Spotify Dataset Analysis

“ Who needs therapy when you have Spotify ☺”

Group 15

Nihar Khillar

Parveen Mittal

Sahejbir Singh Kumar

Simran Sharma

TABLE OF CONTENTS

LINK TO POWERPOINT PRESENTATION RECORDING

https://cometmail-my.sharepoint.com/:v:/g/personal/pxk220018_utdallas_edu/EcxPXbxPriJIiBuVv2IGMjkBi1x6_Ydwnw1pJOqSZHO1oQ?e=2cPAXq

EXECUTIVE SUMMARY

Music is something that is close to each one of us. Be it tough times, be it easy times, be it happy or be it sad times, there's music for each of those occasions. Still, we know so little about music. We all have our favorite songs, podcast or audio book but we fail to describe perfectly why do we like them. Through this project, we try to do so through the analysis of the Spotify data set.

We had set 3 clear objectives for this project:

- Whether music features impact the popularity of a track: The business relevance is music artists can focus on the key features in future projects to increase popularity
- Build a model to predict the popularity of a track: Business relevance: Music producers and Spotify can select music compositions with high probability of popularity
- Make clusters to club similar tracks together based on music features: Business relevance: Can be used to suggest songs to users based on their past listening record

We followed the below process for carrying out the project:

1. Data collection: secondhand data set was collected from Kaggle
2. Data cleaning and preprocessing: In this step, we focused on 3 objectives:
    a. Finding the appropriate subset of data from the large dataset since the original dataset had over 580,000 rows of data. We did so by selecting the data of songs released between 16th April 2020 to 16th April 2021
    b. Getting rid of the duplicate data and rows with missing values (NA or blank spaces)
    c. Convert the numerical popularity score (0-100) to a categorical variable popularity category which will be the dependent variable going forward
3. Exploratory data analysis: In this step, we focused on using graphical tools and statistical summaries available in R to get a better sense of the data as well as identifying key patterns and features in the data set
4. Data Modeling: We focused on three tasks in this step:
    a. Through decision tree and logistic regression models, understand the key music features which explain most of the track's popularity
    b. Build multiple models which can predict the track's popularity based on the relevant input features and test these models with the validation dataset
    c. Through clustering, create clusters of similar tracks (K means) which can act as a recommendation to listeners based on their music preferences
5. Sharing our insights and key takeaways

PROJECT MOTIVATION/ BACKGROUND

- There has always been a close connection between music and society. The music both produces and reflects societal conditions, particularly those that either support or resist social change. The way that the majority of people receive music has changed dramatically since the introduction of recording techniques in the second part of the 20th century. The majority of people have instant access to all genres of music, day and night.
- Music can help us sleep, get pumped up for school and work, calm us down after a bad day, keep our spirits up when we're feeling low, and encourage socialization between people.
- With easy access to smart applications and the creation of music according to genres liked by the people, the music industry is dominating the various industries by providing society with a source of entertainment and stress-free days.
- When talking about online music streaming, one cannot ignore the contributions of Spotify to the music industry. It has attracted millions of subscribers to try and pay for the music they listen to, reducing piracy substantially.
- A huge library of tracks, a great experience across all connected devices, playlist creation tools, and befitting personalised recommendations make Spotify stand out and give it a monthly active user base of 381 million. [1]
- With availability in more than 170 countries[2] at the moment, the Spotify business model has the upper hand over many others because of the significant number of features provided and the delivery of music with no delay.
- The humongous use of the Spotify application led us to choose this domain for our analysis and derive some interesting insights by building the prediction model.

PROJECT FLOW

The R codes have been presented along with key outputs and our observations for each process of the data analysis process

DATA DESCRIPTION

The second hand dataset was sourced from Kaggle. Link is given below. (https://www.kaggle.com/datasets/lehaknarnauli/spotify-datasets?select=tracks.csv)

The following libraries were used in the analysis in R: moments, ggplot2, GGally, rpart, rpart.plot, caret, dplyr, funModeling, randomForest, adabag

```
spotify.df <- read.csv("Spotify.csv")

View(spotify.df)
```

[1] https://www.feedough.com/how-does-spotify-make-money/
[2] https://www.feedough.com/how-does-spotify-make-money/

#Understanding the dataset

summary(spotify.df)

str(spotify.df)

sapply(colnames(spotify.df), function(x) class(spotify.df[[x]]))

lapply(spotify.df, unique)

View(spotify.df)

dim(spotify.df)

```
      id                name             popularity      duration_ms        explicit
 Length:586672      Length:586672      Min.   :  0.00   Min.   :   3344   Min.   :0.00000
 Class :character   Class :character   1st Qu.: 13.00   1st Qu.: 175093   1st Qu.:0.00000
 Mode  :character   Mode  :character   Median : 27.00   Median : 214893   Median :0.00000
                                       Mean   : 27.57   Mean   : 230051   Mean   :0.04409
                                       3rd Qu.: 41.00   3rd Qu.: 263867   3rd Qu.:0.00000
                                       Max.   :100.00   Max.   :5621218   Max.   :1.00000
   artists           id_artists        release_date        danceability        energy
 Length:586672      Length:586672      Length:586672      Min.   :0.0000   Min.   :0.000
 Class :character   Class :character   Class :character   1st Qu.:0.4530   1st Qu.:0.343
 Mode  :character   Mode  :character   Mode  :character   Median :0.5770   Median :0.549
                                                          Mean   :0.5636   Mean   :0.542
                                                          3rd Qu.:0.6860   3rd Qu.:0.748
                                                          Max.   :0.9910   Max.   :1.000
      key            loudness            mode          speechiness      acousticness
 Min.   : 0.000   Min.   :-60.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 2.000   1st Qu.:-12.891   1st Qu.:0.0000   1st Qu.:0.0340   1st Qu.:0.0969
 Median : 5.000   Median : -9.243   Median :1.0000   Median :0.0443   Median :0.4220
 Mean   : 5.222   Mean   :-10.206   Mean   :0.6588   Mean   :0.1049   Mean   :0.4499
 3rd Qu.: 8.000   3rd Qu.: -6.482   3rd Qu.:1.0000   3rd Qu.:0.0763   3rd Qu.:0.7850
 Max.   :11.000   Max.   :  5.376   Max.   :1.0000   Max.   :0.9710   Max.   :0.9960
 instrumentalness      liveness          valence            tempo        time_signature
 Min.   :0.0000000   Min.   :0.0000   Min.   :0.0000   Min.   :  0.0   Min.   :0.000
 1st Qu.:0.0000000   1st Qu.:0.0983   1st Qu.:0.3460   1st Qu.: 95.6   1st Qu.:4.000
 Median :0.0000245   Median :0.1390   Median :0.5640   Median :117.4   Median :4.000
 Mean   :0.1134508   Mean   :0.2139   Mean   :0.5523   Mean   :118.5   Mean   :3.873
 3rd Qu.:0.0095500   3rd Qu.:0.2780   3rd Qu.:0.7690   3rd Qu.:136.3   3rd Qu.:4.000
 Max.   :1.0000000   Max.   :1.0000   Max.   :1.0000   Max.   :246.4   Max.   :5.000
```

```
              id             name       popularity      duration_ms         explicit
     "character"      "character"        "integer"        "integer"        "integer"
         artists       id_artists     release_date     danceability           energy
     "character"      "character"      "character"        "numeric"        "numeric"
             key         loudness             mode      speechiness     acousticness
       "integer"        "numeric"        "integer"        "numeric"        "numeric"
instrumentalness         liveness          valence            tempo   time_signature
       "numeric"        "numeric"        "numeric"        "numeric"        "integer"
```

There were 586672 rows and 20 columns in the data set. There were 3 major categorical variables which were of interest in the analysis: mode, key and explicit while other numerical features of importance were duration_ms, danceability, energy, loudness, mode, speechiness, acousticness,instrumentalness, liveness, valence, tempo and time_signature. The brief description of these features are as under

- Duration_ms: Duration of track in milliseconds
- Explicit: Whether the track has explicit lyrics (1: Y, 0: N)
- Danceability: Track suitability for dancing
- Energy: Perceptual measure of intensity and activity
- Key: Overall key of the track
- Loudness: Overall loudness in decibels (dB)
- Mode: Depicts the modality (major 1 or minor 0) of a track
- Speechiness: Measure of presence of spoken words in a track
- Acousticness: Whether the track is acoustic.
- Instrumentalness: Measures whether a track contains no vocals
- Liveness: Detects the presence of live audience in the recording
- Valence: Describes the musical positiveness conveyed by a track
- Tempo: overall estimated tempo of a track in beats per minute
- Time_signature: stimated overall time signature of a track
- Popularity: Measure the popularity from 0 to 100 based on number of times the tracks have been played

DATA CLEANING AND PRE PROCESSING

The latest release date of the track in the dataset was 16th April 2021. We decided to include a year data i.e. between 16th April 2020 and 16th April 2021 for this analysis to avoid running into system crashes in the later steps. The dataset was reduced to 17328 rows now. Initial dataset had 586672 rows as mentioned earlier.

```
spotify2.df<-spotify.df[spotify.df$release_date >= "2020-04-16" & spotify.df$release_date <= "2021-04-16", ]

#Understanding the trimmed dataset

summary(spotify2.df)

str(spotify2.df)

lapply(spotify2.df, unique)

head(spotify2.df)

glimpse(spotify2.df)

colnames(spotify2.df)

dim(spotify2.df)
```

