Capstone Project 2: Creating a Recommendation System for DJ Mixes using a Spotify User's preferences

Author: Jin No
Date: July 2019
Github: https://github.com/Jwn758/SpringboardProjects

Table of Contents:

1. Introduction

I love being recommended and recommending new songs & artists to friends. But in today's data-driven world, recommendation engines control the new content we see and hear. Whether it be a new product on Amazon, a new show on Netflix, or a new song in your discover weekly Spotify playlist, recommendation engines may know you better than anyone else.

Let's take a look at Spotify's Discover Weekly feature as an example of a recommendation algorithm. It combines various methods, including content-based filtering methods, which looks at the raw audio features of a song to recommend new songs that are similar to songs you've listened to and liked before. A closer exploration of Spotify's recommendation algorithm can be found here: https://medium.com/datadriveninvestor/behind-spotify-recommendation-engine-a9b5a27a935

Yet even with Spotify's Discover Weekly feature, there aren't any good recommendation systems for sets/mixes. When it comes to electronic music, I love recommending sets/mixes rather than individual songs because I believe they can more accurately convey the artist's style. For example, an artist might play unreleased edits, IDs (i.e. unidentifiable songs, typically consisting of an unreleased song by the artist or an artist with a similar style), and play a myriad of songs ranging from their entire discography rather their most recent album. The set/mix might also include some mixing techniques (i.e. transitioning from one song to another) or other live elements that you won't hear by streaming an individual song. Often times, it's a combination of these additional "elements" to a set/mix that I want to recommend…and today's recommendation engines simply don't have the means to capture this type of data and recommend a set/mix.

For my 2nd Capstone Project, I try to solve for this by utilizing content-based filtering and collaborative filtering methods to create a recommendation algorithm for sets/mixes. For this, I will need data on both a user's music preferences and relevant data for sets/mixes. The biggest pitfall today is the lack of available data on sets/mixes and the ability to stream the actual set/mix itself. To combat this, I use Spotify's Web API to pull a user's top artists and created an automated-web scraper that searches for a Spotify user's top artists on a 3rd party website that lists the artist's most recent sets/mixes and the individual songs played in each set/mix along with other relevant metadata about the set/mix itself.

- Problem: Creating a recommendation engine for sets/mixes using a Spotify user's preferences and data scraped from 3rd party resources
- Dataset: Data queried from Spotify's Web API, data scraped from 1001tracklist.com

1.1 Data Source and Importation

The data needed to build an effective recommendation system includes two kinds of inputs: data on **users** and data on **items**:

Users: Spotify Users

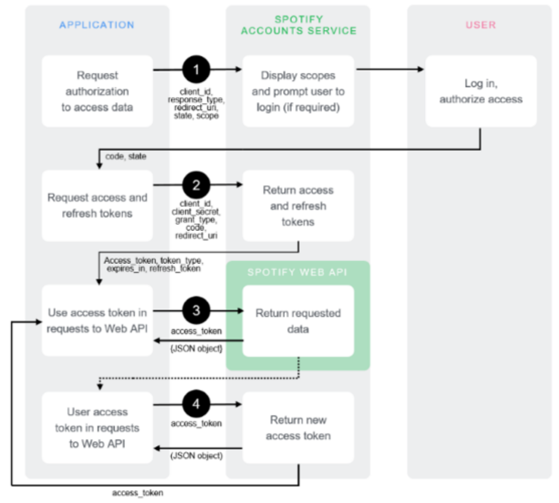
- **Dataset Description**: JSON Objects containing a Spotify User's top songs, top artists, and saved songs pulled directly from Spotify
- **Dataset Source**: Spotify's Web API

Items: Sets/Mixes

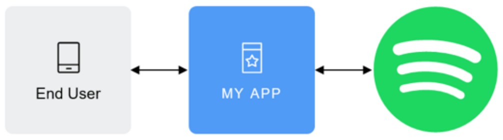
- **Dataset Description**: Dataframe containing up to 30 of the most recent sets/mixes played by the Spotify user's top electronic artists with relevant metrics such as total views, total likes, individual songs played, genres, artists of other songs played, location, etc.
- **Dataset Source**: Data scraped via a custom-built automated web-scraper utilizing Selenium and BeautifulSoup on a 3rd party website that incentivizes users to list and identify songs played in a set/mix

