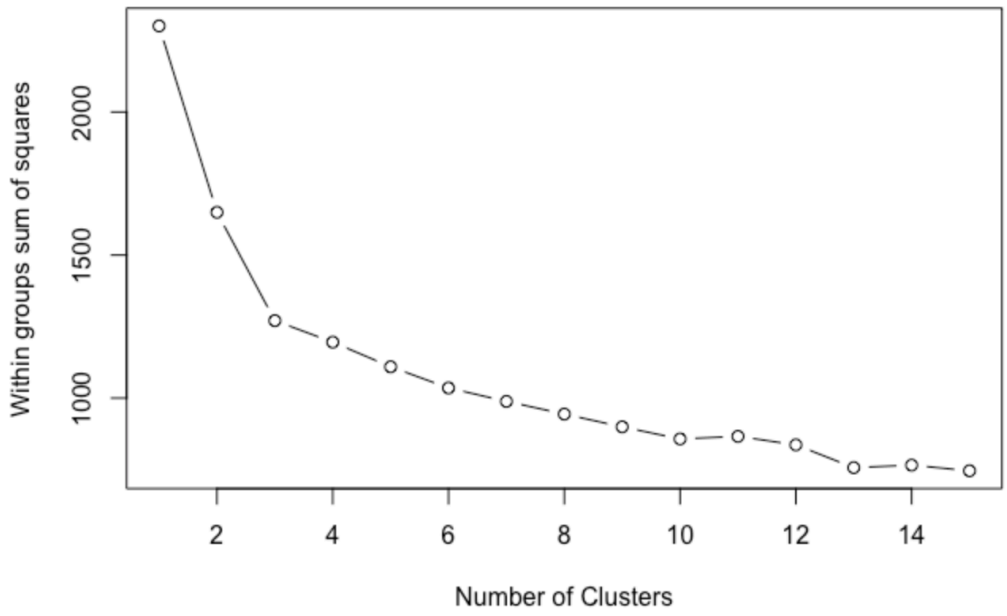



*Figure 1.10 Elbow point*

### Performance Metrics:

At first, I tried different methods to find the optimal number of cluster. For example according to Elbow Method, first Within sum of squares is calculated. Plotting this value with a range of numbers for clusters, will generate the following plot.

In the figure1.10, 4 will be a good number for clusters because after that point there is no drastic change.

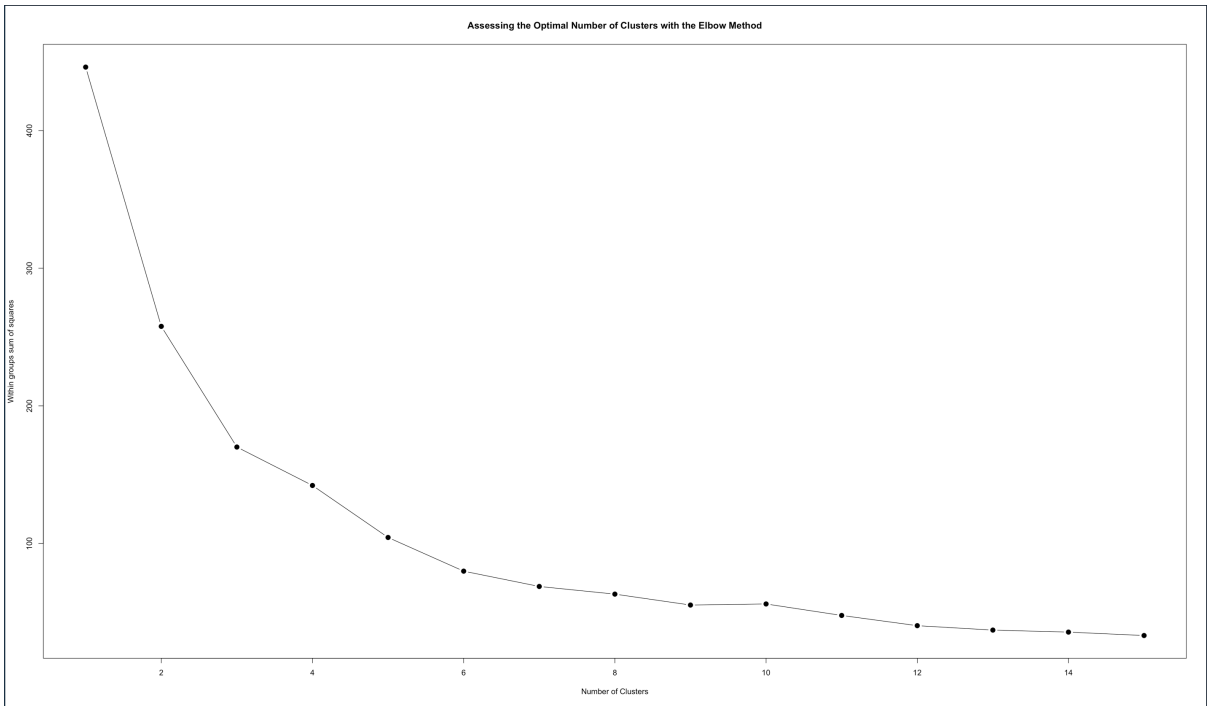



*Figure 1.11 Elbow point in normalised music data*

Figure 1.11 is for normalised music data. As the plot shows, between 4 and 6 clusters is considered optimal clustering.

After experimenting with different number of clusters in different plots based on different variables, I reached to the conclusion that due to high dimension of the data frame, elbow point is not the best approach for optimal number of cluster. I decided to plot the diagram based on different variables and then using the amount of overlapping between clusters to determine an optimal number for clusters.

At last I can check the quality of my clustering by within cluster sum of square error which indicates the distance of each member of the cluster from the mean (centre of cluster).

$$\sum_{x_i \in c} (x_i - \bar{x})^2$$

In my analysis 47.4 % was the value for within cluster sum of square error divided by total sum of square error. Less value of this shows more similarity within clusters.

### 1.3.4 Visualisation of the results in R

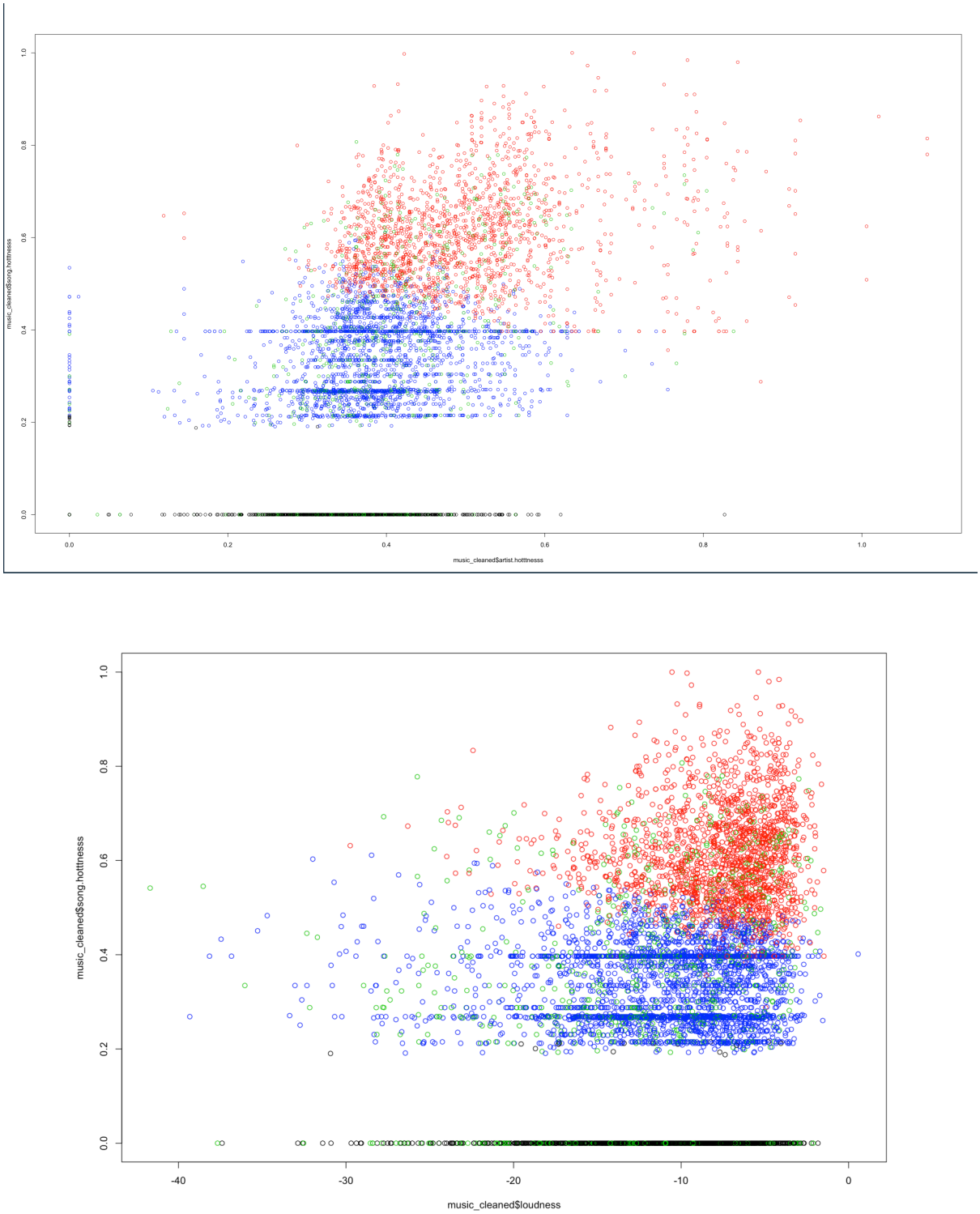






*Figure 1.12 K-means Clusters analysis for song popularity and artist popularity*


*Figure 1.13 K-means Clusters analysis for song popularity and song tempo*


*Figure 1.14 K-means Clusters and analysis for song popularity and song loudness*

Table 1-3 number of K-means Clusters

| Analysis Parameter | Number of clusters |
|---|---|
| Song popularity - Artist Popularity | 4 |
| Song popularity - Tempo | 7 |
| Song popularity - Loudness | 6 |

I managed to cluster songs with their 11 musical attributes that highly related to songs.
I used different number of clusters depending on different attributes analysis to have well-separated clusters. For example 4 clusters for song and artist popularity analysis divided songs to 4 groups. First 0 popularity, second below than 0.5 and third more than 0.5 and one overlapping. The group contains more popular songs shows that artist popularity is high in these songs and the loudness tend to be higher where low popularity songs which has lees than 0.5 on their popularity tend to have more broad range for their loudness.
The clusters can represent a playlist of songs that have similar musical attributes. Each pair can represent an specific playlist for example if a playlist based on tempo of songs desired, clustering with tempo can be used. In this case it will have seven different playlist that has similar musical attributes.

## 1.4 Data mining using SAS Enterprise Miner

### 1.4.1 The application of clustering with K-means Algorithm in SAS Enterprise Miner

My objective is to use K-means unsupervised algorithm to form clusters based on musical attributes and popularity. I am considering 11 different attributes to form my clusters. Songs falling into the same cluster will have similar musical characteristics (more familiar to their listeners) and have similar characteristics such as popularity rate, level of loudness and tempo.
K-means clusters can be viewed in relation to different attributes of the song and the key point is the notion of determining optimal number of k or number of clusters.
One of the application of this analysis would be creating playlists. Depending on the type of the playlist, it can have group of songs. These songs have similar musical features whether it be loudness, tempo, or popularity rate.
Another application of clustering can be exploring data for later analysis. For example, exploring characteristics of popular songs and use this cluster as a training set for another

supervised machine learning algorithm and since they have similar characteristics in 11 dimensions, later supervised learning results can be improved.

### 1.4.2 Train and test the model

Since I am doing a clustering I am using all of data set and there is no training and testing in my analysis.

