Analysis of Spotify data

Introduction

The succesfulness of the song is determined by various audio features which makes it to go up the ladder in the Top Billboard charts. The goal of the analysis is to determine the audio features of the song and their contibution to the Successfulness of the song. The various audio features include the danceability level, energy, tempo, valence, acousticness, liveliness, loudness etc. which contributes a major factor to make a song successful.

Here we see the analysis of the data that was gathered using Billboard's Charts from Jan 1, 2000 - Dec. 31, 2005. The data set includes 350 songs that peaked within the Top 10 during that time. It also has 241 songs that peaked between 30-40 on the charts during that time.

```r
library(httr)
library(jsonlite)
library(curl)
```

```
##
## Attaching package: 'curl'
```

```
## The following object is masked from 'package:httr':
##
##     handle_reset
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.1
```

```r
library(reshape2)
library(stringr)
#install.packages("fuzzyjoin",repos='http://cran.us.r-project.org')
#install.packages("stringdist",repos='http://cran.us.r-project.org')
library(stringdist)
```

```
## Warning: package 'stringdist' was built under R version 3.3.1
```

```r
library(fuzzyjoin)
```

```
## Warning: package 'fuzzyjoin' was built under R version 3.3.1
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.1
```

```r
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.3.1
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

Spotify API configuration

The Spotify API configuration is the first and foremost step to form an interface with Spotify and R for analysis purpose. The key configurations are the Client key and client secret key given while creating the application in Spotify developer console.

The Request and Response urls are configured to get the playlist IDs and track IDS of the song that we have imported in our playlist. The authentication key called Oauth token is used to authenticate our user profile and request before it gives its response.

```
response = POST('https://accounts.spotify.com/api/token',
             accept_json(),
             authenticate(spotifyKey, spotifySecret),
             body = list(grant_type = 'client_credentials'),
             encode = 'form',
             verbose()
)
```

```
spotifyID <- "22vj7pwybpow4rrgw6acnxolq"
spotifyPlaylist <- "7ckg79YQQcX6yiCGNbrRBx"
artists_name <- data.frame()
name_track <- data.frame()
for(offset in c(0,100,200,300,400,500)){
        playlistTracksURL <- paste("https://api.spotify.com/v1/users/",
                                   spotifyID,
                                   "/playlists/",
                                   spotifyPlaylist,
                                   "/tracks?offset=",offset,
                                   sep="")



        HeaderValue = paste0('Bearer ', mytoken)
        getTracks <- GET(url=playlistTracksURL, add_headers(Authorization = Header
Value))
        info <- jsonlite::fromJSON(toJSON(content(getTracks)))
        track <- unlist(info$items$track$id)
        name <- unlist(info$items$track$name)
        name_track1 <- as.data.frame(cbind(as.character(name),as.character(track))
)
        name_track <- as.data.frame(rbind(name_track,name_track1))

        artists <- as.character()
        if(offset == 500){
                for(i in 1:82){

                        artists <- paste0(artists,unlist(info$items$track$artists[[i]][4][[1]][1]),sep=",")


                }} else {
                        for(i in 1:100){

                                artists <- paste0(artists,unlist(info$items$track$artists[[i]][4][[1]][1]),sep=",")



                        }
                }




        artists <- as.data.frame(unlist(strsplit(artists, ",", fixed = TRUE)))

        artists_name <- as.data.frame(rbind(artists_name,artists))


}

artists_name <- as.data.frame(artists_name[-c(334,335,344,348,349),])
name_track <- as.data.frame(cbind(name_track,artists_name))
colnames(name_track) <- c("Songs","TrackID","Artist")
name_track <- name_track[order(name_track$Songs),]
trackIDs <- unlist(as.character(name_track$TrackID))
```

```
audio_feat_op <- data.frame()
track_str <- trackIDs[1]
for (i in 2:100) {

  track_str <- paste(track_str, trackIDs[i], sep=",")
}

trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))

track_str <- trackIDs[101]
for (j in 102:200){
        track_str <- paste(track_str, trackIDs[j], sep=",")
}
trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
```

```
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))

track_str <- trackIDs[201]
for (j in 202:300){
        track_str <- paste(track_str, trackIDs[j], sep=",")
}
trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))

track_str <- trackIDs[301]
for (j in 302:400){
        track_str <- paste(track_str, trackIDs[j], sep=",")
}
trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))

track_str <- trackIDs[401]
for (j in 402:500){
        track_str <- paste(track_str, trackIDs[j], sep=",")
}
trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))

track_str <- trackIDs[501]
for (j in 502:582){
        track_str <- paste(track_str, trackIDs[j], sep=",")
}
trackFeaturesURL <- paste("https://api.spotify.com/v1/audio-features/?ids=",
                           track_str,
                           sep="")

getSongFeats <- GET(url=trackFeaturesURL, add_headers(Authorization = HeaderValue))
info2 <- jsonlite::fromJSON(toJSON(content(getSongFeats)))
audio_features <- as.data.frame(info2)
audio_feat_op <- as.data.frame(rbind(audio_feat_op,audio_features))
song_feat <- as.data.frame(cbind(name_track,audio_feat_op))
rownames(song_feat) <- 1:nrow(song_feat)
```

```
colnames(song_feat)[1] <- "Title"
```

Importing the songs to be analysed

The list of songs that were to be analysed based and determine its stand on succesfulness in Billboard is imported and the songs that were not on Spotify were removed from our file.

```
song_list <- song_list[-c(1,19,96,102,189,295,339,345,498,507,519,591),]
rownames(song_list) <- 1:nrow(song_list)
song_list$Title <- gsub( "[^[:alnum:],]", " ", song_list$Title )
song_feat$Title <- gsub( "[^[:alnum:],]", " ", song_feat$Title )
```

Merging songs list with their features

The song file and the songs from the spotify playlist are merged on the basis of the title.

```
songs <- unique(stringdist_inner_join(stringdist_semi_join(song_list,song_feat,by=c("Title","Artist"),max_dist=200),stringdist_semi_join(song_feat,song_list,by=c("Title","Artist"),max_dist=200),by=c('Title','Artist')))

songs <- data.frame(songs$Title.x,songs$Artist.x,songs$Peak,as.Date(songs$Date.Entered,format = '%m/%d/%y'),as.factor(songs$Successful),songs$Title.y,as.character(songs$TrackID),songs$Artist.y,unlist(songs$audio_features.danceability),unlist(songs$audio_features.energy),unlist(songs$audio_features.key),unlist(songs$audio_features.loudness),as.factor(unlist(songs$audio_features.mode)),unlist(songs$audio_features.speechiness),unlist(songs$audio_features.acousticness),unlist(songs$audio_features.instrumentalness),unlist(songs$audio_features.liveness),unlist(songs$audio_features.valence),unlist(songs$audio_features.tempo),unlist(songs$audio_features.type),unlist(songs$audio_features.id),unlist(songs$audio_features.uri),unlist(songs$audio_features.track_href),unlist(songs$audio_features.analysis_url),unlist(songs$audio_features.duration_ms),as.factor(unlist(songs$audio_features.time_signature)))