1.2 Data Cleaning (Spotify)

Overview of Spotify's Web API
To access a Spotify user's music preferences, one must create and register an app and request authorization from both Spotify and the user


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
(JSON object)
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token
Authorization Code Flow

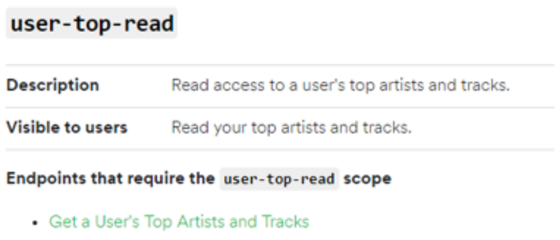
- Server: the Spotify server
- Client: your application
- Resource: the end user data and controls



The diagrams above illustrate how a API call was processed to access user data.

For the purposes of this report, the following should be noted:

- Different databases accessed via different scopes: Since we are accessing a Spotify User's top songs, top artists, and saved songs as inputs to our recommendation engine, the following scopes were specified in our API call:
  - User-library-read
  - User-top-read

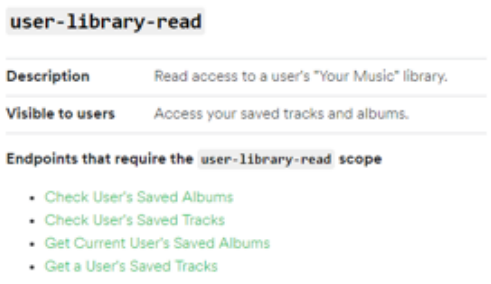


`user-top-read`

Description	Read access to a user's top artists and tracks.
Visible to users	Read your top artists and tracks.

Endpoints that require the `user-top-read` scope

- Get a User's Top Artists and Tracks



`user-library-read`

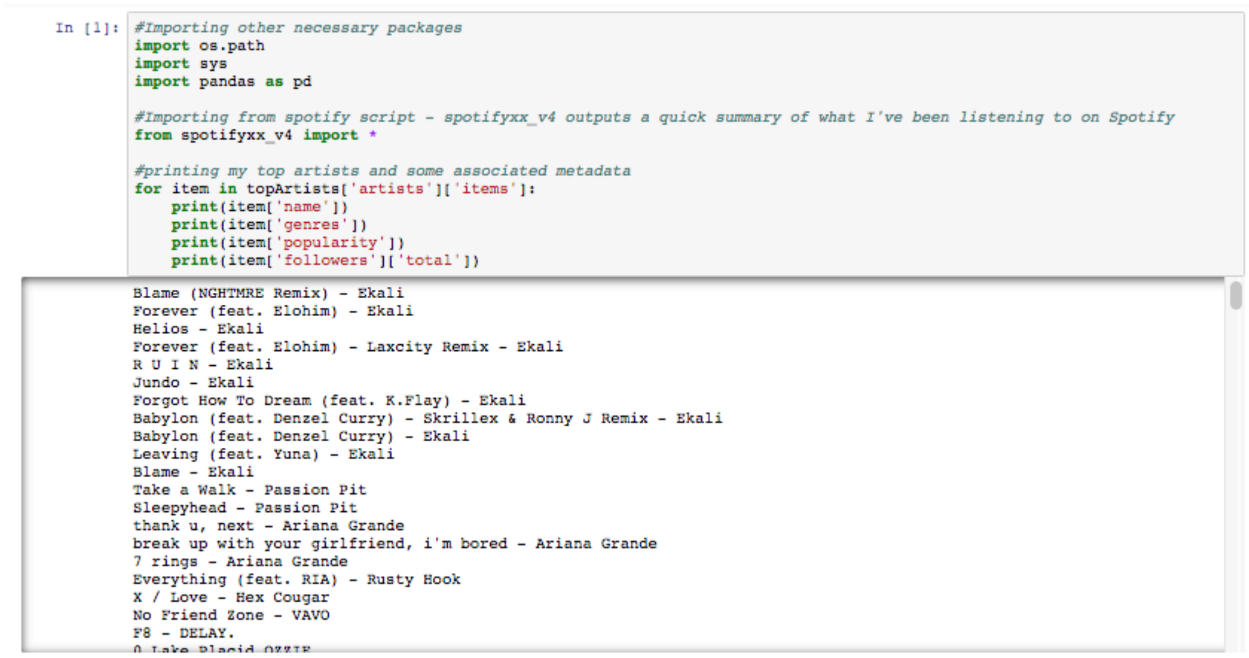
Description	Read access to a user's "Your Music" library.
Visible to users	Access your saved tracks and albums.