DATA CLEANING

#Checking for NA values in the data set

colSums(is.na(spotify2.df))

#Checking for blank spaces in the numeric/ integer columns

nrow(spotify2.df[spotify2.df$popularity=="",])

nrow(spotify2.df[spotify2.df$duration_ms=="",])

nrow(spotify2.df[spotify2.df$explicit=="",])

nrow(spotify2.df[spotify2.df$danceability=="",])

nrow(spotify2.df[spotify2.df$energy=="",])

nrow(spotify2.df[spotify2.df$key=="",])

nrow(spotify2.df[spotify2.df$mode=="",])

nrow(spotify2.df[spotify2.df$speechiness=="",])

nrow(spotify2.df[spotify2.df$acousticness=="",])

nrow(spotify2.df[spotify2.df$instrumentalness=="",])

nrow(spotify2.df[spotify2.df$liveness=="",])

nrow(spotify2.df[spotify2.df$valence=="",])

nrow(spotify2.df[spotify2.df$tempo=="",])

nrow(spotify2.df[spotify2.df$time_signature=="",])

#Checking for duplicated values

sum(duplicated(spotify2.df))

#Counting unique songs in the data set

length(unique(spotify2.df$id))

```
> colSums(is.na(spotify2.df)) # No NA values
              id             name       popularity      duration_ms         explicit
               0                0                0                0                0
         artists       id_artists     release_date     danceability           energy
               0                0                0                0                0
             key         loudness             mode      speechiness     acousticness
               0                0                0                0                0
instrumentalness         liveness          valence            tempo   time_signature
               0                0                0                0                0
  popularity_cat
               0
```

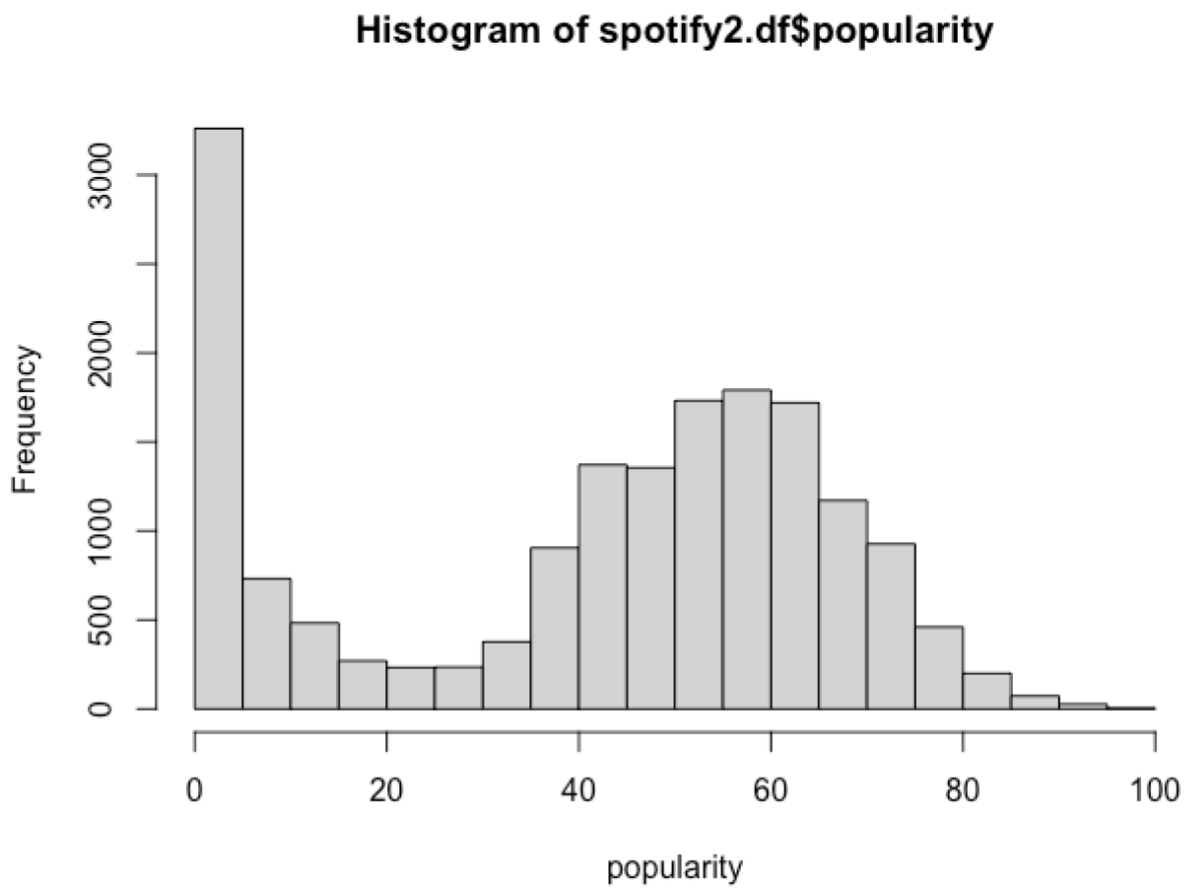
```
> nrow(spotify2.df[spotify2.df$popularity=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$duration_ms=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$explicit=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$danceability=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$energy=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$key=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$mode=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$speechiness=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$acousticness=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$instrumentalness=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$liveness=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$valence=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$tempo=="",])
[1] 0
> nrow(spotify2.df[spotify2.df$time_signature=="",])
[1] 0
```

```
> sum(duplicated(spotify2.df))
[1] 0
> length(unique(spotify2.df$id))   #counting unique songs
[1] 17328
```

There were no NAs or blank spaces and no duplicate data in the dataset.

DATA PREPROCESSING

```
hist(spotify2.df$popularity, breaks=20, xlab = "popularity")
```



In the above histogram, we observed that there were more than 3000 songs in the data set with popularity score in the range of 0 to 5. We decided to remove the songs with 0 popularity score. We wanted to analyze the songs which had been listened to at least once. We replaced the 0 values with NA first and then removed the NA value rows. About 1940 rows were removed leaving us with the final dataset of 15388 rows.

```
spotify2.df["popularity"][spotify2.df["popularity"] == 0] <- NA

summary(spotify2.df$popularity)

spotify2.df <- spotify2.df[complete.cases(spotify2.df),]

dim(spotify2.df)
```

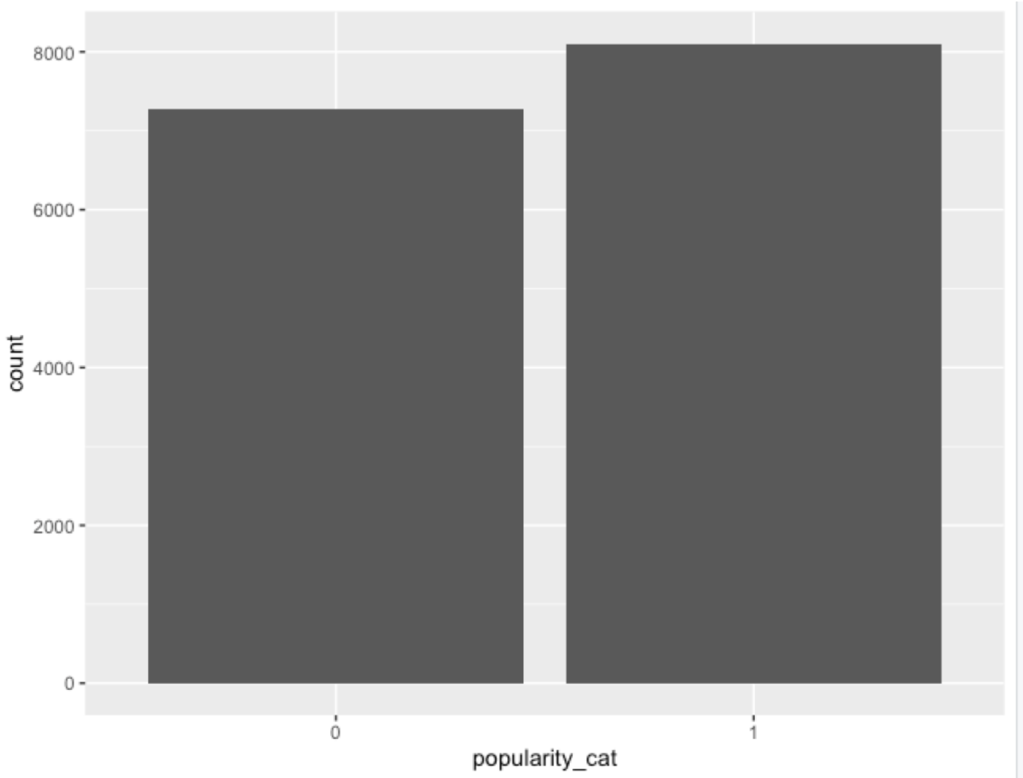
```
> summary(spotify2.df$popularity)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   1.00   37.00   52.00   46.54   63.00  100.00    1940 
> spotify2.df <- spotify2.df[complete.cases(spotify2.df),]
> dim(spotify2.df)
[1] 15388    20
```

Using mutate, we would add popularity_cat in the dataset based on the popularity score. We decided that a song would be a hit if it had a popularity score of over 50 and not hit if score is less than equal to 50.

spotify2.df<-spotify2.df %>% mutate(popularity_cat = case_when(popularity <= 50 ~ 0, popularity > 50 ~ 1))

ggplot(spotify2.df, aes(factor(popularity_cat)))+geom_bar()+xlab("popularity_cat")

table(spotify2.df$popularity_cat)




```
> table(spotify2.df$popularity_cat)

   0    1 
7282 8106 
```

There were 7282 not hit songs and 8106 hit songs in the dataset.