songs <- setNames(songs, c("Title x","Artist x","Peak","Date Entered","Successfulness","Title y","Track ID","Artist y","Danceability","Energy","Key","Loudness","Mode","Speechiness","Acousticness","Instrumentalness","Liveliness","Valence","Tempo","Type","ID","URI","Track_href","Analysis_url","Duration","TimeSignature"))
```

Determining the structure of the songs data

The structure of the songs data can be analysed by using the str() function which displays the classes of each column.

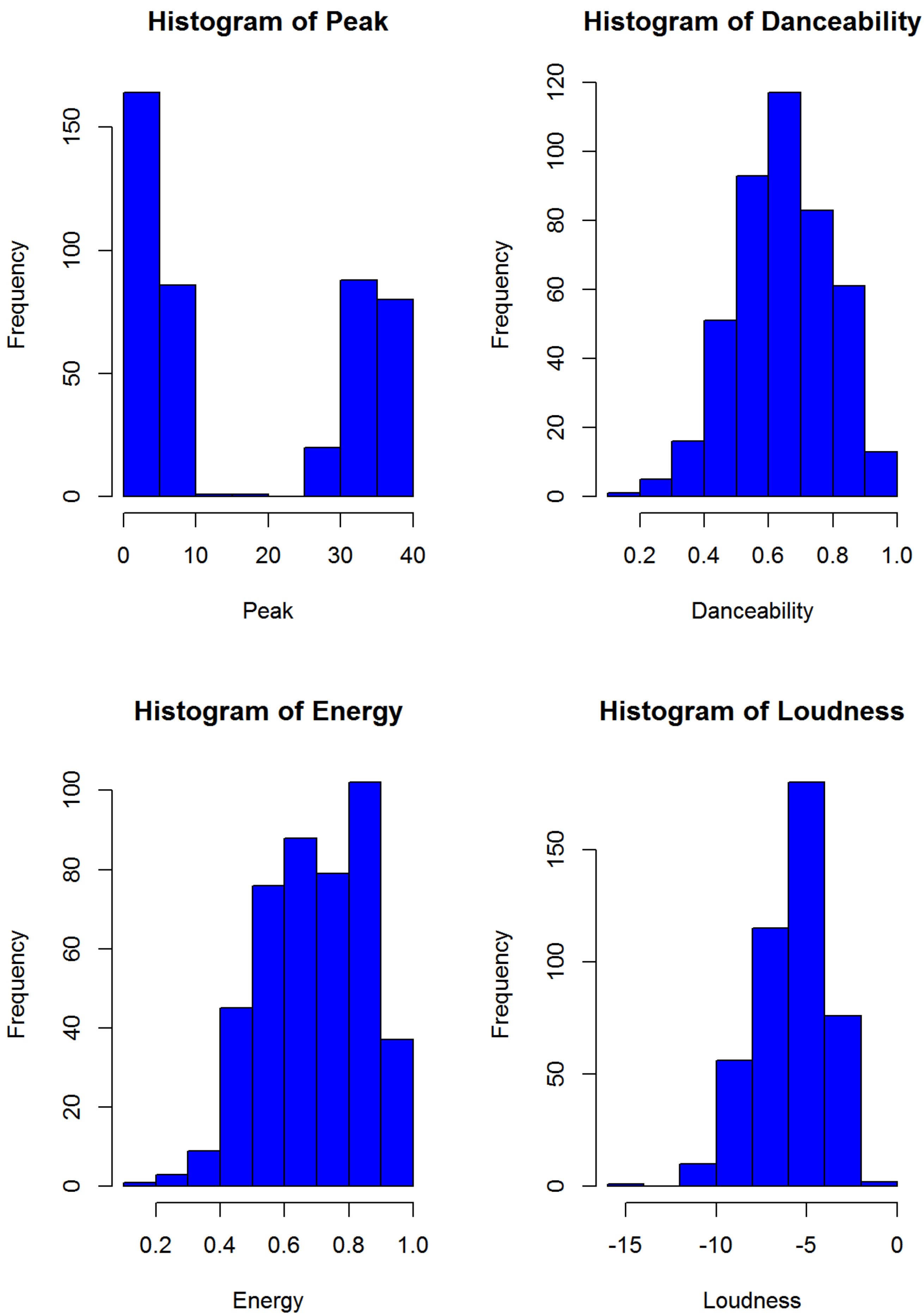
```
str(head(songs,1))
```

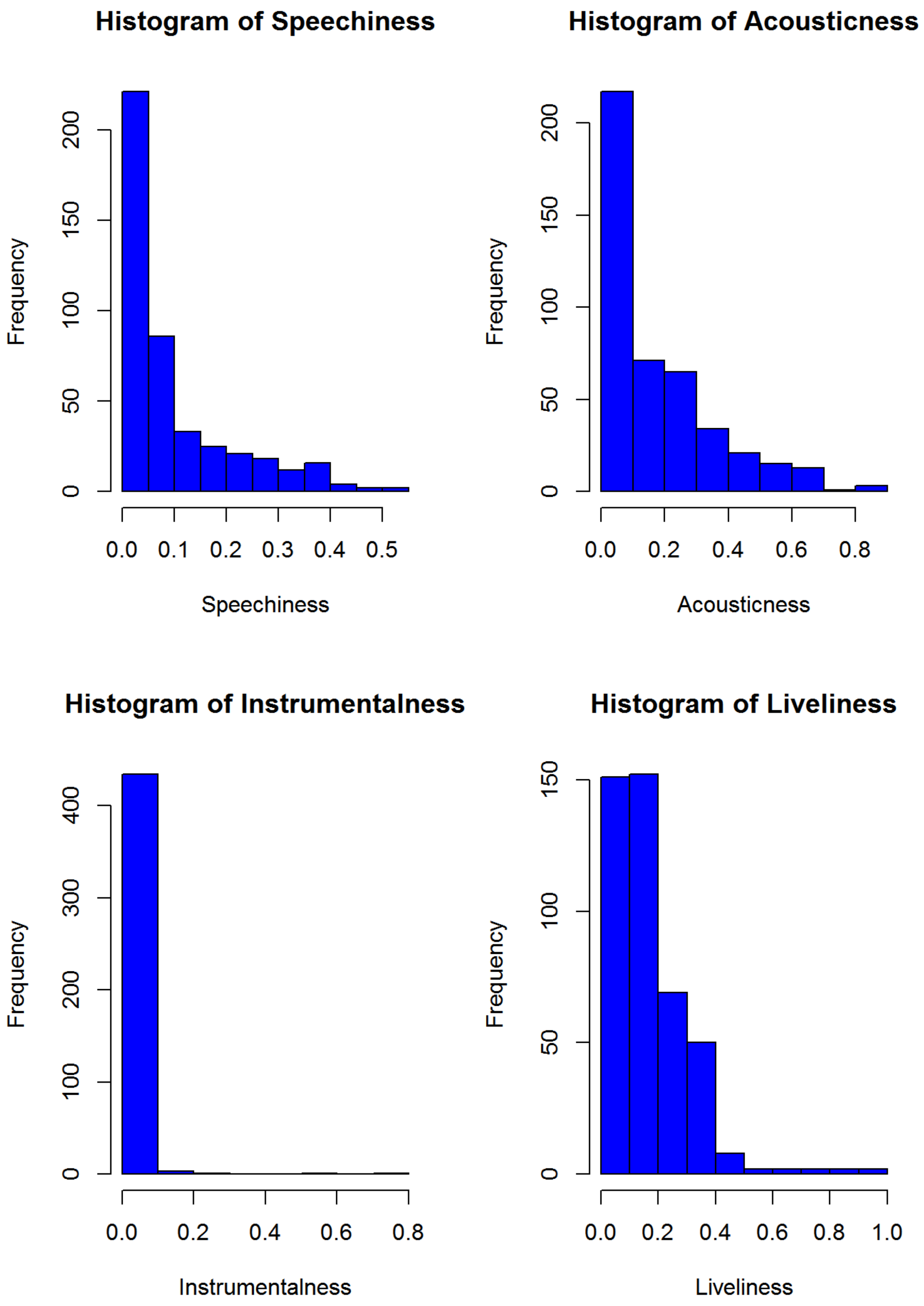
```
## 'data.frame':    1 obs. of  26 variables:
##  $ Title x         : Factor w/ 430 levels " There s Gotta Be  More To Life",..:
1
##  $ Artist x        : Factor w/ 308 levels "112","3 Doors Down",..: 266
##  $ Peak            : int 30
##  $ Date Entered    : Date, format: "2003-09-20"
##  $ Successfulness  : Factor w/ 2 levels "0","1": 1
##  $ Title y         : Factor w/ 430 levels " There s Gotta Be  More To Life",..:
1
##  $ Track ID        : Factor w/ 440 levels "00Mb3DuaIH1kjrwOku9CGU",..: 172
##  $ Artist y        : Factor w/ 308 levels "112","3 Doors Down",..: 101
##  $ Danceability    : num 0.58
##  $ Energy          : num 0.922
##  $ Key             : int 1
##  $ Loudness        : num -3.79
##  $ Mode            : Factor w/ 2 levels "0","1": 2
##  $ Speechiness     : num 0.0958
##  $ Acousticness    : num 0.0917
##  $ Instrumentalness: num 0
##  $ Liveliness      : num 0.189
##  $ Valence         : num 0.676
##  $ Tempo           : num 89.8
##  $ Type            : Factor w/ 1 level "audio_features": 1
##  $ ID              : Factor w/ 440 levels "00Mb3DuaIH1kjrwOku9CGU",..: 172
##  $ URI             : Factor w/ 440 levels "spotify:track:00Mb3DuaIH1kjrwOku9CGU
",..: 172
##  $ Track_href      : Factor w/ 440 levels "https://api.spotify.com/v1/tracks/00M
b3DuaIH1kjrwOku9CGU",..: 172
##  $ Analysis_url    : Factor w/ 440 levels "https://api.spotify.com/v1/audio-anal
ysis/00Mb3DuaIH1kjrwOku9CGU",..: 172
##  $ Duration        : int 200387
##  $ TimeSignature   : Factor w/ 3 levels "3","4","5": 2
```

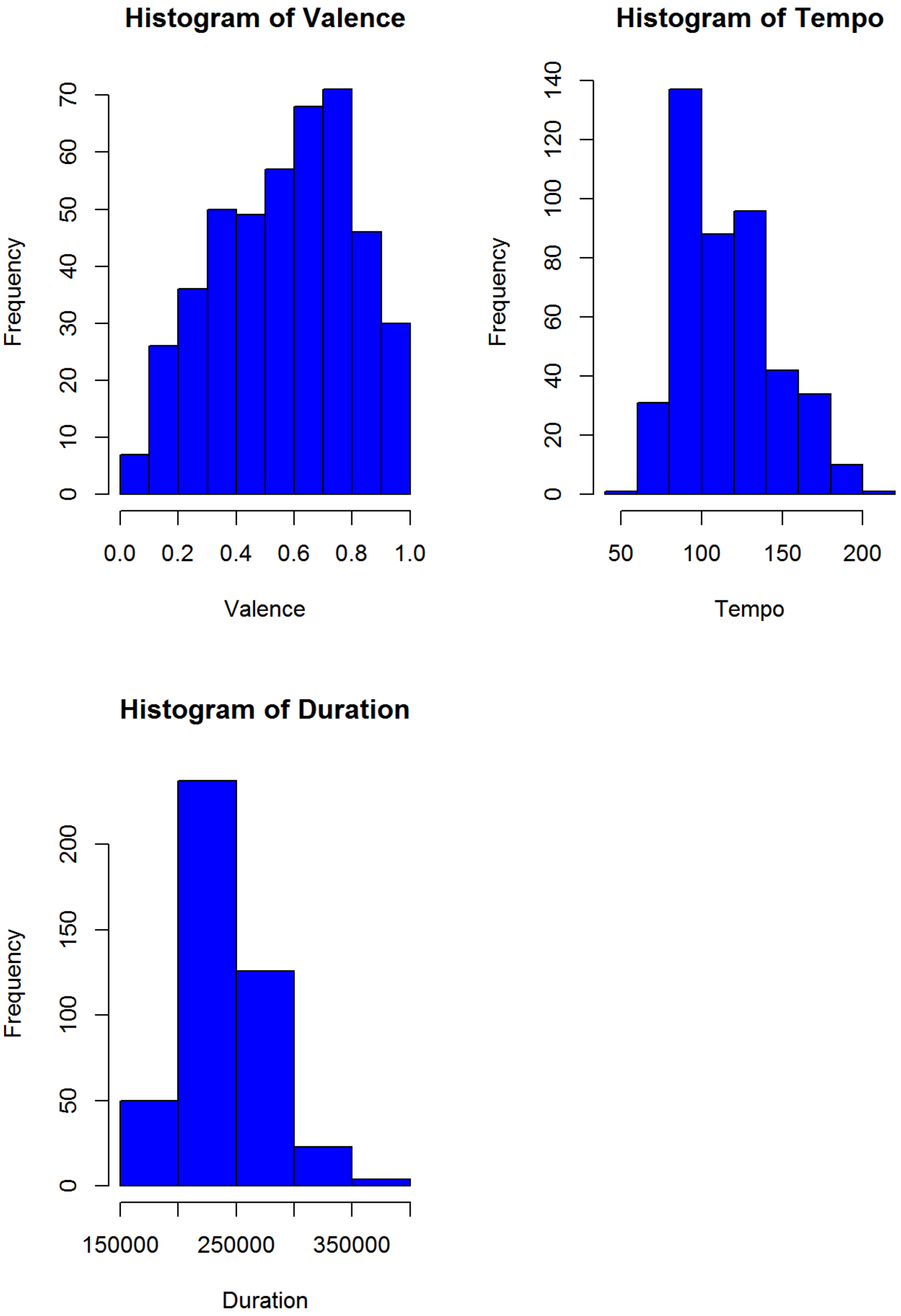
From the structure of song data, we find that the audio features of numeric datatype are Danceability, Energy, acousticness, Valence, tempo, Instrumentalness, Speechiness, loudness, liveness, peak and duration of the song. The categorical values are succesfulness, mode, key and time signature.

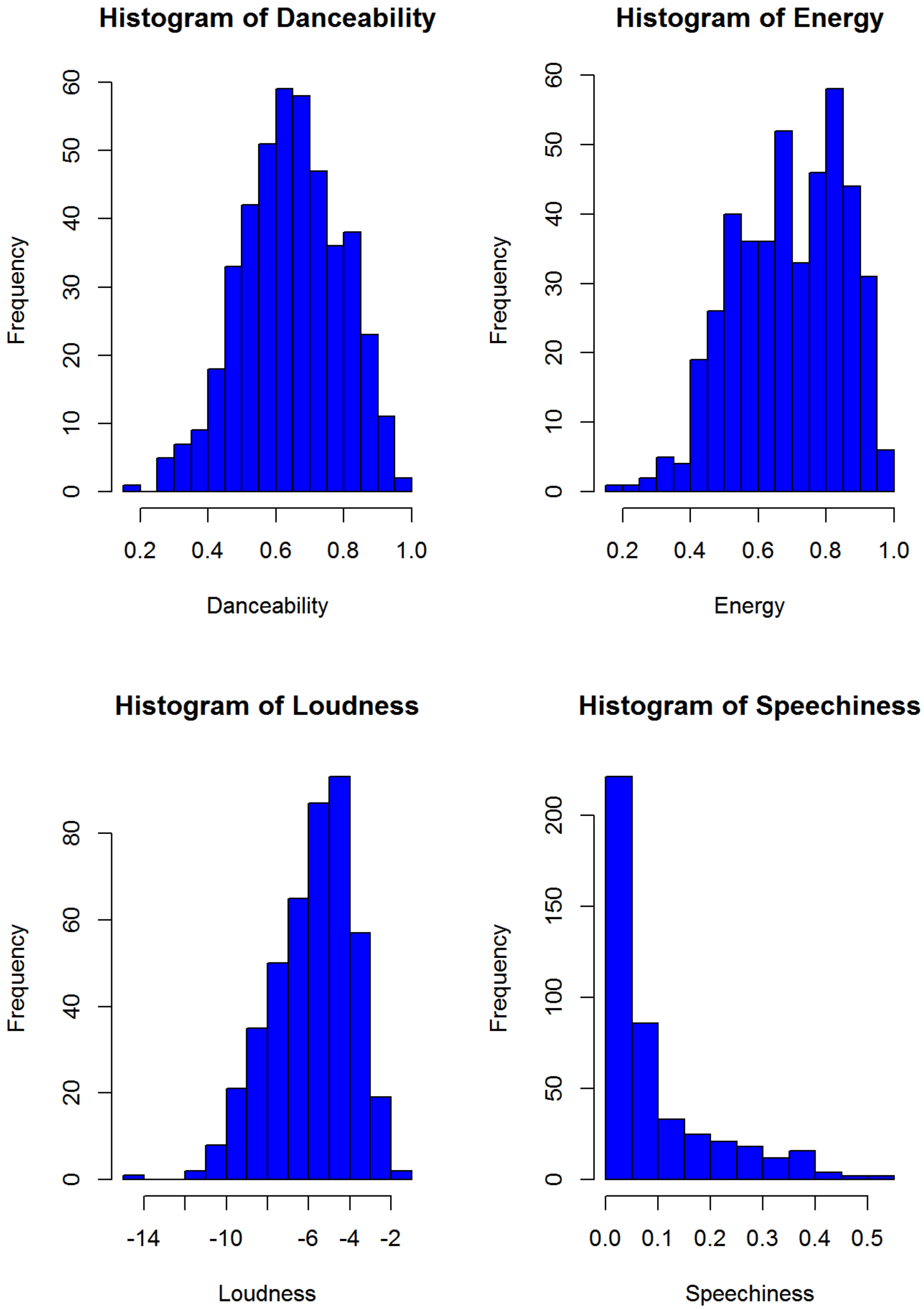
Plotting the histogram with 8 and 16 bins

The histogram is one of the effective ways of visualizing numeric data. Here, the histograms of various audio features of numeric type are plotted with different bins such as 8 and 16.


Histogram of Peak