Endpoints that require the `user-library-read` scope

- Check User's Saved Albums
- Check User's Saved Tracks
- Get Current User's Saved Albums
- Get a User's Saved Tracks

Example API Request using my Spotify Account

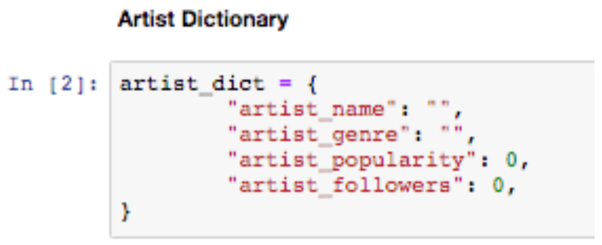
For this report, I used my personal Spotify data to illustrate how a Spotify user's data will be accessed and then stored into a dataframe to build out my recommendation engine:



```
In [1]: #Importing other necessary packages
        import os.path
        import sys
        import pandas as pd

        #Importing from spotify script - spotifyxx_v4 outputs a quick summary of what I've been listening to on Spotify
        from spotifyxx_v4 import *

        #printing my top artists and some associated metadata
        for item in topArtists['artists']['items']:
            print(item['name'])
            print(item['genres'])
            print(item['popularity'])
            print(item['followers']['total'])
```

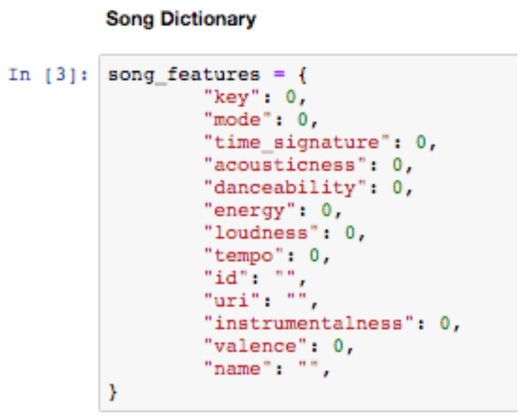
```
Blame (NGHTMRE Remix) - Ekali
Forever (feat. Elohim) - Ekali
Helios - Ekali
Forever (feat. Elohim) - Laxcity Remix - Ekali
R U I N - Ekali
Jundo - Ekali
Forgot How To Dream (feat. K.Flay) - Ekali
Babylon (feat. Denzel Curry) - Skrillex & Ronny J Remix - Ekali
Babylon (feat. Denzel Curry) - Ekali
Leaving (feat. Yuna) - Ekali
Blame - Ekali
Take a Walk - Passion Pit
Sleepyhead - Passion Pit
thank u, next - Ariana Grande
break up with your girlfriend, i'm bored - Ariana Grande
7 rings - Ariana Grande
Everything (feat. RIA) - Rusty Hook
X / Love - Hex Cougar
No Friend Zone - VAVO
F8 - DELAY.
0 Lake Placid OZZIE
```

Getting Unique List of Songs, Artists, and Audio Features of Songs
Creating dictionaries for Artists and Songs with the following keys:



Artist Dictionary

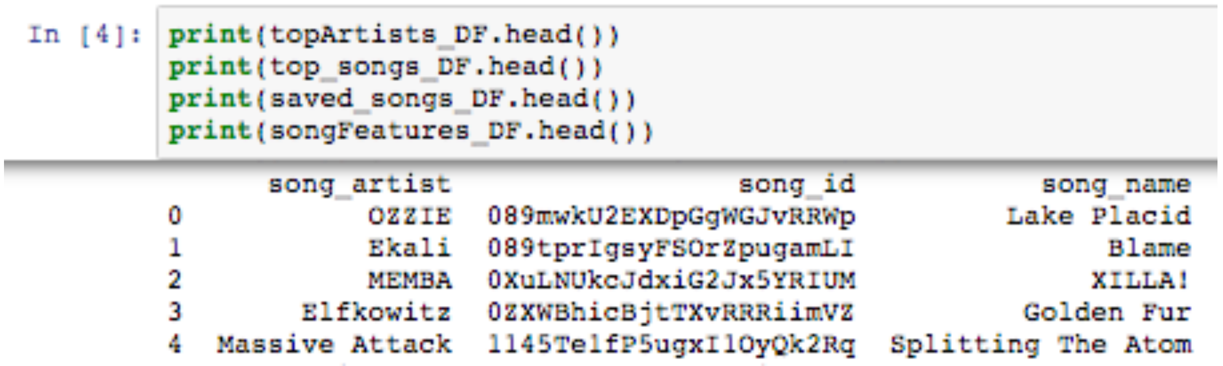
```
In [2]: artist_dict = {
                "artist_name": "",
                "artist_genre": "",
                "artist_popularity": 0,
                "artist_followers": 0,
        }
```



Song Dictionary

```
In [3]: song_features = {
                "key": 0,
                "mode": 0,
                "time_signature": 0,
                "acousticness": 0,
                "danceability": 0,
                "energy": 0,
                "loudness": 0,
                "tempo": 0,
                "id": "",
                "uri": "",
                "instrumentalness": 0,
                "valence": 0,
                "name": "",
        }
```

Converting personal Spotify data into dataframes:



```
In [4]: print(topArtists_DF.head())
        print(top_songs_DF.head())
        print(saved_songs_DF.head())
        print(songFeatures_DF.head())
```

```
     song_artist                 song_id         song_name
0          OZZIE  089mwkU2EXDpGgWGJvRRWp       Lake Placid
1          Ekali  089tprIgsyFSOrZpugamLI             Blame
2          MEMBA  0XuLNUkcJdxiG2Jx5YRIUM            XILLA!
3       Elfkowitz  0ZXWBhicBjtTXvRRRiimVZ        Golden Fur
4  Massive Attack  1145TelfP5ugxI1OyQk2Rq  Splitting The Atom
```

Additional data cleansing and manipulation include:

- Creating different flags for different genres (e.g. if artist was an electronic artist)
- Creating unique lists of songs and artists to use in my automated web scraper (using Selenium and BeautifulSoup)
- Flagging artists to use for 1001tracklists.com based on availability of data

For the purposes of this report, I looked at the following EDM artists that I listened to:

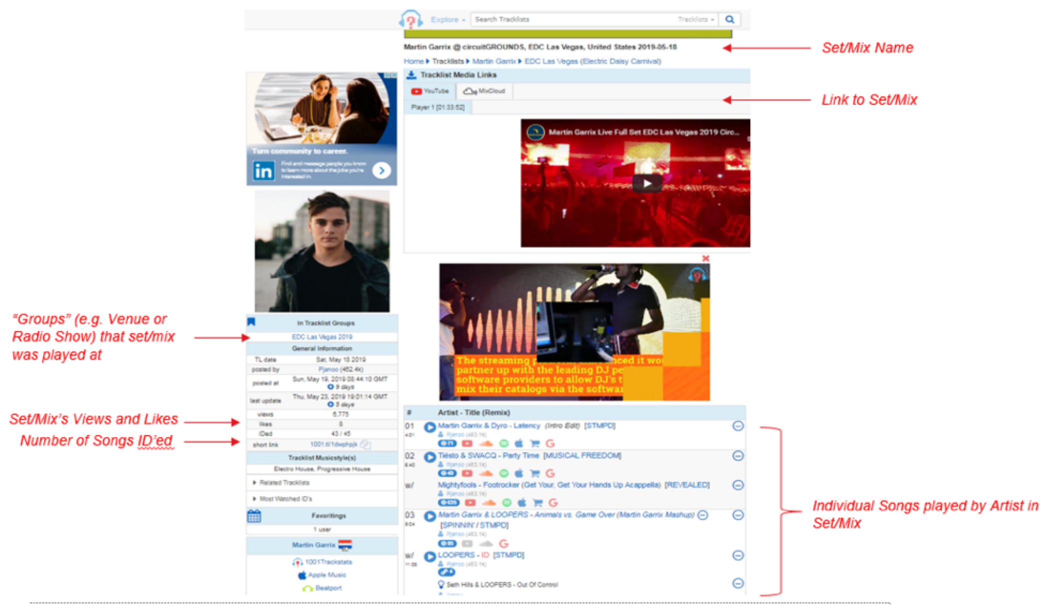
- Ekali
- San Holo
- JOYRYDE
- Flosstradamus

1.3 Data Cleaning (1001tracklists.com)

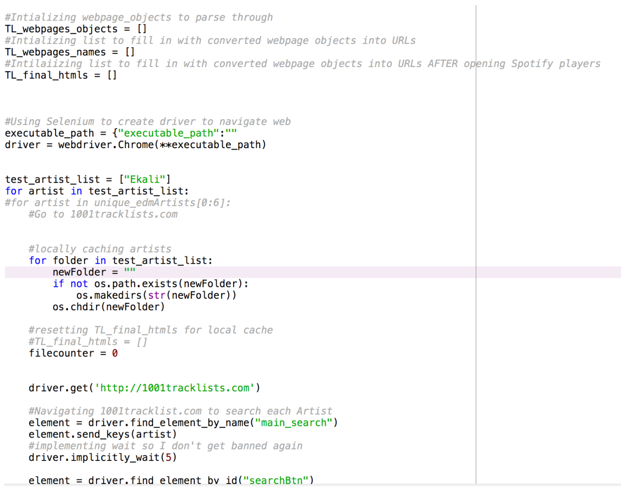
Overview of 1001tracklists.com

1001tracklist.com is a 3rd party resource where users provide a link to the actual set/mix (e.g. Youtube link, Soundcloud link, Mixcloud link, etc.) and list out the artists and name of the individual songs played in the set/mix. Other relevant metadata that was scraped include total webpage views, date played, total set/mix likes, genres, duration of each individual song, number of individual songs played and number of songs actually identified.

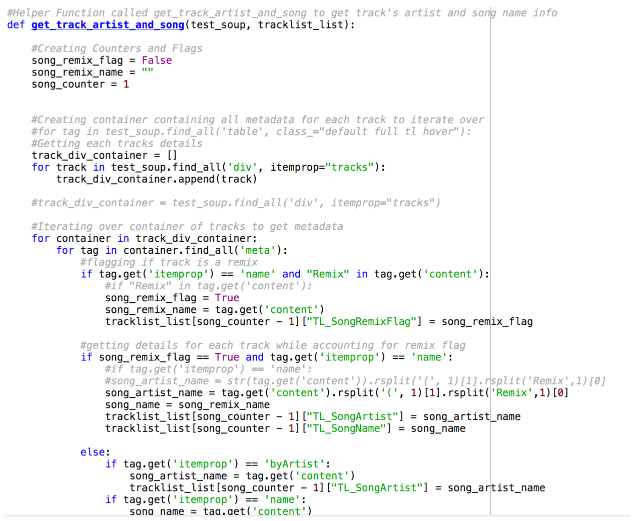
Shown below is an example webpage from 1001tracklists.com:


Set/Mix Name
Link to Set/Mix
"Groups" (e.g. Venue or Radio Show) that set/mix was played at
Set/Mix's Views and Likes
Number of Songs ID'ed
Individual Songs played by Artist in Set/Mix

