RockNRolla: Music Discovery and Perfomance Estimation App

Adarsh Goyal, Anand Deshmukh, Anurag Soni, Meena Kewlani, Yash Ambegaokar

Purdue University, Department of Management, 403 W. State Street, West Lafayette, IN 47907
goyal45@purdue.edu; deshmuk6@purdue.edu; soni16@purdue.edu; mkewlani@purdue.edu; yambegao@purdue.edu

Abstract

The RockNRolla team is building an app that helps artists/music producers to develop songs that would increase its chances of entering the "Top 200 charts" list across various regions as well as increase its following by users. Our App also uses a recommendation engine to help users discover songs based on their listening history as well as songs they haven't heard before but could like.

We aim to connect with streaming services to help them attract both parties (Artists/Music Producers and Users) to their platform and build a stronger network effect. We can use our app to persuade artists to partner with a streaming service, which in turn will prompt users to flock to it as well. This is our value proposition to the streaming service we collaborate with.

We used song attributes, user listening history, clustering techniques, etc. to suggest tracks to users as well as estimate how a song with a specific set of attributes would perform on the charts and the user response to it.

Keywords:

Music recommendation, Song recommendation, Song performance prediction, Spotify Kaggle Data, Fuzzy clustering, k-means clustering, Supervised Clustering

Business Problem

The music streaming industry is burgeoning with players like Spotify, Apple Music, Pandora, SoundCloud, Tidal or so many more. It's a highly competitive arena and streaming services need to discover newer ways to attract us both sides of the network – Users and Artists.

Our app aims to solve this problem in a two-step approach:

1. Facilitate artists/music producers to compose tracks that would help them enter and stay in the top music charts.
2. Build a recommendation engine that increases user discovery of songs based on listening history or moods/genre.

Who are we?

The RockNRolla company who has built this app that can solve the business problem mentioned above.

We are making this pitch to Spotify to convince them to share their music, charts and user data with us, so that we may build a more robust model.

The benefits for a streaming service to partner with RockNRolla are:

1. We help artists/music producers compose songs that would make it to the top world charts, get streamed often by users, stay longer on playlists, etc.
2. In lieu of our help, we can drive artists to the streaming service and based on the leverage we have, we can also persuade them to release their singles/albums exclusively on Spotify or the streaming service that partners with us.
3. This increased interest from artists will drive more users to the platform.

Additionally, by building a robust music discovery and recommendation engine, we want to increase the User's patronage to the streaming service as well.

This will create strong, positive network effect that benefit the streaming service that partners with us.

We plan to use quantified music metrics, user listening history, determination of user's interest in genres of music, etc. to build models that help us achieve our goals. Yes, this business problem is amenable to an analytics solution.

Analytics Problem

1. Classification: Classification of songs in our database into one of four genres:
    a. Workout
    b. Party
    c. Dinner
    d. Sleep
2. Develop an algorithm to compute the relative rating of a song, customized for a given user and user input.
3. Predict a list of songs a user would like based on the relative ratings for them.
4. Build a prediction model to forecast how a song with specific attributes would perform based on the following metrics for commercial success and user patronage:
    a. Number of times the song would be played
    b. The percentage length of the song that would be heard by users
    c. Number of days the song would be on a user's playlist
    d. Number of days in top 200 chart, per region
    e. Number of streams per region per day, for the lifetime of a song