Frequency
0
50
100
150
0
10
20
30
40
Peak
Histogram of Danceability
Frequency
0
20
40
60
80
100
120
0.2
0.4
0.6
0.8
1.0
Danceability
Histogram of Energy
Frequency
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1.0
Energy
Histogram of Loudness
Frequency
0
50
100
150
-15
-10
-5
0
Loudness


Histogram of Speechiness
Frequency
0
50
100
150
200
0.0
0.1
0.2
0.3
0.4
0.5
Speechiness
Histogram of Acousticness
Frequency
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
Acousticness
Histogram of Instrumentalness
Frequency
0
100
200
300
400
0.0
0.2
0.4
0.6
0.8
Instrumentalness
Histogram of Liveliness
Frequency
0
50
100
150
0.0
0.2
0.4
0.6
0.8
1.0
Liveliness


Histogram of Valence
Frequency
0
10
20
30
40
50
60
70
0.0
0.2
0.4
0.6
0.8
1.0
Valence
Histogram of Tempo
Frequency
0
20
40
60
80
100
120
140
50
100
150
200
Tempo
Histogram of Duration
Frequency
0
50
100
150
200
150000
250000
350000
Duration

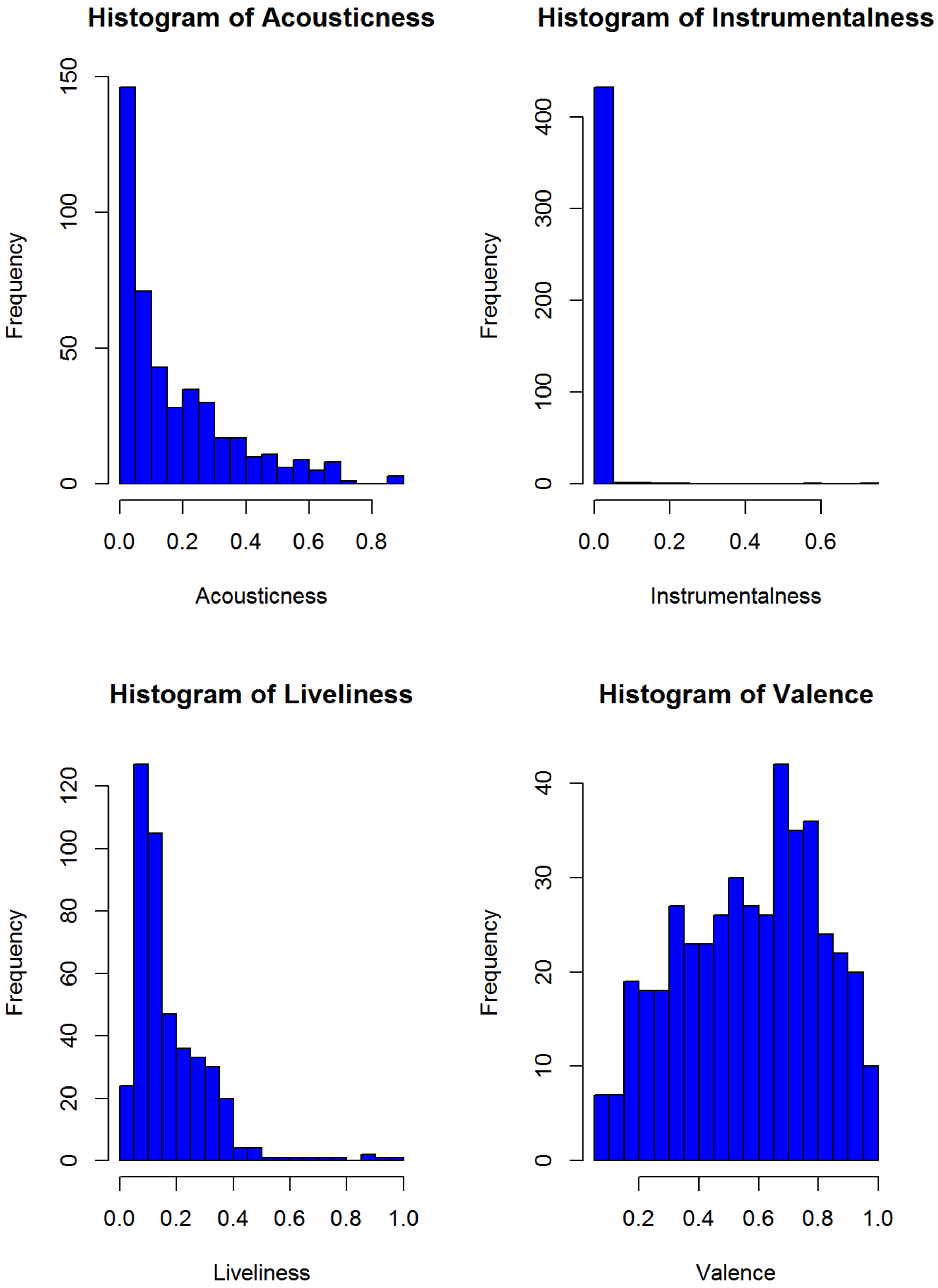


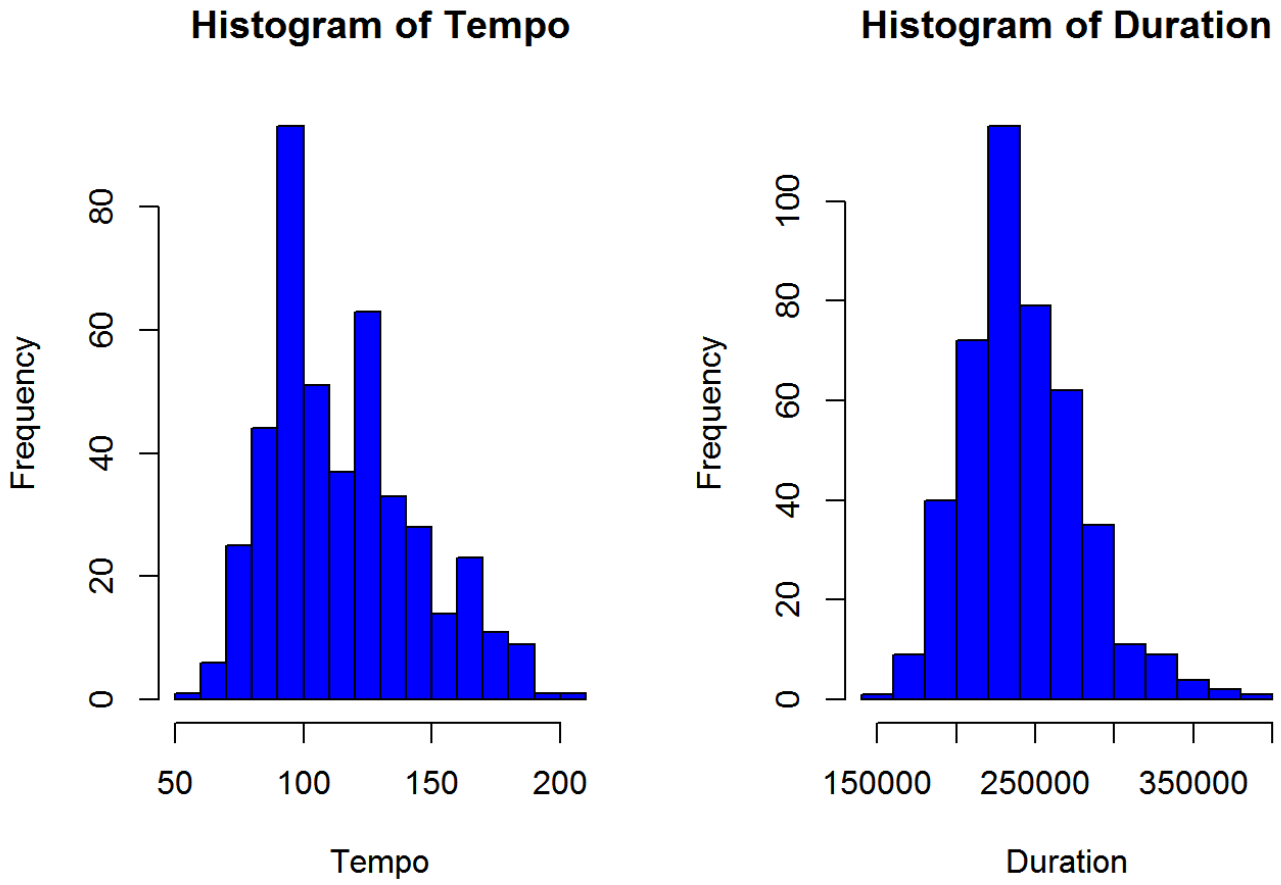
Histogram of Danceability

Histogram of Energy

Histogram of Loudness

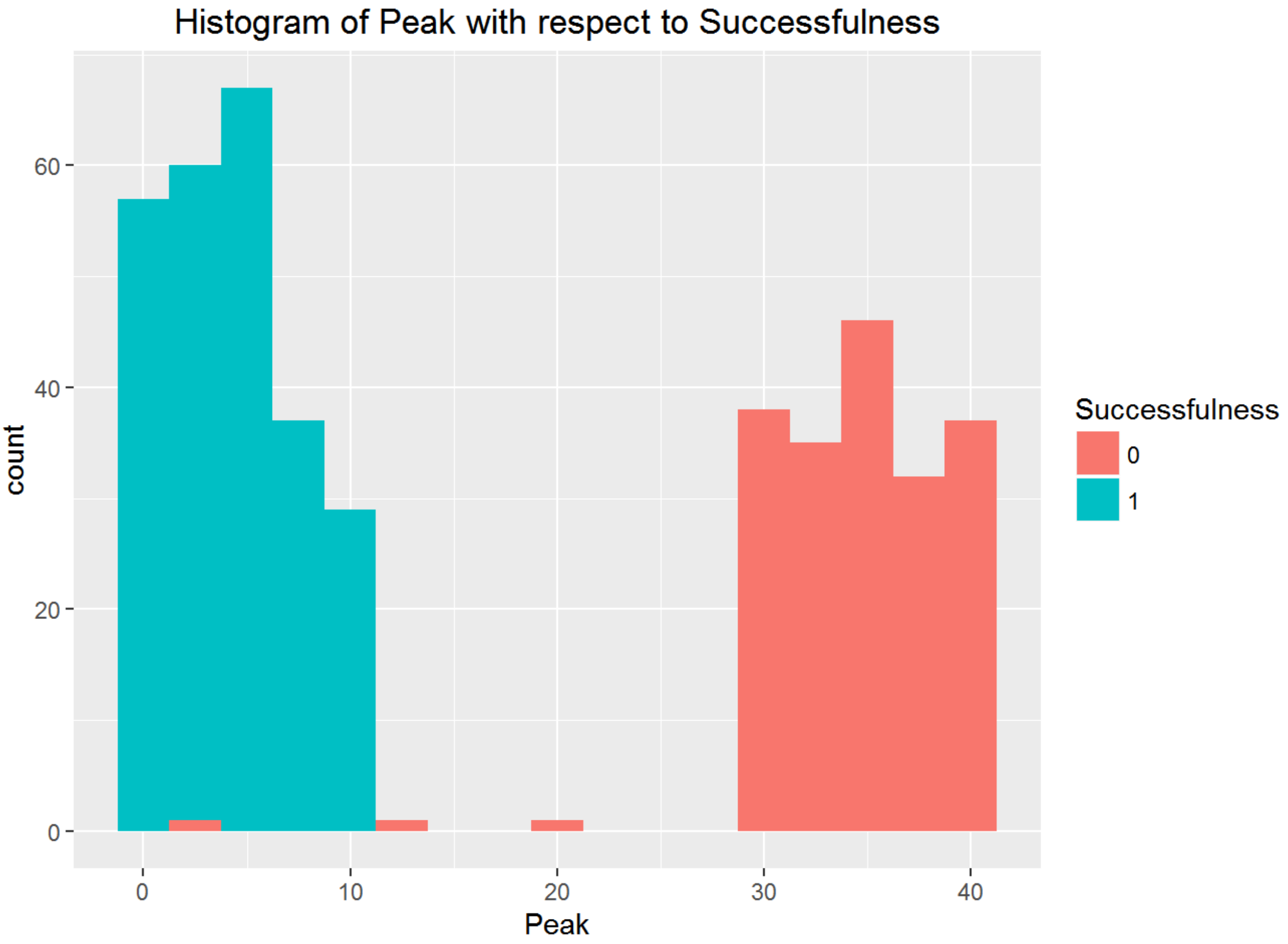
Histogram of Speechiness

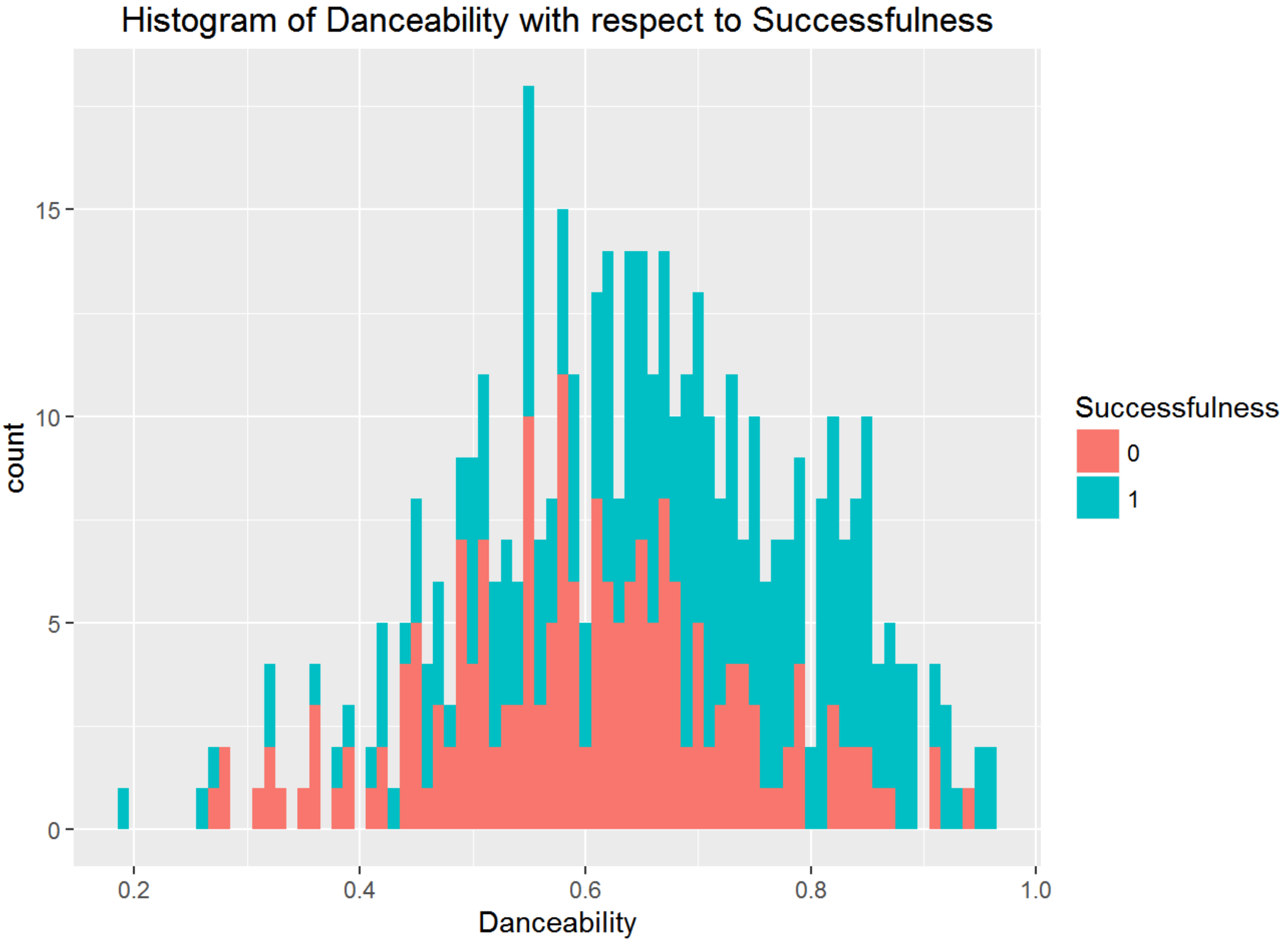

Histogram of Acousticness
Frequency
0
50
100
150
0.0
0.2
0.4
0.6
0.8
Acousticness
Histogram of Instrumentalness
Frequency
0
100
200
300
400
0.0
0.2
0.4
0.6
Instrumentalness
Histogram of Liveliness
Frequency
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0
Liveliness
Histogram of Valence
Frequency
0
10
20
30
40
0.2
0.4
0.6
0.8
1.0
Valence

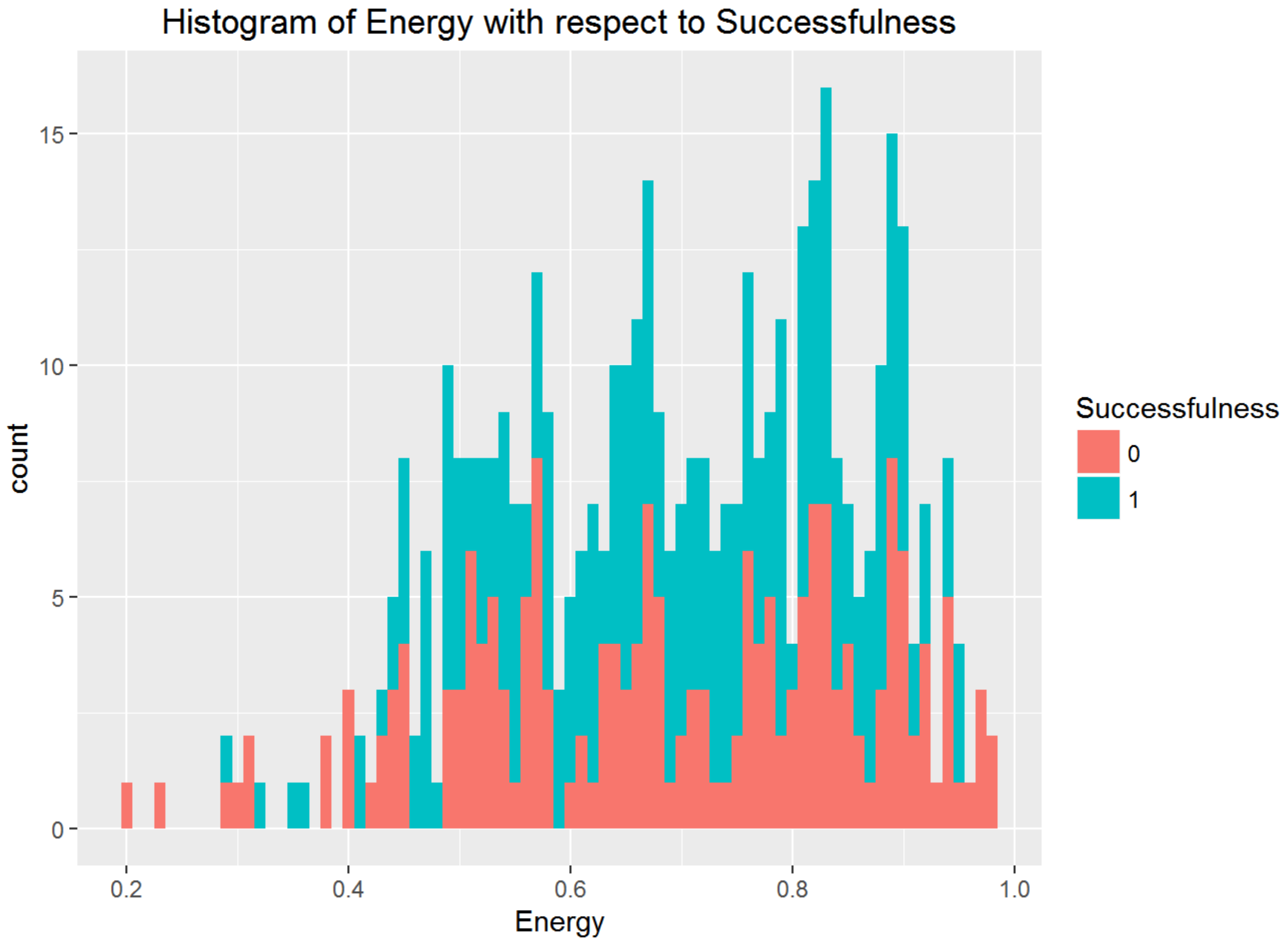


Plotting Stacked histogram based on Succesfulness

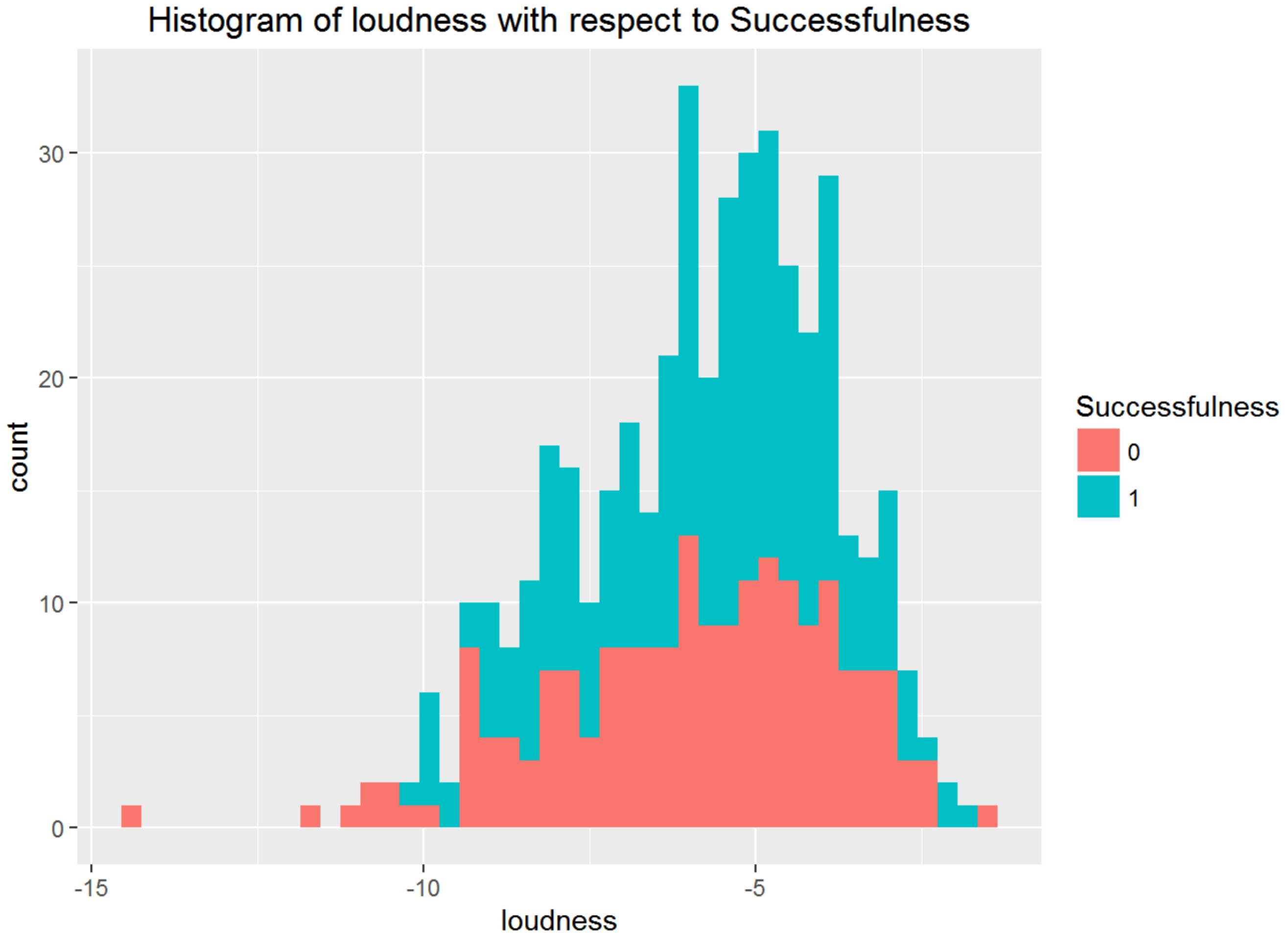
The stacked histograms for the numerical values in songs data are plotted with regards to the succesfulness of the song.


Histogram of Peak with respect to Successfulness

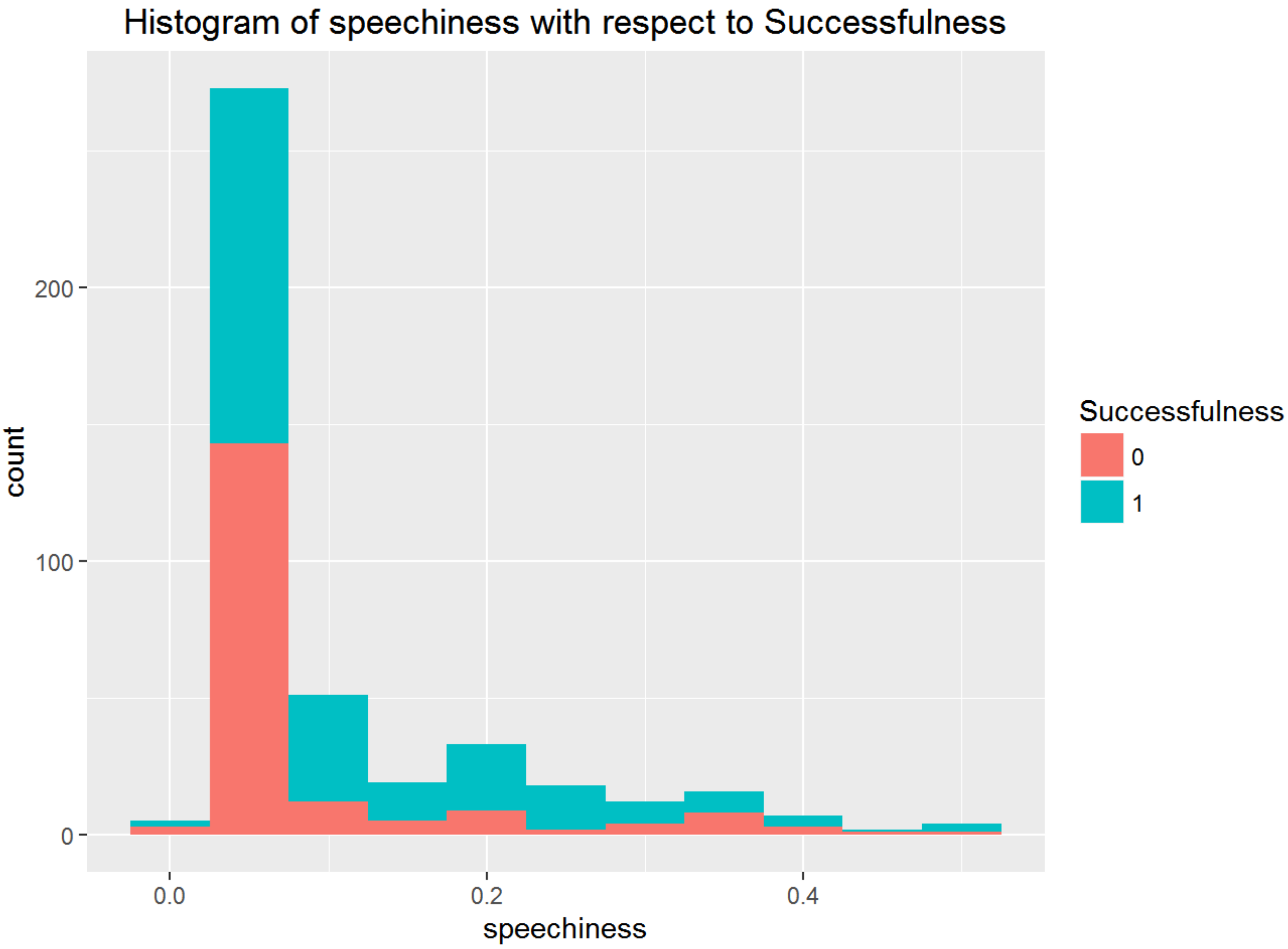

Histogram of Danceability with respect to Successfulness

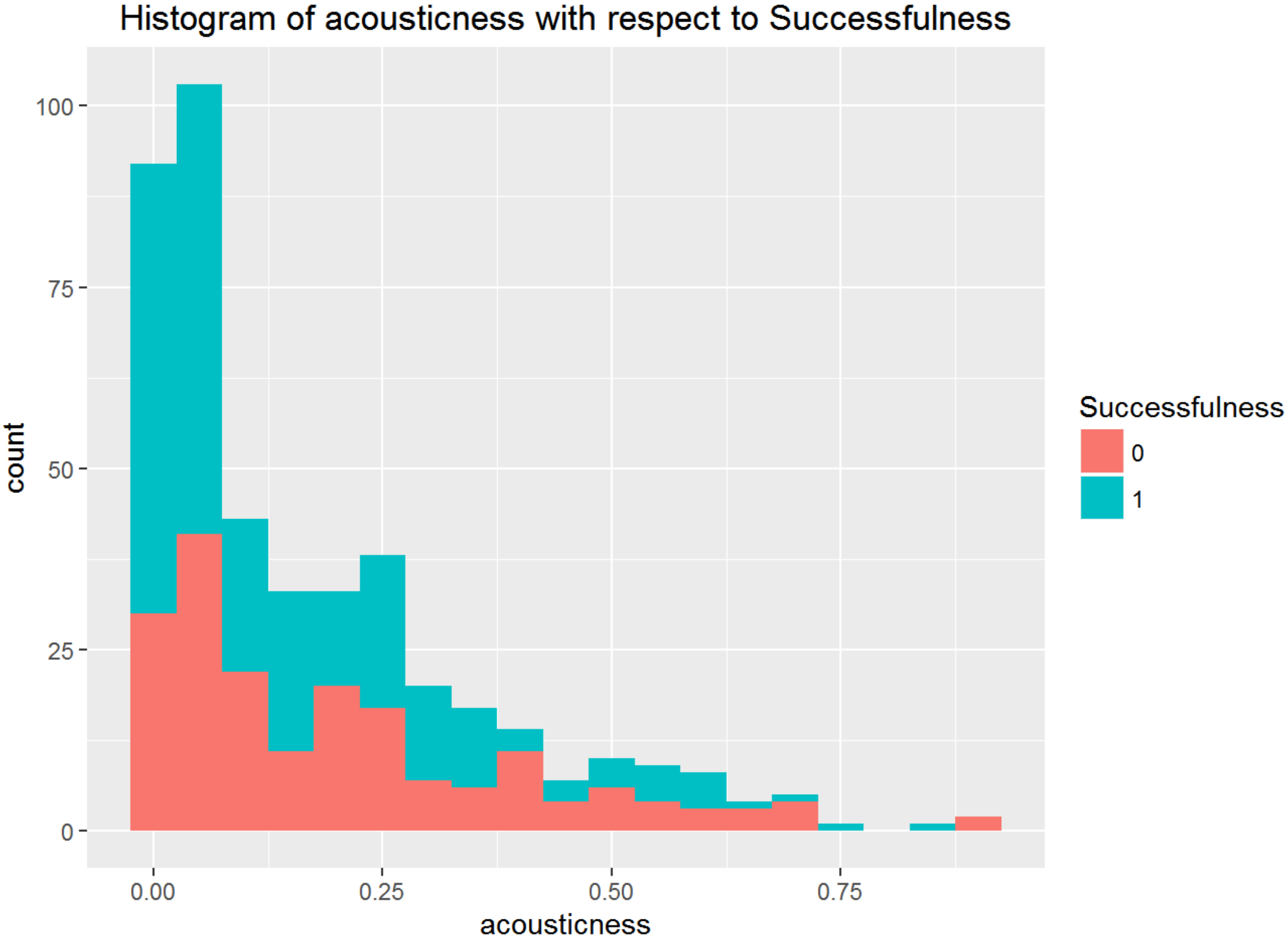


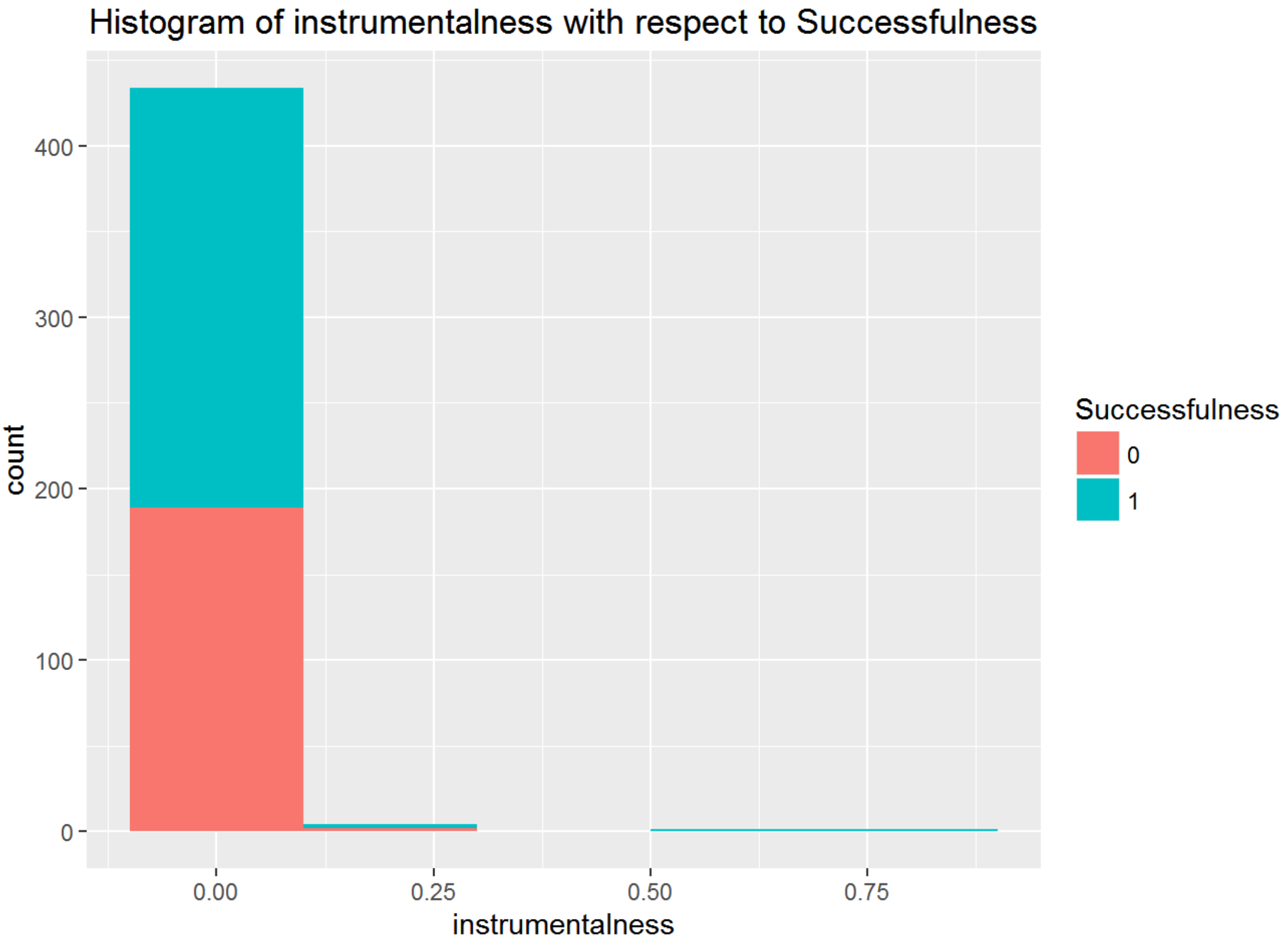
Histogram of Energy with respect to Successfulness

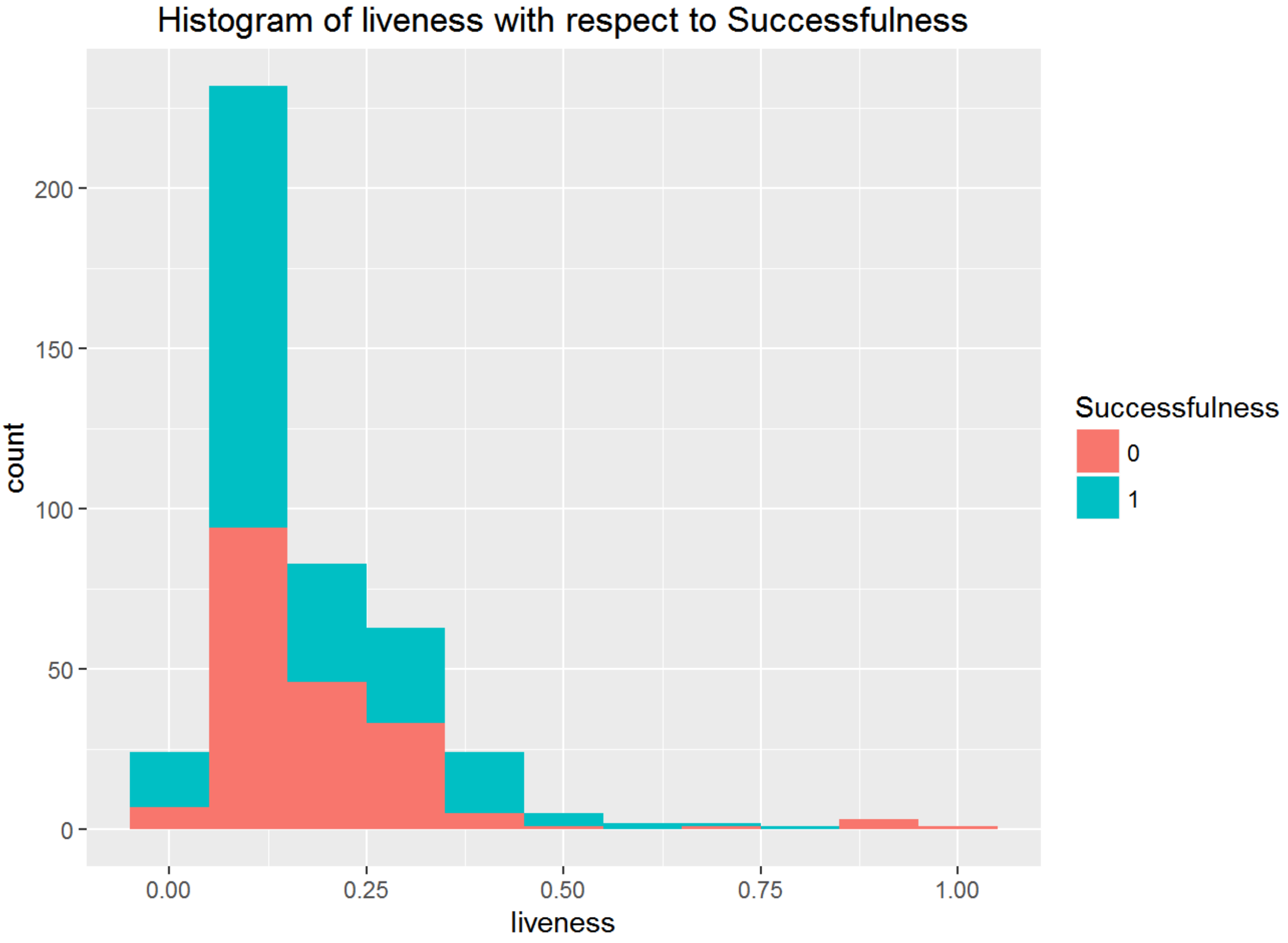


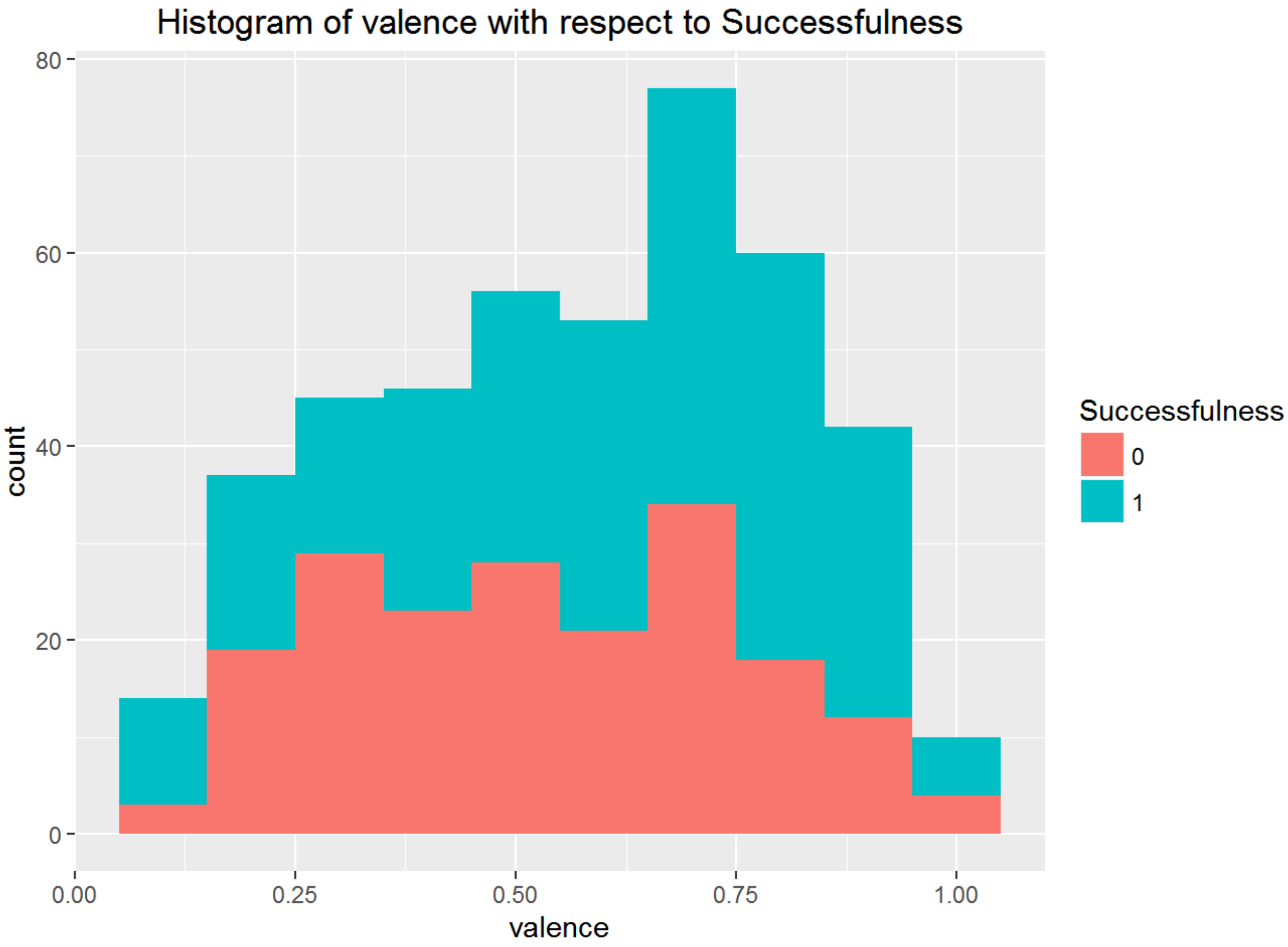
Histogram of loudness with respect to Successfulness

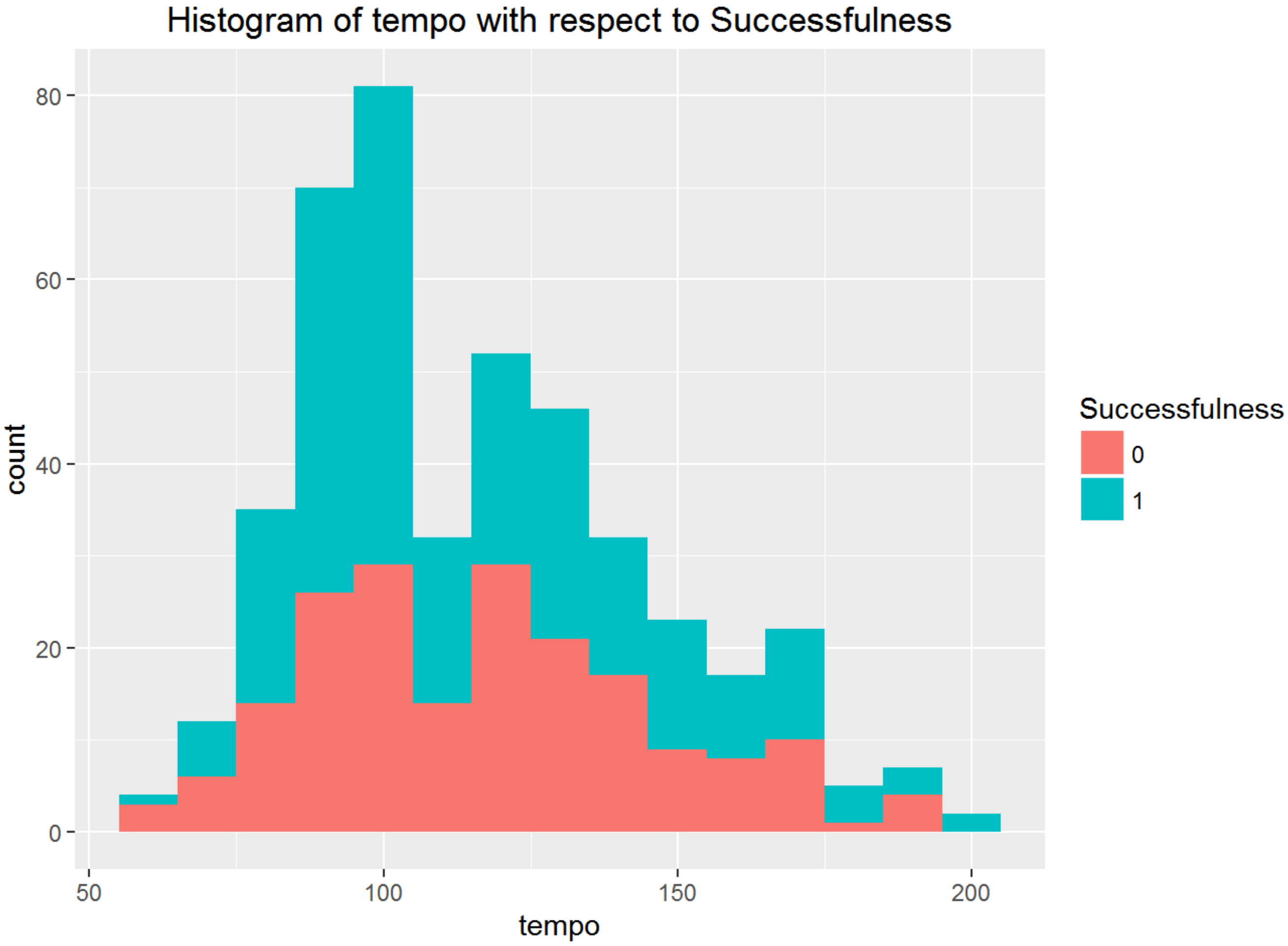

Histogram of speechiness with respect to Successfulness

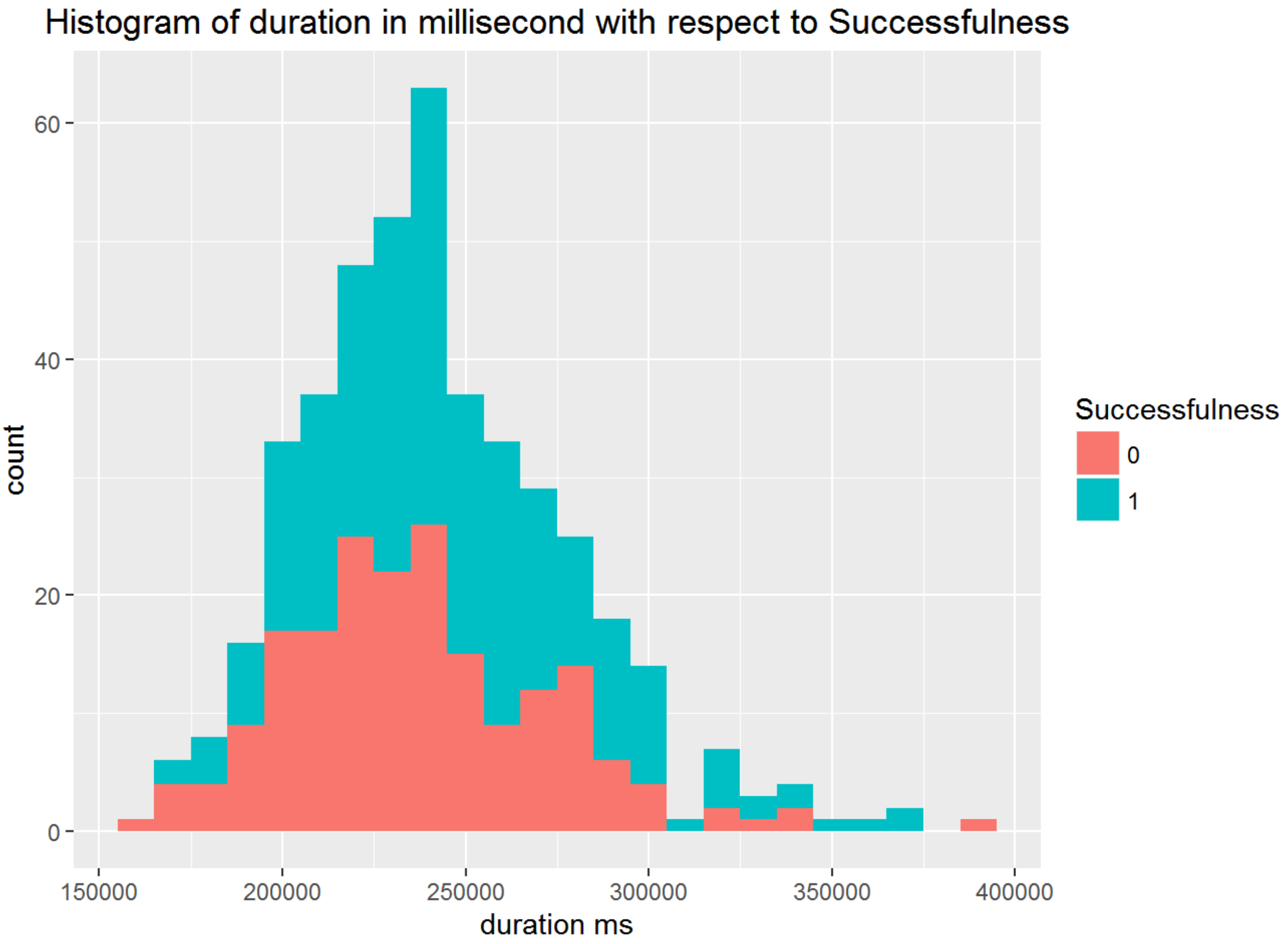

Histogram of acousticness with respect to Successfulness


Histogram of instrumentalness with respect to Successfulness
count
400
300
200
100
0
0.00
0.25
0.50
0.75
instrumentalness
Successfulness
0
1


Histogram of liveness with respect to Successfulness
count
200
150
100
50
0
0.00
0.25
0.50
0.75
1.00
liveness
Successfulness
0
1


Histogram of valence with respect to Successfulness
80
60
40
20
0
count
0.00
0.25
0.50
0.75