EXPLORATORY DATA ANALYSIS

numerical_cols = c("duration_ms","danceability","energy","loudness","speechiness","acousticness" ,"instrumentalness","liveness","valence","tempo","popularity")

spotify2_num.df<-spotify2.df %>% select(numerical_cols)

data.frame(mean=sapply(spotify2_num.df, mean,na.rm=TRUE),

median=sapply(spotify2_num.df, median,na.rm=TRUE),

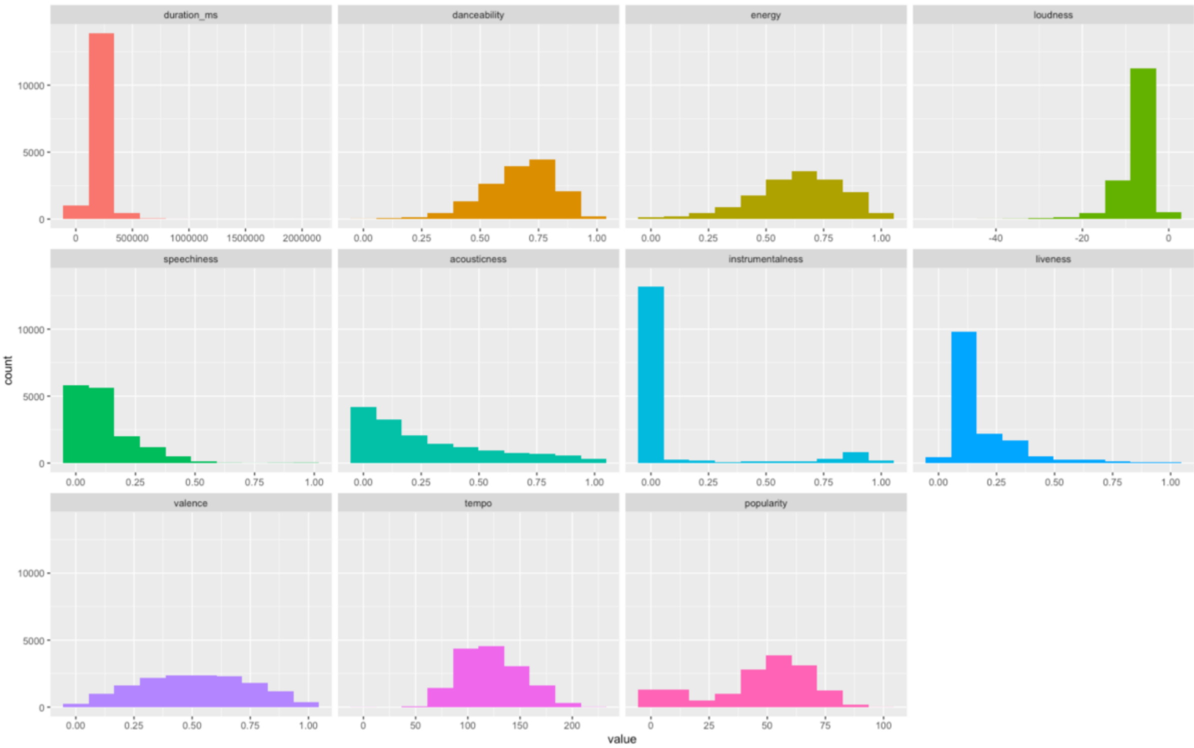
min=sapply(spotify2_num.df, min,na.rm=TRUE),

max=sapply(spotify2_num.df, max,na.rm=TRUE),

standardev=sapply(spotify2_num.df,sd,na.rm=TRUE))

	mean	median	min	max	standardev
duration_ms	197493.77410970	192650.00000000	23534.0	2059336.000	69572.7428111
danceability	0.66939446	0.68800000	0.0	0.986	0.1517994
energy	0.63415982	0.65100000	0.0	1.000	0.1960134
loudness	-7.49118339	-6.71600000	-51.8	1.509	3.8526665
speechiness	0.12267376	0.07020000	0.0	0.966	0.1195391
acousticness	0.27290187	0.17700000	0.0	0.996	0.2708028
instrumentalness	0.09731392	0.00000133	0.0	1.000	0.2605267
liveness	0.18146670	0.12200000	0.0	0.992	0.1476501
valence	0.50822166	0.51000000	0.0	0.990	0.2341557
tempo	122.04801118	122.00050000	0.0	220.470	29.1366760
popularity	46.53782168	52.00000000	1.0	100.000	22.2820925

Duration_ms had a very high standard deviation and high range (Max values were for yearbook and new years mix which tend to club multiple tracks together which increased the length of the tracks)



#Skewness check

skewness(spotify2_num.df$duration_ms,na.rm=TRUE)

skewness(spotify2_num.df$danceability,na.rm=TRUE)

skewness(spotify2_num.df$energy,na.rm=TRUE)

skewness(spotify2_num.df$loudness,na.rm=TRUE)

skewness(spotify2_num.df$speechiness,na.rm=TRUE)

skewness(spotify2_num.df$acousticness,na.rm=TRUE)

skewness(spotify2_num.df$instrumentalness,na.rm=TRUE)

skewness(spotify2_num.df$liveness,na.rm=TRUE)

skewness(spotify2_num.df$valence,na.rm=TRUE)

skewness(spotify2_num.df$tempo,na.rm=TRUE)

skewness(spotify2_num.df$popularity,na.rm=TRUE)

Based on histograms observation and from skewness metric, comments on the skewness of the data frame variables are given below:

- Almost symmetric: Valence, tempo
- Moderately skewed: Acousticness, popularity, energy, danceability
- Highly skewed: duration_ms, loudness, speechiness, liveness, instrumentalness

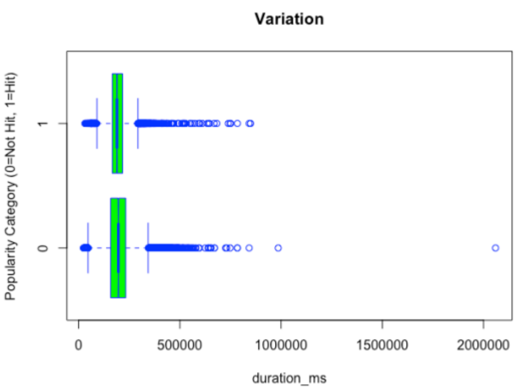
We checked the spread of data for popularity_cat across multiple features.

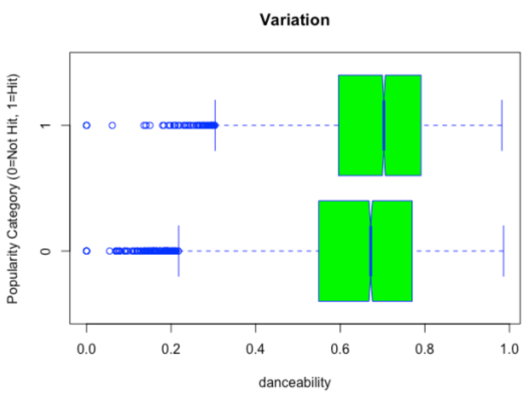
```
#boxplot

boxplot(duration_ms~popularity_cat, data = spotify2.df,

    main = "Variation",

    xlab = "duration_ms",

    ylab = "Popularity Category (0=Not Hit, 1=Hit)",

    col = "green",

    border = "blue",

    horizontal = TRUE,

    notch = TRUE

)

boxplot(danceability~popularity_cat, data = spotify2.df,

    main = "Variation",

    xlab = "danceability",

    ylab = "Popularity Category (0=Not Hit, 1=Hit)",

    col = "green",

    border = "blue",

    horizontal = TRUE,

    notch = TRUE

)

boxplot(energy~popularity_cat, data = spotify2.df,

    main = "Variation",

    xlab = "energy",

    ylab = "Popularity Category (0=Not Hit, 1=Hit)",

    col = "green",

    border = "blue",

    horizontal = TRUE,
```

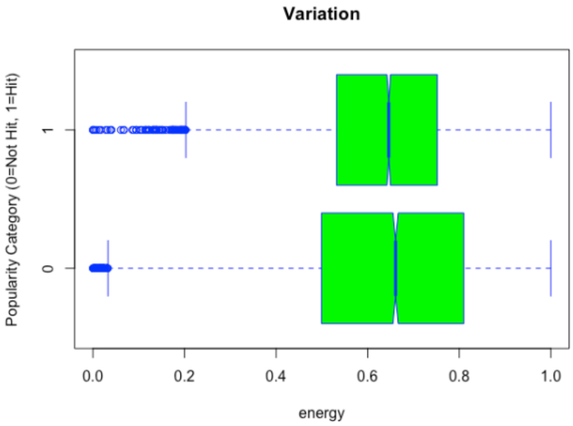
```
    notch = TRUE
)
boxplot(loudness~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "loudness",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
    notch = TRUE
)
boxplot(speechiness~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "speechiness",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
    notch = TRUE
)
boxplot(acousticness~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "acousticness",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
```