*SAS Enterprise Miner* is using Hierarchical clustering for automatic clustering and if I change it to user specific clustering with my desired number of cluster, *SAS* will run k-mean clustering algorithm. It uses Ward's method which is agglomerative clustering algorithm which starts with n clusters and then progress until it reaches to 1 cluster.

Automatic clustering is not practical for large number of data due to different factors such as cpu time. I cleaned my data and used SAS for k-means clustering.

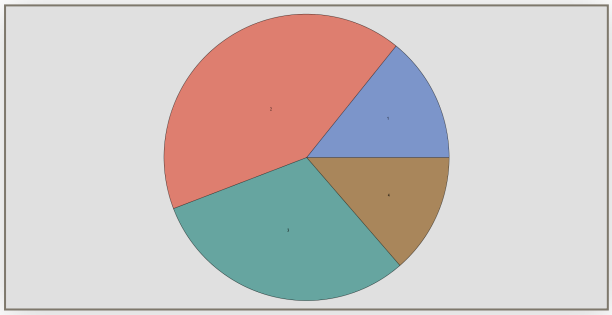
### 1.4.3 Visualisation of the results in SAS Enterprise Miner

For normalising data in SAS macro and standard procedure can be used. I used my normalised data which each value falls between the range of -1 and 1.

I used 4 number of clusters and here is the result:

Mean Statistics

| Clustering Criterion | Maximum Relative Change in Cluster Seeds | Improvement in Clustering Criterion | Segment Id | Frequency of Cluster | Root-Mean-Square Standard Deviation | Maximum Distance from Cluster Seed | Nearest Cluster | Distance to Nearest Cluster | artist.hotttnesss | bars_start | beats_start | duration | end_of_fade_in | loudness | song.hotttnesss | start_of_fade_out | tatums_start | tempo | time_signature |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.178935 | 0.037563 | . | 1 | 944 | 0.20954 | 2.871964 | 4 | 0.995568 | -0.27694 | -0.98423 | -0.92605 | -0.77335 | -0.96586 | 0.41933 | -0.36077 | -0.77983 | -0.94891 | -0.14998 | -1 |
| 0.178935 | 0.037563 | . | 2 | 2753 | 0.166417 | 2.397571 | 4 | 0.638902 | -0.13615 | -0.96658 | -0.93944 | -0.73196 | -0.96268 | 0.603553 | 0.054456 | -0.73906 | -0.9596 | -0.07085 | 0.013077 |
| 0.178935 | 0.037563 | . | 3 | 2004 | 0.174862 | 2.884931 | 4 | 0.6114 | -0.37522 | -0.96256 | -0.93744 | -0.73222 | -0.96199 | 0.501239 | -0.79015 | -0.73937 | -0.95881 | -0.12142 | 0.020126 |
| 0.178935 | 0.037563 | . | 4 | 907 | 0.18814 | 2.964776 | 3 | 0.6114 | -0.31102 | -0.94156 | -0.89983 | -0.74425 | -0.93283 | 0.174653 | -0.31543 | -0.75219 | -0.92515 | -0.29056 | -0.04998 |



Variable Importance

| Variable Name | Label | Number of Splitting Rules | Number of Surrogate Rules | Importance |
|---|---|---|---|---|
| loudness | | 7 | 4 | 1.00000 |
| tempo | | 3 | 10 | 0.94252 |
| song_hotttnesss | song.hotttnesss | 6 | 0 | 0.89100 |
| duration | | 0 | 10 | 0.78919 |
| artist_hotttnesss | artist.hotttnesss | 3 | 7 | 0.76083 |
| start_of_fade_out | | 0 | 8 | 0.75327 |
| beats_start | | 0 | 10 | 0.74315 |
| tatums_start | | 0 | 7 | 0.73031 |
| time_signature | | 1 | 2 | 0.65138 |
| bars_start | | 0 | 8 | 0.45177 |
| end_of_fade_in | | 0 | 7 | 0.23274 |

*Figure 1.15 K-means with 4 clusters in SAS Enterprise Miner*

There are different numbers of songs in each of these clusters. Based on the mean value of each attribute in each cluster, it can be seen that more popular songs are in second cluster, faster songs are more likely to be in second cluster and louder songs are in second cluster as well so it can be seen that these these attributes are correlated to each other. From the output

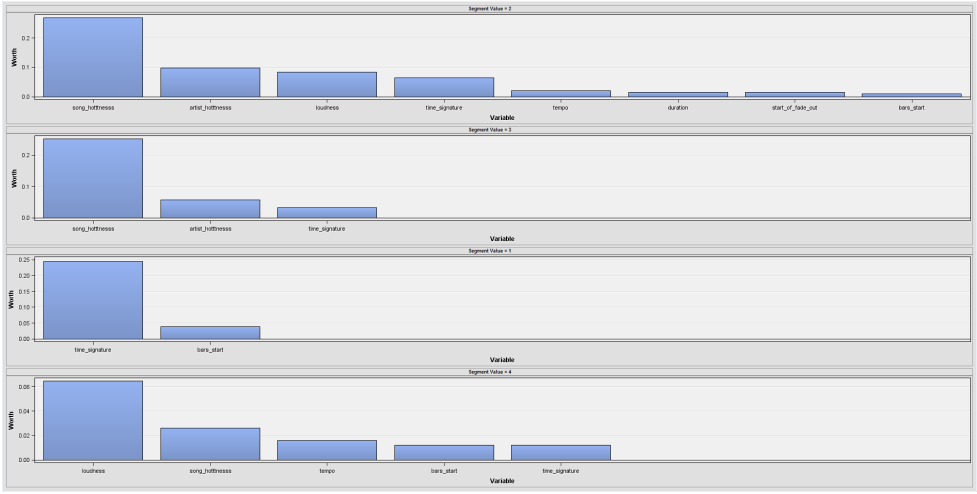


*Figure 1.16 Segment Profile Analysis (variable worth)*

window, it can be inferred that loudness , tempo and popularity of a song are top 3 most important elements of making clusters.

Lastly, the quality of each clusters can be checked by Mean Statistics. I evaluate the cluster by Root mean square. It has been used in clustering and it is a very popular performance metric. The only disadvantage for this performance metric is that it is sensitive to outlier. (Mean absolute square can be used as an alternative)

Here the Root mean squared is less than 20% for each cluster which shows the similarity within clusters.

After running segment profile node, I could visualise more information about clusters.

As it can be seen from Figure1.16, the weight that has been given to attributes in clustering is depicted.

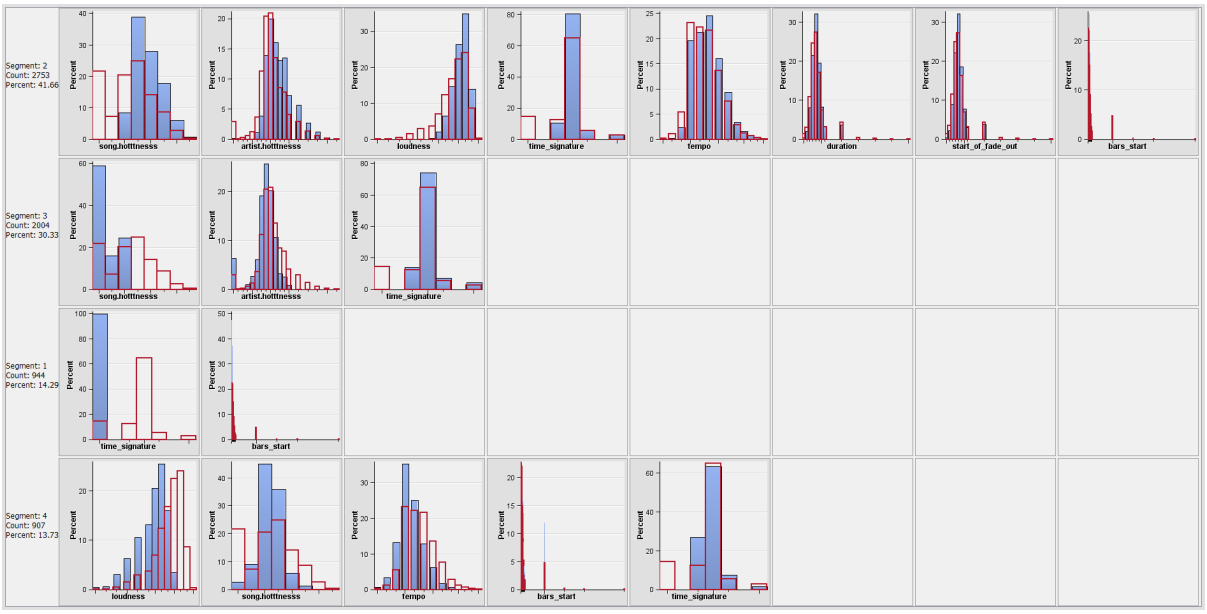



*Figure 1.17 Segment Profile Analysis*

Figure 1.17 visualise the difference between each cluster and the whole data set. The red bars indicate overall average which means some attributes are more likely to be found in some cluster/segments.

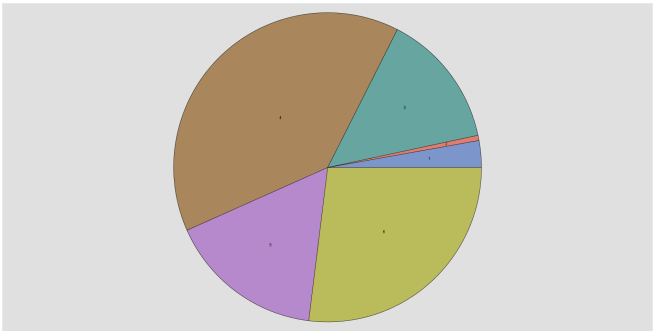
Notable Observation:

- Cluster 2 has the most popular songs and the highest tempo and loudness in this cluster is in the middle.
- Cluster 3 and 1 have the most unpopular songs.
- Cluster 4 has average songs in terms of loudness and song popularity
- Due to defined threshold there are only two attributes of time_signature and bars_start in Cluster 1.

For getting more detailed cluster I clustered my dataset one more time with 6 cluster. The process is almost Identical, but for 6 cluster. I briefly explain notable parts.