1.00
valence
Successfulness
0
1


Histogram of tempo with respect to Successfulness
80
60
40
20
0
count
50
100
150
200
tempo
Successfulness
0
1

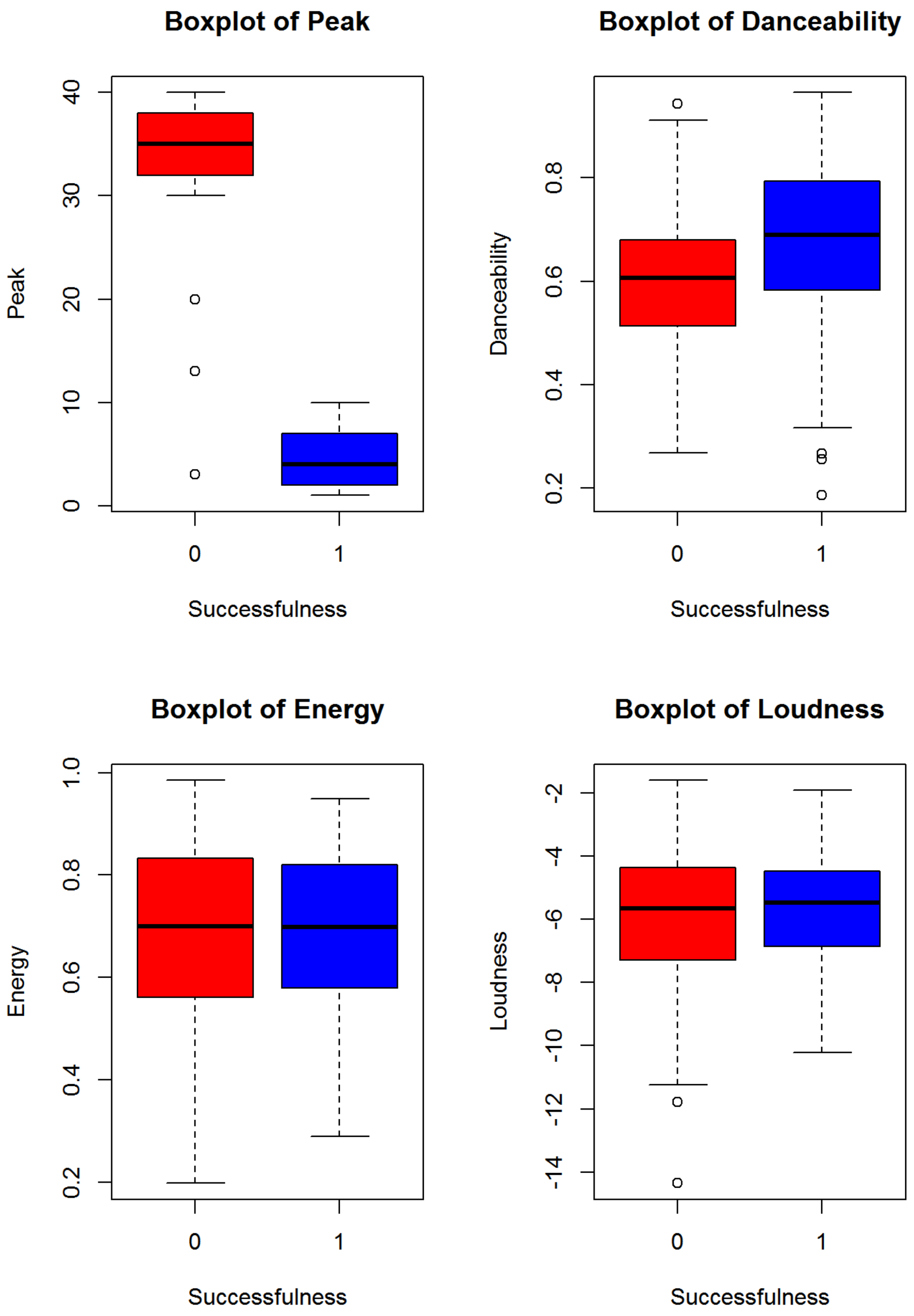


Inference:

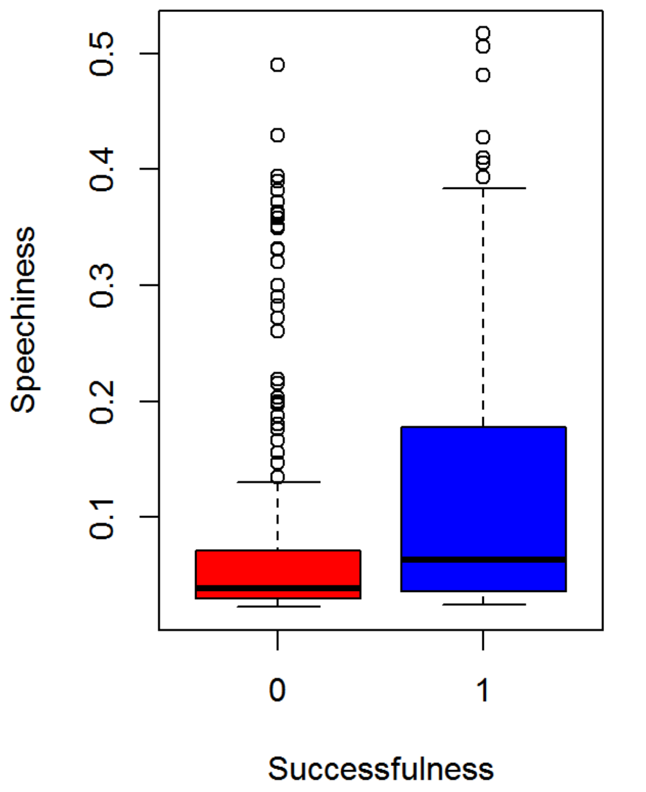
1. We can see that for peak and succesfulness, the songs which peaks in top 11 are considered successful and the songs which are after that are considered unsuccessful.
2. The Danceability of successful songs are significantly high between 0.5 and 0.9 whereas the is no signifacnt difference between sucessful and unsuccessful songs at lower danceability rates.
3. The speechiness and acousticness are near to 0 for most songs and the count is high for succesful songs.

Plotting of stacked Boxplots based on Succesfulness

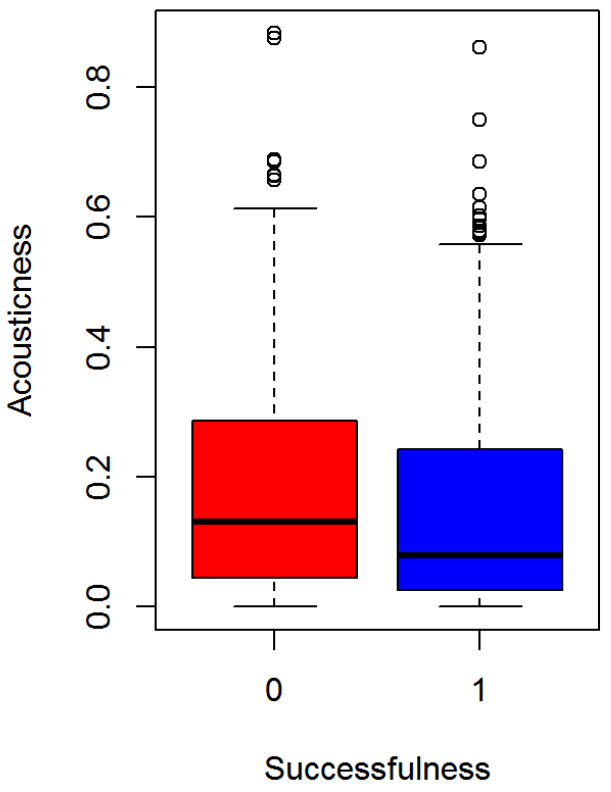
The stacked boxplots for the numerical values in songs data are plotted with regards to the succesfulness of the song.


Boxplot of Peak
Peak
0
10
20
30
40
0
1
Successfulness
Boxplot of Danceability
Danceability
0.2
0.4
0.6
0.8
0
1
Successfulness
Boxplot of Energy
Energy
0.2
0.4
0.6
0.8
1.0
0
1
Successfulness
Boxplot of Loudness
Loudness
-14
-12
-10
-8
-6
-4
-2
0
1
Successfulness

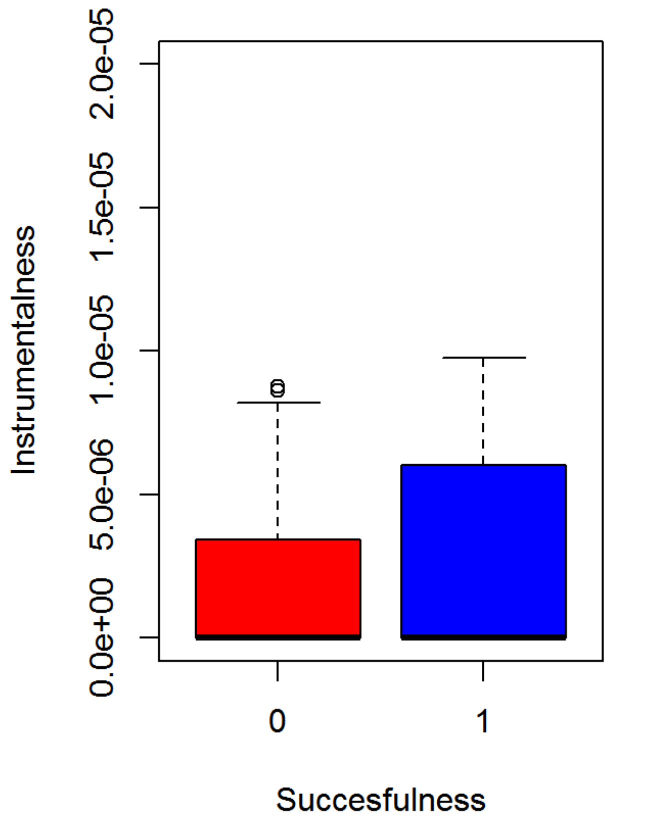
Boxplot of Speechiness



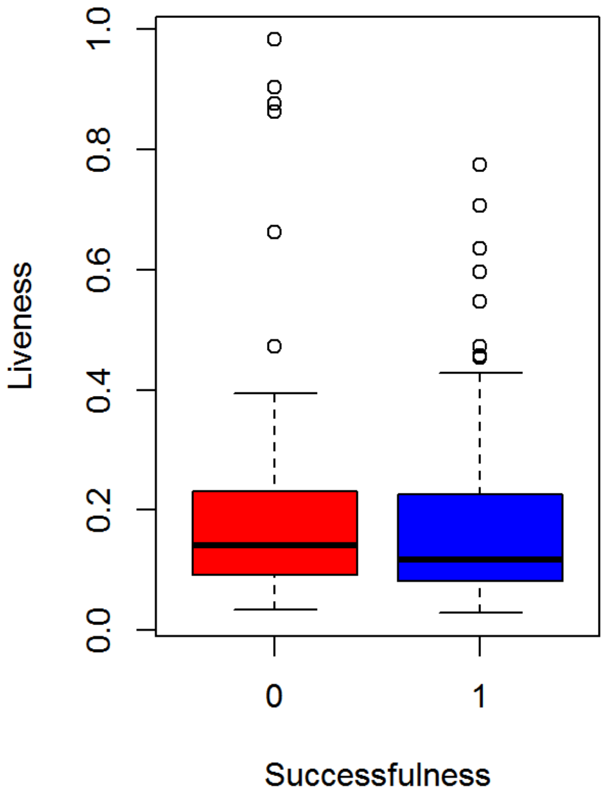
Boxplot of Acousticness

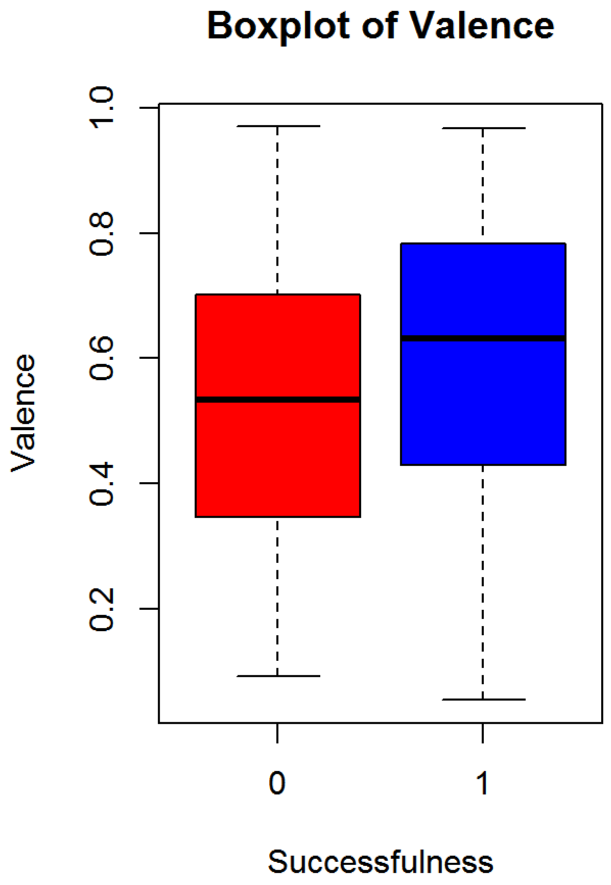


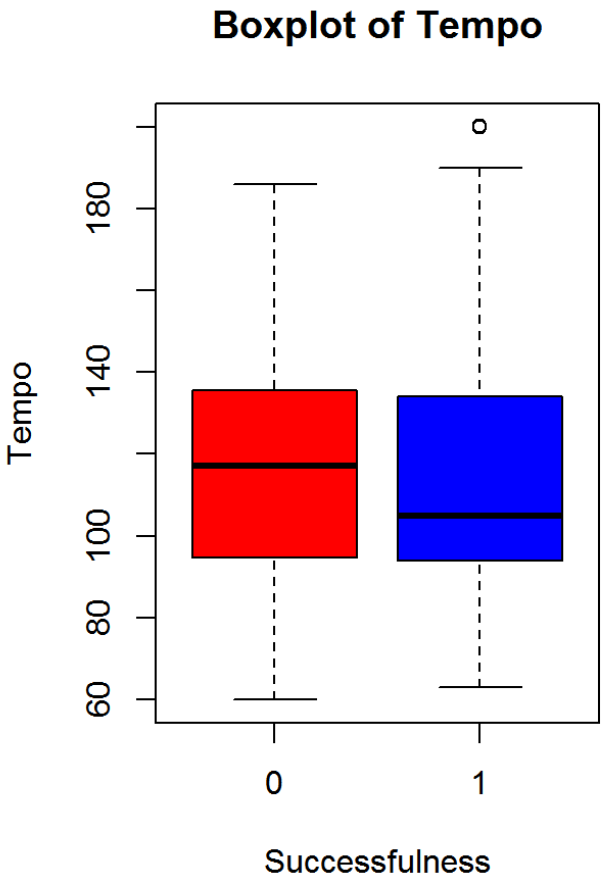
Boxplot of Instrumentalness

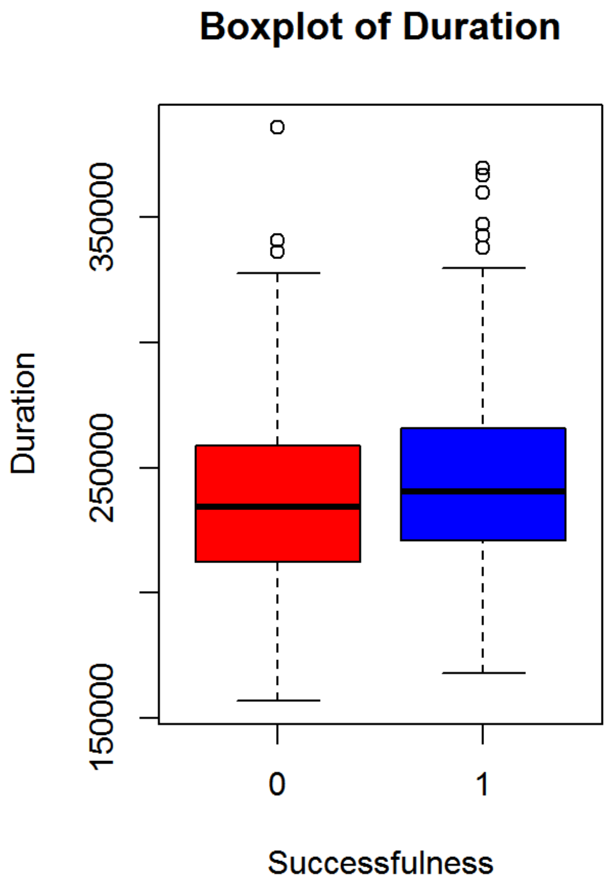


Boxplot of Liveness




Boxplot of Valence


Boxplot of Tempo

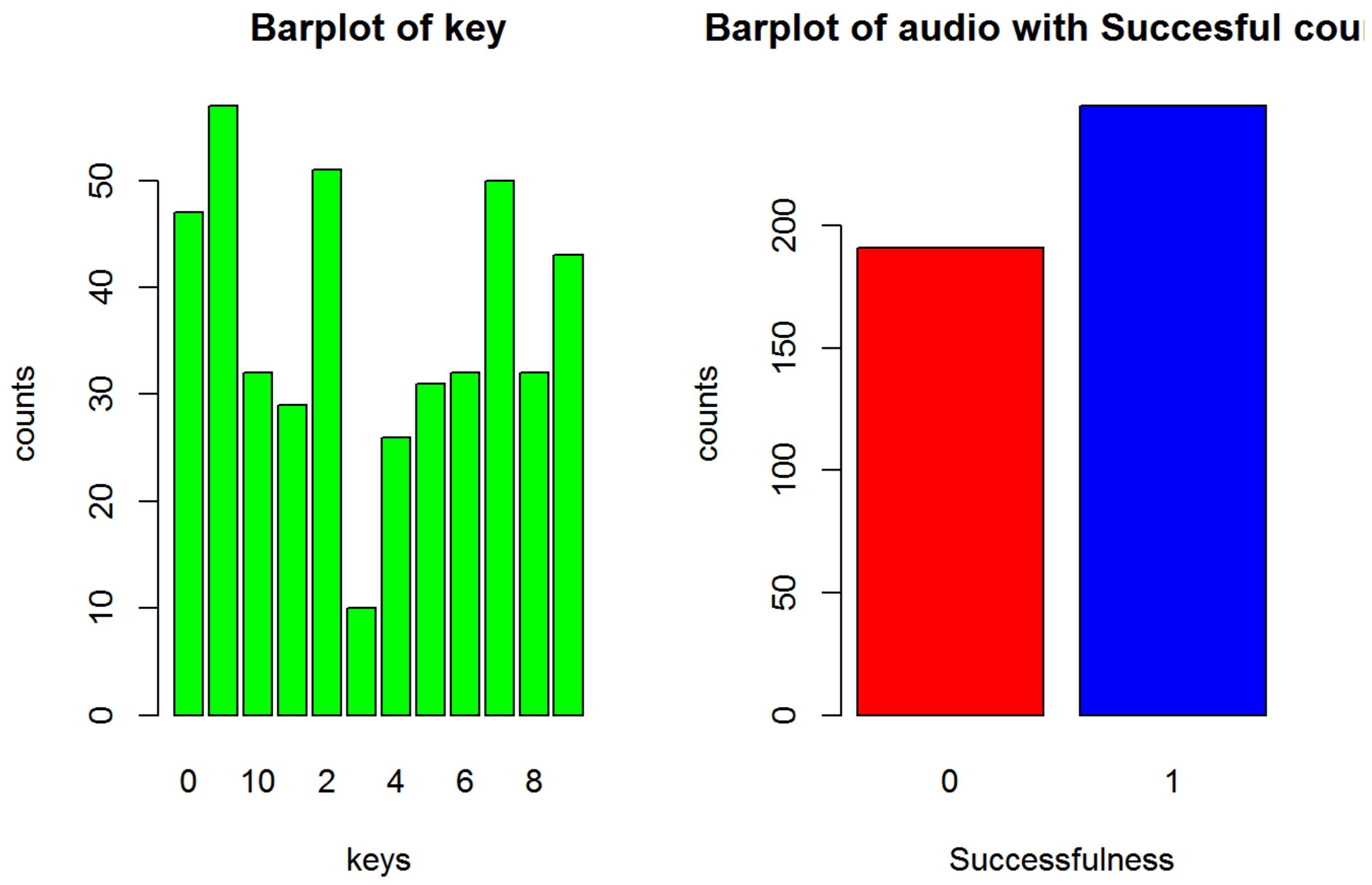

Boxplot of Duration

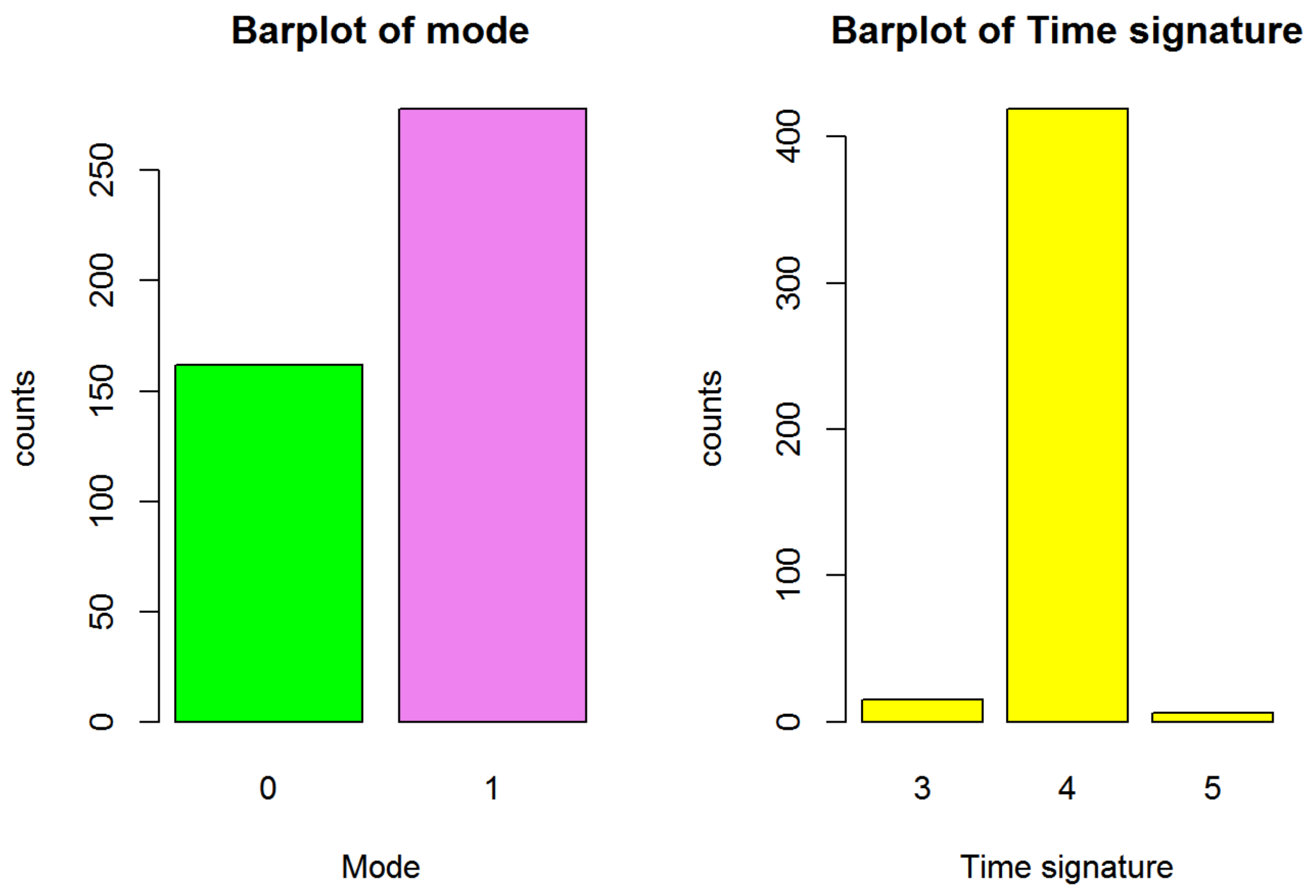
Inferences: 1. The median of Peak for successful songs lies around 5 which indicates that most of the songs had hit the 5th position in the Billboard chart. The peak value for the unsuccessful songs lies around 35 with possible outliers at 15 which can be rated as quite popular but not a hit.

2. The median to the danceability feature of songs are in the range of 0.7 which refers to good value with respect to succesfulness.
3. The energy for succesful and unsuccesful songs doesn't differ quite much and the median values are almost same.
4. A good valence value accounts to the song being more succesful. The median value for a succesful song with regards to valence is 0.56.
5. The tempo of the song lies around 90 for succesful song which accounts for good beats to make the music enjoyable.

Barplots based on categorical values

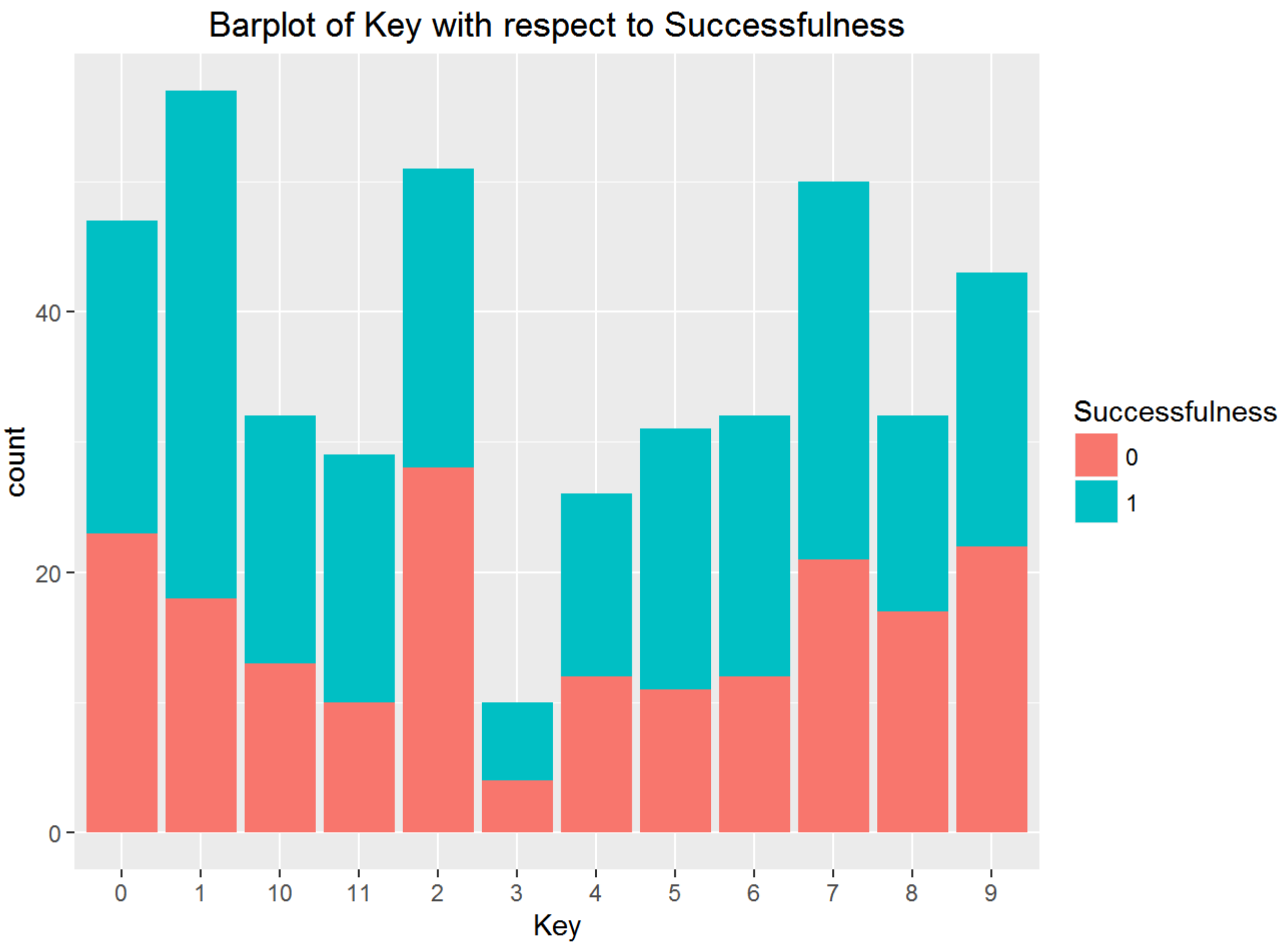
The analysis of categorical values for audio features of a song is best plotted by barplots. Here, the catgorical audio features such as mode, succesfulness, key and time signature are plotted.

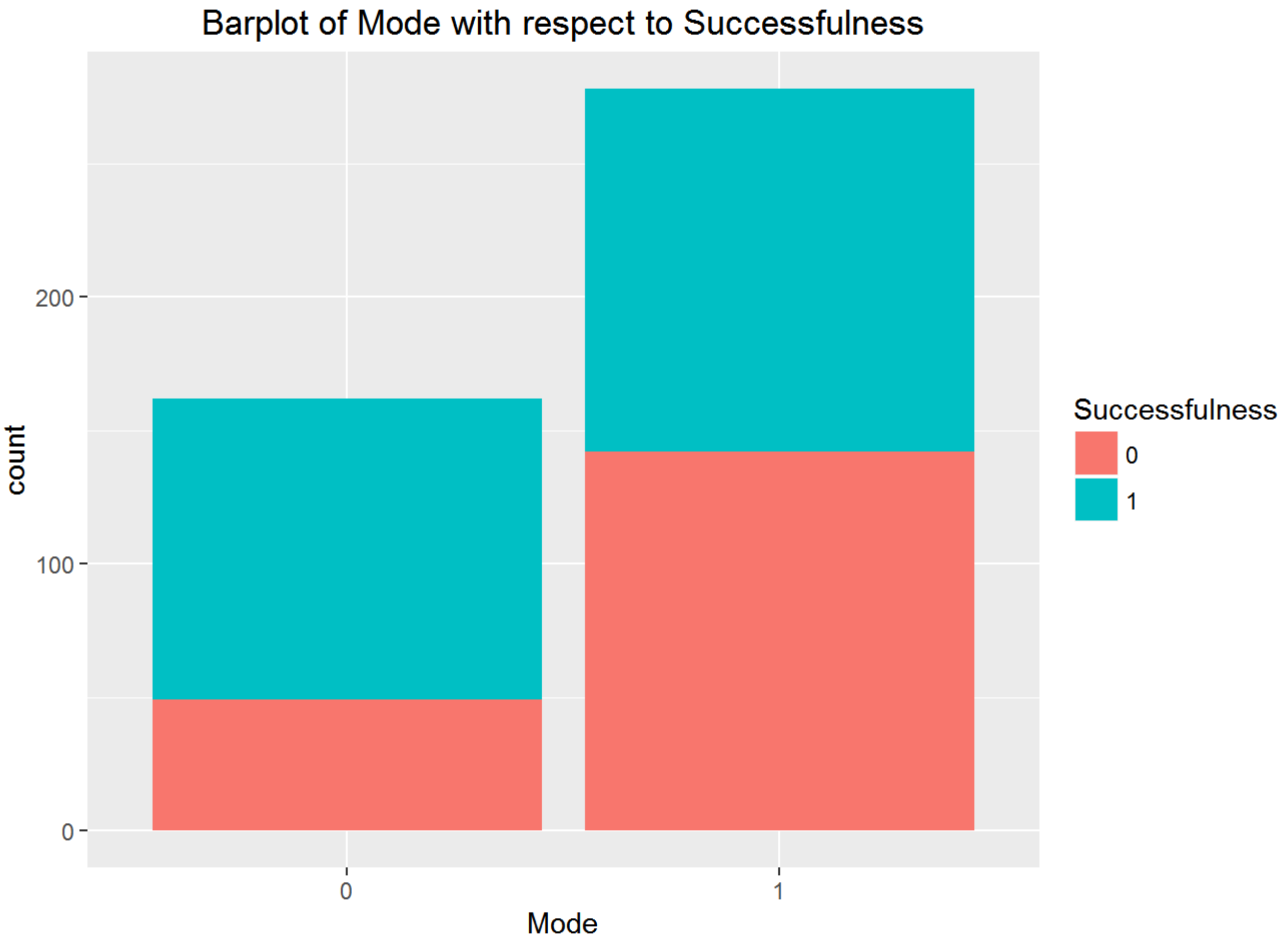