Data

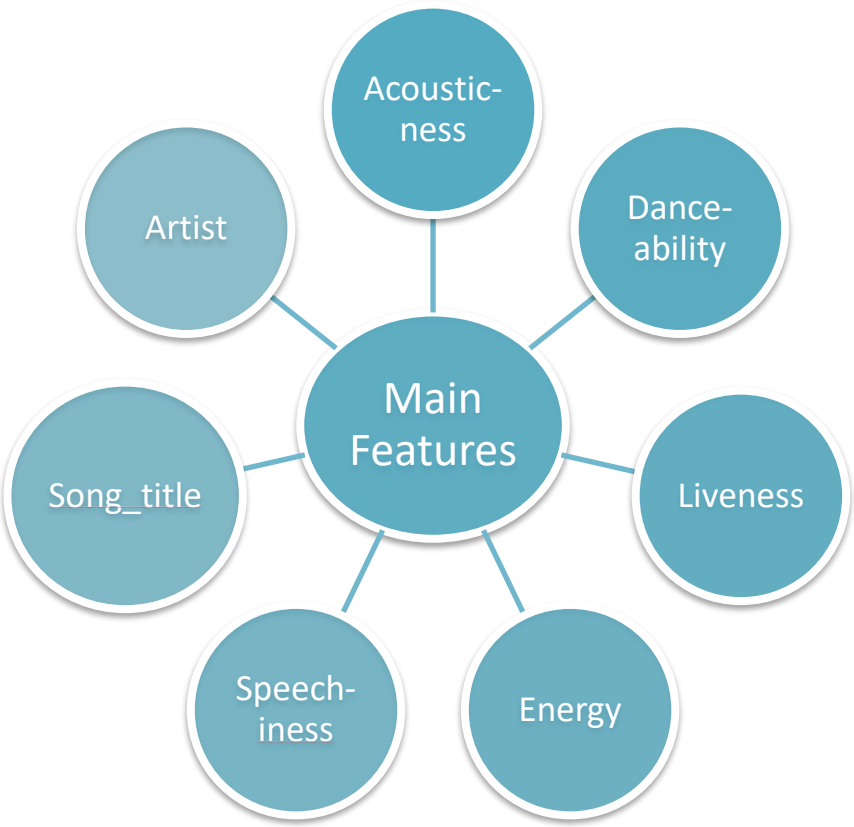
Sources of Data:

1. Dataset 1: Spotify Song attributes
   https://www.kaggle.com/geomack/spotifyclassification
   2,000+ songs

2. Dataset 2: Audio Features for Playlist Creation
   https://www.kaggle.com/aniruddhaachar/audio-features
   1,000+ songs

3. Dataset 3: Spotify's Worldwide Daily Song Ranking
   https://www.kaggle.com/edumucelli/spotifys-worldwide-daily-song-ranking
   Over a million entries

The attributes of songs that we were interested in dataset 1 (Spotify Song attributes) were:



The main difference between datasets 1 and 2 were that in datasets 2 the songs were also classified into the following four genres:

1. Workout
2. Party
3. Dinner
4. Sleep

Data Manipulation #1: Hence, we needed to classify the list of 2,000+ songs in dataset 1 as either of these 4 genres.

Once classified, we appended the dataset 1 and 2 together to get a list of 3,000+ songs, after removing the duplicates.

Data Manipulation #2: Inner join the "Spotify's Worldwide Daily Song Ranking" (dataset 3) dataset to this appended dataset of 3,000 songs

A very crucial element of our analysis required us to have User Listening Habits data, so that we could model our recommendation and music discovery algorithms to incorporate user preferences, likes/dislikes, songs heard/unheard, demographics, etc. However, this information was unavailable.

Since the importance of this information couldn't be disregarded, we engineered the information ourselves. This leads to the 3rd and probably most critical data manipulation act.

Data Manipulation #3:

- Need:
  - Lack of access to Spotify's user data
  - To develop recommendation engine based on user's past preferences and playlist history
- How: Random generation of 100 users' data to generate the following information:
  - Song heard/unheard
  - Number of times a song is played
  - Percentage length of the song the user usually listens to
  - Number of days on user's playlist
- Assumptions:
  - Song heard/unheard – Assumed 30% of songs are heard
  - Other metrics – Skewed normal distribution

We didn't have to do any scaling or cleaning of data.

Methodology Selection

Methodology to solve the Genre Classification problem:

- Two ways to do this:
  - Unsupervised Clustering –
    - K-means: divides songs into categories objectively but
    - Pros: No accuracy is involved
    - Cons:
      - Illegible to users (no understandable divides)
      - Data needs to be standardized (only numeric data)
      - We require to re-run the code (not scalable)
  - Supervised classification - Random forest for classification
    - Pros:
      - Understandable by users
      - Scalable
      - Any type of data can be used (categorical or numeric)
    - Cons: Might need larger data for accuracy
- Picked Supervised Classification – Random forest model – Nature of our problem

- Why R? – easily available and easy to use packages (lot of material and support online)

Methodology for building the enterprise level functionalities: Help facilitate composition of a song

- We clustered songs based on attributes using k-means clustering. 5 clusters created.
- Once user defines the attributes of a song, the song falls into amongst those cluster based on Fuzzy Clustering
- Fuzzy clustering gives a probability of a song falling in each pre-defined cluster.
- Each cluster has songs that fall in the "Top 200 chart" for various global regions. These songs are sorted according to the number of days they've been in the top 200 charts.
- We calculated the expected "length of stay in charts" for the new input attributes by summation (multiplying the average length of stay for all songs in that cluster * probability of new song lying in that cluster)
- We then check how the new song will perform over the lifetime by calculating the number of streams it generates after regular intervals
- We plot the graph (which is displayed on the app)