Mean Statistics

| Clustering Criterion | Maximum Relative Change in Cluster Seeds | Improvement in Clustering Criterion | Segment Id | Frequency of Cluster | Root-Mean-Square Standard Deviation | Maximum Distance from Cluster Seed | Nearest Cluster | Distance to Nearest Cluster | artist.hotttnesss | bars_start | beats_start | duration | end_of_fade_in | loudness | song.hotttnesss | start_of_fade_out | tatums_start | tempo | time_signature |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.170426 | 0.043348 | . | 1 | 186 | 0.227933 | 2.760649 | 5 | 1.082514 | -0.27223 | -0.91607 | -0.93716 | -0.72606 | -0.96938 | 0.435804 | -0.34615 | -0.73361 | -0.95774 | -0.09377 | 1 |
| 0.170426 | 0.043348 | . | 2 | 36 | 0.34712 | 1.94218 | 5 | 1.211685 | -0.32603 | -0.81063 | -0.19752 | -0.59544 | -0.52091 | 0.059363 | -0.39935 | -0.6099 | -0.22621 | -0.33602 | -0.42593 |
| 0.170426 | 0.043348 | . | 3 | 934 | 0.204031 | 1.676563 | 5 | 0.981805 | -0.27763 | -0.98597 | -0.93452 | -0.77513 | -0.96875 | 0.420691 | -0.36249 | -0.78143 | -0.95694 | -0.14778 | -1 |
| 0.170426 | 0.043348 | . | 4 | 2588 | 0.156822 | 2.328709 | 5 | 0.606412 | -0.12904 | -0.96733 | -0.93891 | -0.73357 | -0.96233 | 0.609671 | 0.065179 | -0.7406 | -0.95924 | -0.06958 | -0.01906 |
| 0.170426 | 0.043348 | . | 5 | 1082 | 0.164752 | 2.40482 | 6 | 0.589027 | -0.31704 | -0.9505 | -0.92141 | -0.7379 | -0.94645 | 0.240078 | -0.32529 | -0.74587 | -0.94669 | -0.27328 | -0.04713 |
| 0.170426 | 0.043348 | . | 6 | 1782 | 0.159919 | 1.669659 | 5 | 0.589027 | -0.37689 | -0.96562 | -0.93947 | -0.73523 | -0.96382 | 0.508758 | -0.81909 | -0.74229 | -0.96027 | -0.11233 | -0.02768 |



Variable Importance

| Variable Name | Label | Number of Splitting Rules | Number of Surrogate Rules | Importance |
|---|---|---|---|---|
| tempo | | 0 | 7 | 1.00000 |
| loudness | | 3 | 2 | 0.98246 |
| song_hotttnesss | song.hotttnesss | 4 | 1 | 0.88912 |
| duration | | 0 | 7 | 0.82488 |
| start_of_fade_out | | 0 | 6 | 0.82021 |
| artist_hotttnesss | artist.hotttnesss | 0 | 4 | 0.74581 |
| time_signature | | 2 | 0 | 0.71238 |
| beats_start | | 0 | 5 | 0.63320 |
| tatums_start | | 2 | 3 | 0.63314 |
| end_of_fade_in | | 0 | 6 | 0.49076 |
| bars_start | | 1 | 1 | 0.33813 |

*Figure 1.18 K-means with 6 clusters in SAS Enterprise Miner*

Notable Observation:

- Cluster 6 is more likely to have popular and loud songs
- Cluster 2 only have 36 observation and has the slowest songs in the dataset
- The start time of each beat is longer than any other clusters.

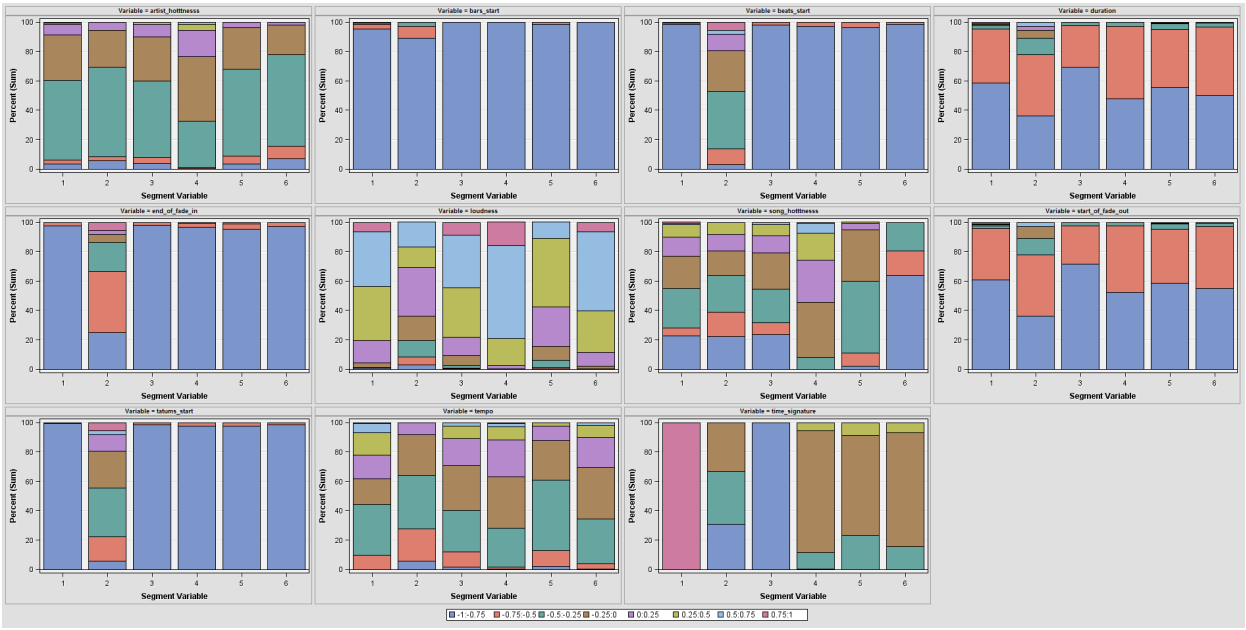



*Figure 1.19 Attribute percentage in each segment/cluster*

Figure 1.19 shows a different visualisation of portions of music features in each cluster. For example, segment 1 has different range of songs with different popularity value but the popularity value in cluster 6 is more consistent and dominated by the range of between 0.5 to 0.75.

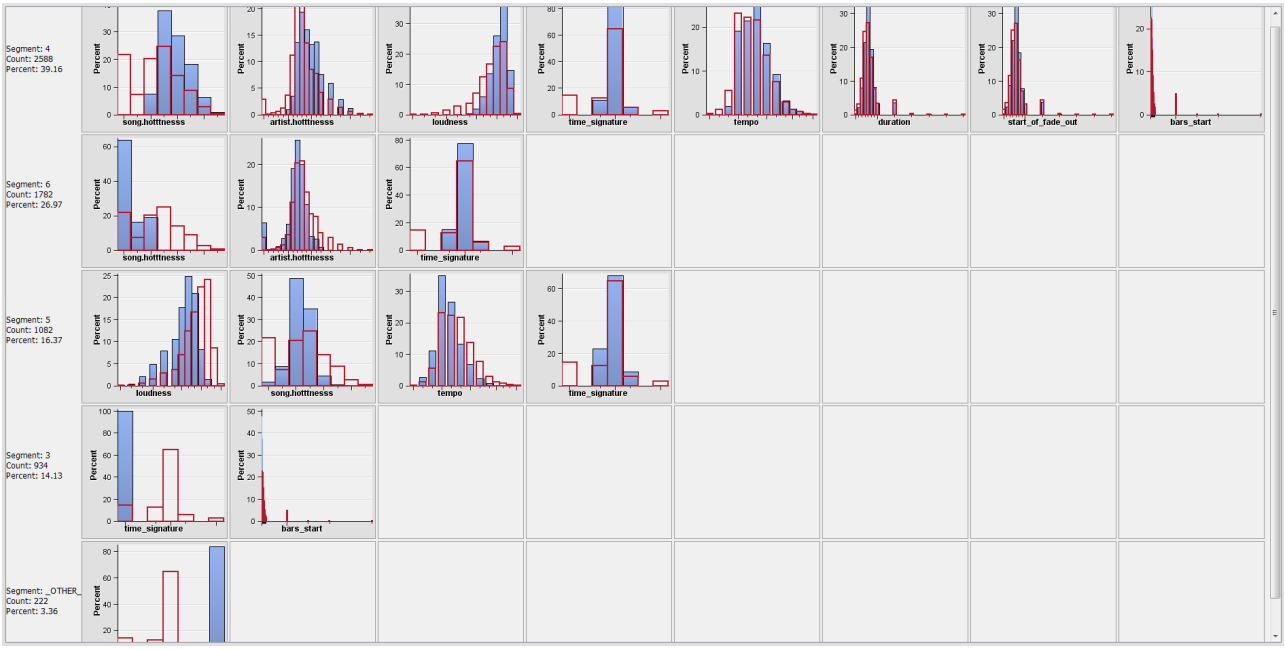



*Figure 1.20 Segment Profile Analysis*

**Evaluate Clustering Algorithms:**

- **Extrinsic:**

I can use clusters in another task as an input to another algorithm for example doing a classification within clusters to raise the accuracy of prediction with supervised learning.
I can also use clusters define outliers.

- **Intrinsic:**

They help to find patterns and relationships within the dataset and this can be quantitative and qualitative. I can use visualisation tools to find these patters within clusters.
I have both method in this research. I searched for patterns in clustering and among clusters and then I the last part of this research I will use clustering to improve classification tree.

### 1.5 Results analysis and Discussion

In clustering, each group should be well-separated and have considerable amount of observations in them. In non hierarchical clustering the number of cluster should be specified before using the algorithm and in R I have used many ways and lastly I use the visualisation of data to specify my cluster. In SAS I could use CCC[5] plot in hierarchical clustering to see the optimal number of clusters and change it according. For example, if optimal number of was picked from a local maximum.

#### 1.5.1 Comparison between observed results in R and *SAS Enterprise Miner*

*SAS* is more reliant on GUI[6] while R is mostly reliant on coding. I used same number of cluster in each tools in my analysis and the final results were almost the same. I managed to clusters songs which have certain level of popularity, tempo, and loudness into different clusters and both SAS and R provide me with well-separated clusters holding considerable amount of observation.
The main difference in SAS and R is representation. I could almost have the same analytics but in different representation and sometimes each tools have different functionality. For example segment plot in SAS gave the percentage of different range in different attributes in a dataset. This task is not simply provided in R.

[5] Cubic Clustering Criterion

[6] Graphical User Interface

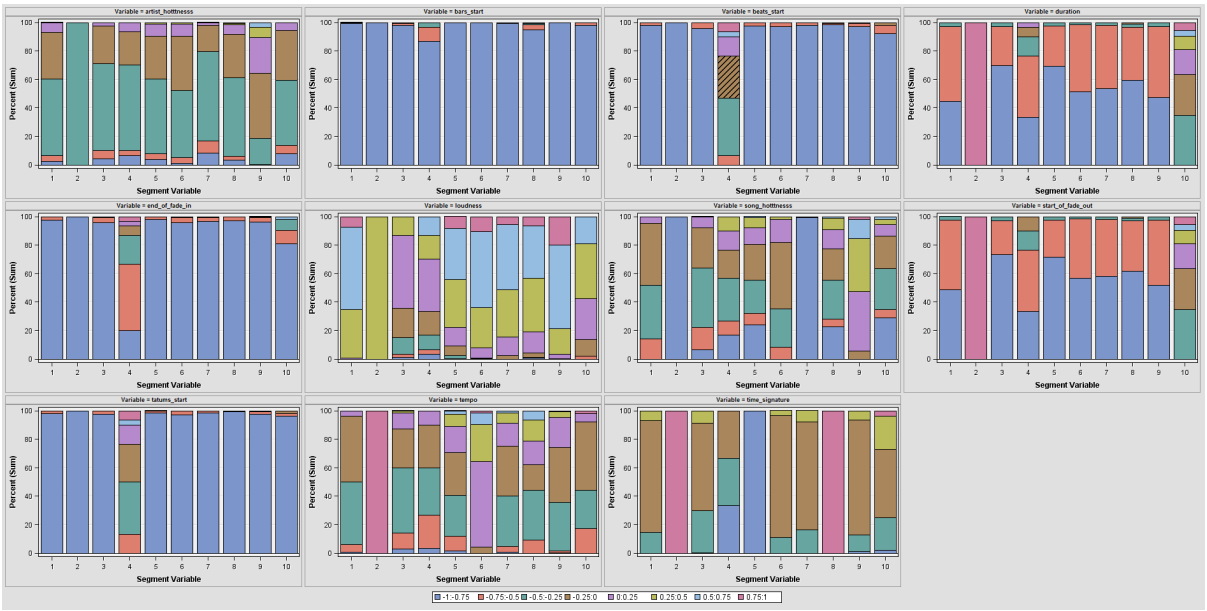



*Figure 1.21 Segment Plot for 10 Cluster in R*

## 2.1 Implementation of Supervised Learning in R (Decision Tree Algorithm)

### 2.1.1 Application of Classification in R with Decision tree Algorithm

In this section my objective is to predict the popularity of a songs based on its musical features. Due to my last analysis with unsupervised learning, I am aware of correlated variables. I am considering all 11 musical attributes to generate general models with higher level of accuracy. I am using confusion matrix for determining the level of accuracy, precision and recall of my model.
In this analysis, my I use 0.5 as a threshold for a song to be popular. If a song has a popularity of less than 0.5 in this analysis, I will classify that song as less popular song. ( low popularity) and I will classify songs with 0 popularity as unknown.
My primary variable for song popularity in music data set is a continuous variable, therefore regression tree will be more accurate option but to have more general model, I will transform this attribute to three class. High, Low, and Unknown songs.
I used 0.5 threshold to have more general model. I experiment with the model and higher threshold will result in higher level of accuracy but I decided to have generic model.