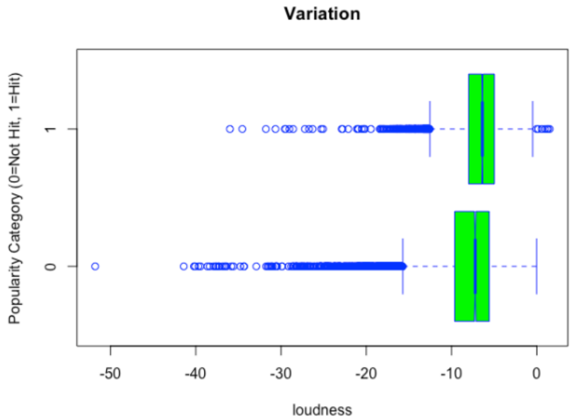
```
    notch = TRUE
)
boxplot(instrumentalness~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "instrumentalness",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
    notch = TRUE
)
boxplot(liveness~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "liveness",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
    notch = TRUE
)
boxplot(valence~popularity_cat, data = spotify2.df,
    main = "Variation",
    xlab = "valence",
    ylab = "Popularity Category (0=Not Hit, 1=Hit)",
    col = "green",
    border = "blue",
    horizontal = TRUE,
```

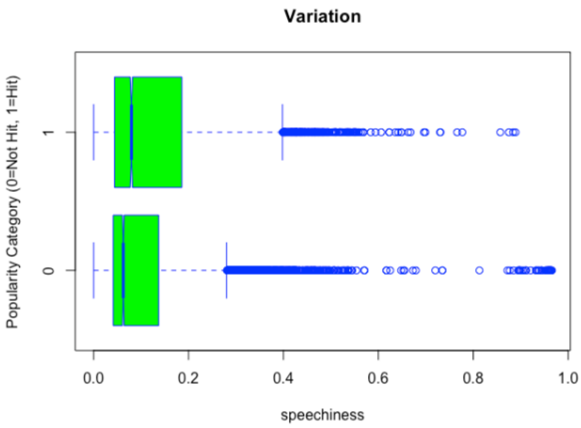
```
	notch = TRUE
)
boxplot(tempo~popularity_cat, data = spotify2.df,
	main = "Variation",
	xlab = "tempo",
	ylab = "Popularity Category (0=Not Hit, 1=Hit)",
	col = "green",
	border = "blue",
	horizontal = TRUE,
	notch = TRUE
)
boxplot(popularity~popularity_cat, data = spotify2.df,
	main = "Variation",
	xlab = "popularity",
	ylab = "Popularity Category (0=Not Hit, 1=Hit)",
	col = "green",
	border = "blue",
	horizontal = TRUE,
	notch = TRUE
)
```

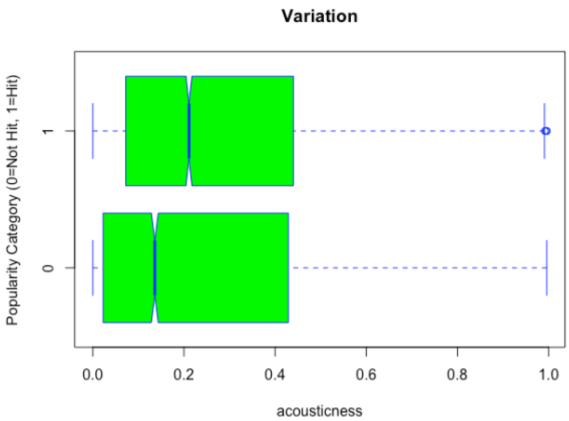


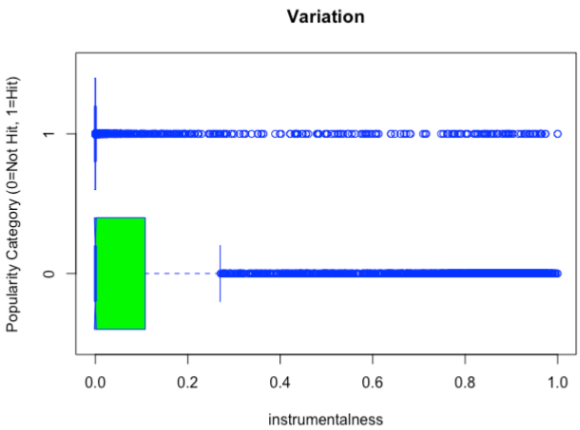


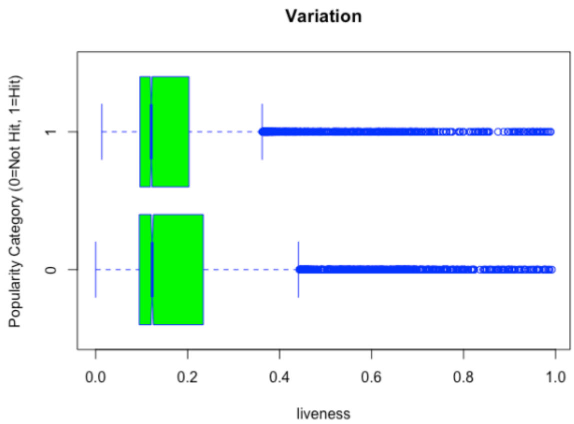

Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
0.0
0.2
0.4
0.6
0.8
1.0
energy

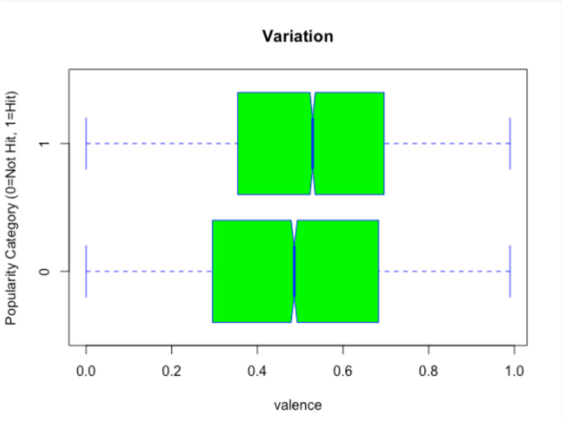

Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
-50
-40
-30
-20
-10
0
loudness

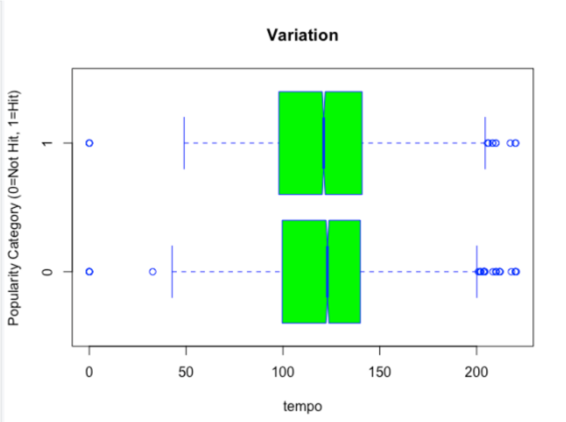

Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
0.0
0.2
0.4
0.6
0.8
1.0
speechiness

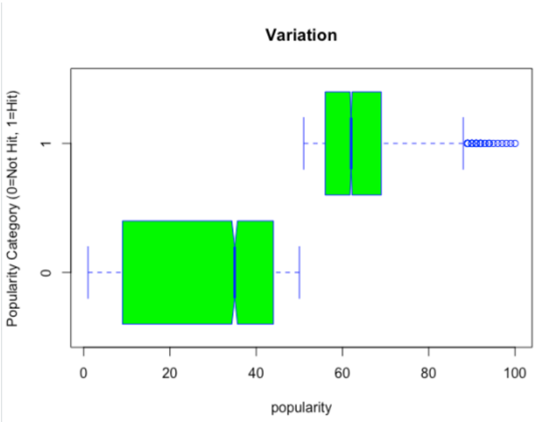

Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
0.0
0.2
0.4
0.6
0.8
1.0
acousticness


Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness

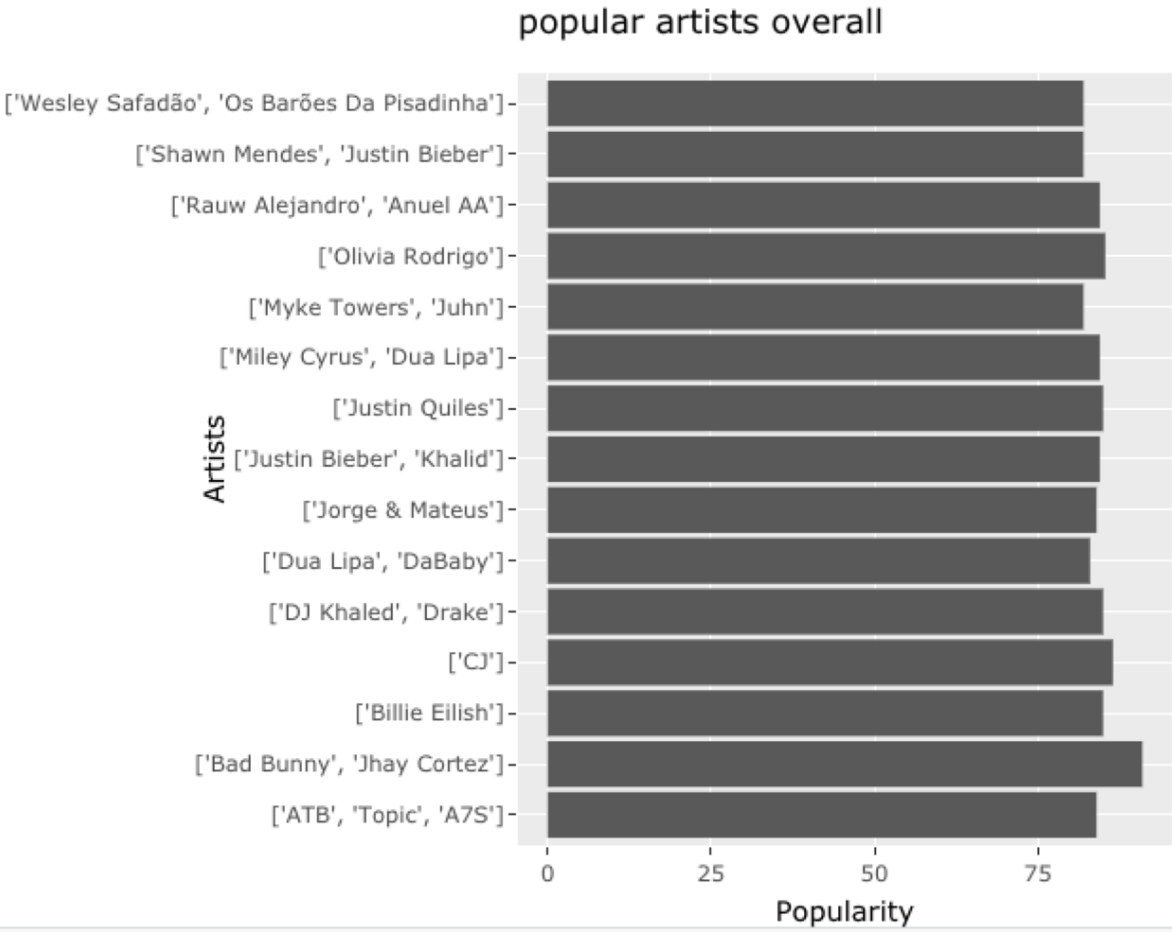

Variation
Popularity Category (0=Not Hit, 1=Hit)
1
0
0.0
0.2
0.4
0.6
0.8
1.0
liveness








```
#finding popular artists

popular_artists <- spotify2.df %>% group_by(Artist = artists) %>%

  summarise(No_of_tracks = n(),Popularity = mean(popularity))  %>%

  filter(No_of_tracks > 1) %>%

  arrange(desc(Popularity)) %>%

  top_n(15, wt = Popularity) %>%

  ggplot(aes(x = Artist, y = Popularity)) +

  geom_bar(stat = "identity") +

  coord_flip() + labs(title = "popular artists overall", x = "Artists", y = "Popularity")

ggplotly(popular_artists)
```




```
# top artists in based on popularity

top_10_artists <- spotify2.df %>%

  group_by(Artist = artists) %>%

  summarise(No_of_tracks = n(), Popularity = mean(popularity)) %>%

  filter(No_of_tracks > 1) %>%

  arrange(desc(Popularity)) %>%

  top_n(10, wt = Popularity)
```

	Artist	No_of_tracks	Popularity
	<chr>	<int>	<dbl>
1	['Bad Bunny', 'Jhay Cortez']	2	91
2	['CJ']	2	86.5
3	['Olivia Rodrigo']	3	85.3
4	['Billie Eilish']	2	85
5	['DJ Khaled', 'Drake']	2	85
6	['Justin Quiles']	2	85
7	['Justin Bieber', 'Khalid']	2	84.5
8	['Miley Cyrus', 'Dua Lipa']	2	84.5
9	['Rauw Alejandro', 'Anuel AA']	2	84.5
10	['ATB', 'Topic', 'A7S']	2	84
11	['Jorge & Mateus']	2	84

We have listed the top artists above with at least 2 songs in the time period so as to exclude the one hit wonders from this analysis.

DATA MODELING

classification_cols = c("duration_ms","danceability","energy","loudness","speechiness","acousticness"

,"instrumentalness","liveness","valence","tempo","mode","key","explicit","popularity_cat")

partition

spotify_class.df<-spotify2.df %>% select(classification_cols)

set.seed(1)

train.index <- sample(c(1:dim(spotify_class.df)[1]), dim(spotify_class.df)[1]*0.6)

train.df <- spotify_class.df[train.index,]

valid.df <- spotify_class.df[-train.index,]

We had split the dataset into training and validation data sets in 60:40 ratio

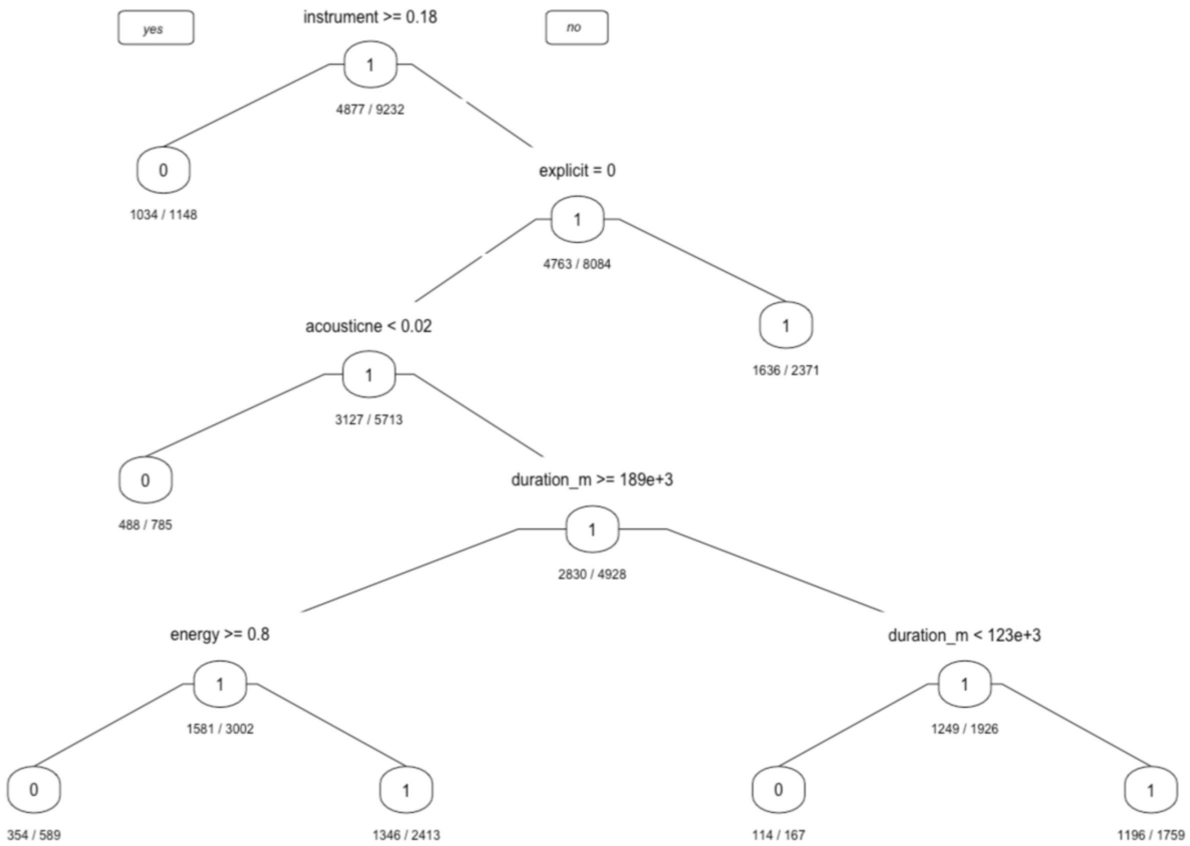
DECISION TREE MODELS

#decision tree

default.ct <- rpart(popularity_cat ~ ., data = train.df ,method = "class")

plot tree

```
prp(default.ct, type = 1, extra = 2, under = TRUE, split.font = 1, varlen = -10)

# count number of leaves

length(default.ct$frame$var[default.ct$frame$var == "<leaf>"])
```



There were 7 leaves in the default decision tree.

```
# classify records in the data.

default.ct.point.pred.train <- predict(default.ct,train.df,type = "class")

# generate confusion matrix for training data

confusionMatrix(default.ct.point.pred.train, as.factor(train.df$popularity_cat))

default.ct.point.pred.valid <- predict(default.ct,valid.df,type = "class")

confusionMatrix(default.ct.point.pred.valid, as.factor(valid.df$popularity_cat))
```

```
> confusionMatrix(default.ct.point.pred.train, as.factor(train.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1990  699
         1 2365 4178

               Accuracy : 0.6681
                 95% CI : (0.6584, 0.6777)
    No Information Rate : 0.5283
    P-Value [Acc > NIR] : < 0.00000000000000022
```

```
> confusionMatrix(default.ct.point.pred.valid, as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1343  472
         1 1584 2757

               Accuracy : 0.666
```

#deeper tree

deeper.ct <- rpart(popularity_cat ~ ., data = train.df, method = "class", cp = -1, minsplit = 1)

length(deeper.ct$frame$var[deeper.ct$frame$var == "<leaf>"])

There were 1684 leaves in the deeper tree which clearly indicate towards excessive overfitting of data.

repeat the code for the validation set, and the deeper tree

deeper.ct.point.pred.train <- predict(deeper.ct,train.df,type = "class")

confusionMatrix(deeper.ct.point.pred.train, as.factor(train.df$popularity_cat))

deeper.ct.point.pred.valid <- predict(deeper.ct,valid.df,type = "class")

confusionMatrix(deeper.ct.point.pred.valid, as.factor(valid.df$popularity_cat))

```
> confusionMatrix(deeper.ct.point.pred.train, as.factor(train.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4350   41
         1    5 4836

               Accuracy : 0.995
```

```
> confusionMatrix(deeper.ct.point.pred.valid, as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1846 1155
         1 1081 2074

               Accuracy : 0.6368
```

The accuracy was 0.995 for the deeper tree with the training dataset but crashed to 0.6368 with the validation dataset. That clearly suggested the excessive overfitting of data in the original model.