Model Building

For Classification Analytics Probem: Random forest (Package: *randomForest*):

- Dataset 2 (Audio Features for Playlist Creation) has the 'class' variable which is the target variable. Need to predict the value of class in Dataset 1 (Spotify Song attributes) based on attributes
- The appended dataset (Dataset 1 + 2) is divided into test and train datasets (70:30).
- Number of trees optimized for train dataset based on expected error from each iteration of model
- Model accuracy is checked by OOB error estimate (lesser the better) and the confusion matrix gives an idea of misclassification
- Variable importance based on Mean Gini Index tells us which are the important variables
- Cross-validate the results on the test dataset, check same metrics and accuracy
- Predict response variable on Dataset 1. Check accuracy of the model by checking If the overlapping song are rightly classified. In this way, the model is integrated back into the problem.
- **Assumption**: Each song in the data has only one class

To cluster songs based on attributes – Fuzzy Clustering (Package: fclust):

- We expect an intermingling effect of the various attributes and metrics that characterize a song.

Functionality: What the App offers

A. Assistance in developing a song:
   Use attributes of songs and list of top 200 songs per region - per day to predict:
   a. Number of times the song would be played
   b. The percentage length of the song that would be heard by users
   c. Number of days the song would be on a user's playlist
   d. Number of days in top 200 chart, per region
   e. Number of streams per region per day, for the lifetime of a song

B. Recommendation Engine for Users:
   a. User selection of filters:
      i. Genre
      ii. Heard/unheard
   b. Surprise me section: Unheard songs are suggested

We've used conditional logic in the following situations:

- Heard/Not heard
- Genre classification

Some R Packages we've used are:

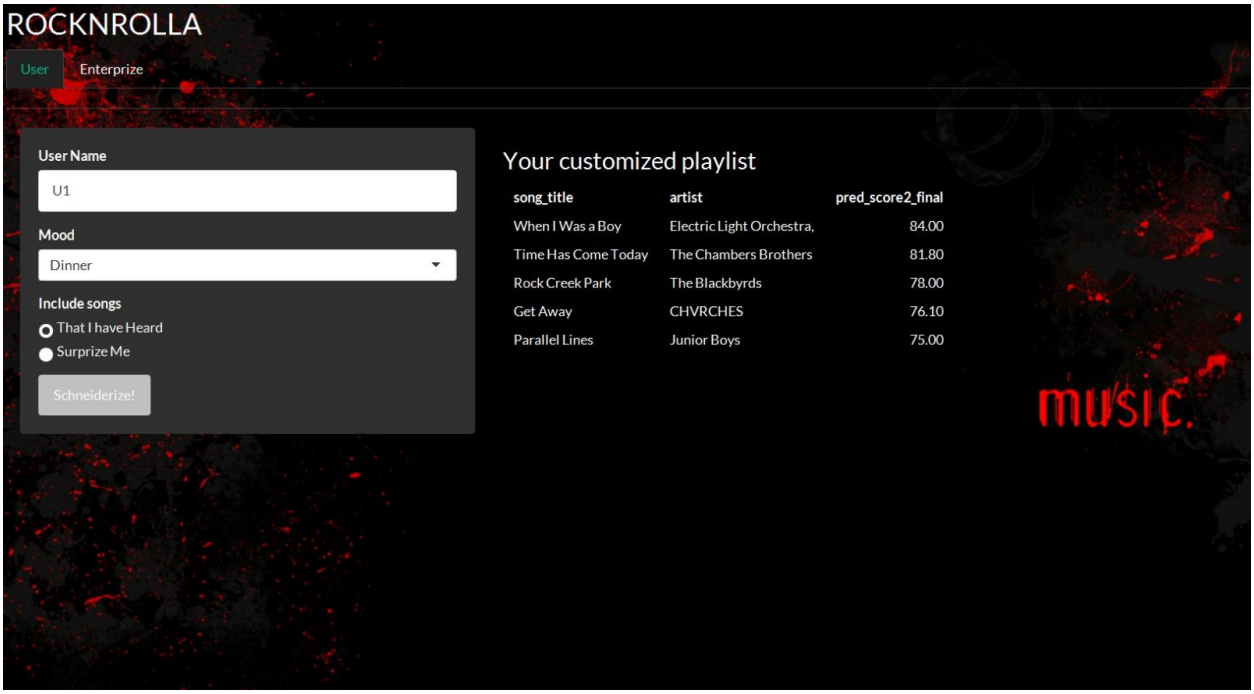
- library(shiny)
- library(shinysky)
- library(plotly)
- library(ggplot2)
- library(sqldf)
- library(devtools)
- library(hexbin)
- library(data.table)
- library(fclust)
- #install.packages("shinydashboard")
- #install.packages("shinythemes")
- library(shinydashboard)
- library(shinythemes)