Shown below are some snapshots of the code used to build out my web scraper to collect the necessary data on each set:

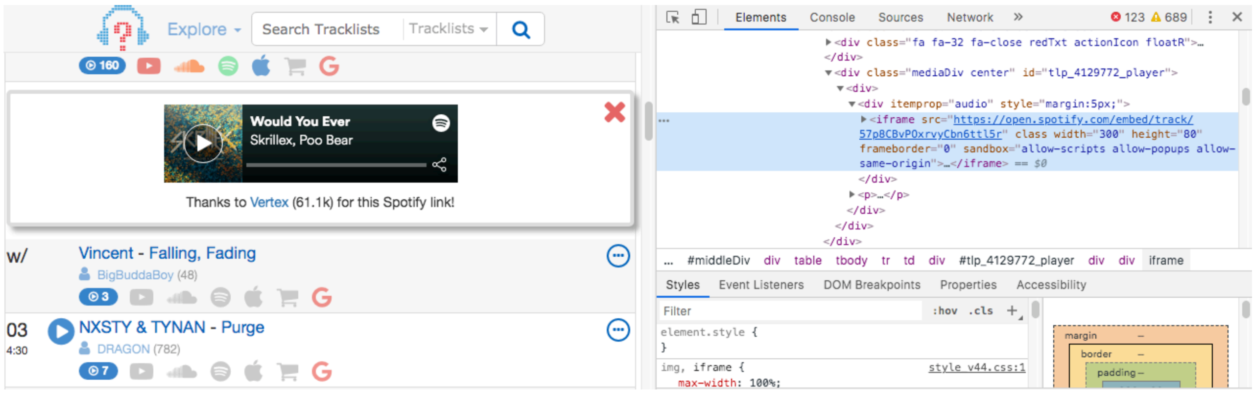


```
#Intializing webpage_objects to parse through
TL_webpages_objects = []
#Intializing list to fill in with converted webpage objects into URLs
TL_webpages_names = []
#Intilaiizing list to fill in with converted webpage objects into URLs AFTER opening Spotify players
TL_final_htmls = []



#Using Selenium to create driver to navigate web
executable_path = {"executable_path":""
driver = webdriver.Chrome(**executable_path)


test_artist_list = ["Ekali"]
for artist in test_artist_list:
#for artist in unique_edmArtists[0:6]:
    #Go to 1001tracklists.com


    #locally caching artists
    for folder in test_artist_list:
        newFolder = ""
        if not os.path.exists(newFolder):
            os.makedirs(str(newFolder))
        os.chdir(newFolder)

    #resetting TL_final_htmls for local cache
    #TL_final_htmls = []
    filecounter = 0


    driver.get('http://1001tracklists.com')

    #Navigating 1001tracklist.com to search each Artist
    element = driver.find_element_by_name("main_search")
    element.send_keys(artist)
    #implementing wait so I don't get banned again
    driver.implicitly_wait(5)

    element = driver.find_element_by_id("searchBtn")
```



```
#Helper Function called get_track_artist_and_song to get track's artist and song name info
def get_track_artist_and_song(test_soup, tracklist_list):

    #Creating Counters and Flags
    song_remix_flag = False
    song_remix_name = ""
    song_counter = 1


    #Creating container containing all metadata for each track to iterate over
    #for tag in test_soup.find_all('table', class_="default full tl hover"):
    #Getting each tracks details
    track_div_container = []
    for track in test_soup.find_all('div', itemprop="tracks"):
        track_div_container.append(track)

    #track_div_container = test_soup.find_all('div', itemprop="tracks")

    #Iterating over container of tracks to get metadata
    for container in track_div_container:
        for tag in container.find_all('meta'):
            #flagging if track is a remix
            if tag.get('itemprop') == 'name' and "Remix" in tag.get('content'):
                #if "Remix" in tag.get('content'):
                song_remix_flag = True
                song_remix_name = tag.get('content')
                tracklist_list[song_counter - 1]["TL_SongRemixFlag"] = song_remix_flag

            #getting details for each track while accounting for remix flag
            if song_remix_flag == True and tag.get('itemprop') == 'name':
                #if tag.get('itemprop') == 'name':
                #song_artist_name = str(tag.get('content')).rsplit('(', 1)[1].rsplit('Remix',1)[0]
                song_artist_name = tag.get('content').rsplit('(', 1)[1].rsplit('Remix',1)[0]
                song_name = song_remix_name
                tracklist_list[song_counter - 1]["TL_SongArtist"] = song_artist_name
                tracklist_list[song_counter - 1]["TL_SongName"] = song_name

            else:
                if tag.get('itemprop') == 'byArtist':
                    song_artist_name = tag.get('content')
                    tracklist_list[song_counter - 1]["TL_SongArtist"] = song_artist_name
                if tag.get('itemprop') == 'name':
                    song_name = tag.get('content')
```