#Pruned tree

set.seed(1)

cv.ct <- rpart(popularity_cat ~ ., data = train.df, method = "class", cp = 0.00001, minsplit = 1, xval = 5)

printcp(cv.ct)

```
           CP nsplit rel error  xerror     xstd
1  0.21125144      0  1.000000 1.00000 0.011014
2  0.02192882      1  0.788749 0.79013 0.010668
3  0.01377727      3  0.744891 0.77015 0.010611
4  0.00711825      6  0.703559 0.72744 0.010475
5  0.00665901      7  0.696441 0.72239 0.010457
6  0.00482204      9  0.683123 0.71504 0.010431
7  0.00401837     10  0.678301 0.71183 0.010419
8  0.00367394     12  0.670264 0.71091 0.010416
9  0.00344432     13  0.666590 0.70402 0.010391
10 0.00275545     14  0.663146 0.70448 0.010393
11 0.00260237     15  0.660390 0.70471 0.010393
12 0.00252583     18  0.652583 0.70448 0.010393
13 0.00241102     19  0.650057 0.70448 0.010393
14 0.00229621     22  0.642250 0.70723 0.010403
15 0.00221967     26  0.633065 0.70677 0.010401
16 0.00206659     30  0.623651 0.70379 0.010390
17 0.00195178     32  0.619518 0.70333 0.010388
18 0.00183697     36  0.610333 0.70333 0.010388
19 0.00172216     37  0.608496 0.70586 0.010398
20 0.00165327     41  0.600459 0.70195 0.010383
21 0.00160735     47  0.590356 0.70333 0.010388
22 0.00153081     50  0.585534 0.70379 0.010390
23 0.00149254     56  0.576349 0.70379 0.010390
24 0.00137773     58  0.573364 0.69874 0.010371
25 0.00126292     63  0.566475 0.69759 0.010367
26 0.00122465     73  0.552928 0.69552 0.010359
27 0.00114811     77  0.547876 0.69552 0.010359
28 0.00107157     89  0.533869 0.69208 0.010346
29 0.00103330     93  0.528817 0.69208 0.010346
30 0.00101033     97  0.524684 0.69208 0.010346
31 0.00099502    106  0.513203 0.69254 0.010347
32 0.00096441    112  0.507233 0.69254 0.010347
```

From the above diagram, we found that when cp value was 0. 00107157, the xerror was minimum but rose after that. The desired cp for the best pruned tree had been found.

pruned.ct <- prune(cv.ct, cp = 0.00107157)

length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])

printcp(pruned.ct)

There were 90 leaves in the best pruned tree.

classify records in the training data.

pruned.ct.point.pred.train <- predict(pruned.ct,train.df,type = "class")

generate confusion matrix for training data

confusionMatrix(pruned.ct.point.pred.train, as.factor(train.df$popularity_cat))

repeat the code for the validation set, and the pruned tree

pruned.ct.point.pred.valid <- predict(pruned.ct,valid.df,type = "class")

confusionMatrix(pruned.ct.point.pred.valid, as.factor(valid.df$popularity_cat))

```
> confusionMatrix(pruned.ct.point.pred.train, as.factor(train.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2796  766
         1 1559 4111

               Accuracy : 0.7482
```

```
> confusionMatrix(pruned.ct.point.pred.valid, as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1670  746
         1 1257 2483

               Accuracy : 0.6746
```

random forest

rf <- randomForest(as.factor(popularity_cat) ~ ., data = train.df, ntree = 500,

mtry = 4, nodesize = 5, importance = TRUE)

variable importance plot

varImpPlot(rf, type = 1)

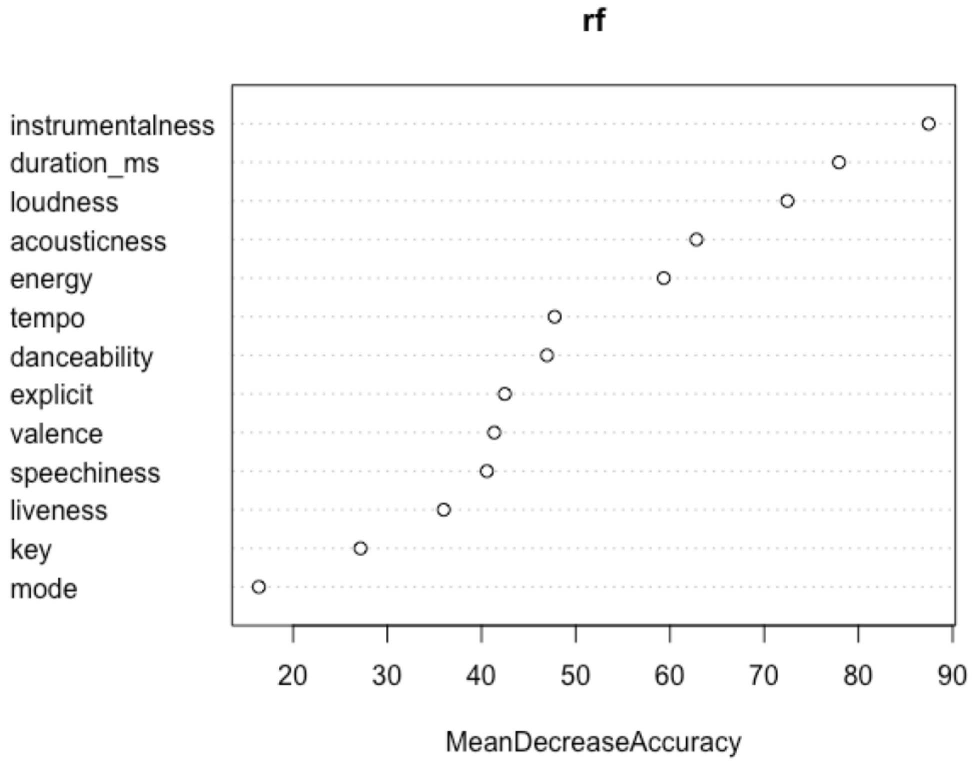
confusion matrix

rf.pred1 <- predict(rf, train.df)

confusionMatrix(rf.pred1, as.factor(train.df$popularity_cat))

rf.pred <- predict(rf, valid.df)

confusionMatrix(rf.pred, as.factor(valid.df$popularity_cat))




```
> confusionMatrix(rf.pred1, as.factor(train.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4328   38
         1   27 4839

               Accuracy : 0.993
```

```
> confusionMatrix(rf.pred, as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1691  469
         1 1236 2760

               Accuracy : 0.723
```

#Boosted tree

library(adabag)

train.df$popularity_cat <- as.factor(train.df$popularity_cat)

set.seed(1)

boost <- boosting(popularity_cat ~ ., data = train.df)

pred1 <- predict(boost, train.df)

confusionMatrix(as.factor(pred1$class), as.factor(train.df$popularity_cat))

pred <- predict(boost, valid.df)

confusionMatrix(as.factor(pred$class), as.factor(valid.df$popularity_cat))

```
> confusionMatrix(as.factor(pred$class), as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1560  560
         1 1367 2669

               Accuracy : 0.687
```

```
> confusionMatrix(as.factor(pred1$class), as.factor(train.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2431  785
         1 1924 4092

               Accuracy : 0.7066
```

LOGISTIC REGRESSION MODELS

regression.