Improvements to the App

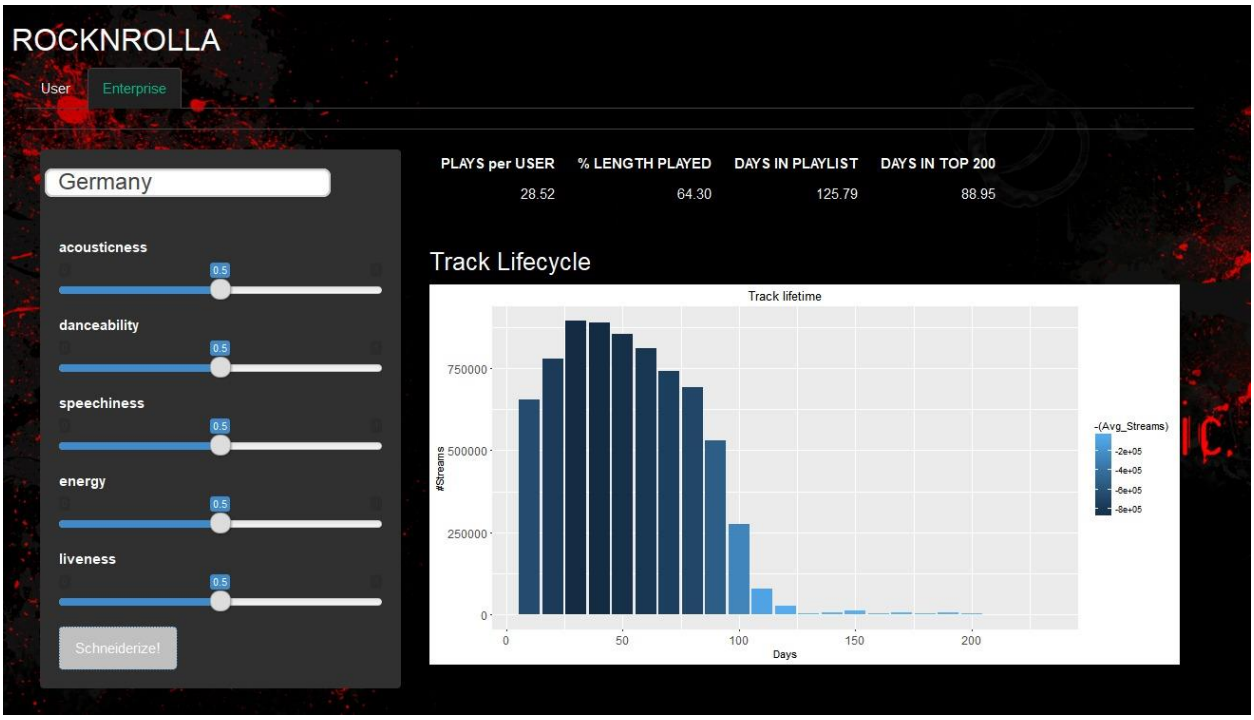
1. Incorporate actual User data – listening history, playlists, like/dislikes, demographics, etc
2. Expand song list
3. Create a logistic regression to predict whether a song will appear on a “Top 200 chart” for the various regions
4. Regression to identify which song attributes impact the acceptance and performance of a song
5. In the Shiny App:
    a. User login and password
    b. “Search by song” feature with autocomplete
    c. Ability to click on a recommended song to open a widget to play the song

GUI Design and Quality

User level interface



Enterprise level interface



Conclusions

Our app works well but there are limitations with our models and appearance. Given data, resources, experience and time we can improve it further. However, we see tremendous potential in how this app can help streaming services increase their networks – Artists and Listeners (Users). We want to partner with a streaming company to build a partnership that propels both entities to the top of the markets.