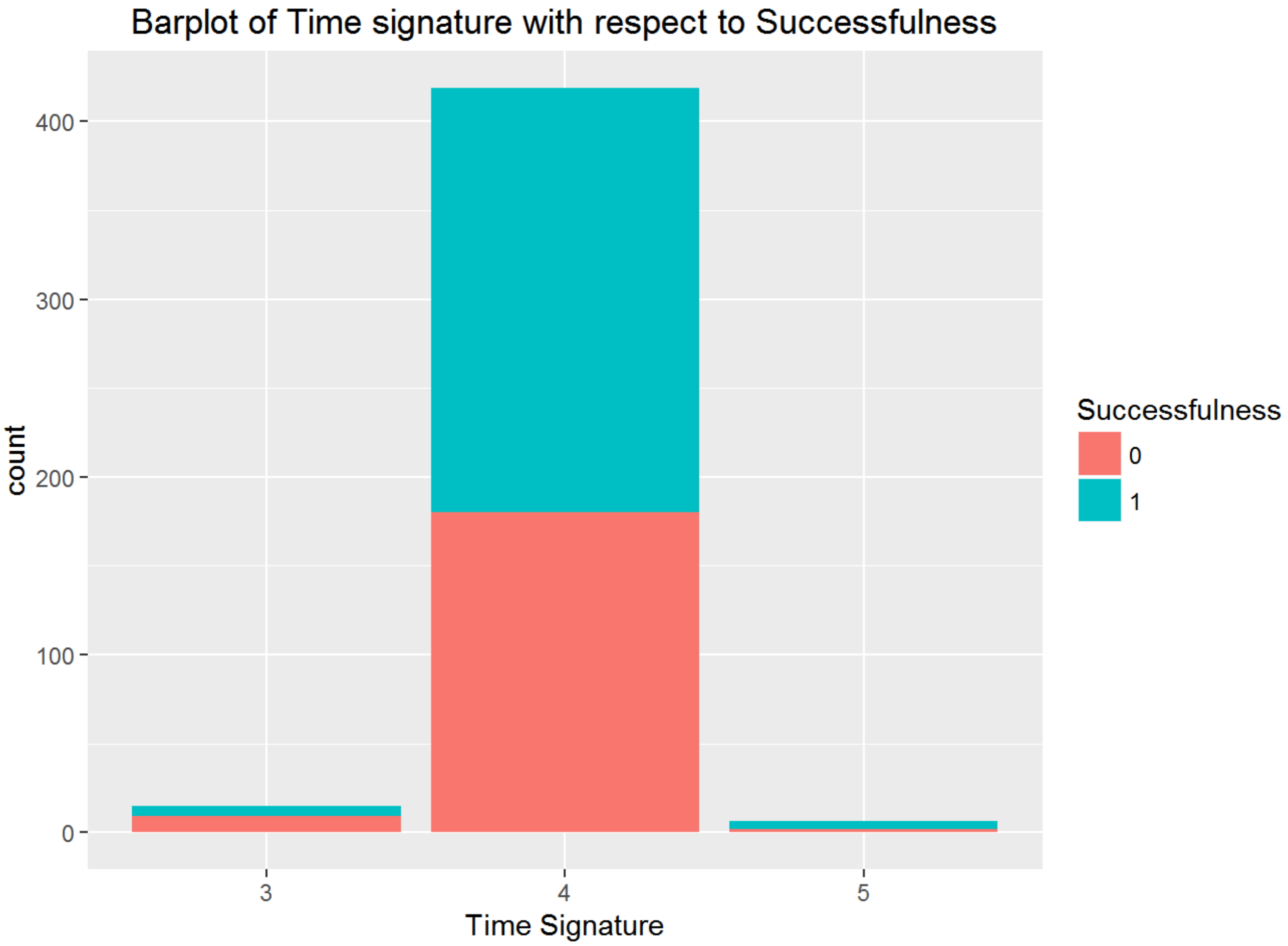


Stacked barplots based on Succesfulness

The stacked barplot is a way of plotting the categorical values with regards to succesfulness of a song.


Barplot of Key with respect to Successfulness
count
0
20
40
0
1
10
11
2
3
4
5
6
7
8
9
Key
Successfulness
0
1


Barplot of Mode with respect to Successfulness
count
0
100
200
0
1
Mode
Successfulness
0
1

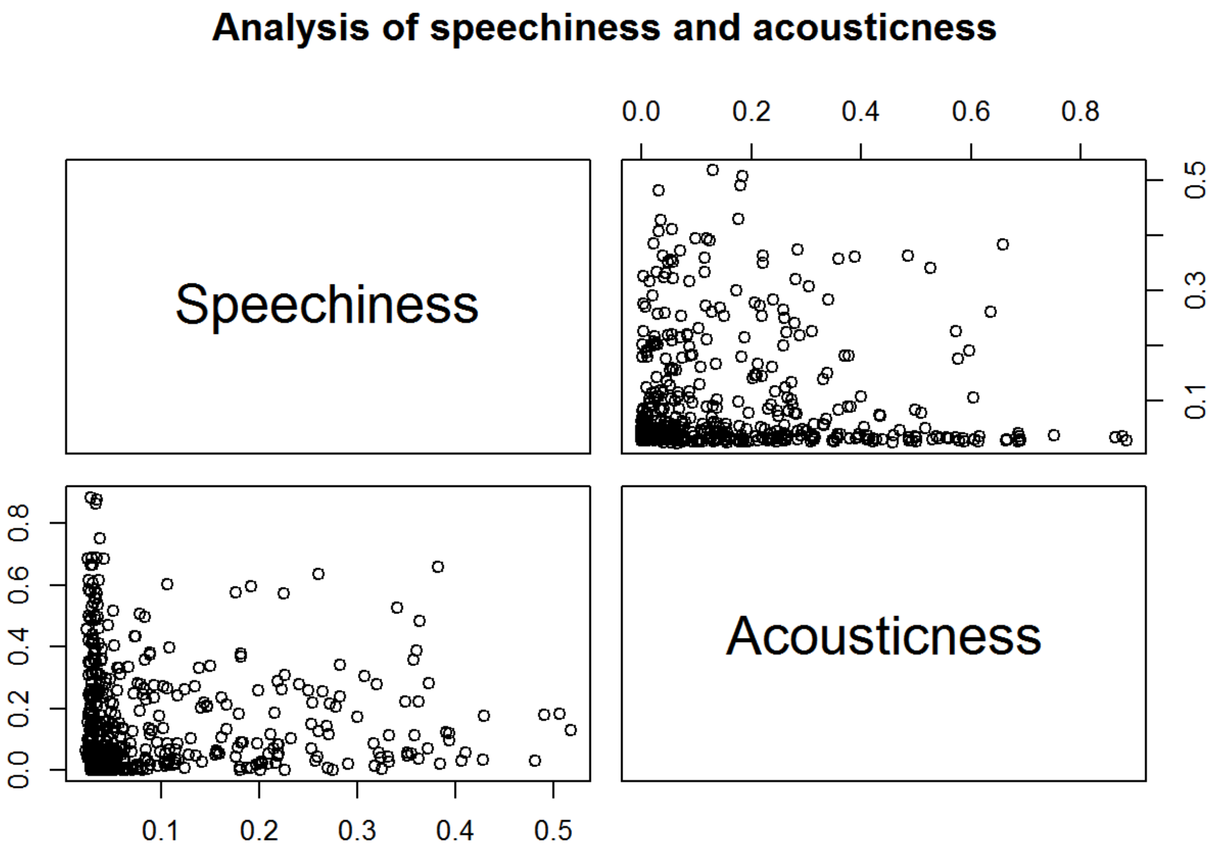


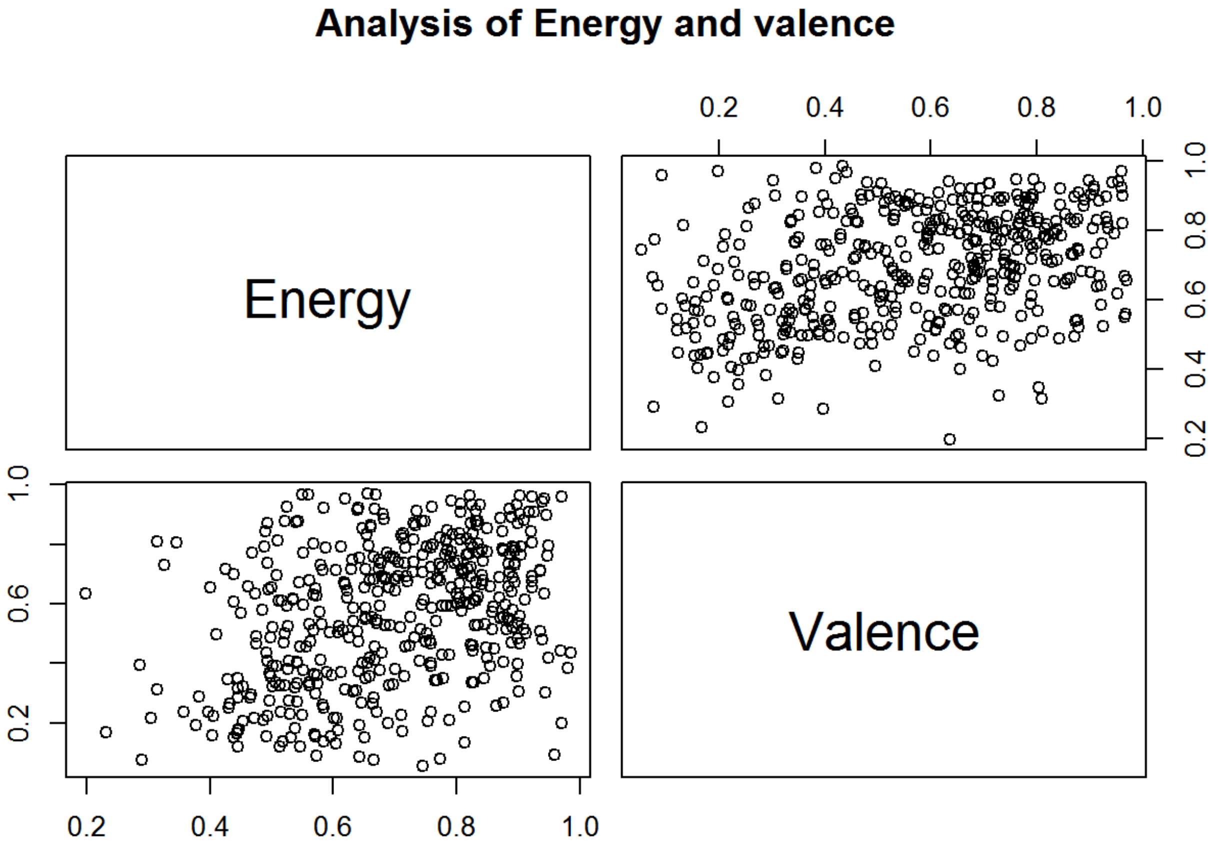
Inferences: 1. The mode of most of the songs are 1 where the successful songs having mode as 1 are higher than that of the unsuccesful songs.

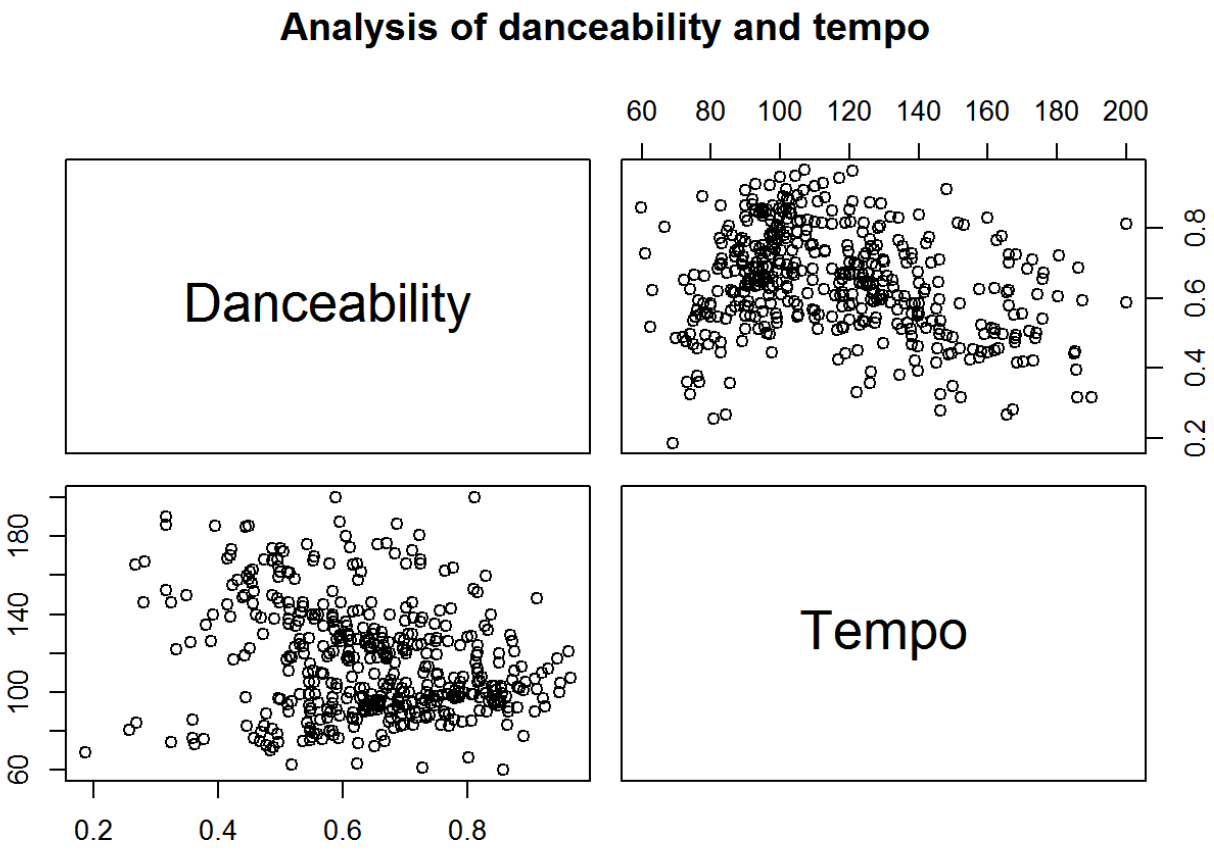
2. The time signature for most of the songs are 4.

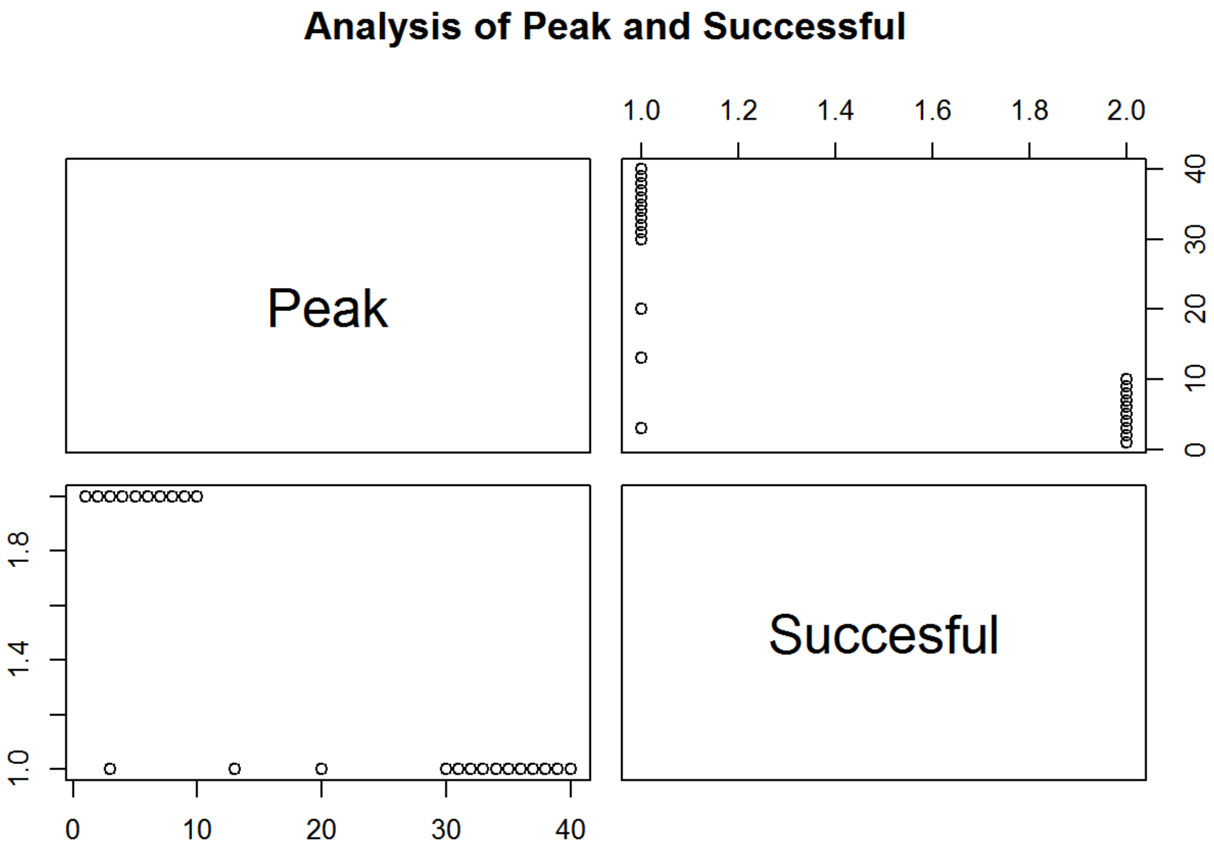
Pairwise Analysis on Audio features

The pairwise analysis is done based on comparison of two audio features and bringing out the relationship between them. Here the pariwise analysis of speechiness and acousticness; energy and valence; danceability and tempo and peak and successfulness are plotted.


Analysis of speechiness and acousticness
Speechiness
Acousticness
0.0
0.2
0.4
0.6
0.8
0.1
0.3
0.5
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5


Analysis of Energy and valence
Energy
Valence
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0
0.2
0.6
1.0
0.2
0.4
0.6
0.8
1.0





Inferences: 1. The speechiness and acousticness are similar in points being plotted. Most of the songs have speechiness and acousticness less than 0.1 and gradually spreads out at 0.1,0.2 and 0.3.

2. The more the energy, more is the valence for a song. Most of the songs have energy and valence features around 0.6 to 1.0.
3. The tempo of the music i.e the beats determin the bass of the song which also determines the dance moves that occur. Most of the songs with tempo around 90 to 100 have higher danceablity rates of more than 0.6 to 1.0 .
4. The pairwise analysis of peak and succesfulness often determines how the sngs have reached out when it hits the top of the billboard charts. The peaks of 1 to 10 determin the success rate of the song as most successfull. The songs of peak above 11 have moderate success and the unsuccesful songs are placed mostly in the 30 to 40th position in billboard.

Prediction of Successfulness of songs

The successfulness of the songs cn be predicted as supervised learning i.e classification. In Classification, the songs are classified using labels, ‘0’ which denotes unsuccessful and ‘1’ which denotes successful. One part of the song dataset is trained according to the classification models and the other part of the data set acts as test dataset. By the method of cross validation and classification models, prediction of successful and unsuccesful songs are made.