### 2.1.2 Explanation of the experimental procedure

After bringing up my normalised dataset from last section to R environment, I removed cluster column. I then defined popularity attribute which is based on the value of song.hotttness. If songhotttnesss is more than 0.5 the class number in popularity column is 2 if it is 0.5 or less than 0.5 it will get number 1 for popularity attribute and songs with the value of zero get zero in popularity column. Then I removed song.hotttnesss. Then to have a classification problem I change the class value to be a factor and then I sample music data to 90 percent of training set and 10 percent of testing set. Then I used “party” package and its “ctree” functionality to generate my model and decision tree.

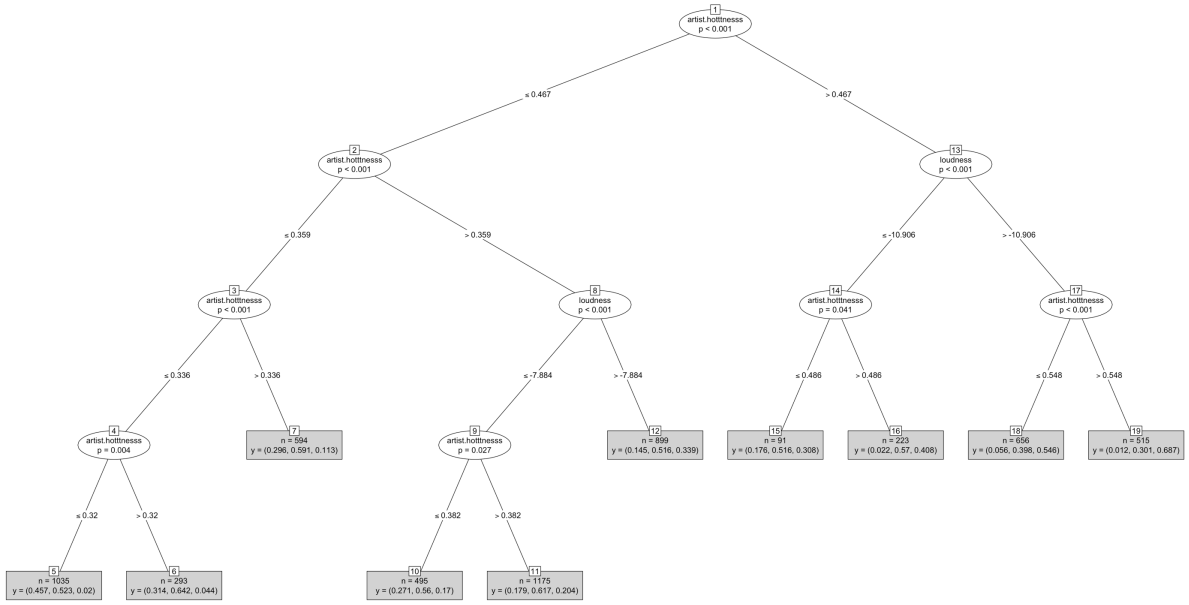



*Figure 2.1 Decision Tree with Party Package*

As it can be seen from figure 2.1, artist.hotttnesss was the most significant attribute to determine the value of song popularity. Then in the following nodes Artist popularity gets repeated along with loudness indicating the importance of these two attributes since they are used in different places for splitting.

It's nice to have a policy for determining the quality of nodes in decision tree. One of the simple policies is defining a threshold and then halt the growth of the tree if the population size was smaller than threshold. Another method is defining purity threshold and halt the growth, if the purity was smaller than minimum threshold. The important concept is to find the best quality split. Another approach is test statistics significance of the null hypothesis. In other words the left and right subgroup should be meaningful different and for that the null hypothesis should be rejected with high confidence (low value of p-value).

The association in "ctree" is done by a p-value and then a binary split implemented and recursively repeat last steps.

The split is done on attributes that has the lowest p-value and in this three the p-value of artist.hotttnesss is relatively small (0.001) and explains about the quality of split in root node.

**Performance metrics:**

I have used confusion matrix to check the accuracy, precision and recall of the model.

Accuracy:
The overall percentage of correct predictions compared to the class variable.
This can be misleading in some cases that some class variable are not found by model, or since I am using random training set and validation set, the distribution of each classes is random and imbalanced, therefore, I have used Precision and Recall metrics as well.
Precision:
Correct prediction over all prediction which gives in other words it gives the accuracy of prediction set.
Recall:
Correct prediction over all relative class variable in other words it gives the accuracy of prediction in general.
I have three distinct values for my class variable here is the formula for precision and recall in case of multi-value class variable:

$$\text{Precision}_i = \frac{M_{ii}}{\sum_j M_{ji}}$$

$$\text{Recall}_i = \frac{M_{ii}}{\sum_j M_{ij}}$$

And the accuracy remains the same as binary class variable as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy: 52.37%
Precision: 0 for Unknown, 54.29% for Low, and 45.03% for High which means the model is better in predicting songs with low popularity and could not find unknown songs. It is worth mentioning that these are in predicted set.
Recall: 0 for Unknown, 81.92% for Low, and 39.59% for High which means the model managed to predict 81.92% of all songs with class low (1) and 39.59% of popular song, which is relatively low, could not find unknown songs.

### 2.1.2.1 Enhancing classification

In this section I try to use different packages and different techniques to enhance the accuracy of my classification ion model.

#### 2.1.2.1.1 Decision Tree with RPART

In this section I am setting 70% of my data to be training set and 30% of my data to be testing set (Holdout Estimation). Then I use "RPART" package to produce the classification tree based on musical attribute. I use RPART.PLOT package to plot the classification tree.

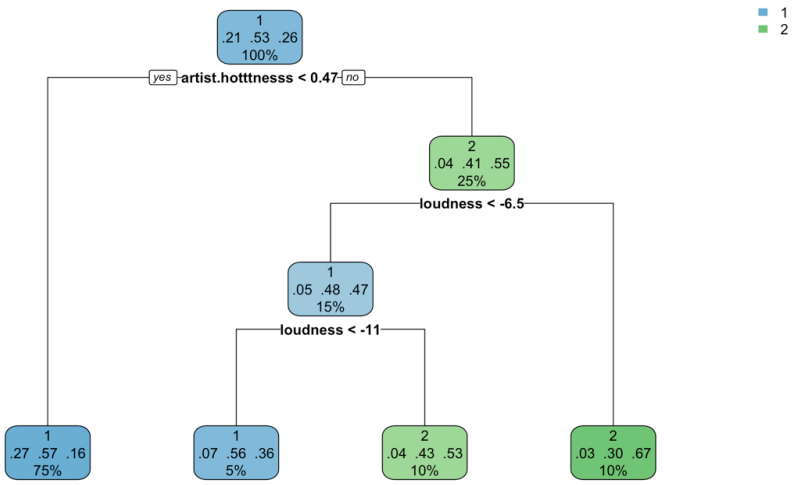



*Figure 2.2 Classification Tree with RPART.PLOT*

As it can be seen from figure 2.2 the decision tree is smaller than using "ctree" and the percentage in the leaf indicate the number of observation in this subgroup compared to all observation.
Then I used complexity parameter to analysis overfitting. I am looking to see if the error gets more as I train my model.

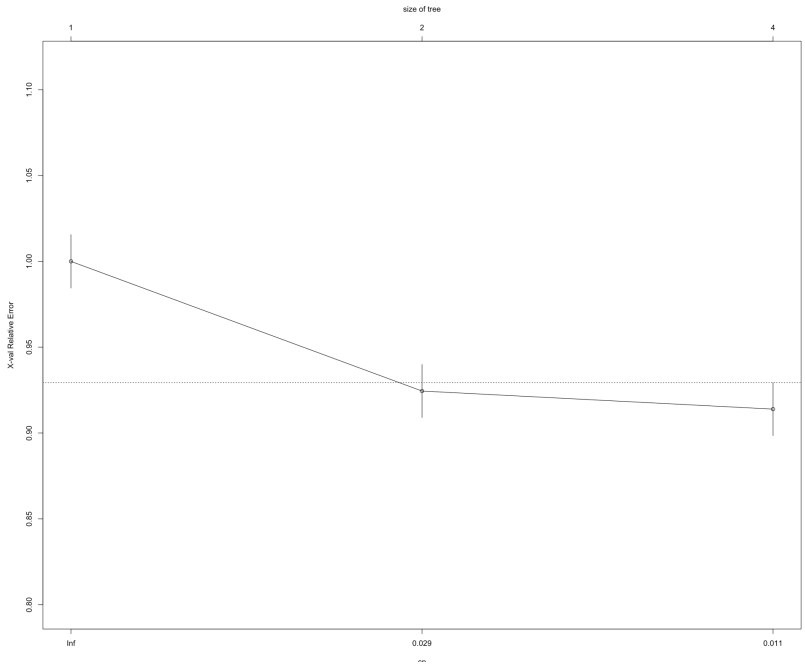



*Figure 2.3 Complexity parameter as a measure for pruning*

Figure 2.3 shows the amount of error is not getting worse, therefore, pruning is not changing the accuracy.
Then I prune the tree with "xerror" value provided by "RPART" package. The pruned tree is exactly the same and the validation set accuracy is the as before pruning which was 55.34%.

Precision: 0 for Unknown, 85.61%, for Low, and 43.24% for High which means the model is very good in predicting the low popular song and could not find unknown songs. It is worth mentioning that these are all in predicted set.
Recall: 0 for Unknown, 54.93% for Low, and 56.97% for High which means the model predict 54.93% of songs with low value in their popularity and 56.99 of songs with high value in popularity.

#### 2.1.2.1.2 Hybrid Approach (Combination of Supervised and Unsupervised Models)

In this section I will be using clustering to enhance the performance of classification tree to predict the popularity of the song.
I am setting 70% of my data to be training set and 30% of my data to be testing set (Holdout Estimation). I will be using dataset form clustering from the first part of this research.remove clusters attribute. Then I normalise training set and testing set. Then I use elbow method to find the optimal number of cluster (with within sum of square error performance metric for clusters) in both sets. I do the k-means clustering with 6 clusters and then add the number of cluster of each song to the training set and testing set. I set cluster attributes to be factor since I do not want have it as a continuous variable in the dataset. Then I calculate the popularity attribute and remove song.hotttnesss and the rest of the procedure is the same as before classification tasks. Here is the results:

It can be seen later later in the section 2.1.3 visualisation of the result in R, figure 2.4, that cluster 4 can determine 61.8% of popular songs (with high value) then in node 14 all of the songs are unknown. The accuracy of this model is 57.17% percent. The result for precision and recall are as below:

Precision:
Unknown: 22.53%
Low: 57.10%
High: 61.80%
Recall:
Unknown: 0.03%
Low: 75.79%
High: 66.21%

From the above number I can say this method has raised the overall performance of classification tree. This model will need more analysis for predicting unknown songs but for songs with low popularity, it classify 57.10% of results from its prediction set and 75.79% of all songs to low which is considerable. For popular songs the precision of 61.80% and recall of 66.21% are considerable as well.