logit.reg <- glm(popularity_cat ~ ., data = train.df, family = "binomial")

options(scipen=999)

summary(logit.reg)

formula(logit.reg)

logit.reg.pred <- predict(logit.reg, valid.df, type = "response") logit.reg.pred.classes <- ifelse(logit.reg.pred > 0.5, 1, 0)

confusionMatrix(as.factor(logit.reg.pred.classes), as.factor(valid.df$popularity_cat))

```
> formula(logit.reg)
popularity_cat ~ duration_ms + danceability + energy + loudness +
    speechiness + acousticness + instrumentalness + liveness +
    valence + tempo + mode + key + explicit
```

```
> confusionMatrix(as.factor(logit.reg.pred.classes), as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1376  508
         1 1551 2721

               Accuracy : 0.6655
```

model selection

full.logit.reg <- glm(popularity_cat ~ ., data = train.df, family = "binomial")

empty.logit.reg <- glm(popularity_cat ~ 1,data = train.df, family= "binomial")

summary(empty.logit.reg)

backwards = step(full.logit.reg)

summary(backwards)

backwards.reg.pred <- predict(backwards, valid.df, type = "response")

backwards.reg.pred.classes <- ifelse(backwards.reg.pred > 0.5, 1, 0)

confusionMatrix(as.factor(backwards.reg.pred.classes), as.factor(valid.df$popularity_cat))

```
> confusionMatrix(as.factor(backwards.reg.pred.classes), as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1367  506
         1 1560 2723

               Accuracy : 0.6644
```

```
> formula(backwards)
popularity_cat ~ duration_ms + danceability + energy + loudness +
    acousticness + instrumentalness + liveness + explicit
```

forwards =
step(empty.logit.reg,scope=list(lower=formula(empty.logit.reg),upper=formula(full.logit.reg)), direction="forward",trace=0)

formula(forwards)

forwards.reg.pred <- predict(forwards, valid.df, type = "response")

forwards.reg.pred.classes <- ifelse(forwards.reg.pred > 0.5, 1, 0)

confusionMatrix(as.factor(forwards.reg.pred.classes), as.factor(valid.df$popularity_cat))

```
> confusionMatrix(as.factor(forwards.reg.pred.classes), as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1367  506
         1 1560 2723

               Accuracy : 0.6644
```

```
> formula(forwards)
popularity_cat ~ instrumentalness + explicit + loudness + energy +
    acousticness + duration_ms + liveness + danceability
```

stepwise =
step(empty.logit.reg,scope=list(lower=formula(empty.logit.reg),upper=formula(full.logit.reg)), direction="both",trace=1)

formula(stepwise)

stepwise.reg.pred <- predict(stepwise, valid.df, type = "response")

stepwise.reg.pred.classes <- ifelse(stepwise.reg.pred > 0.5, 1, 0)

confusionMatrix(as.factor(stepwise.reg.pred.classes), as.factor(valid.df$popularity_cat))

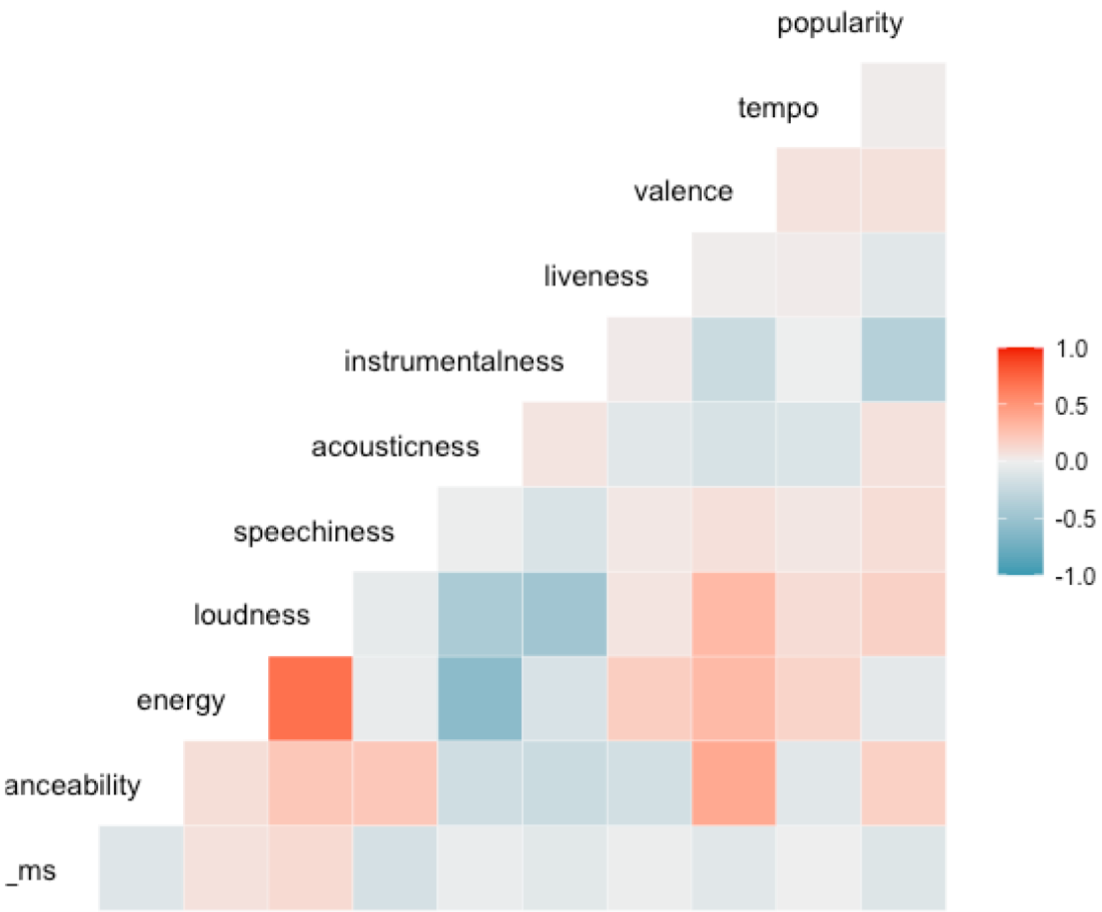
```
> confusionMatrix(as.factor(stepwise.reg.pred.classes), as.factor(valid.df$popularity_cat))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1367  506
         1 1560 2723

               Accuracy : 0.6644
```

```
> formula(stepwise)
popularity_cat ~ instrumentalness + explicit + loudness + energy +
    acousticness + duration_ms + liveness + danceability
```

#Checking for multi collinearity using correlation plots

forcorr_cols=c("duration_ms","danceability","energy","loudness","speechiness","acousticness"

```
,"instrumentalness","liveness","valence","tempo","popularity")

Spotify_Corr.df=spotify2.df %>% select(forcorr_cols)

Cor_spotify=data.frame(cor(Spotify_Corr.df, use = "complete.obs"))

Cor_spotify

plot(Spotify_Corr.df)

plot(Cor_spotify)

ggcorr(Spotify_Corr.df, hjust = 1)
```



From the correlation table, we found that the most highly positively correlated features were loudness and energy with correlation of 0.69 (It was intuitive as higher energy tracks would tend to be louder). Most highly negatively correlated features were acousticness and energy with a value of 0.6.

FINAL MODEL SELECTION

The accuracies obtained in all the models are given below:

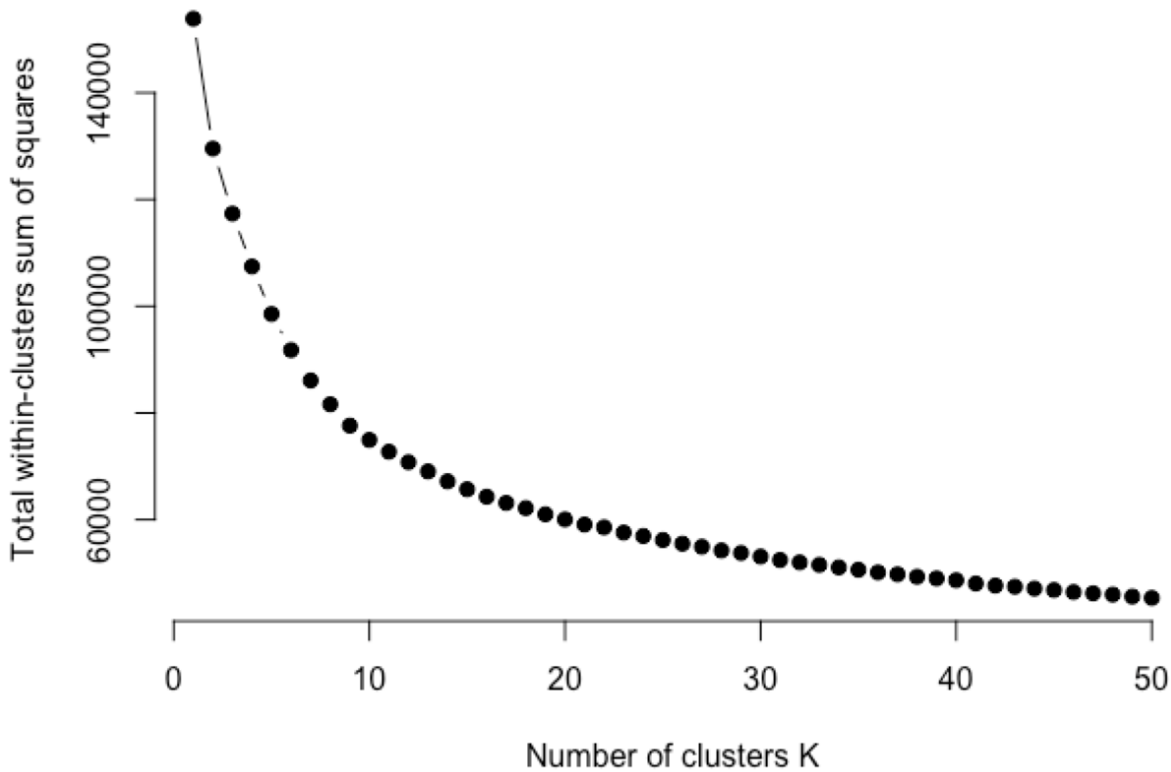
Class	Accuracy obtained on training dataset	Accuracy obtained on validation dataset (from Confusion matrix)
Default decision tree	0.6681	0.666
Deeper tree	0.995	0.6368
Best pruned decision tree	0.7482	0.6746
Random forest	0.993	0.7238
Boosted tree	0.7066	0.687
Default logistic regression	-	0.6655
Backward selection regression	-	0.6644
Forward selection regression	-	0.6644
Stepwise selection regression	-	0.6644