```r
#install.packages("rplot.part")
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.1
```

```r
#install.packages("ROCR")
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.3.1
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.3.1
```

```
## 
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
## 
##     lowess
```

```r
#install.packages("e1071")
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.1
```

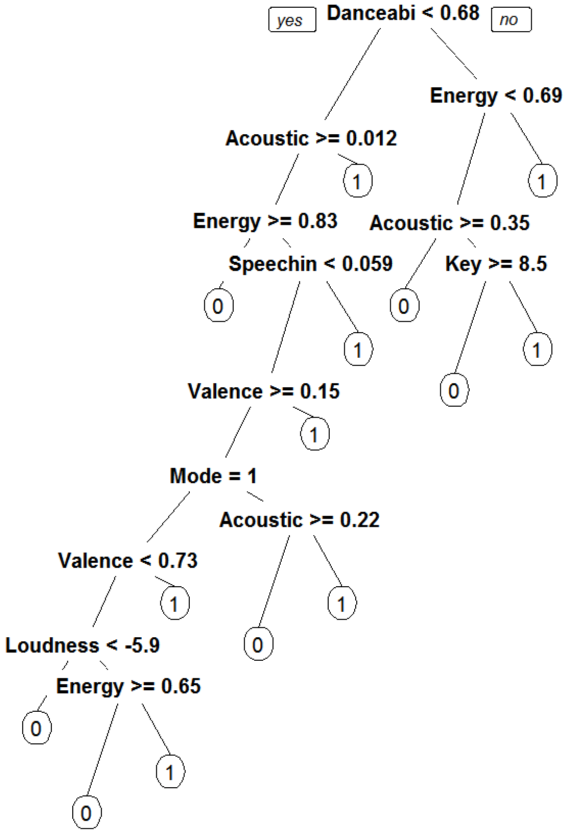
```r
library(class)
```

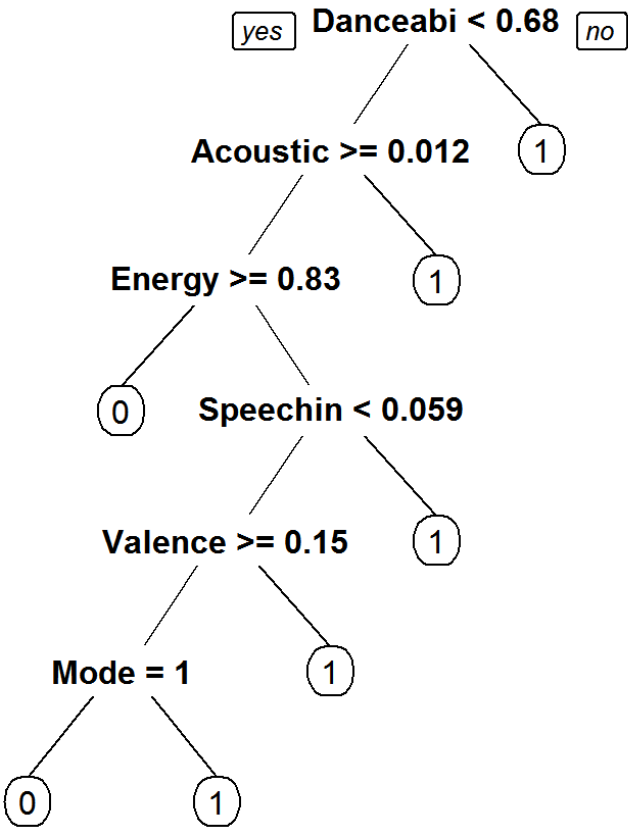
Decision tree

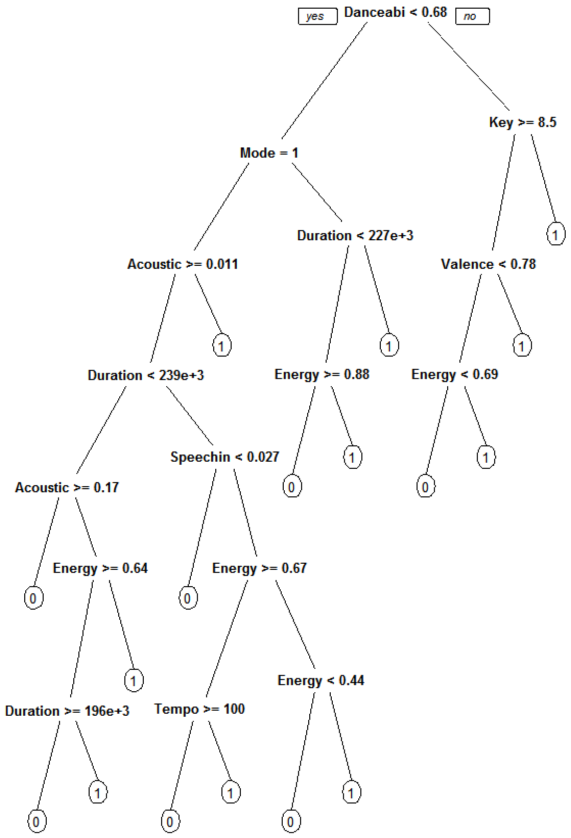
The decision tree is one of the classification model which partitions data according to class. This supervised learning model uses recursive partioning function which partitions the succesfulness of the song recursively according to its features. The decision tree can also be pruned to avoid overfitting the dataset and can be shown.

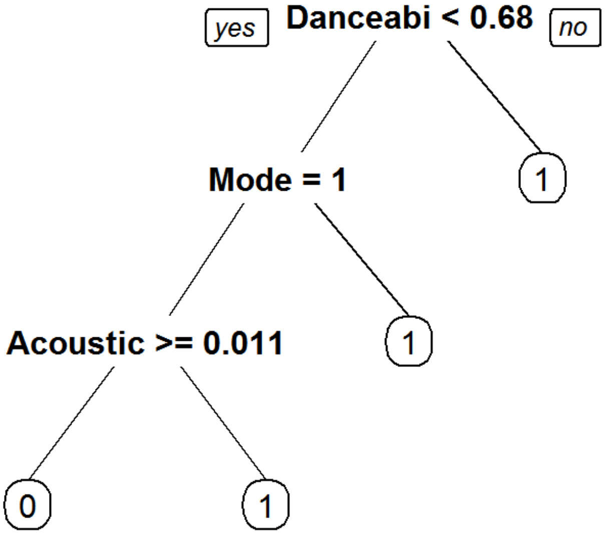
The decision tree partions from the root node which can be any of the features of the song to branch out leaf nodes according to yes/no condition. For example, in the songs dataset, the audio feature, Danceability is the root node which partions itself based on succesfulness of the song i.e if the Danceability is less or greater than 0.68, it branches out into two leaf nodes based on the condition.

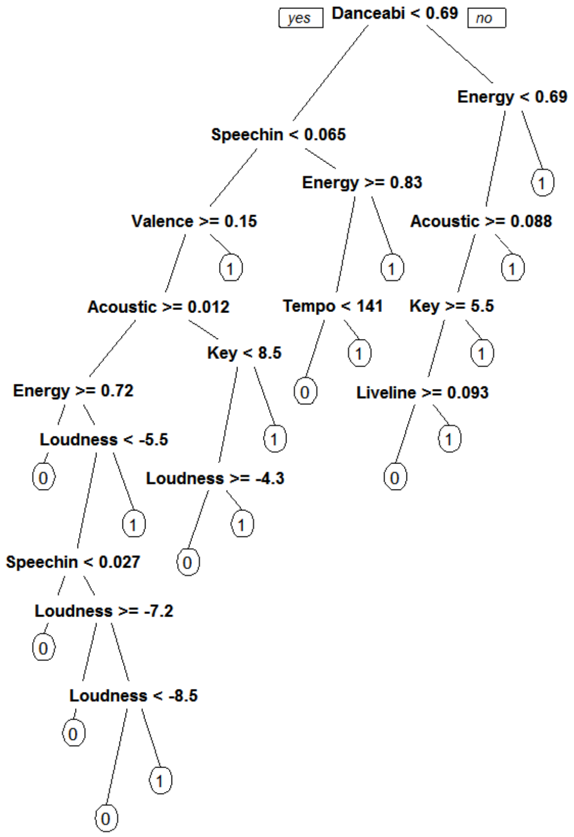
```
dtree <- function(x){
       n <- nrow(songs)
shuffled <- songs[sample(n),]
set.seed(1)
accs <- rep(0,x)
for (i in 1:x)
{
        indices <- (((i-1) * round((1/x)*nrow(shuffled))) + 1):((i*round((1/x) * n
row(shuffled))))
        train <- shuffled[-indices,]
        test <- shuffled[indices,]
        tree <- rpart(Successfulness ~ Danceability+Energy+Key+Loudness+Mode+Speec
hiness+Acousticness+Instrumentalness+Liveliness+Valence+Tempo+Duration+TimeSignatur
e, train, method = "class")
        pred <- predict(tree,test,type="class")
        conf <- table(test$Successfulness,pred)
        accs[i] <- sum(diag(conf))/sum(conf)
        accs[i]
        par(mfrow=c(1,2))
        prp(tree)
        pruned <- prune(tree,cp=tree$cptable[which.min(tree$cptable[,"xerror"]),"C
P"])
        prp(pruned)
}
mean(accs)
}
dtree(10)
```

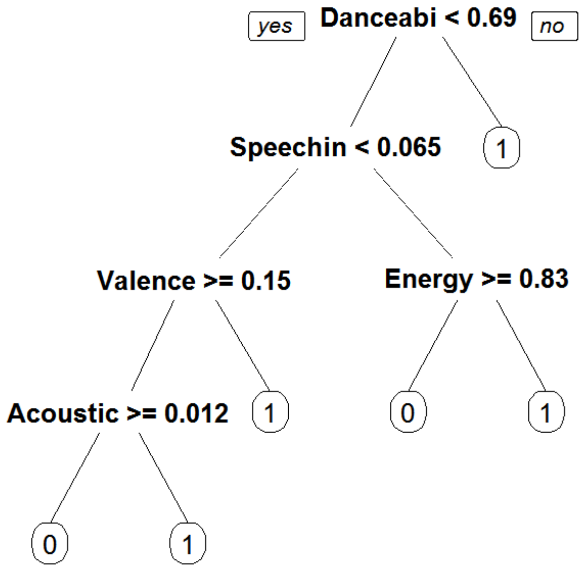

yes
Danceabi < 0.68
no
Energy < 0.69
Acoustic >= 0.012
1
1
Energy >= 0.83
Acoustic >= 0.35
Speechin < 0.059
Key >= 8.5
0
0
1
1
0
Valence >= 0.15
1
Mode = 1
Acoustic >= 0.22
Valence < 0.73
1
1
Loudness < -5.9
0
Energy >= 0.65
0
1
0

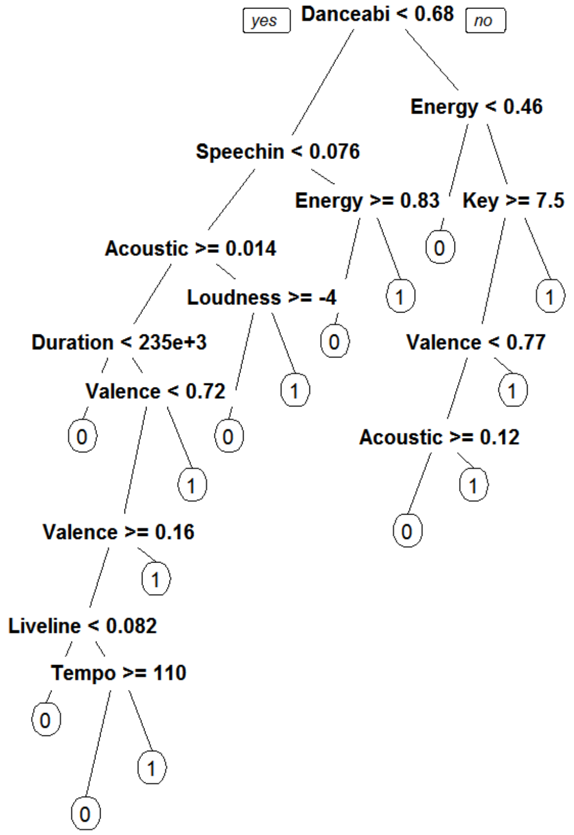

yes
Danceabi < 0.68
no
Acoustic >= 0.012
1
Energy >= 0.83
1
0
Speechin < 0.059
Valence >= 0.15
1
Mode = 1
1
0
1

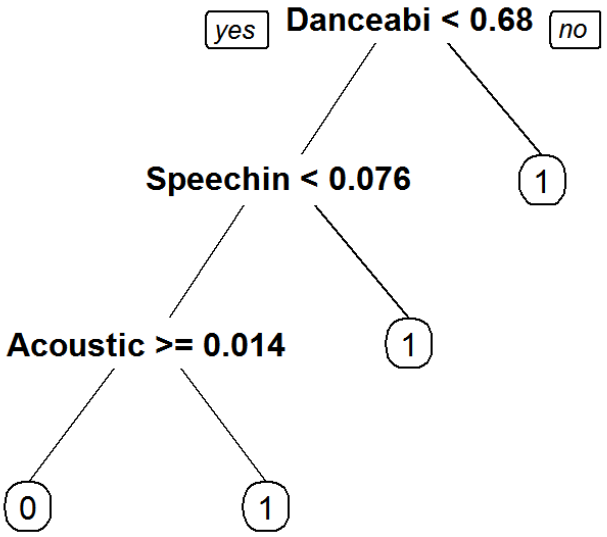

yes
Danceabi < 0.68
no
Key >= 8.5
Mode = 1
Duration < 227e+3
1
Acoustic >= 0.011
Valence < 0.78
1
1
1
Duration < 239e+3
Energy >= 0.88
Energy < 0.69
Speechin < 0.027
1
1
Acoustic >= 0.17
0
0
Energy >= 0.64
Energy >= 0.67
0
0
1
Energy < 0.44
Duration >= 196e+3
Tempo >= 100
1
1
1
0
0
0

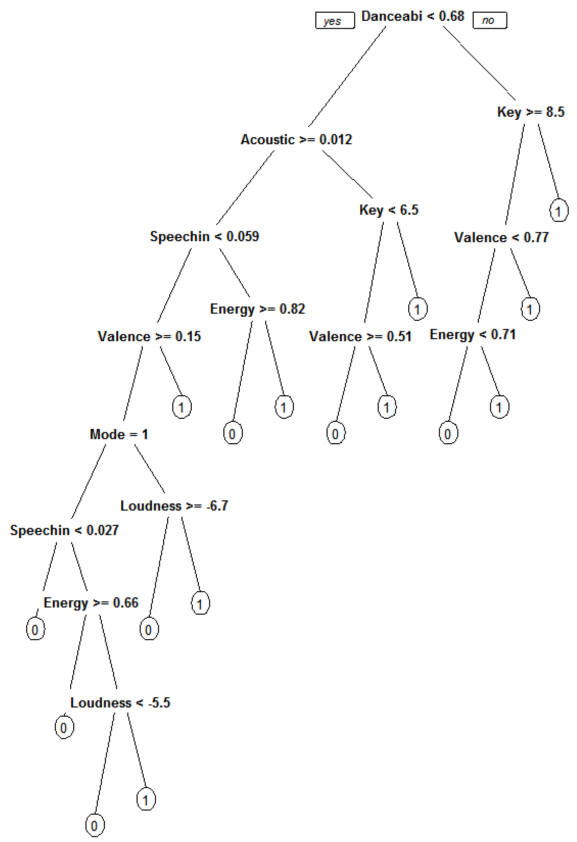

yes
Danceabi < 0.68
no
Mode = 1
1
Acoustic >= 0.011
1
0
1

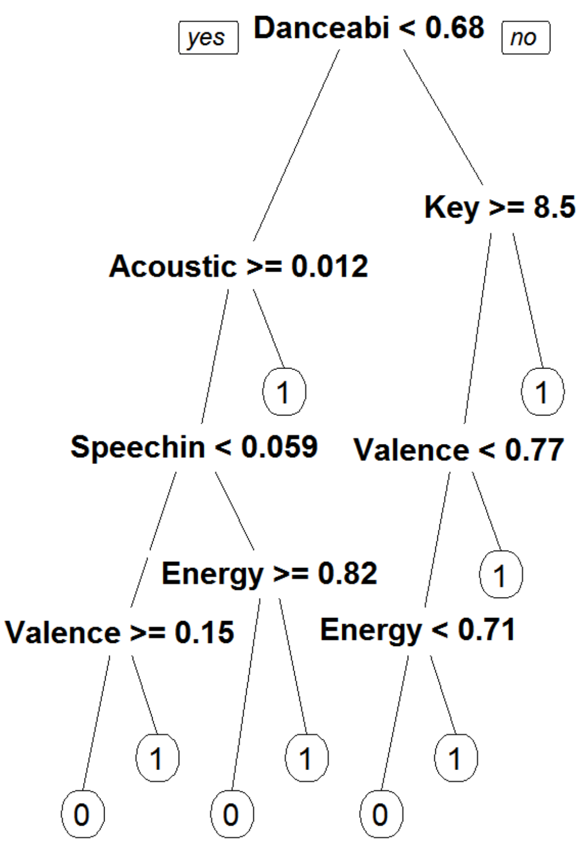

yes
Danceabi < 0.69
no
Energy < 0.69
Speechin < 0.065
1
Energy >= 0.83
Valence >= 0.15
Acoustic >= 0.088
1
1
1
Acoustic >= 0.012
Tempo < 141
Key >= 5.5
Key < 8.5
1
1
0
Energy >= 0.72
Liveline >= 0.093
Loudness < -5.5
1
1
0
Loudness >= -4.3
0
1
1
Speechin < 0.027
0
Loudness >= -7.2
0
Loudness < -8.5
0
1
0

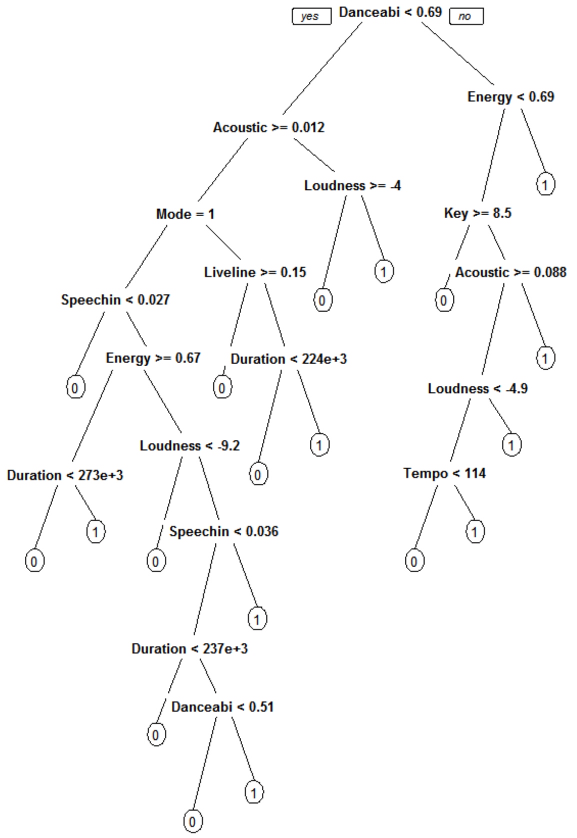

yes
Danceabi < 0.69
no
Speechin < 0.065
1
Valence >= 0.15
Energy >= 0.83
Acoustic >= 0.012
1
0
1
0
1

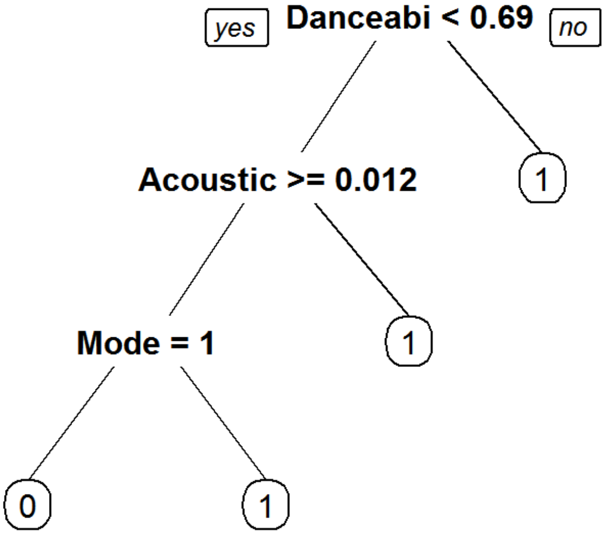

yes
Danceabi < 0.68
no
Energy < 0.46
Speechin < 0.076
Energy >= 0.83
Key >= 7.5
Acoustic >= 0.014
0
Loudness >= -4
1
1
Duration < 235e+3
0
Valence < 0.77
Valence < 0.72
1
1
0
0
Acoustic >= 0.12
1
1
0
Valence >= 0.16
1
Liveline < 0.082
Tempo >= 110
0
1
0

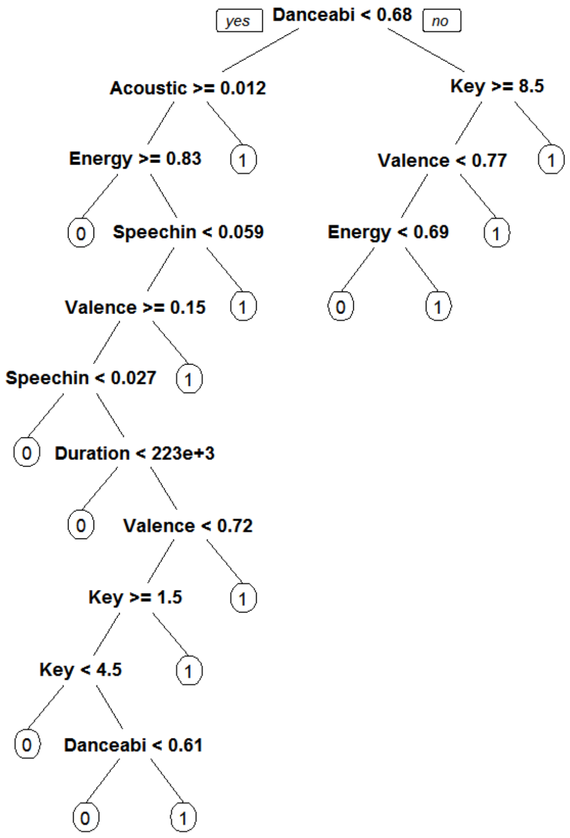

yes
Danceabi < 0.68
no
Speechin < 0.076
1
Acoustic >= 0.014
1
0
1

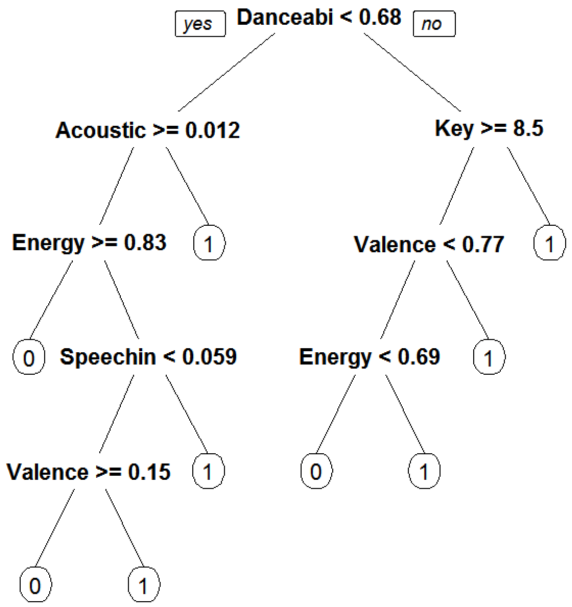

yes
Danceabi < 0.68
no
Key >= 8.5
Acoustic >= 0.012
1
Key < 6.5
Valence < 0.77
Speechin < 0.059
1
1
Energy >= 0.82
Valence >= 0.51
Energy < 0.71
Valence >= 0.15
1
0
1
0
1
0
1
Mode = 1
Loudness >= -6.7
Speechin < 0.027
Energy >= 0.66
1
0
0
Loudness < -5.5
0
1
0

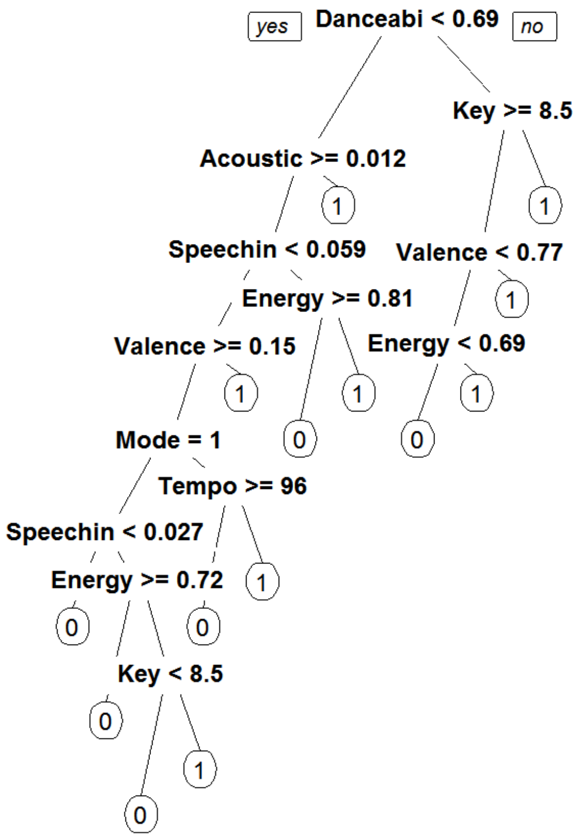

yes
Danceabi < 0.68
no
Key >= 8.5
Acoustic >= 0.012
1
1
Speechin < 0.059
Valence < 0.77
Energy >= 0.82
1
Valence >= 0.15
Energy < 0.71
1
1
1
0
0
0

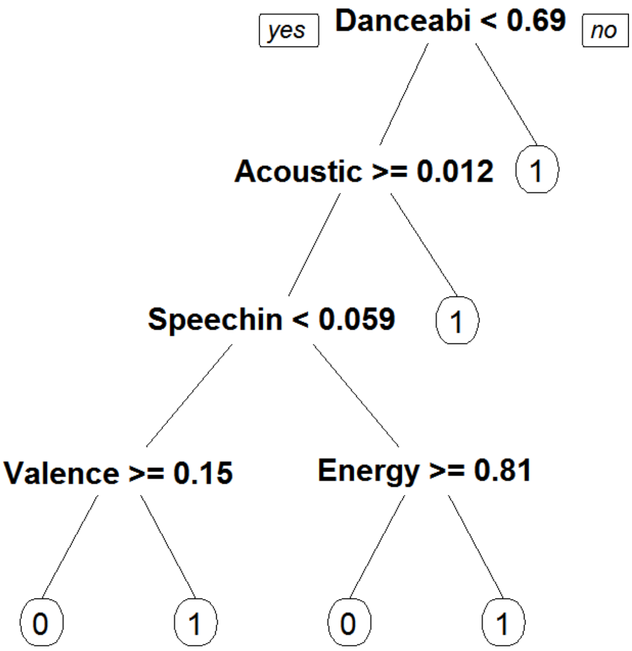

yes
Danceabi < 0.69
no
Energy < 0.69
Acoustic >= 0.012
Loudness >= -4
1
Key >= 8.5
Mode = 1
Liveline >= 0.15
1
0
0
Acoustic >= 0.088
Speechin < 0.027
Energy >= 0.67
Duration < 224e+3
0
1
Loudness < -4.9
0
Loudness < -9.2
1
1
0
Duration < 273e+3
Tempo < 114
1
Speechin < 0.036
1
0
0
0
1
Duration < 237e+3
Danceabi < 0.51
0
1
0

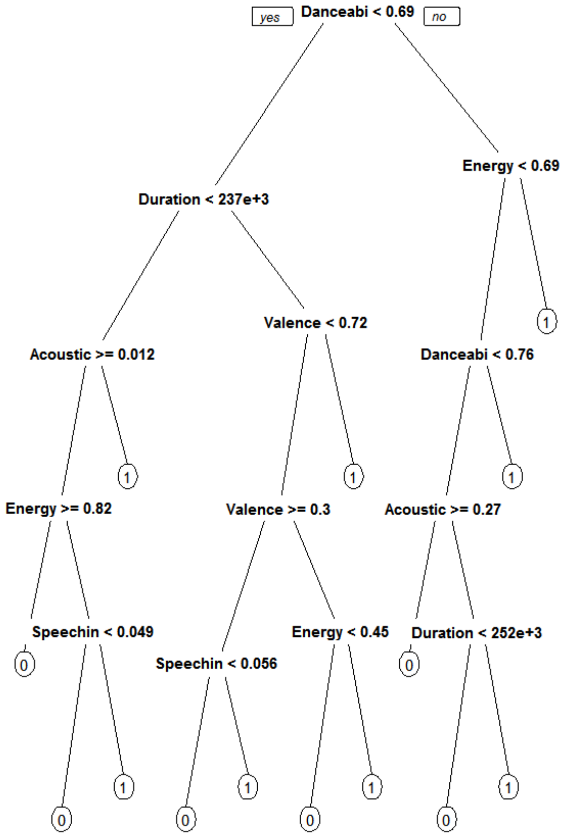

yes
Danceabi < 0.69
no
Acoustic >= 0.012
1
Mode = 1
1
0
1

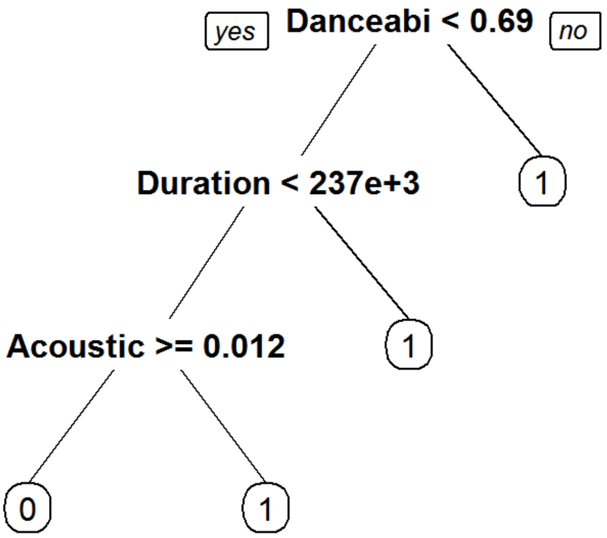

yes
Danceabi < 0.68
no
Acoustic >= 0.012
Key >= 8.5
Energy >= 0.83
1
Valence < 0.77
1
0
Speechin < 0.059
Energy < 0.69
1
Valence >= 0.15
1
0
1
Speechin < 0.027
1
0
Duration < 223e+3
0
Valence < 0.72
Key >= 1.5
1
Key < 4.5
1
0
Danceabi < 0.61
0
1

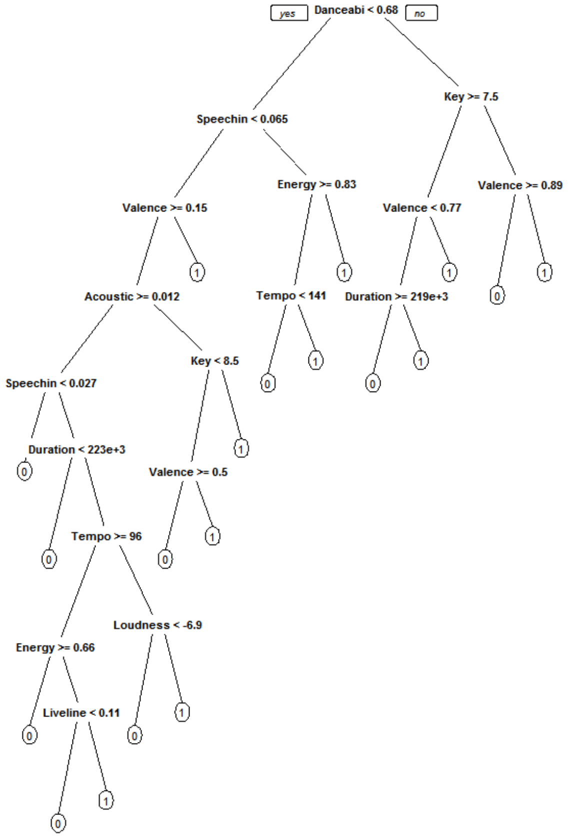

yes
Danceabi < 0.68
no
Acoustic >= 0.012
Key >= 8.5
Energy >= 0.83
1
Valence < 0.77
1
0
Speechin < 0.059
Energy < 0.69
1
Valence >= 0.15
1
0
1
0
1

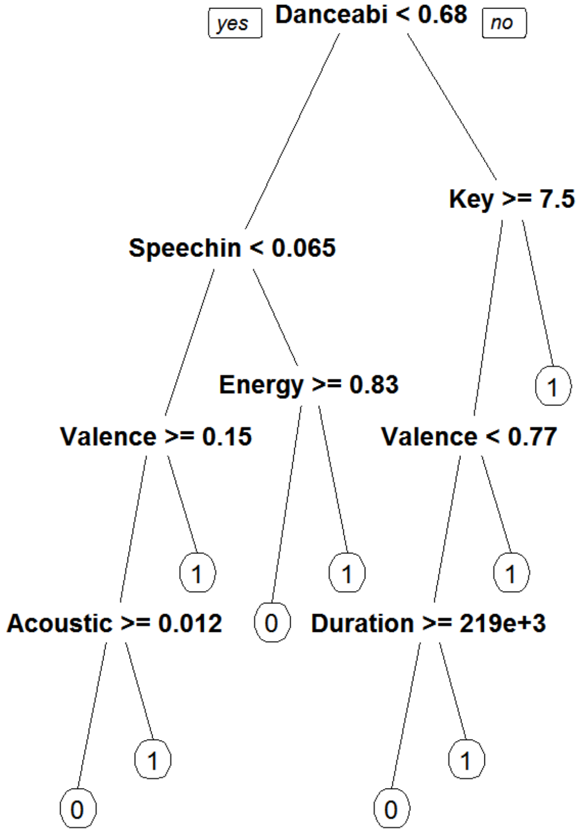

yes
Danceabi < 0.69
no
Key >= 8.5
Acoustic >= 0.012
1
1
Speechin < 0.059
Valence < 0.77
Energy >= 0.81
1
Valence >= 0.15
Energy < 0.69
1
1
1
Mode = 1
0
0
Tempo >= 96
Speechin < 0.027
Energy >= 0.72
1
0
0
Key < 8.5
0
1
0


yes
Danceabi < 0.69
no
Acoustic >= 0.012
1
Speechin < 0.059
1
Valence >= 0.15
Energy >= 0.81
0
1
0
1


yes
Danceabi < 0.69
no
Duration < 237e+3
Energy < 0.69
Acoustic >= 0.012
Valence < 0.72
Danceabi < 0.76
Energy >= 0.82
Valence >= 0.3
Acoustic >= 0.27
Speechin < 0.049
Speechin < 0.056
Energy < 0.45
Duration < 252e+3


yes
Danceabi < 0.69
no
Duration < 237e+3
Acoustic >= 0.012


yes
Danceabi < 0.68
no
Speechin < 0.065
Key >= 7.5
Energy >= 0.83
Valence >= 0.89
Valence >= 0.15
Valence < 0.77
Acoustic >= 0.012
Tempo < 141
Duration >= 219e+3
Key < 8.5
Speechin < 0.027
Duration < 223e+3
Valence >= 0.5
Tempo >= 96
Loudness < -6.9
Energy >= 0.66
Liveline < 0.11

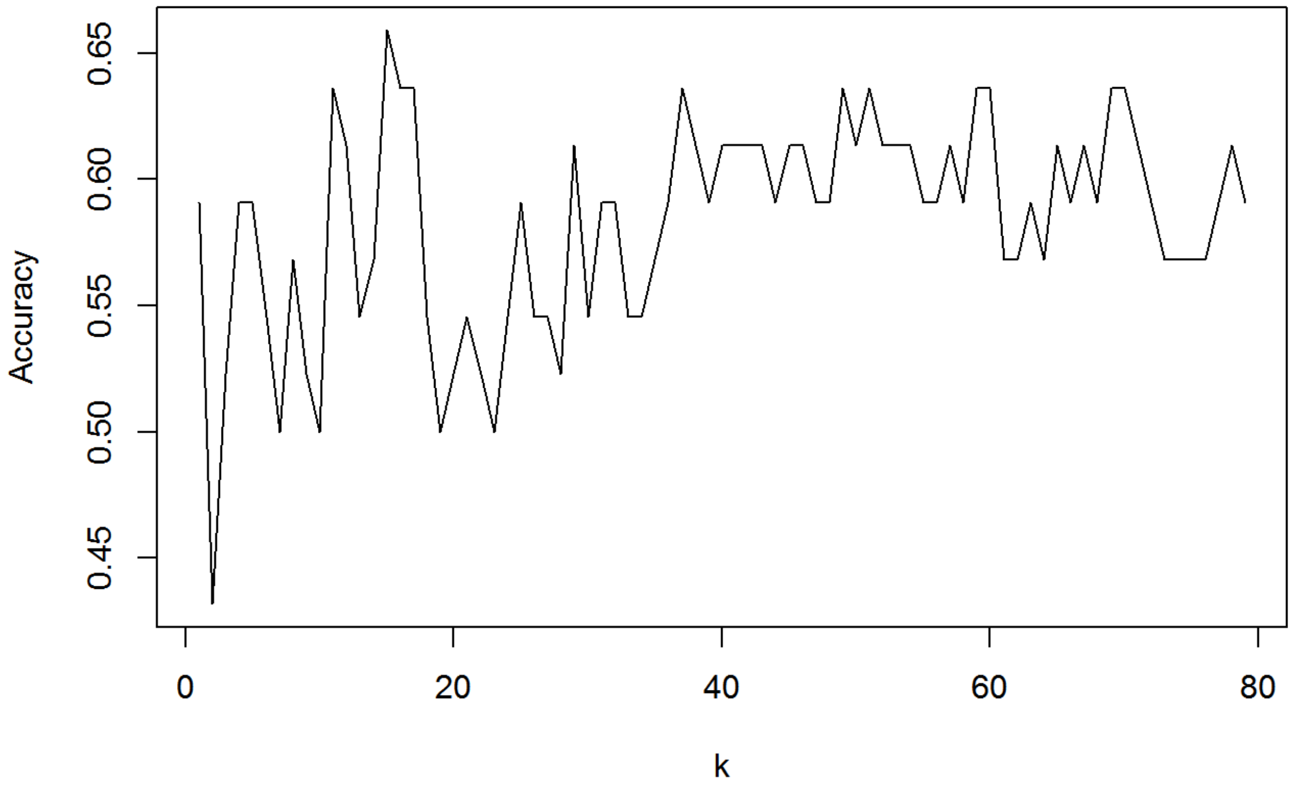

yes
Danceabi < 0.68
no
Key >= 7.5
Speechin < 0.065
Energy >= 0.83
Valence >= 0.15
Valence < 0.77
Acoustic >= 0.012
Duration >= 219e+3

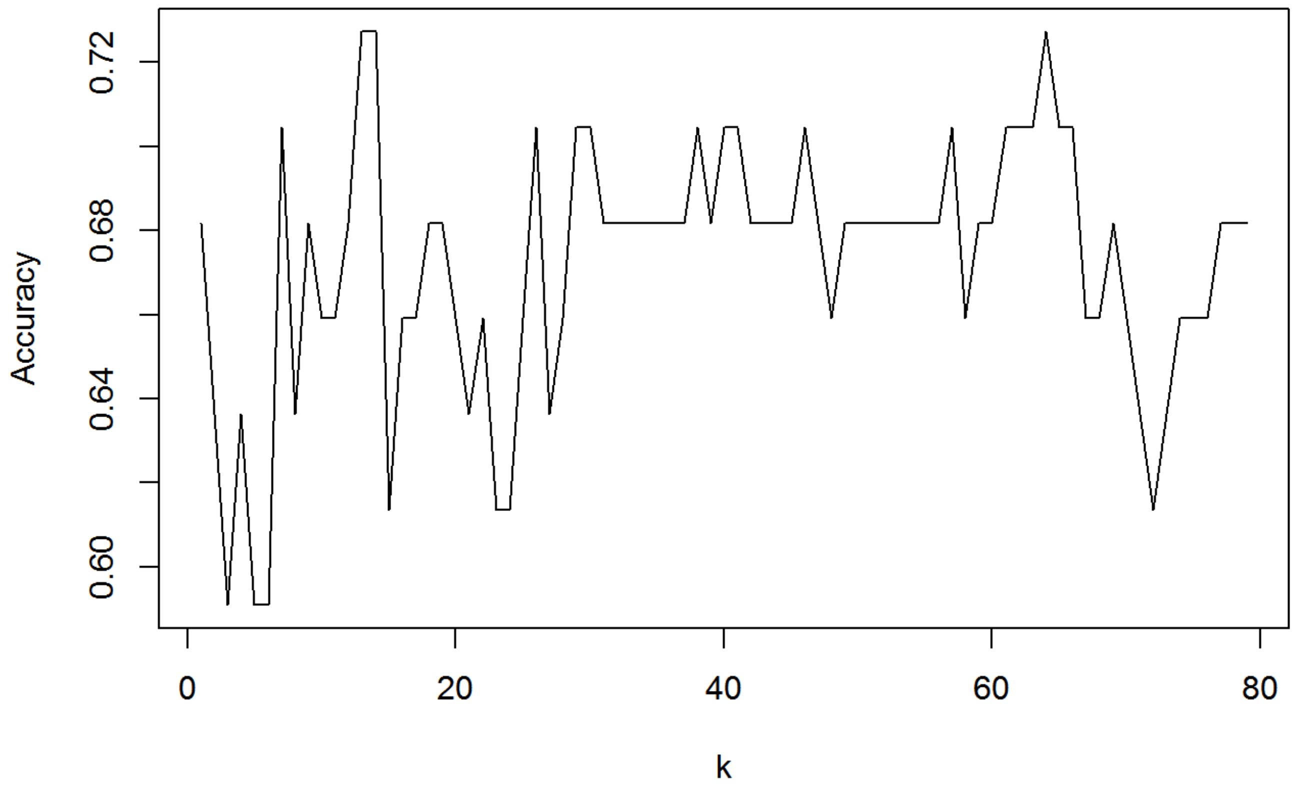
```
## [1] 0.6613636
```

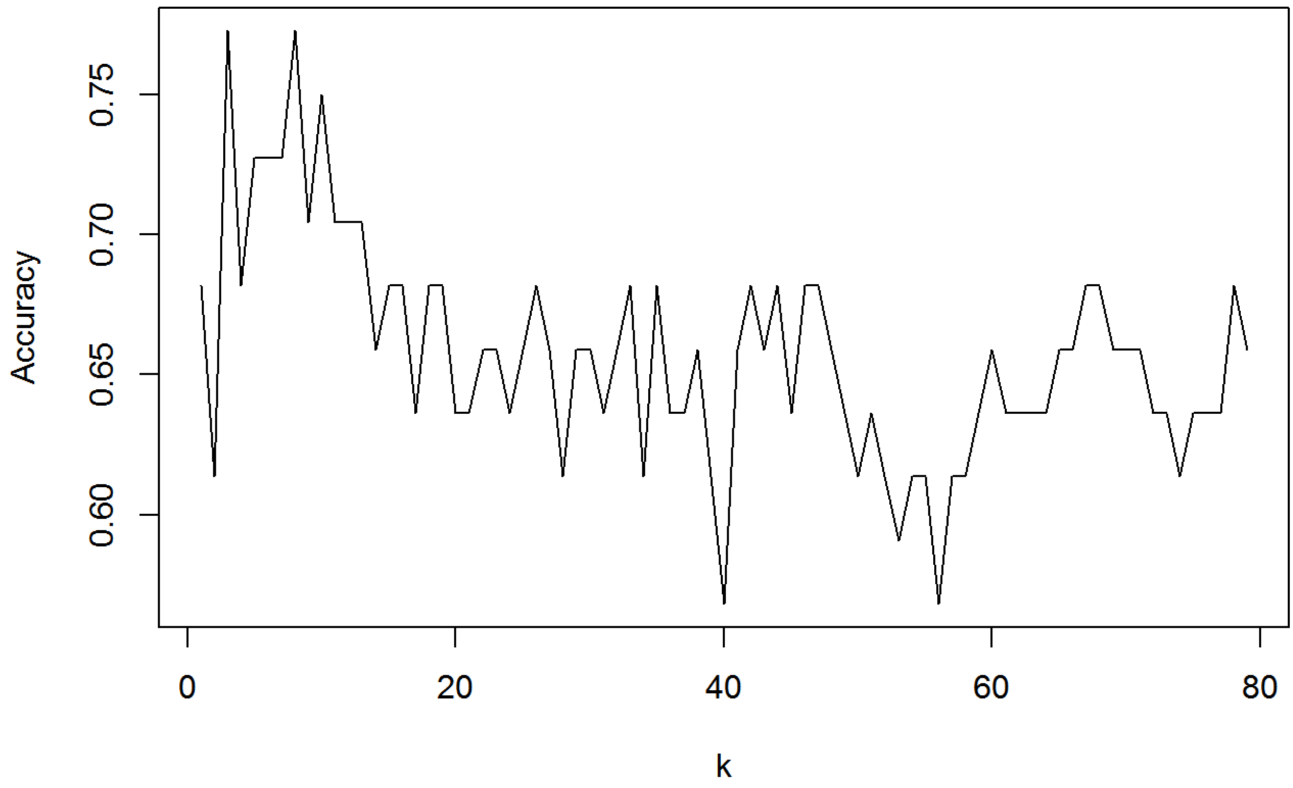
K Nearest Neighbours

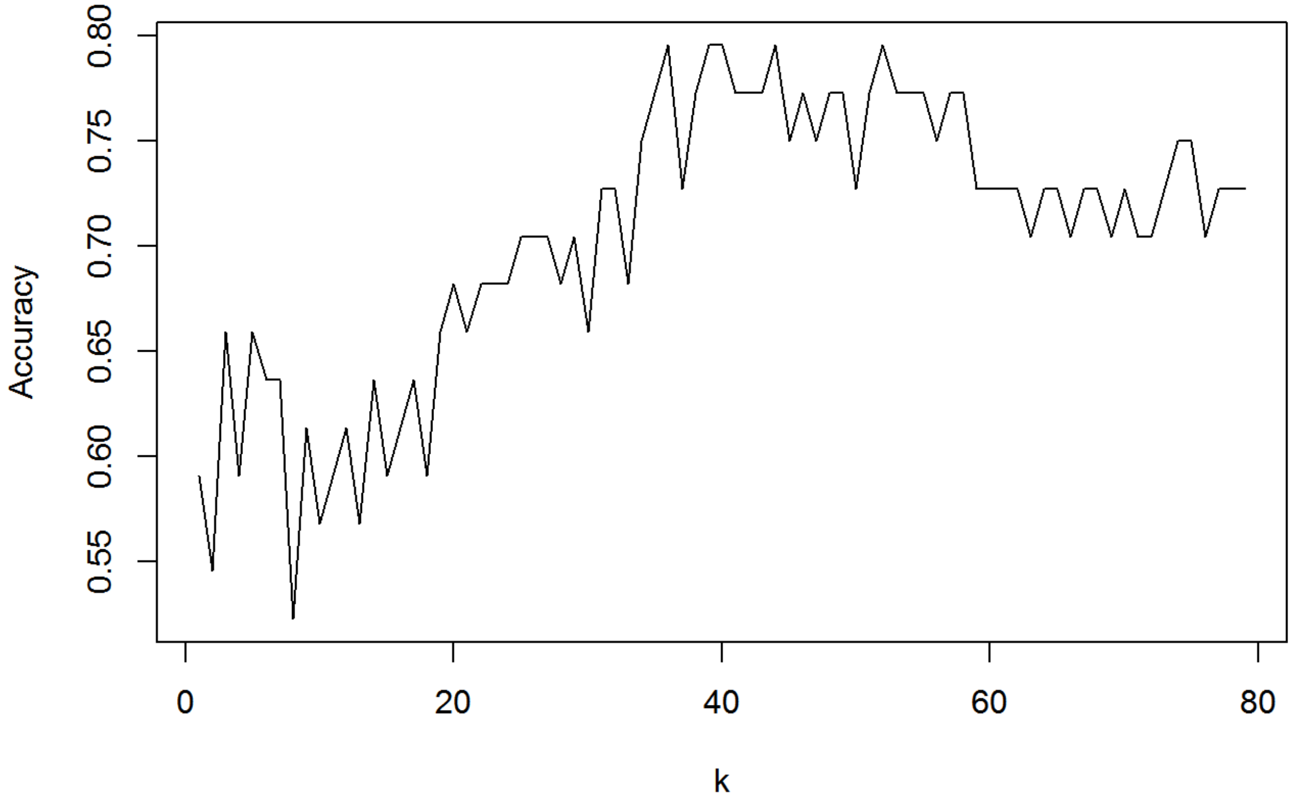
K Nearest Neighbour is another classification model which predicts the test song based on the Kth nearest neighbour(Euclidean Distance) from the training dataset. Based on K values, the Euclidean distance of test dataset is measured from the K nearest neighbours of training dataset and classified as Successful or unsuccessful accordingly. The audio features are normalized to fit the scale and avoid Scaling issue. Here the Kvalue is determined by a certain range of values with regards to training dataset. The k value with the highest accuracy (as shown in plots) is taken for every iteration (cross validation) and the accuracies are found out for every iteration. The mean of accuracy is calculated.

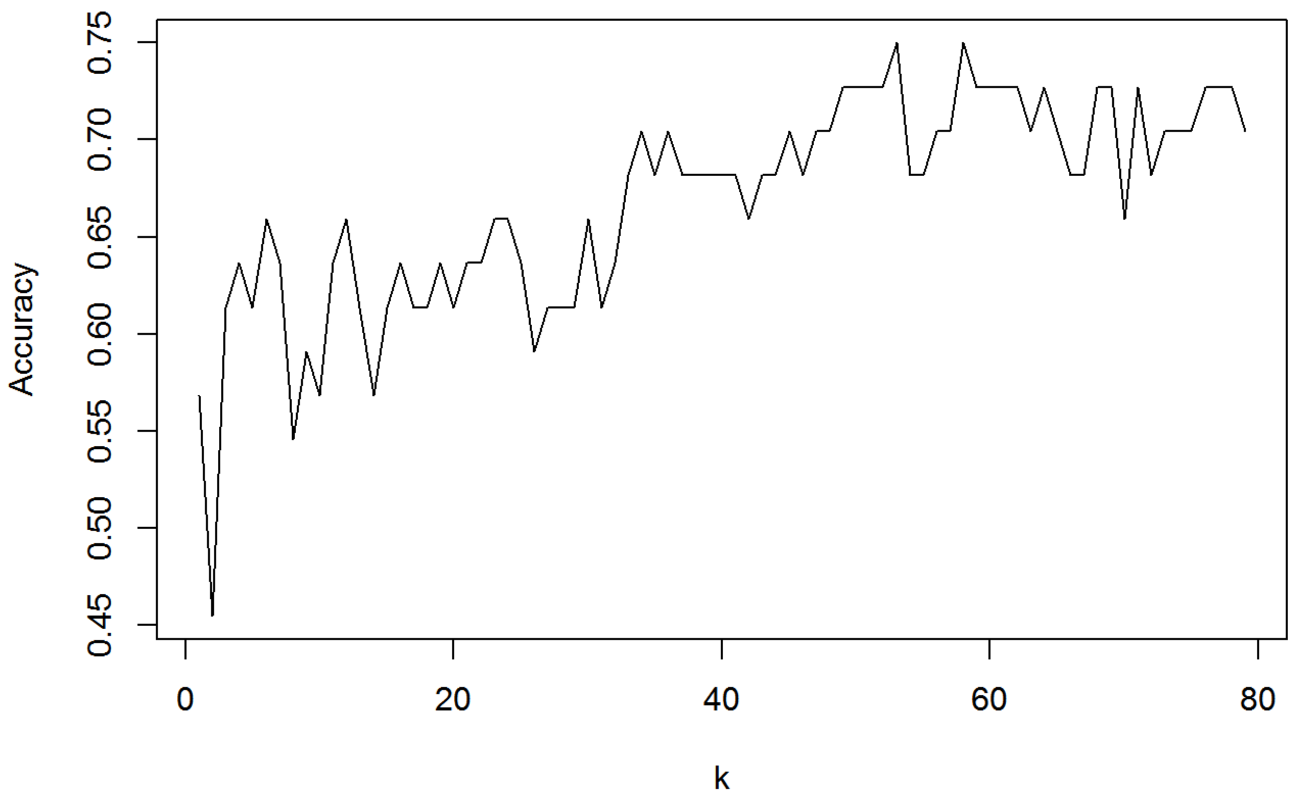
```
m <- nrow(songs)
shuffled <- songs[sample(m),]
set.seed(1)
accs <- rep(0,10)
for(i in 1:10){
indices <- (((i-1) * round((1/10)*nrow(shuffled))) + 1):((i*round((1/10) * nrow(sh
uffled))))
train <- shuffled[-indices,]
test <- shuffled[indices,]
# Normalizing key
min_Key <- min(train$Key)
max_Key <- max(train$Key)
train$Key <- (train$Key - min_Key) / (max_Key - min_Key)
test$Key <- (test$Key - min_Key) / (max_Key - min_Key)