## 2.1.3 Visualisation of the results in R

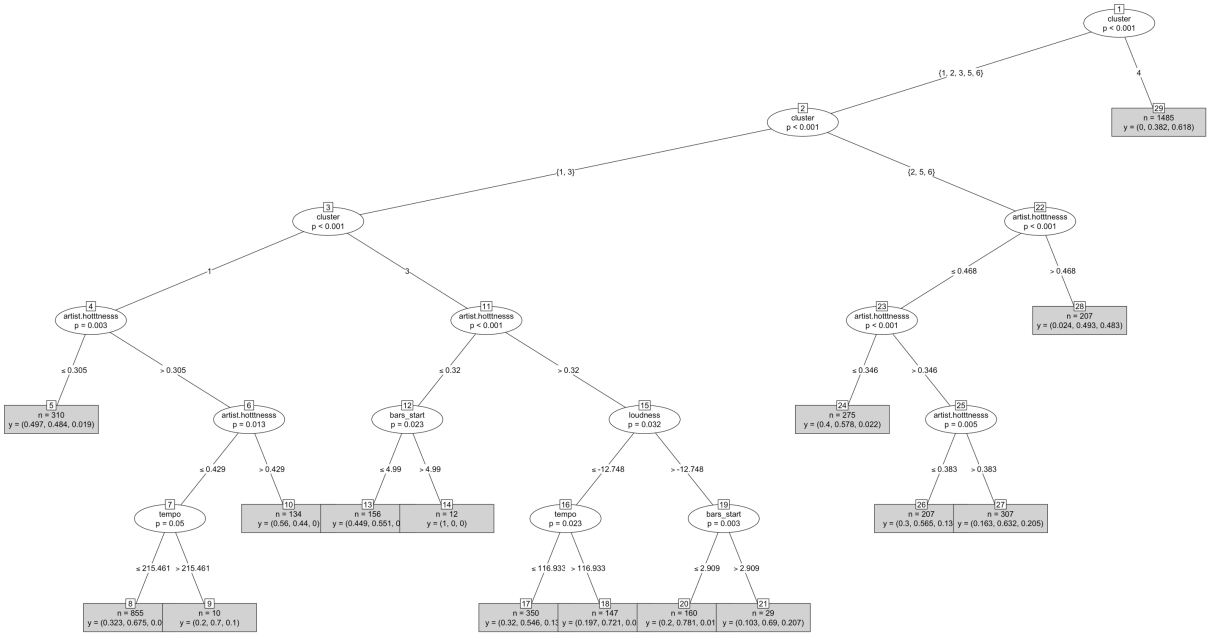



*Figure 2.4 Decision tree with hybrid approach*

## 2.2 Data mining using SAS Enterprise Miner

### 2.2.1 The application of classification tree in SAS Enterprise Miner

My objective is to use decision tree algorithm in SAS to predict popularity of the song. The data is the same data that had been used in R implementation with 11 variable (song.hotttness is replaced by categorical attribute, popularity, containing 0 for unknown, 1 for low popularity, and 2 for high popularity). I will try to have an optimised split in terms of purity and error.

One of the application of this model is to find which attributes are more influential for determining the popularity of a song and also to define how much these attributes are important and what is the relationship between these attributes.At the end I can use this model to predict the popularity of a song.

### 2.2.2 Train and test the model

In this section I try to use SAS Enterprise Miner and its built-in functionality to have more accurate model of my dataset. In this scenario I will not use test set because I am interested in finding a model with high level of accuracy and I try to avoid overfitting (with validation set) and I try to find a model that fits best my dataset but not overfit training data.

Ideally if very large data set is available, it is better to have test set to generalise the model for new data but in this section I am interested in training the model with train set and then fine-tuning it with validation set to compare it to my previous model and check if I can have quality split with pure nodes and minimal error. I set the training set to 90% and validation set to 10%.

### 2.2.3 Visualisation of the results in SAS Enterprise Miner

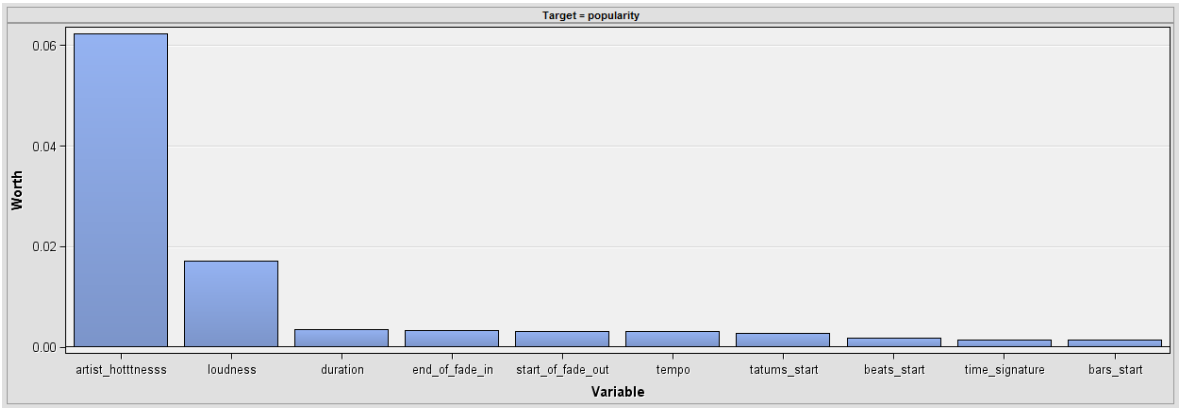



*Figure 2.5 Variable worth for Classification Tree*

I used statexplore node to explore my dataset for decision tree.
As it can be seen from statexplore result, artist.hotttnesss is by far the most important variable determining popularity of the song.
After partitioning data I ran the decision tree node.

| Variable | Role | Mean | Standard Deviation | Non Missing | Missing | Minimum | Median | Maximum | Skewness | Kurtosis |
|---|---|---|---|---|---|---|---|---|---|---|
| artist_hotttnesss | INPUT | 0.405341 | 0.130252 | 5944 | 0 | 0 | 0.394881 | 1.082503 | -0.05564 | 3.046657 |
| bars_start | INPUT | 1.063753 | 1.696891 | 5944 | 0 | 0 | 0.78334 | 59.74354 | 14.39844 | 347.1287 |
| beats_start | INPUT | 0.437233 | 0.546688 | 5944 | 0 | 0.01887 | 0.33213 | 12.24583 | 7.946676 | 113.5052 |
| duration | INPUT | 239.336 | 107.5691 | 5944 | 0 | 3.5522 | 225.593 | 1686.752 | 3.215951 | 25.99275 |
| end_of_fade_in | INPUT | 0.746564 | 1.801848 | 5944 | 0 | 0 | 0.194 | 36.49 | 6.400444 | 72.78849 |
| loudness | INPUT | -10.2101 | 5.199315 | 5944 | 0 | -41.691 | -9.148 | 0.566 | -1.30727 | 2.496565 |
| start_of_fade_out | INPUT | 230.5264 | 105.7295 | 5944 | 0 | 3.552 | 216.712 | 1686.752 | 3.299848 | 27.02138 |
| tatums_start | INPUT | 0.302245 | 0.527233 | 5944 | 0 | 0.01569 | 0.18969 | 12.24583 | 9.007928 | 133.8041 |
| tempo | INPUT | 123.8158 | 34.4762 | 5944 | 0 | 26.663 | 120.731 | 258.677 | 0.487015 | 0.186085 |
| time_signature | INPUT | 3.592026 | 1.244585 | 5944 | 0 | 1 | 4 | 7 | -0.61557 | 1.379747 |

*Figure 2.6 Output from statexplore*

| Target | Target Label | Fit Statistics | Statistics Label | Train | Validation | Test |
|---|---|---|---|---|---|---|
| popularity | | _NOBS_ | Sum of Frequencies | 5944 | 664 | . |
| popularity | | _MISC_ | Misclassification Rate | 0.426817 | 0.444277 | . |
| popularity | | _MAX_ | Maximum Absolute Error | 0.983607 | 0.966817 | . |
| popularity | | _SSE_ | Sum of Squared Errors | 3327.547 | 384.5531 | . |
| popularity | | _ASE_ | Average Squared Error | 0.186605 | 0.193049 | . |
| popularity | | _RASE_ | Root Average Squared Error | 0.431978 | 0.439373 | . |
| popularity | | _DIV_ | Divisor for ASE | 17832 | 1992 | . |
| popularity | | _DFT_ | Total Degrees of Freedom | 11888 | . | . |

*Figure 2.7 Fit stats for Classification Tree.*

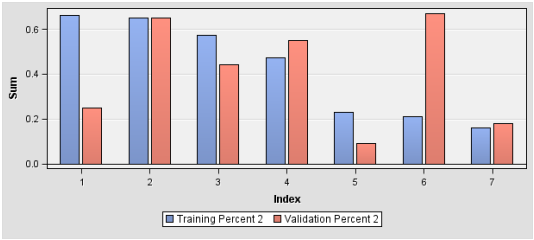



*Figure 2.8 Leaf Statistics*

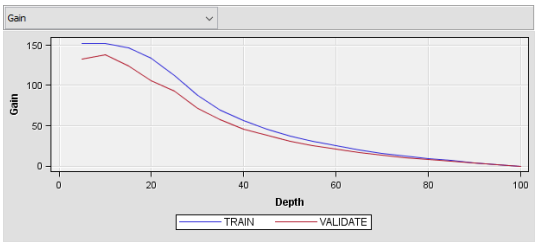



*Figure 2.9 Score Ranking Overlay for Popularity based on Gain or cumulative percent response*

Figure 2.8 shows the number of predicted and existed cases in each leaf and the difference show between two bars indicate error and as it can be seen leaf 2 has small error and leaf 6 has large error.
Figure 2.9 shows actual proportion with the primary outcome with the primary outcome at each decile (fraction of ranked data)
It is worth mentioning that *SAS Enterprise Miner* generates optimal tree using log worth for ordering in splitting. (SAS provide the functionality to create the tree and prune it manually as well)