- **Random forest model offers the highest accuracy on the validation dataset and is selected as the model for prediction of track popularity**
- Interesting takeaway: The selection algorithms didn't improve the performance of the default regression model
- Key features which influence popularity of a track based on Decision tree, Randomforest and regression were:
  - instrumentalness, explicit, energy, acousticness, loudness
  - Duration_ms (This was surprising since we do not usually consider the length of a track while listening to music

CLUSTERING WITH K MEANS

```
cluster_cols=c("id","duration_ms","danceability","energy","loudness","speechiness","acousticness"
        ,"instrumentalness","liveness","valence","tempo")

spotify_cluster.df <- spotify2.df %>% select(cluster_cols)

row.names(spotify_cluster.df) <- spotify_cluster.df[,1]

spotify_cluster.df <- spotify_cluster.df[,-1]

View(spotify_cluster.df)

summary(spotify_cluster.df)

spotify_cluster.df.norm <- sapply(spotify_cluster.df, scale)

row.names(spotify_cluster.df.norm) <- row.names(spotify_cluster.df)
```

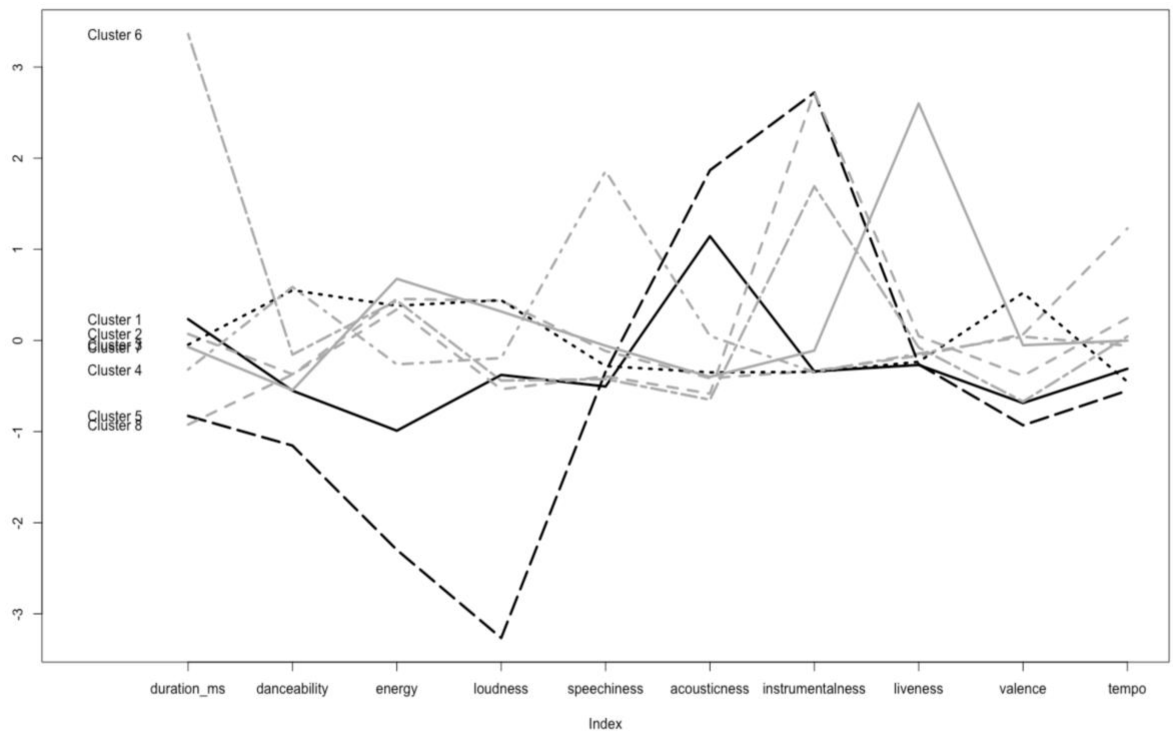
```
# PLotting elbow chart

k.max <- 50

data <- spotify_cluster.df.norm

wss <- sapply(1:k.max,

            function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})

wss

plot(1:k.max, wss,

     type="b", pch = 19, frame = FALSE,

     xlab="Number of clusters K",

     ylab="Total within-clusters sum of squares")
```



From the above elbow plot, we concluded that 8 clusters would be appropriate for performing the k means algorithm.

```
km <- kmeans(spotify_cluster.df.norm,8)

#Plots for g

# plot an empty scatter plot
```

```
plot(c(0), xaxt = 'n', ylab = "", type = "l",
    ylim = c(min(km$centers), max(km$centers)), xlim = c(0, 10))
# label x-axes
axis(1, at = c(1:10), labels = names(spotify_cluster.df))


# plot centroids
for (i in c(1:8))
  lines(km$centers[i,], lty = i, lwd = 3, col = ifelse(i %in% c(1, 3, 5),
                                                        "black", "dark grey"))
# name clusters
text(x =0.3, y = km$centers[, 1], labels = paste("Cluster", c(1:8)))
```



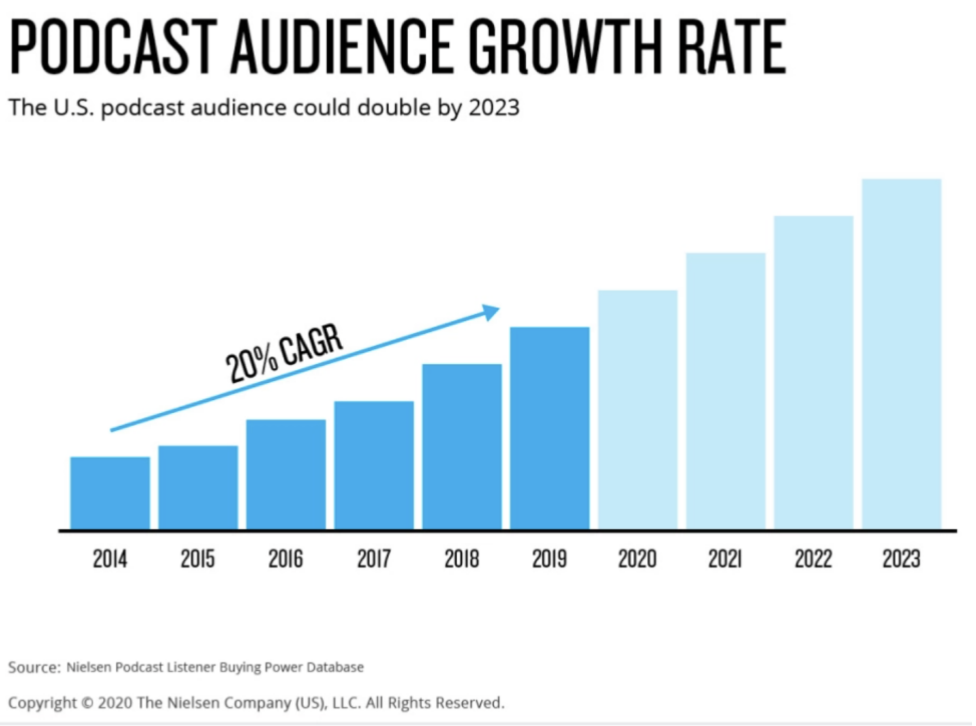
Cluster Name	Interpretation of characteristics with respect to other clusters
Cluster 1	High duration_ms and low instrumentalness (which means spoken words are more prominent compared to instruments. This cluster might contain audiobooks and podcasts.
Cluster 2	Low acousticness but very high instrumentalness. This cluster might contain rock songs.
Cluster 3	These stores sell the highest Original jeans with second highest sales in Fashion jeans and average sales in other jean types. The reasons for the extraordinarily high

	sales should be studied to find out if the learnings can help boost the sales in other clusters.
Cluster 4	Highest speechiness with high danceability. This cluster might have vocal focused tracks with less instruments but good danceability.
Cluster 5	Low duration_ms, lowest loudness and highest instrumentalness. This might contain theme music like Indiana Jones music
Cluster 6	Highest duration, high speechiness. This might contain Marathon tracks of artists and Yearbook mix
Cluster 7	Highest liveness. This might contain Live recordings like concert
Cluster 8	Lowest duration but high energy and high instrumentalness. This might contain rock music

- Valence and tempo have good variation across clusters

KEY TAKEAWAYS

- For popularity prediction, random forest had the highest accuracy on the validation dataset and hence the model can be used for future predictions of track popularity
- From the logistic regression model, we inferred that acoustics tracks with high loudness and danceability with explicit lyrics can become popular songs on Spotify
- Duration_ms also play a major role in the popularity. This would not have been the case a few years but a rise of popularity of audio books, talk shows and podcasts might have increased the significance of duration of the tracks in popularity as all these new listening areas have higher duration compared to regular music tracks. Nielson research shared below shows CAGR growth of about 20% for podcast listeners in the US.



Source: https://www.nielsen.com/insights/2020/podcast-content-is-growing-audio-engagement/