# Normalizing Loudness
min_Loud <- min(train$Loudness)
max_Loud <- max(train$Loudness)
train$Loudness <- (train$Loudness - min_Loud) / (max_Loud - min_Loud)
test$Loudness <- (test$Loudness - min_Loud) / (max_Loud - min_Loud)

#Normailizing Tempo
min_Tempo <- min(train$Tempo)
max_Tempo <- max(train$Tempo)
train$Tempo <- (train$Tempo - min_Tempo) / (max_Tempo - min_Tempo)
test$Tempo <- (test$Tempo - min_Tempo) / (max_Tempo - min_Tempo)

# Normailizing Duration
min_Dur <- min(train$Duration)
max_Dur <- max(train$Duration)
train$Duration <- (train$Duration - min_Dur) / (max_Dur - min_Dur)
test$Duration <- (test$Duration - min_Dur) / (max_Dur - min_Dur)

#Normalizing Time signature
min_time <- min(as.numeric(train$TimeSignature))
max_time <- max(as.numeric(train$TimeSignature))
train$TimeSignature <- (as.numeric(train$TimeSignature) - min_time) / (max_time - m
in_time)
test$TimeSignature <- (as.numeric(test$TimeSignature) - min_time) / (max_time - min
_time)

train_labels <- as.factor(train[,5])
test_labels <- as.factor(test[,5])

knn_train <- train
knn_test <- test

knn_train$Successfulness <- NULL
knn_test$Successfulness <- NULL

range <- 1:round(0.2 * nrow(knn_train))
accsk <- rep(0, length(range))
for (k in range) {
        predk <- knn(train = knn_train[,c(8:18,24,25)], test = knn_test[,c(8:18,24
,25)], cl = train_labels, k = k)
        confk <- table(test_labels,predk)
        accsk[k] <- sum(diag(confk))/sum(confk)

}

plot(range, accsk, xlab = "k",ylab="Accuracy",type='l')
pred <- knn(train=knn_train[,c(8:18,24,25)], test = knn_test[,c(8:18,24,25)], cl =
train_labels,k=which.max(accsk))
conf <- table(test_labels,pred)
accs[i] <- sum(diag(conf))/sum(conf)
mean_acc <- mean(accs)
}
```

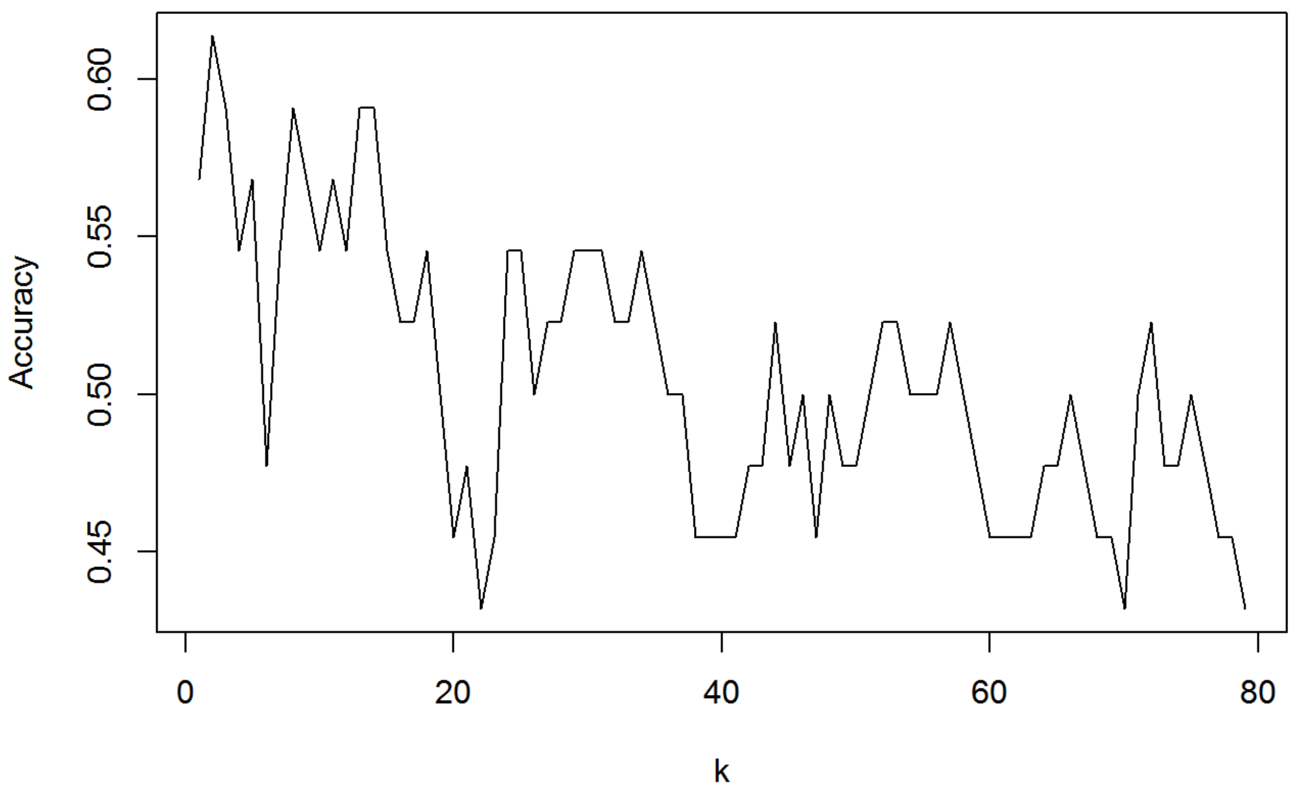


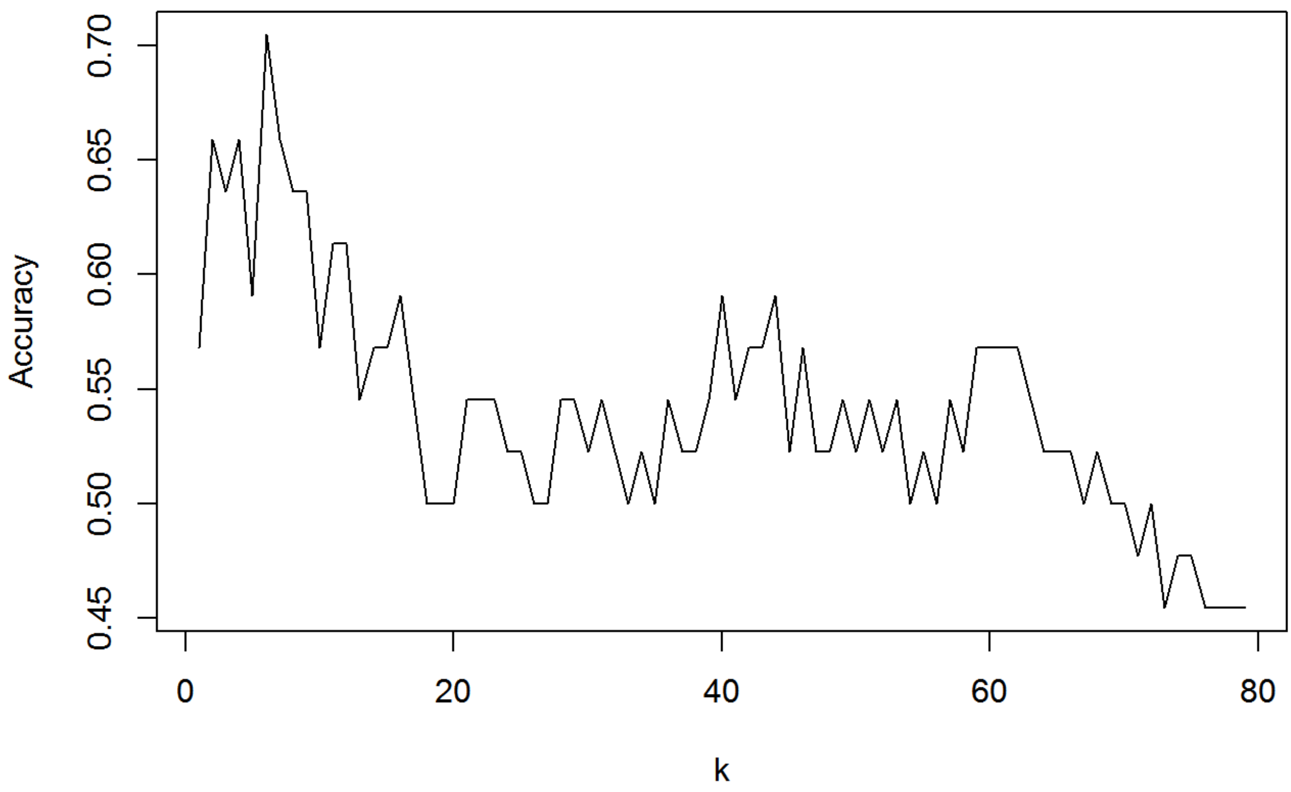

Accuracy
0.60
0.64
0.68
0.72
0
20
40
60
80
k

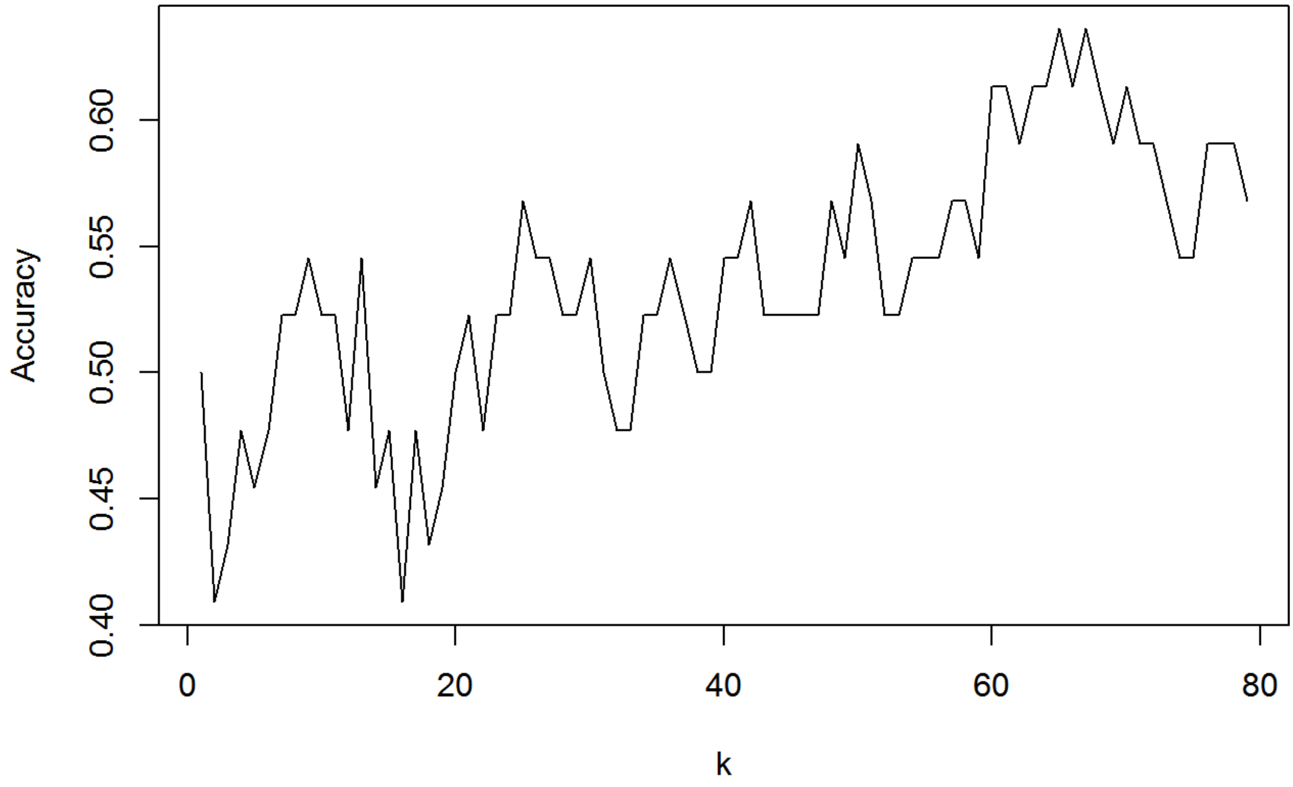

Accuracy
0.60
0.65
0.70
0.75
0
20
40
60
80
k

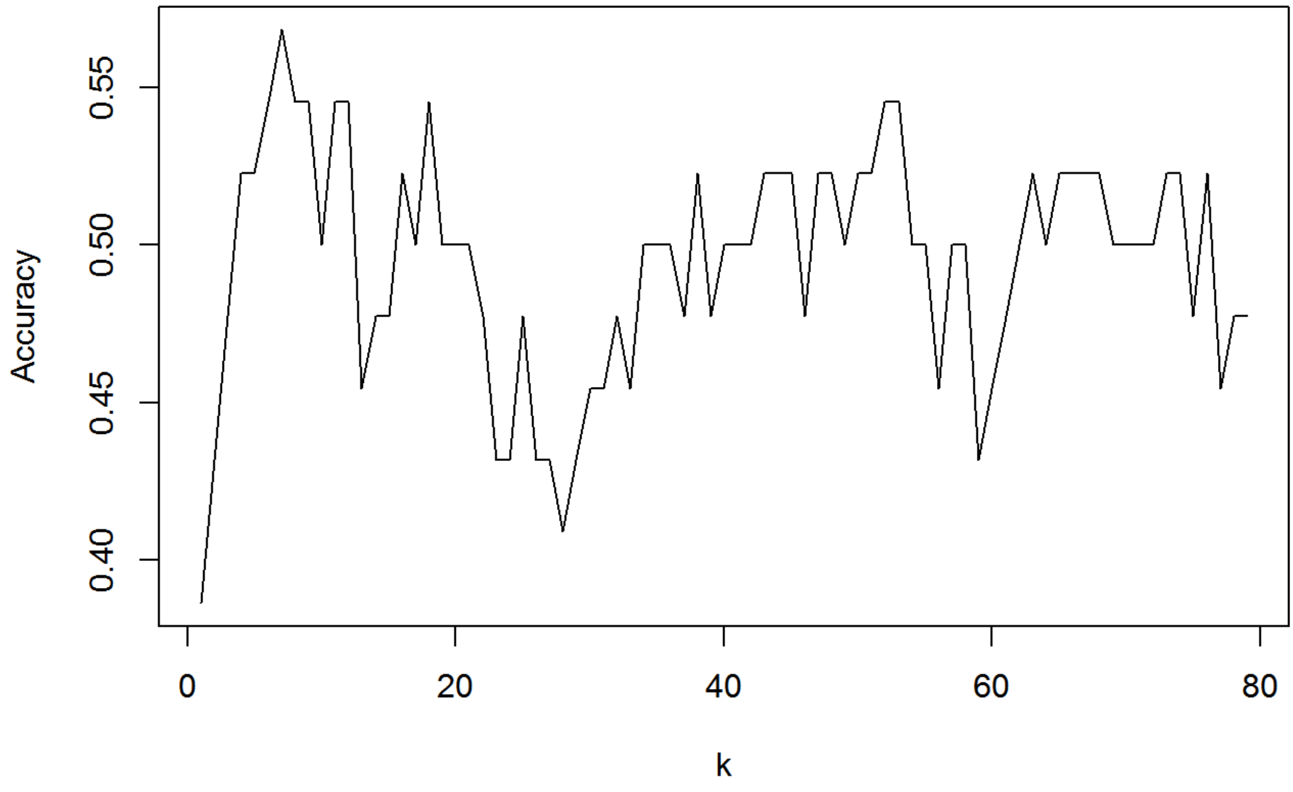

Accuracy
0.55
0.60
0.65
0.70
0.75
0.80
0
20
40
60
80
k

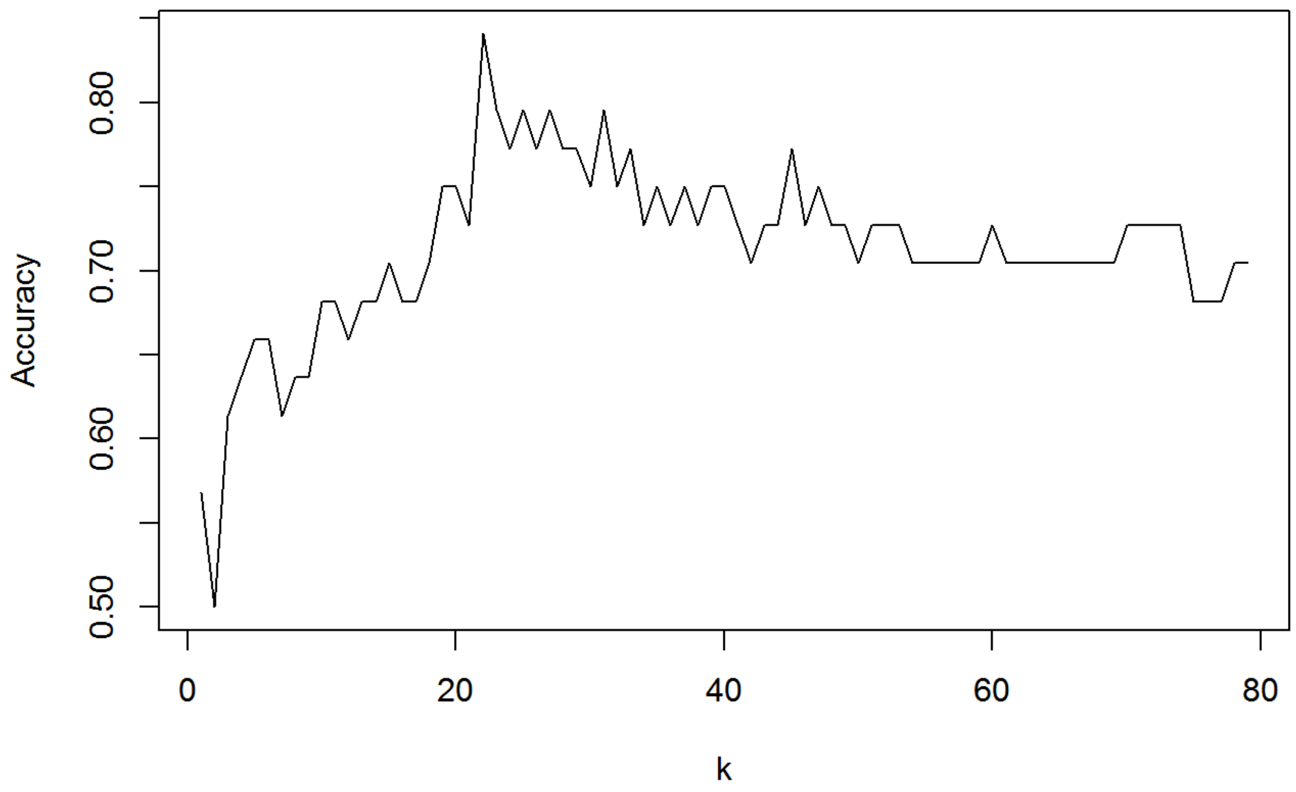

Accuracy
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0
20
40
60
80
k


Accuracy
0.45
0.50
0.55
0.60
0
20
40
60
80
k


Accuracy
0.45
0.50
0.55
0.60
0.65
0.70
0
20
40
60
80
k


Accuracy
0.40
0.45
0.50
0.55
0.60
0
20
40
60
80
k


Accuracy
0.40
0.45
0.50
0.55
0
20
40
60
80
k



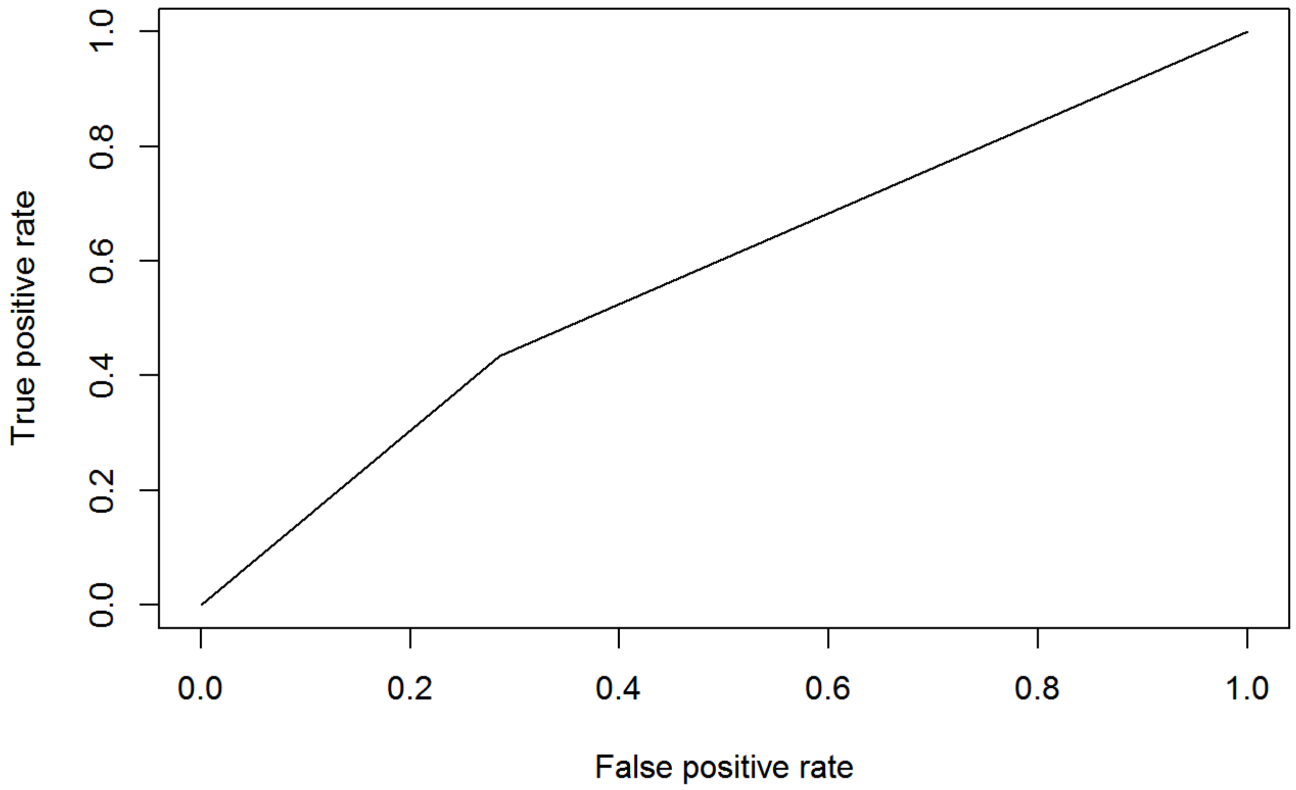
```
mean_acc
```

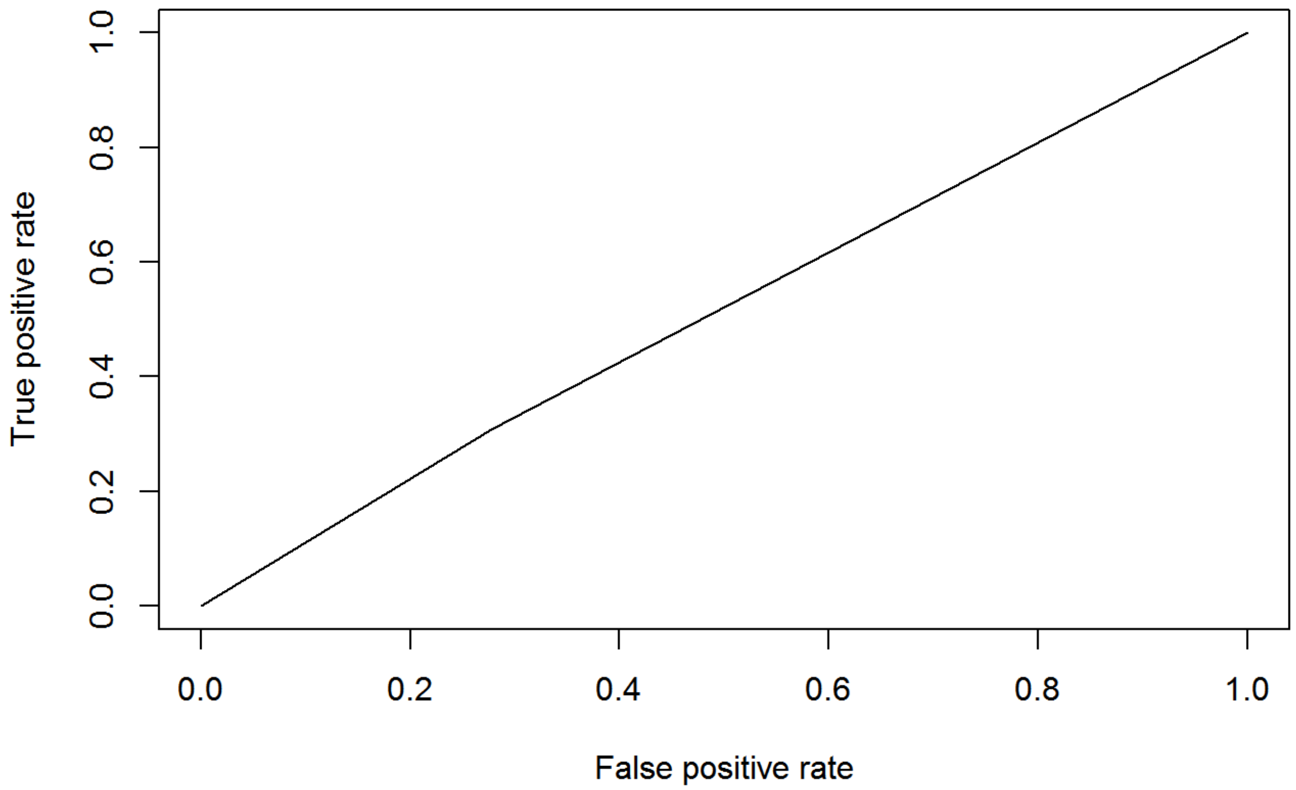
```
## [1] 0.6863636
```

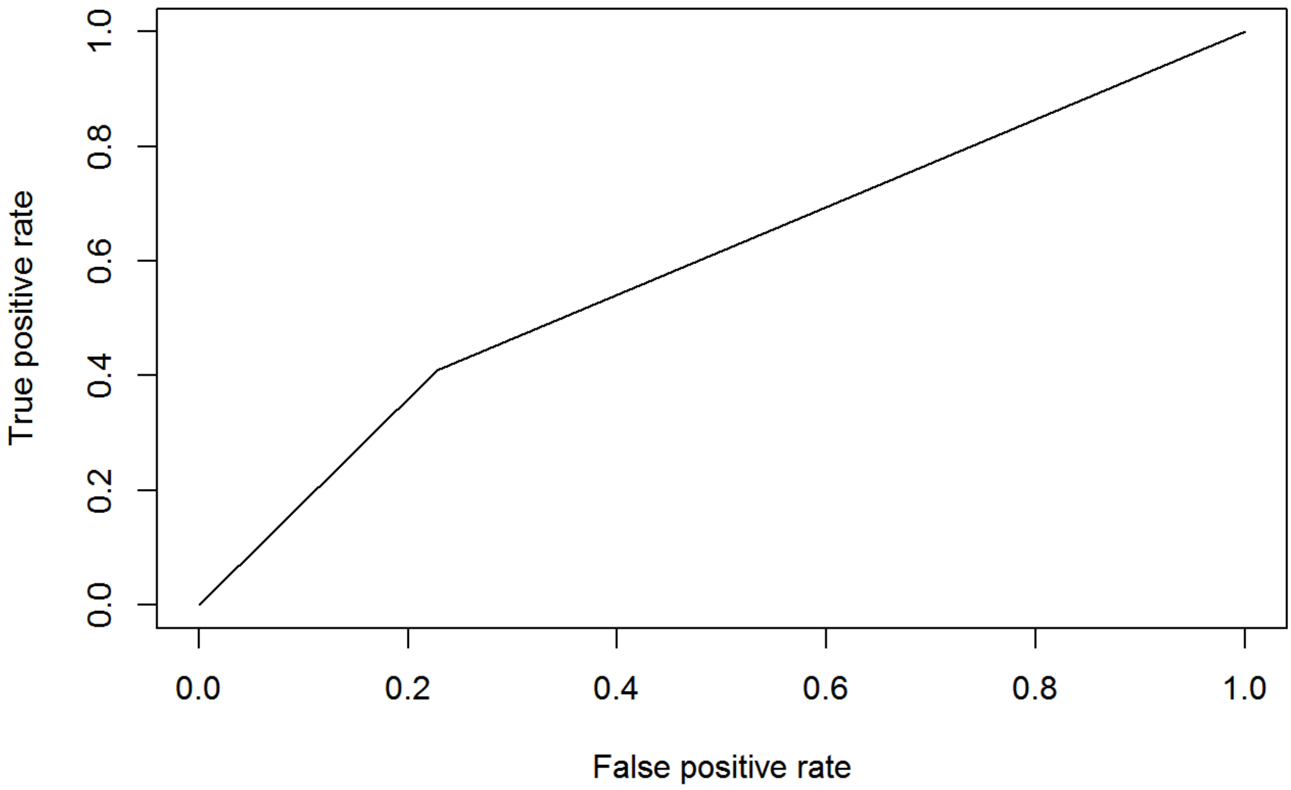
Naive Bayes

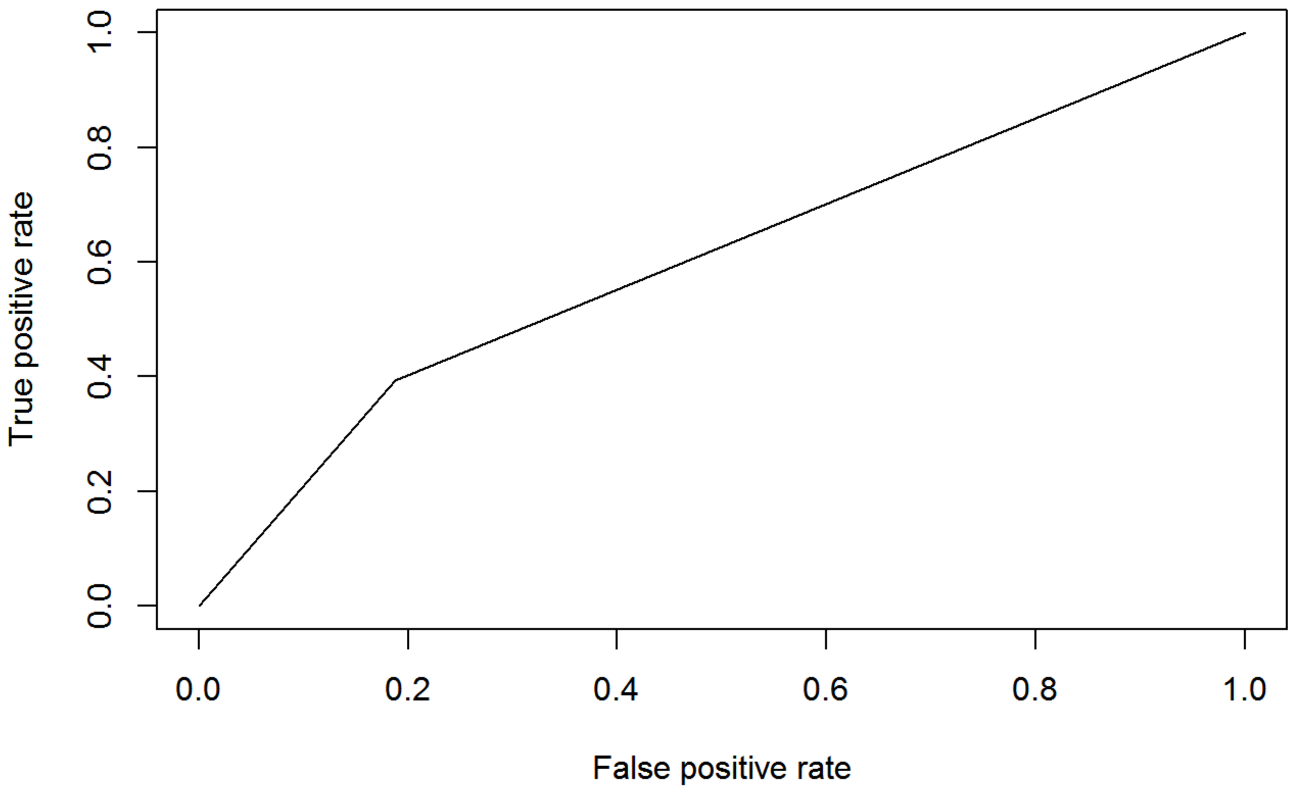
Naive Bayes is based on Bayes Theorem which denotes that the probability of one set is based on the conditional probablity of another set. Naive Bayes comes with strong independence assumptions. Here the probability of succesfulness of the songs, whether it is successful or unsuccessful is based on the conditional probablities of audio features such as Danceabiltiy, Energy, Valence etc. Here the ROC curve denotes the true positive rates against false positive rates.

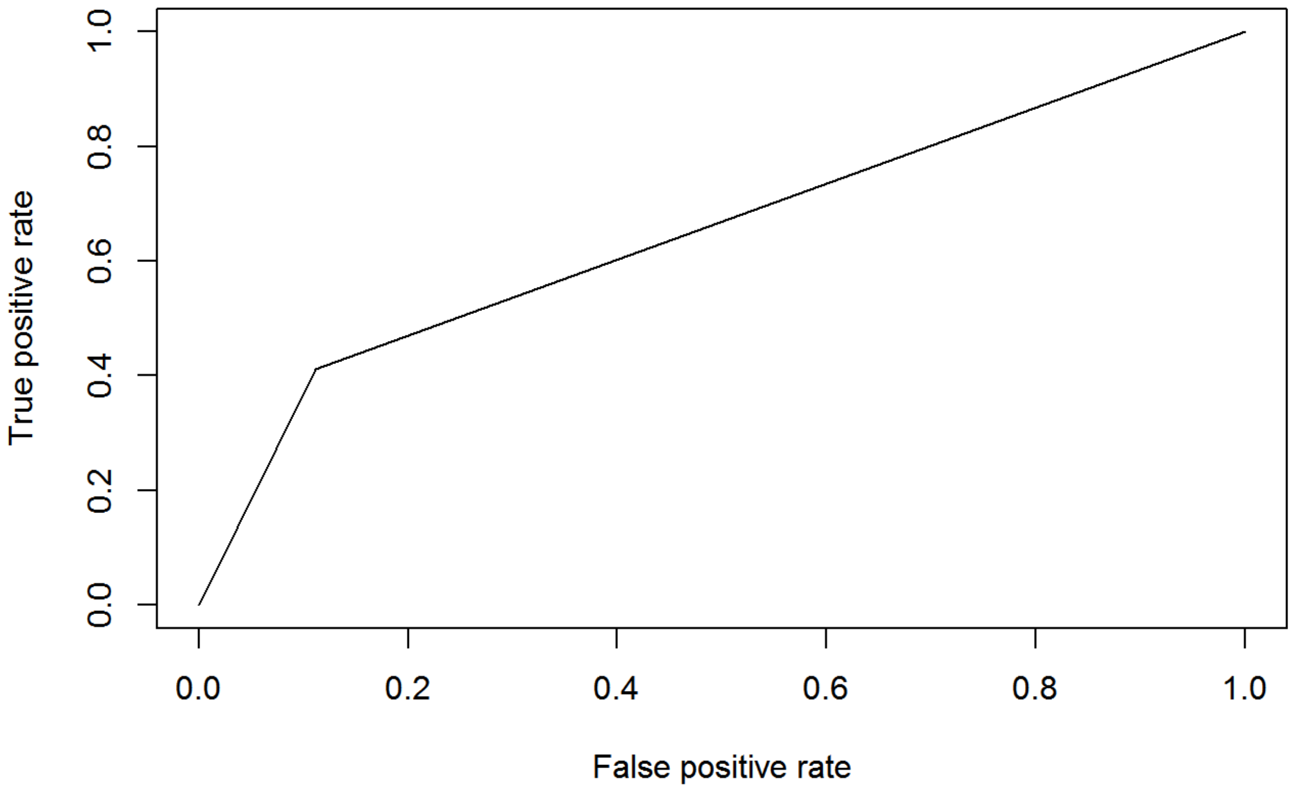
```
nb <- function(y){
       p <- nrow(songs)
shuffled <- songs[sample(p),]
set.seed(1)
accs <- rep(0,y)
for (i in 1:y)
{
        indices <- (((i-1) * round((1/y)*nrow(shuffled))) + 1):((i*round((1/y) * n
row(shuffled))))
        train <- shuffled[-indices,]
        test <- shuffled[indices,]
# Normalizing key
min_Key <- min(train$Key)
max_Key <- max(train$Key)
train$Key <- (train$Key - min_Key) / (max_Key - min_Key)
test$Key <- (test$Key - min_Key) / (max_Key - min_Key)