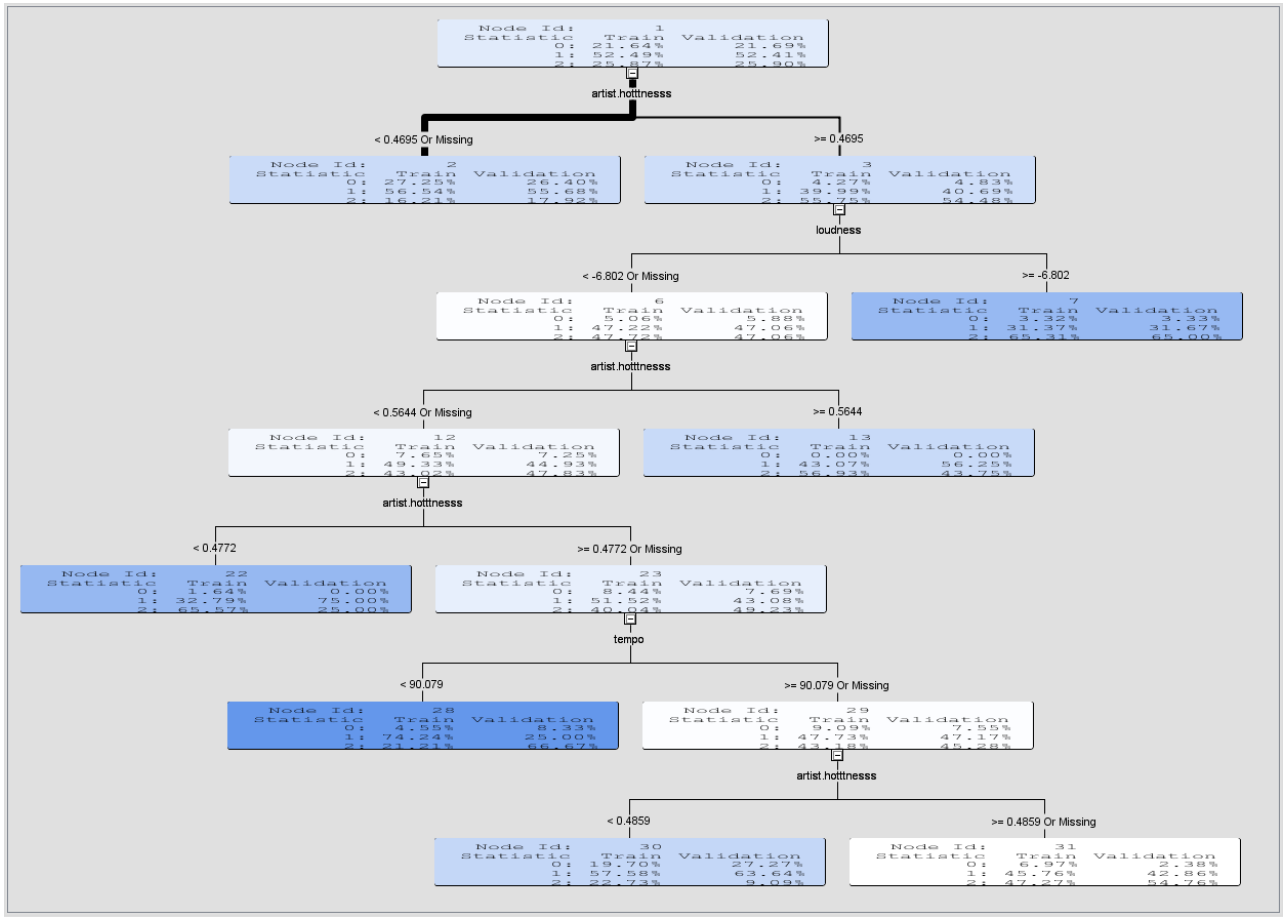



*Figure 2.10 Decision Tree with SAS Enterprise Miner*

The first node in Decision tree shows the 21.69% of songs popularity value is unknown, 52.41% has low popularity, 25.90% has high popularity. The thickness of the edge is indicator of number of observations and the intensity of colour indicates the intensity of prediction for a certain categorical value of 0, 1, or 2 in validation set. It has 7 leafs. The most Decision tree shows artist.hotttnesss is the most important variable music attribute in predicting the popularity of songs.

Each node can be better for certain prediction. For example, Node 22 is a important for predicting songs with low popularity because it managed to have 75% of them in validation set.

Tree Leaf Report

| Node Id | Depth | Training Observations | Training Percent 2 |
|---|---|---|---|
| 2 | 1 | 4491 | 0.16 |
| 7 | 2 | 663 | 0.65 |
| 31 | 6 | 330 | 0.47 |
| 13 | 3 | 267 | 0.57 |
| 28 | 5 | 66 | 0.21 |
| 30 | 6 | 66 | 0.23 |
| 22 | 4 | 61 | 0.66 |

*Figure 2.11 Output of Decision tree- Leaf Report*

In this model artist.hotttnesss, loudness, tempo are important for splitting, so they are influential in determining popularity. Other attribute that considered influential are mention is figure 2.12

Variable Importance

| Variable Name | Label | Number of Splitting Rules | Number of Surrogate Rules | Importance | Validation Importance | Ratio of Validation to Training Importance |
|---|---|---|---|---|---|---|
| artist_hotttnesss | artist.hotttnesss | 4 | 1 | 1.0000 | 1.0000 | 1.0000 |
| loudness | | 1 | 3 | 0.8875 | 0.8598 | 0.9689 |
| duration | | 0 | 5 | 0.8652 | 0.8541 | 0.9871 |
| start_of_fade_out | | 0 | 3 | 0.8481 | 0.8541 | 1.0070 |
| beats_start | | 0 | 2 | 0.8377 | 0.8134 | 0.9710 |
| bars_start | | 0 | 4 | 0.2540 | 0.2604 | 1.0251 |
| tempo | | 1 | 1 | 0.2528 | 0.2096 | 0.8292 |
| tatums_start | | 0 | 2 | 0.2401 | 0.2097 | 0.8737 |
| end_of_fade_in | | 0 | 1 | 0.2024 | 0.2119 | 1.0467 |

*Figure 2.12 Output of Decision tree- Variable Importance*

### 2.2.4 Results analysis and Discussion

Fit stats in figure 2.7 gives information about quality of classification tree. Different type of error can be seen. Misclassification Rate is 42% in training set (one of the most important performance metrics) and 44% in validation set which indicated the percentage when the model predicted wrong. Average Square error is another indicator of quality of decision tree which is 18% for training set and 19% for validation set. These measurements are performance metrics in this model. Also Classification Table is available in output windows in decision tree which shows values for confusion matrix. Target percentage is precision and

outcome percentage is recall. At the end I used Event classification table for calculating accuracy.

Classification Table

Data Role=TRAIN Target Variable=popularity Target Label=' '

| Target | Outcome | Target Percentage | Outcome Percentage | Frequency Count | Total Percentage |
|---|---|---|---|---|---|
| 0 | 1 | 26.8224 | 96.4230 | 1240 | 20.8614 |
| 1 | 1 | 56.8029 | 84.1667 | 2626 | 44.1790 |
| 2 | 1 | 16.3746 | 49.2198 | 757 | 12.7355 |
| 0 | 2 | 3.4822 | 3.5770 | 46 | 0.7739 |
| 1 | 2 | 37.3959 | 15.8333 | 494 | 8.3109 |
| 2 | 2 | 59.1219 | 50.7802 | 781 | 13.1393 |

Data Role=VALIDATE Target Variable=popularity Target Label=' '

| Target | Outcome | Target Percentage | Outcome Percentage | Frequency Count | Total Percentage |
|---|---|---|---|---|---|
| 0 | 1 | 26.0148 | 97.9167 | 141 | 21.2349 |
| 1 | 1 | 55.1661 | 85.9195 | 299 | 45.0301 |
| 2 | 1 | 18.8192 | 59.3023 | 102 | 15.3614 |
| 0 | 2 | 2.4590 | 2.0833 | 3 | 0.4518 |
| 1 | 2 | 40.1639 | 14.0805 | 49 | 7.3795 |
| 2 | 2 | 57.3770 | 40.6977 | 70 | 10.5422 |

As it can be seen from classification table, the model did not predict unknown songs. The precision for low songs is 55.16% and for highly popular songs is 57.37% in validation set . The recall was 85.91% for songs with low popularity and 40.69% for highly popular songs. The accuracy is 77.80% in validation set. These observations show the model has an acceptable level of accuracy and the recall value shows that 85.91% low popular songs can be predicted correctly.

In Event Classification Table there are values for confusion matrix for training set and validation set.

Data Role=TRAIN Target=popularity Target Label=' '

| False Negative | True Negative | False Positive | True Positive |
|---|---|---|---|
| 757 | 3866 | 540 | 781 |

Data Role=VALIDATE Target=popularity Target Label=' '

| False Negative | True Negative | False Positive | True Positive |
|---|---|---|---|
| 102 | 440 | 52 | 70 |

Validation set:
Precision: 57.37%
Recall: 40.69%
Accuracy: 76.80%

Training Set:
Precision: 59%
Recall: 50%
Accuracy: 78.17%

My objective was to train a model with high amount of data to have more accuracy. Training set has 78.17% accuracy and validation set has 76.80% of accuracy. It may seem that training model with more observation can raise the accuracy but it is worth mentioning that this model is not working properly for all class variable (unknown songs) in confusion matrix according to the values for precision and recall.

#### 2.2.4.1 Comparison between observed results in R and *SAS Enterprise Miner*

The accuracy in R implementation with "ctree" was 52.37% for validation set. Using *SAS Enterprise miner* The accuracy is 78% for validation set. I used the same portion for training and test set in both implementation, but in the tree in SAS was pruned while tree that was built with "ctree" was not pruned. In Hybrid approach in R, which had the best performance in R implementation, the accuracy was 57% which is again very lower than the accuracy of 78%. Nevertheless, I believe Hybrid approach is a better model due to other performance metrics that I have used such as Precision and Recall.

In both of these two metrics, Hybrid approach performed better and it shows that SAS model was able to predict one type of outcome while Hybrid approach in R was able to perform better with all class variable types.
The decision tree in R has 4 main musical attributes that are important for splitting and these are the same as 3 constituent musical attribute for decision tree in *SAS Enterprise Miner* which are artist.hotttnesss, loudness and tempo. ( R model has bar_starts as well).
In summary, the overall performance of model was better in Hybrid approach in R, but both model revealed significant information about the important values for predicting popularity of songs and offered an semi accurate model. I believe they accuracy level based on musical feature are significant but certainly can be improved with further analysis.

In general, selecting a tool for data mining is subjective. Both R Programming language and *SAS Enterprise miner* are powerful tools. From my point of view however, SAS is more reliant on GUI[7] while R is based on code and commands. SAS exploits loads of pre-defined functions and default setting, making it easier to start it. It is also capable of sophisticated algorithms and different measurements. R ,however, is more flexible and its open source feature makes it easier to find solution in communities. All in all, I believe both of them are powerful tools and capable of advanced data mining techniques and in the end the result that I have get from using same sample and algorithm was almost the same.

[7] Graphical User Interface

## 3. Conclusion

The objective of this research was to provide analysis on musical features of songs.
In the first part of my research, I used unsupervised learning to categorise songs to create playlist with similar musical features. SAS and R provide me different approach for analysis but the main application is the same.
I used SAS to see Song popularity, loudness and tempo and artist popularity are the most important attributes to determine different clusters. And in my R analysis I managed to plot different clusters to see their different attributes. One of the notable finding was that popular songs are more tend to be louder. Another application of my clustering analysis is to for playlist that have similar musical features. And I managed to show them both in R and SAS.
In SAS I used root mean square as my performance metrics and and in R I used within cluster sum of square and I checked for well-separated clusters visually. I managed to find cluster with considerable amount of similarity which can later be used for further research.
In the second part of my research, I categorised song popularity to three group of high, low and unknown. Then I tried to use classification tree to predict the popularity of the song based on its musical features. In R I used three different approach to predict popularity. The best performance was for the Hybrid approach using a combination of unsupervised learning for clustering songs and then predicting their popularity value. The result was improved for all categories of popularity and it is performs best for songs with low popularity. When I use SAS the accuracy was raised by almost 20% but since accuracy is not comprehensive performance metrics it did not include the fact that the decision tree in SAS was not able to identify unknown songs in terms of popularity. It performance was better songs with lower popularity and worse for songs with higher popularity comparing to the hybrid model, produced in R.
In summary, I have reached to consistent observations in both analysis in SAS and R. I believe this research can be improved by further studies in using hybrid methods.

# Reference :


Dieleman, S. (2014). *Recommending music on Spotify with deep learning*. [online] Available at: http://benanne.github.io/2014/08/05/spotify-cnns.html [Accessed 3 Dec. 2017].

Ellis, D., Whitman, B., Jehan, T. and Lamere, P. (2010). *The Echo Nest Musical Fingerprint*. [online] Available at: https://www.ee.columbia.edu/~dpwe/pubs/EllisWJL10-ENfprint.pdf [Accessed 13 Dec. 2017].

Bertin-Mahieux, T., Ellis, D. and Lamere, P. (2011). *THE MILLION SONG DATASET*. [online] Available at: http://tbertinmahieux.com/Papers/ismir11.pdf [Accessed 1 Dec. 2017].

Jannach, D., Kamehkhosh, I. and Bonnin, G. (2014). *Analyzing the Characteristics of Shared Playlists for Music Recommendation*. [online] Available at: http://ceur-ws.org/Vol-1271/Paper1.pdf [Accessed 1 Dec. 2017].

Pennacchioli, D., Rossetti, G., Pappalardo, L., Pedreschi, D., Giannotti, F. and Coscia, M. (n.d.). *The patterns of musical influence on the Last.Fm social network*. [online] Available at: https://lucapappalardo.files.wordpress.com/2014/08/lastfm1.pdf [Accessed 2 Dec. 2017].