# Normalizing Loudness
min_Loud <- min(train$Loudness)
max_Loud <- max(train$Loudness)
train$Loudness <- (train$Loudness - min_Loud) / (max_Loud - min_Loud)
test$Loudness <- (test$Loudness - min_Loud) / (max_Loud - min_Loud)

#Normailizing Tempo
min_Tempo <- min(train$Tempo)
max_Tempo <- max(train$Tempo)
train$Tempo <- (train$Tempo - min_Tempo) / (max_Tempo - min_Tempo)
test$Tempo <- (test$Tempo - min_Tempo) / (max_Tempo - min_Tempo)

# Normailizing Duration
min_Dur <- min(train$Duration)
max_Dur <- max(train$Duration)
train$Duration <- (train$Duration - min_Dur) / (max_Dur - min_Dur)
test$Duration <- (test$Duration - min_Dur) / (max_Dur - min_Dur)

#Normalizing Time signature
min_time <- min(as.numeric(train$TimeSignature))
max_time <- max(as.numeric(train$TimeSignature))
train$TimeSignature <- (as.numeric(train$TimeSignature) - min_time) / (max_time - min_time)
test$TimeSignature <- (as.numeric(test$TimeSignature) - min_time) / (max_time - min_time)

nb_model <- naiveBayes(Successfulness ~ Danceability+Energy+Key+Loudness+Mode+Speechiness+Acousticness+Instrumentalness+Liveliness+Valence+Tempo+Duration+TimeSignature, train)
pred <- predict(nb_model,test)
conf <- table(test$Successfulness,pred)
accs[i] <- sum(diag(conf))/sum(conf)

# ROC Curve
nb.pred <- prediction(as.numeric(pred), as.numeric(test$Successfulness))
nb.roc.perf <- performance(nb.pred, "tpr", "fpr")
plot(nb.roc.perf,xlab="False positive rate",ylab="True positive rate")
}
mean(accs)
}
nb(10)
```

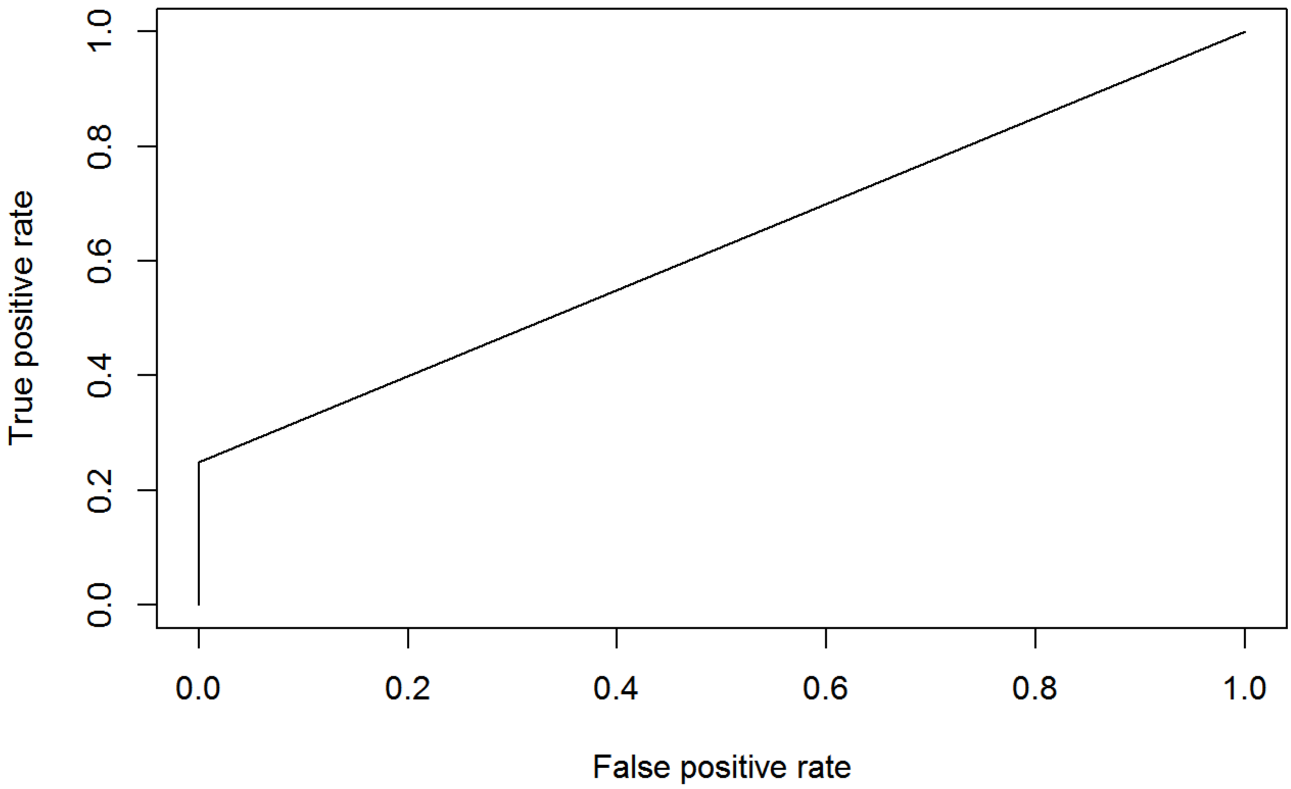

True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

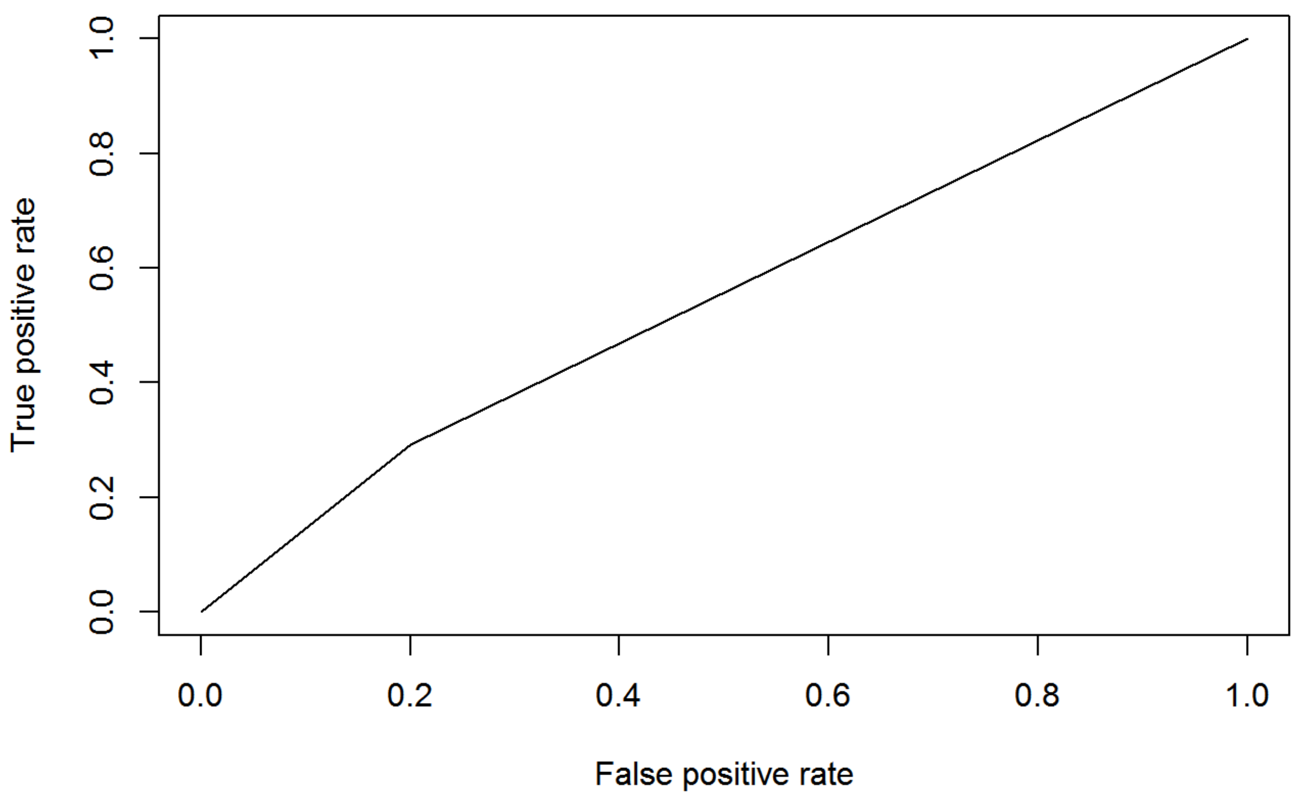

True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

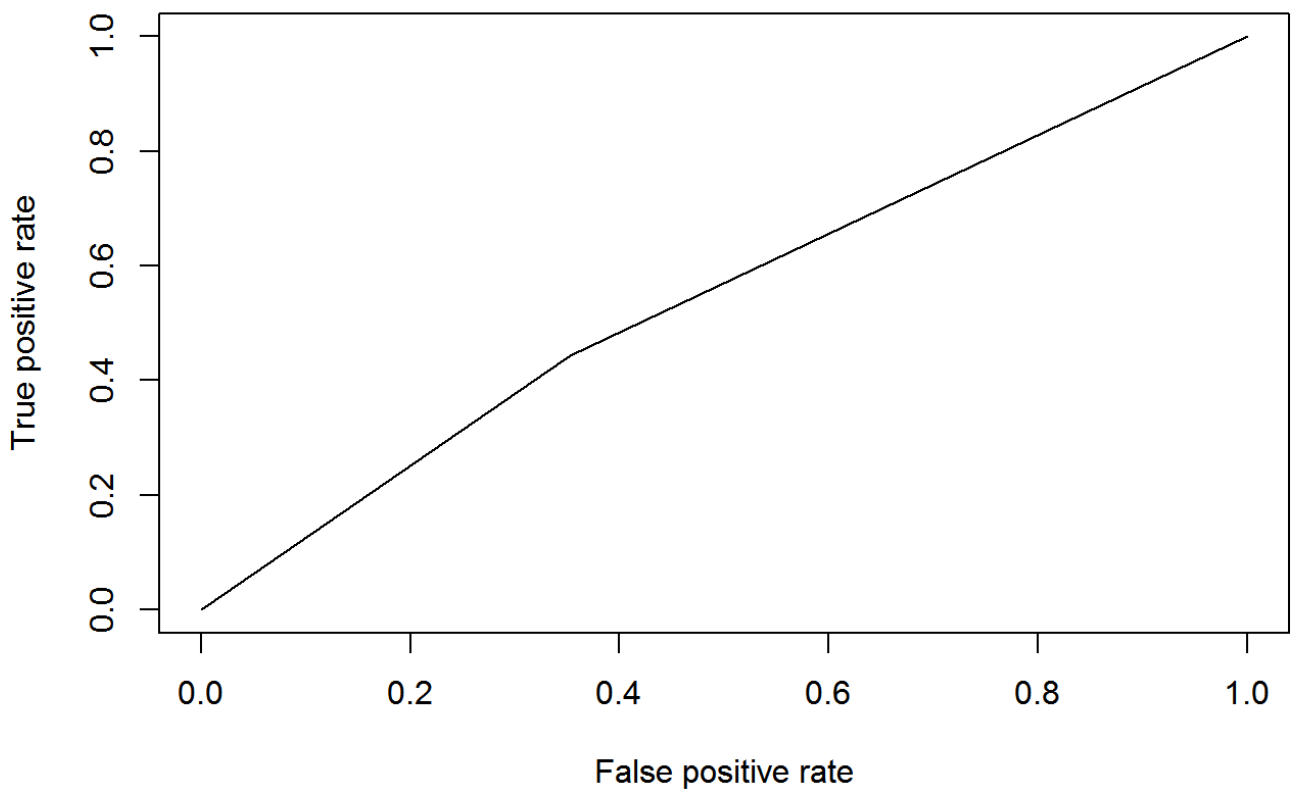

True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

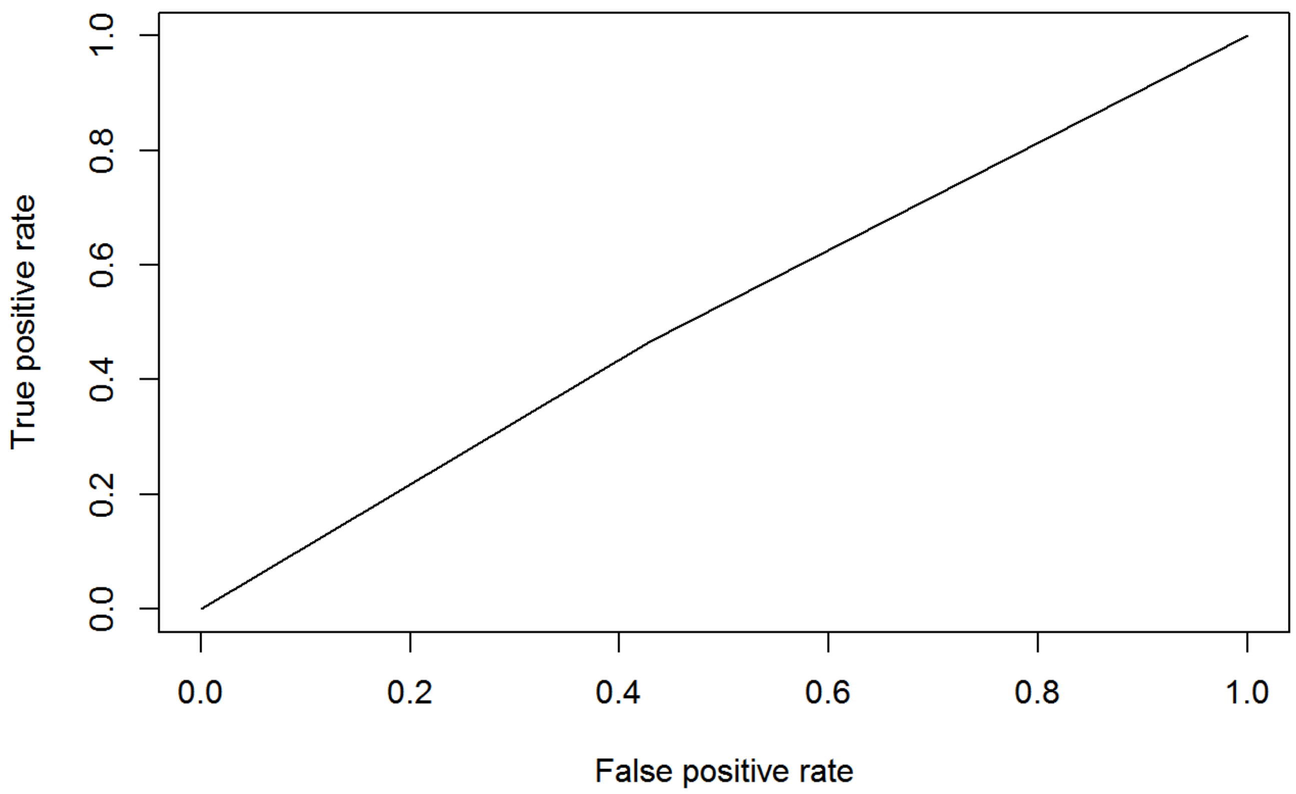

True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

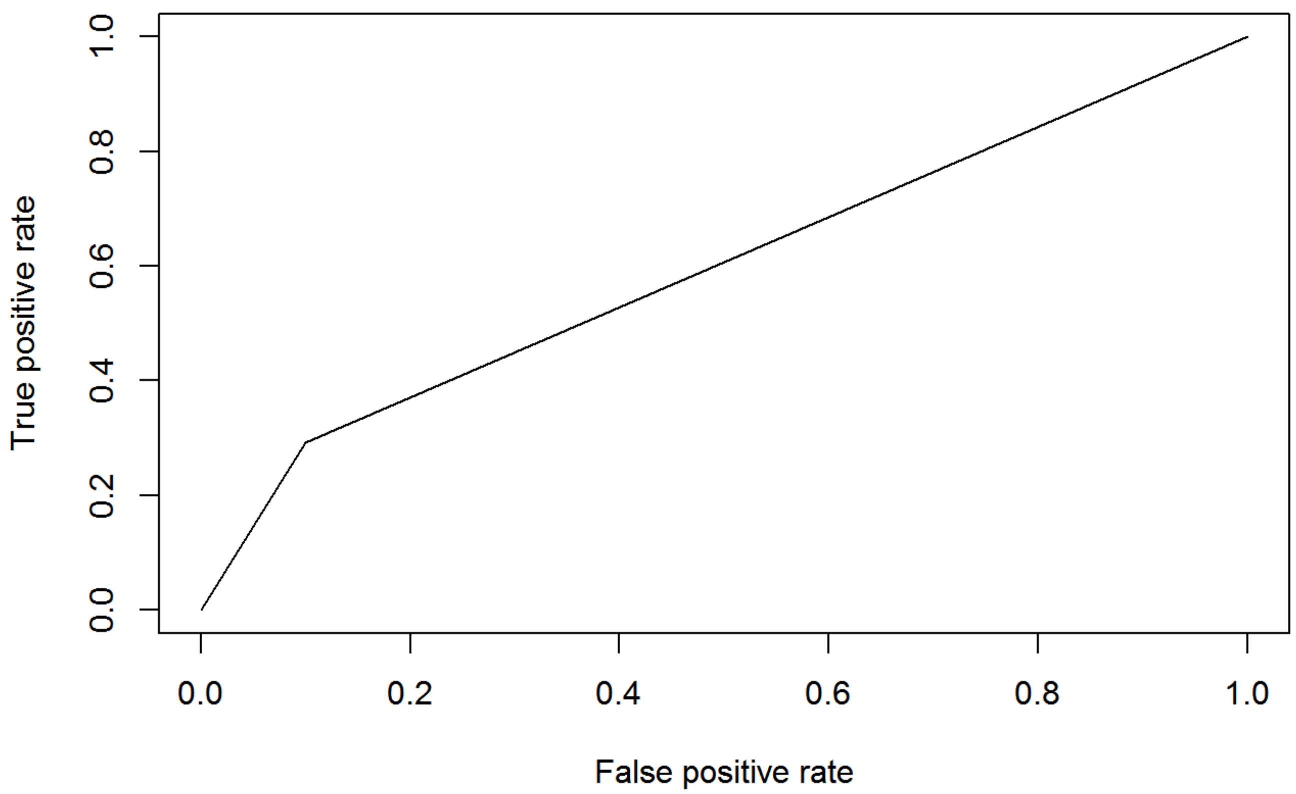

True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

```
## [1] 0.5522727
```

Conclusion: The analysis of various audio features for a set of songs during the late 2000’s period which had appeared in the Billboard chart shows that most succesful songs have higher danceability level of 0.6 and more and tempo of 90 with an adequate energy level which often hits the top 10 position in the Billboard chart. The successfulness of songs is prediction of the test data sets from songs against the trained datasets by the means of classification method i.e Supervised Learning by Decision tree, K Nearest Neighbours and Naive Bayes models. The accuracy of decision tree(0.68) is better than the other two

Loading [Contrib]/a11y/accessibility-menu.js