Çano, E. and Morisio, M. (n.d.). *Music Mood Data Creation Based on LAST.FM Tags*. [online] Available at: http://airccj.org/CSCP/vol7/csit76803.pdf [Accessed 2 Dec. 2017].

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2000). *CRISP-DM 1.0*. [online] Available at: https://www.the-modeling-agency.com/crisp-dm.pdf [Accessed 1 Dec. 2017].

Brocklebank, J. (2017). *Data Mining*. [online] Available at: http://www2.sas.com/proceedings/sugi22/DATAWARE/PAPER128.PDF [Accessed 1 Dec. 2017].

SAS Institute Inc, 2011. Getting Started with SAS® Enterprise Miner™ 7.1. 1st electronic book ed. United States: SAS®.

Jehan, T. (2014). *Analyzer Documentation*. [online] Available at: http://docs.echonest.com.s3-website-us-east-1.amazonaws.com/_static/AnalyzeDocumentation.pdf [Accessed 5 Dec. 2017].

Sanchez, G. (2013). *Handling and Processing Strings in R*. [online] Available at: http://www.gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf [Accessed 6 Dec. 2017].

peng, R. (2016). *Exploratory Data Analysis with R*. [ebook] leanpub, pp.5-17. Available at: https://leanpub.com/exdata [Accessed 9 Dec. 2017].

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K., Studer, M., Roudier, P. and Gonzalez, J. (2017). *Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al*. [online] Available at: https://cran.r-project.org/web/packages/cluster/cluster.pdf [Accessed 2 Dec. 2017].

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. 2nd ed. Springer.

Therneau, T., Atkinson, E. and Foundation, M. (2017). *An Introduction to Recursive Partitioning Using the RPART Routines*. [online] Available at: https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf [Accessed 15 Dec. 2017].

Hothorn, T., Hornik, K. and Zeileis, A. (n.d.). *party: A Laboratory for Recursive Partytioning*. [online] Available at: https://cran.r-project.org/web/packages/party/vignettes/party.pdf [Accessed 2 Dec. 2017].

Zeileis, A., Hothorn, T. and Hornik, K. (n.d.). *party with the mob: Model-Based Recursive Partitioning in R*. [online] Available at: https://cran.r-project.org/web/packages/party/vignettes/MOB.pdf [Accessed 9 Dec. 2017].

Revelle, W. (2017). *An introduction to the psych package: Part II Scale construction and psychometrics*. [online] Available at: https://cran.r-project.org/web/packages/psych/vignettes/overview.pdf [Accessed 7 Dec. 2017].

Milborrow, S. (2016). *Plotting rpart trees with the rpart.plot package*. [online] Available at: http://www.milbo.org/rpart-plot/prp.pdf [Accessed 9 Dec. 2017].

Tibshirani, R., Walther, G. and Hastie, T. (2000). *Estimating the number of clusters in a data set via the gap statistic*. [online] Available at: http://venus.unive.it/romanaz/edami/letture/gap_statistic.pdf [Accessed 2 Dec. 2017].

Fu, W. and Perry, P. (2017). *Estimating the number of clusters using cross-validation*. [online] Available at: https://arxiv.org/pdf/1702.02658.pdf [Accessed 2 Dec. 2017].

Yan, M. (2005). *Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion*. [online] Available at: https://pdfs.semanticscholar.org/8492/87862e01639378d2301fe5489378df4adf59.pdf [Accessed 13 Dec. 2017].

Anon, (n.d.). *STAT 505*. [online] Available at: https://onlinecourses.science.psu.edu/stat505/ [Accessed 18 Dec. 2017].

Erlandson, E. (2016). *Measuring Decision Tree Split Quality with Test Statistic P-Values*. [online] Available at: http://erikerlandson.github.io/blog/2016/05/26/measuring-decision-tree-split-quality-with-test-statistic-p-values/ [Accessed 10 Dec. 2017].

Singh, S. and Giri, M. (2014). *Comparative Study Id3, Cart And C4.5 Decision Tree Algorithm: A Survey*. [online] Available at: http://www.ijaist.com/index.php/publications/2014/2014-july/item/download/538_cda535df3e9ac96a64797dd24c1aa3fe [Accessed 23 Dec. 2017].

Beleites, C., Salzer, R. and Sergo, V. (2013). Validation of Soft Classification Models using Partial Class Memberships: An Extended Concept of Sensitivity & Co. applied to the Grading of Astrocytoma Tissues. [online] Available at: https://arxiv.org/pdf/1301.0264.pdf [Accessed 18 Dec. 2017].

Amigo, E., Gonzalo, J. and Artiles, J. (2008). *A comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints Technical Report*. [online] Available at: http://cs.utsa.edu/~qitian/seminar/Spring11/03_11_11/IR2009.pdf [Accessed 21 Dec. 2017].

Delling, D., Gaertler, M., Gorke, R., Nikoloski, Z. and Wagner, D. (n.d.). *How to Evaluate Clustering Techniques*. [online] Available at: https://pdfs.semanticscholar.org/1466/c1320f9626059e5939d394c601a008c09103.pdf [Accessed 25 Dec. 2017].

Chai, T. and Draxler, R. (2014). *Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature*. [online] Available at: https://www.geosci-model-dev-discuss.net/7/C591/2014/gmdd-7-C591-2014-supplement.pdf [Accessed 19 Dec. 2017].

Georges, J., Thompson, J. and Wells, C. (2010). Applied Analytics Using SAS Enterprise Miner 5.3 Course Notes. Cary, NC: SAS Institute.

Appendix A: R Script for Unsupervised Learning ( Clustering )

```
#installing packages
#for string manipulation
install.packages("stringr")
#for k-means clustering algorithm
install.packages("cluster")
#for filetring based on string in datasets
install.packages("dplyr")
#for plotting
install.packages("psych")
install.packages("ggplot2")

#using respective packages

library(stringr)
library(cluster)
library(dplyr)
library(psych)
library(ggplot2)

#reading data set

music <- read.csv("music.csv", header=T)

#selecting musiclly significant features for analysis

music_cleaned <- music[,c(1, 7, 9, 10, 11, 18, 24,26, 28, 29, 30, 32)]

#check if there is any NA in the dataset
any(is.na(music_cleaned))

#checking the structure and dimention of the dataset

dim(music_cleaned)
str(music_cleaned)

#getting the number  of NAs
sum(is.na(music_cleaned$song.hotttnesss))

#checking the number of missing values for each column
```

```
sum(is.na(music_cleaned$artist.hotttnesss))
sum(is.na(music_cleaned$bars_start))
sum(is.na(music_cleaned$beats_start))
sum(is.na(music_cleaned$duration))
sum(is.na(music_cleaned$end_of_fade_in))
sum(is.na(music_cleaned$loudness))
sum(is.na(music_cleaned$song.hotttnesss))
sum(is.na(music_cleaned$start_of_fade_out))
sum(is.na(music_cleaned$tatums_start))
sum(is.na(music_cleaned$tempo))
sum(is.na(music_cleaned$time_signature))
sum(is.na(music_cleaned$terms))


#using stringr and dplyr retrun a separate dataset if the track has one of the 5 main genre
type in its terms column
#Rock dataset

contains_rock <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "rock"))
nrow(contains_rock)

#Jazz dataset

contains_Jazz <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "jazz"))
nrow(contains_Jazz)

#Pop dataset

contains_Pop <- music_cleaned %>%
filter(str_detect(music_cleaned$terms, "pop"))
nrow(contains_Pop)

#classic dataset

contains_Classic <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "classic"))
nrow(contains_Classic)
#country dataset
```

```
contains_Country <- music_cleaned %>%
  filter(str_detect(music_cleaned$terms, "country"))
nrow(contains_Country)

#calculate the mean for each main genres

Rock_Song_hotness_mean <- mean(contains_rock$song.hotttnesss, na.rm = TRUE)


Jazz_Song_hotness_mean <- mean(contains_Jazz$song.hotttnesss, na.rm = TRUE)


Pop_Song_hotness_mean <- mean(contains_Pop$song.hotttnesss, na.rm = TRUE)


Classic_Song_hotness_mean <- mean(contains_Classic$song.hotttnesss, na.rm = TRUE)


Country_Song_hotness_mean <- mean(contains_Country$song.hotttnesss, na.rm = TRUE)

#3.impute the mean to every NA row
length <- nrow(music_cleaned)

#for Rock Tracks

for(i in 1:length)
{
  if(is.na(music_cleaned$song.hotttnesss[i]))
  {
    if(grepl("rock", music_cleaned$terms[i]))
      {
         music_cleaned$song.hotttnesss[i] <- Rock_Song_hotness_mean
      }
  }
}

#remaining NA rows
#sum(is.na(music_cleaned))
#for Jazz Tracks
```

```
for(i in 1:length)
{
  if(is.na(music_cleaned$song.hotttnesss[i]))
  {
    if(grepl("jazz", music_cleaned$terms[i]))
    {
      music_cleaned$song.hotttnesss[i] <- Jazz_Song_hotness_mean
    }
  }
}

#remaining NA rows
sum(is.na(music_cleaned))

#for POP Tracks

#for(i in 1:length)
#{
#  if(is.na(music_cleaned$song.hotttnesss[i]))
#  {
#    if(grepl("pop", music_cleaned$terms[i]))
#    {
#     music_cleaned$song.hotttnesss[i] <- Pop_Song_hotness_mean
#    }
#  }
#}
#remaining NA rows
#sum(is.na(music_cleaned))
#for Classical Tracks
for(i in 1:length)
{
  if(is.na(music_cleaned$song.hotttnesss[i]))
  {
    if(grepl("classic", music_cleaned$terms[i]))
    {
      music_cleaned$song.hotttnesss[i] <- Classic_Song_hotness_mean
    }
  }
}
#remaining NA rows
sum(is.na(music_cleaned))
```

```
#for country Tracks
for(i in 1:length)
{
  if(is.na(music_cleaned$song.hotttnesss[i]))
  {
    if(grepl("country", music_cleaned$terms[i]))
    {
      music_cleaned$song.hotttnesss[i] <- Country_Song_hotness_mean
    }
  }
}
#remaining NA rows
#sum(is.na(music_cleaned))
#getting the number  of NAs
sum(is.na(music_cleaned$song.hotttnesss))


#eliminating NA left in  the dataset
music_cleaned <- music_cleaned[complete.cases(music_cleaned),]
#checking the number of zero values for the column when zero value is meaningless
sum(music_cleaned$loudness == 0)
sum(music_cleaned$tempo ==0)
#removing zeros in the rows
#music_cleaned <- music_cleaned[-which(music_cleaned$song.hotttnesss == 0),]
#using subset to eliminate zeros (alternative to perivious line)
#music_cleaned <- subset(music_cleaned,song.hotttnesss!=0)
music_cleaned <- subset(music_cleaned,tempo!=0)
music_cleaned <- subset(music_cleaned,loudness!=0)
#num of complete case (without NA)
sum(as.numeric(complete.cases(music_cleaned)))

head(music_cleaned)
#getting the pair of clusters
pairs(music_cleaned)
#finding correlation in pairs of cluster with psych package as a better alternative to previous
line
pairs.panels(music_cleaned)
#scatter plot for a 3 most influentional attributes for song popularity (checking data)
qplot(music_cleaned$artist.hotttnesss, music_cleaned$song.hotttnesss)
qplot(music_cleaned$tempo, music_cleaned$song.hotttnesss)
qplot(music_cleaned$loudness, music_cleaned$song.hotttnesss)
```

```
#backing up the dataset
music_cleaned_backup <- music_cleaned
music_cleaned_backup$terms <- as.factor(music_cleaned_backup$terms)
str(music_cleaned_backup)
#forming mainGenre dataset containing 5 main genre
mainGenre <- rbind(contains_rock, contains_Jazz, contains_Classic, contains_Country)
dim(mainGenre)

#song popularity histogram in different genre
qplot(song.hotttnesss, data = mainGenre, fill = terms)
#Drawing plot for highly related attributes to song popularity in 5 main genre
qplot(song.hotttnesss, loudness, data = mainGenre , color = mainGenre$terms)
qplot(song.hotttnesss, tempo, data = mainGenre , color = mainGenre$terms)

#Drawing plot for highly related attributes to song popularity in each genre
qplot(song.hotttnesss, loudness, data = contains_rock , color = contains_rock$terms)
qplot(song.hotttnesss, loudness, data = contains_Jazz , color = contains_Jazz$terms)
qplot(song.hotttnesss, loudness, data = contains_Classic , color = contains_Classic$terms)
qplot(song.hotttnesss, loudness, data = contains_Country , color =
contains_Country$terms)
qplot(song.hotttnesss, loudness, data = contains_Pop , color = contains_Pop$terms)

qplot(song.hotttnesss, tempo, data = contains_rock , color = contains_rock$terms)
qplot(song.hotttnesss, tempo, data = contains_Jazz , color = contains_Jazz$terms)
qplot(song.hotttnesss, tempo, data = contains_Classic , color = contains_Classic$terms)
qplot(song.hotttnesss, tempo, data = contains_Country , color = contains_Country$terms)
qplot(song.hotttnesss, loudness, data = contains_Pop , color = contains_Pop$terms)


#removing the nominal value
music_cleaned <- music_cleaned[,-c(11)]

#export fully cleaned dataset to a csv file
write.csv(music_cleaned, file='musicCleaned.csv')

#normalizing with range between -1 and 1 (for more accurate)
col <- ncol(music_cleaned)
newmin = -1
newmax= 1
#initializing
```

```
nrml_music <- 0
#normalizing
for(j in 1:col){
  oldmin <- min(music_cleaned[j])
  oldmax <- max(music_cleaned[j])
  current <- music_cleaned[j]
  temp <- ((current-oldmin)/(oldmax-oldmin))*(newmax-newmin) + newmin
  temp <- as.data.frame(temp)
  nrml_music <- cbind(nrml_music, data.frame(temp))
}
#removing extra columns
nrml_music <- nrml_music[,-c(1)]
write.csv(nrml_music, file='normalizedMusic.csv')
#check if the range is correct
min(nrml_music$song.hotttnesss)
max(nrml_music$song.hotttnesss)
#standardisation with substraction of mean and division by standard deviation with scale
function (as an alternative to normalization)
#less accurate in this case
#means = apply(music_cleaned, 2, mean)
#sds = apply(music_cleaned, 2, sd)
#nrml_music = scale(music_cleaned, center = means, scale=sds)
#calculating the distance matrix
distance = dist(nrml_music)
set.seed(123)
#clustring the normalized dataset into three cluster with k-means package
kc <- kmeans(nrml_music,4)
#checking the optimal number of clusters with using within sum of square
mydata <- nrml_music[1:3]
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares",
     main="Assessing the Optimal Number of Clusters with the Elbow Method",
     pch=20, cex=2)
wss

#center of each cluster
kc$centers
```

```
#distribution of terms(genre in clusters)
table(music_cleaned$tempo, kc$cluster)


#results
plot(kc$centers)
# 4 clusters for the relation between artist popularity and song popularity

plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$tempo, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$loudness, data =
music_cleaned,col=kc$cluster)
# 7 clusters for the relationship between popularity of the song and its tempo
kc <- kmeans(nrml_music,9)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$tempo, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$loudness, data =
music_cleaned,col=kc$cluster)
# 6 clusters for the relationship between popularity of the song and its loudness
plot(music_cleaned$song.hotttnesss ~ music_cleaned$loudness, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=kc$cluster)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$tempo, data =
music_cleaned,col=kc$cluster)
#adding each song cluster in song popularity/artist papularity analysis
music_cleaned$cluster <- kc$cluster
#finding cluster with highest song popularity
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=music_cleaned$cluster==3)


#exporting
write.csv(music_cleaned,
file='musicCleanedWithPopularityAndClusterUsingUnsupervisedClustering.csv')
```

Appendix B: R Script for Supervised Learning (Classification ) And Hybrid Approach

```
#installing packages
#for decision tree
install.packages("party")
#for decision tree
installed.packages("rpart")
#for better plotting
install.packages(“rpart.plot")

#loading packages

library(party)
library(rpart)
library(rpart.plot)

#reading the cleaned data with clustering

music_cleaned <- read.csv("musicCleanedWithClusterUsingdClustering.csv", header=T)

#removing row numebr column
music_cleaned <- music_cleaned[,-1]
music_cleaned_new <- music_cleaned

#songs with popularity of unknown, low, and popular --classification rather than regression)

music_cleaned$popularity <- ifelse(music_cleaned$song.hotttnesss > 0.5,
2,ifelse(music_cleaned$song.hotttnesss ==0, 0,1))
#checking songs based on three categories in songs
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=music_cleaned$popularity==0)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=music_cleaned$popularity==1)
plot(music_cleaned$song.hotttnesss ~ music_cleaned$artist.hotttnesss, data =
music_cleaned,col=music_cleaned$popularity==2)
#removing continuous target variable
music_cleaned$song.hotttnesss <- NULL

#backing up cluster for later analysis and then removing it
music_cleaned_cluster<- music_cleaned$cluster
```

```
music_cleaned$cluster <- NULL
str(music_cleaned)

#changing class variable to factor for classification

music_cleaned$popularity <-as.factor(music_cleaned$popularity)

#drawing the decision tree

set.seed(123)
#sampling music data to training and validating set for Estimation --to avoid overfitting--
pd <- sample(2,nrow(music_cleaned), replace=TRUE, prob= c(0.9,0.1))
trainingSet <- music_cleaned[pd==1,]
validationSet <- music_cleaned[pd==2,]
#using party package for classification
popularity_tree <- ctree(popularity ~ . , trainingSet)
plot(popularity_tree, type="simple")
str(music_cleaned)
#training model
trainTab<-table(predict(popularity_tree), trainingSet$popularity)
trainTab
#accuracy of trainig set
sum(diag(trainTab))/sum(trainTab)
#accuracy of classifaction in decision tree on test/validation set
validationTab<-table(predict(popularity_tree, newdata= validationSet),
validationSet$popularity)
#accuracy of classifaction in decision tree on test/validation set
validationTab
accuracy_of_decision_tree = sum(diag(validationTab))/sum(validationTab)


#precision
precision_of_decision_tree = diag(validationTab) / rowSums(validationTab)
#precision for each class
precision_of_decision_tree["0"]
precision_of_decision_tree["1"]
precision_of_decision_tree["2"]
#recall
recall_of_decision_tree <- (diag(validationTab) / colSums(validationTab))
#recall for each class
recall_of_decision_tree["0"]
```

```
recall_of_decision_tree["1"]
recall_of_decision_tree["2"]


#decision tree with rplot and pruning it
set.seed(123)
#sampling music data to training and validating set for Estimation --to avoid overfitting--
# 70% of training set and 30% of validation/testing set- Hold out
pd <- sample(2,nrow(music_cleaned), replace=TRUE, prob= c(0.7,0.3))
trainingSet <- music_cleaned[pd==1,]
validationSet <- music_cleaned[pd==2,]

#training the model
popularity_tree <- rpart(popularity ~ . , data=trainingSet, method = "class")
popularity_tree
rpart.plot(popularity_tree)
#printing complexity parameter
printcp(popularity_tree)

#drawing the complexity parameter

plotcp(popularity_tree)
#classification with rpart package
prediction <- predict(popularity_tree, validationSet, type="class")

#checking the accuracy of model validation/test set

validationTab <-table(validationSet$popularity, predicted = prediction)
sum(diag(validationTab))/sum(validationTab)

#pruning decision tree by the least amount of error

ptree<-
prune(popularity_tree,cp=popularity_tree$cptable[which.min(popularity_tree$cptable[,"xer
ror"]),"CP"])
#plotting new pruned tree
rpart.plot(ptree)

#testing/validation with pruned decision tree
prediction <- predict(ptree, validationSet, type = "class")
```

```
#accuracy on testing/validation data
validationTab <- table(validationSet$popularity, predicted= prediction)
accuracy_of_decision_tree <- (diag(validationTab))/sum(validationTab)

#precision

precision_of_decision_tree <- diag(validationTab) / rowSums(validationTab)
#precision for each class
precision_of_decision_tree["0"]
precision_of_decision_tree["1"]
precision_of_decision_tree[“2"]

#recall

recall_of_decision_tree <- (diag(validationTab) / colSums(validationTab))
#recall for each class
recall_of_decision_tree["0"]
recall_of_decision_tree["1"]
recall_of_decision_tree["2"]


#enhancing with hybrid appraoch
#using clustering as an helping factor in decision making

set.seed(123)
#sampling data frame to 70% of training set and 30% of validation/testing set- Holdout
pd <- sample(2,nrow(music_cleaned_new), replace=TRUE, prob= c(0.7,0.3))
trainingSet <- music_cleaned_new[pd==1,]
validationSet <- music_cleaned_new[pd==2,]

#removing previous clusters

trainingSet$cluster <- NULL
validationSet$cluster <- NULL

#normalisation using scaling

means = apply(trainingSet, 2, mean)
sds = apply(trainingSet, 2, sd)
nrml_music = scale(trainingSet, center = means, scale=sds)
means2 = apply(validationSet, 2, mean)
```

```
sds2 = apply(validationSet, 2, sd)
nrml_music2 = scale(validationSet, center = means2, scale=sds2)

#checking the elbow point for best number of k for k-means clustering

mydata <- trainingSet[1:3]
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares",
     main="Assessing the Optimal Number of Clusters with the Elbow Method",
     pch=20, cex=2)


mydata <- validationSet[1:3]
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares",
     main="Assessing the Optimal Number of Clusters with the Elbow Method",
     pch=20, cex=2)

#clustering each set with efficient number of cluster--from The Elbow Method individually

k <- kmeans(nrml_music,6)
k2 <- kmeans(nrml_music2 ,6)
#adding cluster number to each song of samples

trainingSet$cluster <- k$cluster
validationSet$cluster <- k2$cluster
#changing cluster numbers to be a factor
trainingSet$cluster <-as.factor(trainingSet$cluster)
validationSet$cluster <-as.factor(validationSet$cluster)

#definition of popularity measures

trainingSet$popularity <- ifelse(trainingSet$song.hotttnesss > 0.5,
2,ifelse(trainingSet$song.hotttnesss ==0, 0,1))
```

```
validationSet$popularity <- ifelse(validationSet$song.hotttnesss > 0.5,
2,ifelse(validationSet$song.hotttnesss ==0, 0,1))
#changing the problem from regression to classification

trainingSet$popularity <- as.factor(trainingSet$popularity)
validationSet$popularity <- as.factor(validationSet$popularity)
#removing song popularity

trainingSet$song.hotttnesss <- NULL
validationSet$song.hotttnesss <- NULL
#trainig the model/drawing decision tree

popularity_tree <- ctree(popularity ~ ., trainingSet)
#plotting the tree

plot(popularity_tree, type="simple")

#checking the accuracy for training set
trainTab<-table(predict(popularity_tree), trainingSet$popularity)
trainTab
#accuracy of classifaction in decision tree
sum(diag(trainTab))/sum(trainTab)
validationTab<-table(predict(popularity_tree, newdata= validationSet),
validationSet$popularity)
accuracy_of_decision_tree <- sum(diag(validationTab))/sum(validationTab)

#precision

precision_of_decision_tree <- diag(validationTab) / rowSums(validationTab)
#precision for each class
precision_of_decision_tree["0"]
precision_of_decision_tree["1"]
precision_of_decision_tree[“2"]

#recall

recall_of_decision_tree <- (diag(validationTab) / colSums(validationTab))
#recall for each class
recall_of_decision_tree["0"]
recall_of_decision_tree["1"]
recall_of_decision_tree["2"]
```