The hardest data to scrape were the available Spotify links to each identified track in each set/mix as the links were only shown on actual "clicks". As such, Selenium was used to navigate to each set/mix page and click each Spotify link. Only then was I able to use BeautifulSoup to capture the Spotify unique ID of each identified track which was useful in measuring the set/mix's average Spotify Danceability, Energy, Tempo, and Valence which will play a major role in recommending the right set/mix.



```
▶<div class="fa fa-32 fa-close redTxt actionIcon floatR">…
</div>
▼<div class="mediaDiv center" id="tlp_4129772_player">
  ▼<div>
    ▼<div itemprop="audio" style="margin:5px;">
      ▶<iframe src="https://open.spotify.com/embed/track/
      57p8CBvPOxrvyCbn6ttl5r" class width="300" height="80"
      frameborder="0" sandbox="allow-scripts allow-popups allow-
      same-origin">…</iframe> == $0
      </div>
    ▶<p>…</p>
    </div>
  </div>
</div>
```

```
… #middleDiv div table tbody tr td div #tlp_4129772_player div div iframe
```

```
element.style {
}
img, iframe {                                   style_v44.css:1
  max-width: 100%;
```

Running the automated web scraper

For the purposes of this report, I don't show all the code used in the automated browser navigation and scraping used to obtain the set/mix data. See related files on my github for the actual code. Below I show an example output of the scraper that I've outputted to a local Excel file. Due to the scope of this project, I limited the output to 4-5 sets for the 4 artists previously mentioned but plan to do a more in-depth report on more artists and more sets/mixes.

```
In [11]: #Running the automated web scraper
         Master_DF = pd.read_excel("masterdata0729.xlsx")
         Master_DF = Master_DF.drop_duplicates(subset=["TL_Name","TL_SongName","TL_SongArtist"])
         print(Master_DF.tail())
```

```
     Unnamed: 0                         TL_Name  TL_URL TL_SC_Link  \
746          76  Ekali @ EDC Las Vegas 2019-05-19     NaN       None
747          77  Ekali @ EDC Las Vegas 2019-05-19     NaN       None
748          78  Ekali @ EDC Las Vegas 2019-05-19     NaN       None
750          80  Ekali @ EDC Las Vegas 2019-05-19     NaN       None
751          81  Ekali @ EDC Las Vegas 2019-05-19     NaN       None

    TL_Youtube_Link     TL_Date  TL_Views  TL_Likes  TL_Number_IDed  \
746            None  2019-05-19      8313         8             NaN
747            None  2019-05-19      8313         8             NaN
748            None  2019-05-19      8313         8             NaN
750            None  2019-05-19      8313         8             NaN
751            None  2019-05-19      8313         8             NaN

                 TL_Genres           TL_Sources  TL_Avg_Spotify_Danceability  \
746  Dubstep, Hip Hop, Trap  EDC Las Vegas 2019                      0.58169
747  Dubstep, Hip Hop, Trap  EDC Las Vegas 2019                      0.58169
748  Dubstep, Hip Hop, Trap  EDC Las Vegas 2019                      0.58169
750  Dubstep, Hip Hop, Trap  EDC Las Vegas 2019                      0.58169
751  Dubstep, Hip Hop, Trap  EDC Las Vegas 2019                      0.58169
```

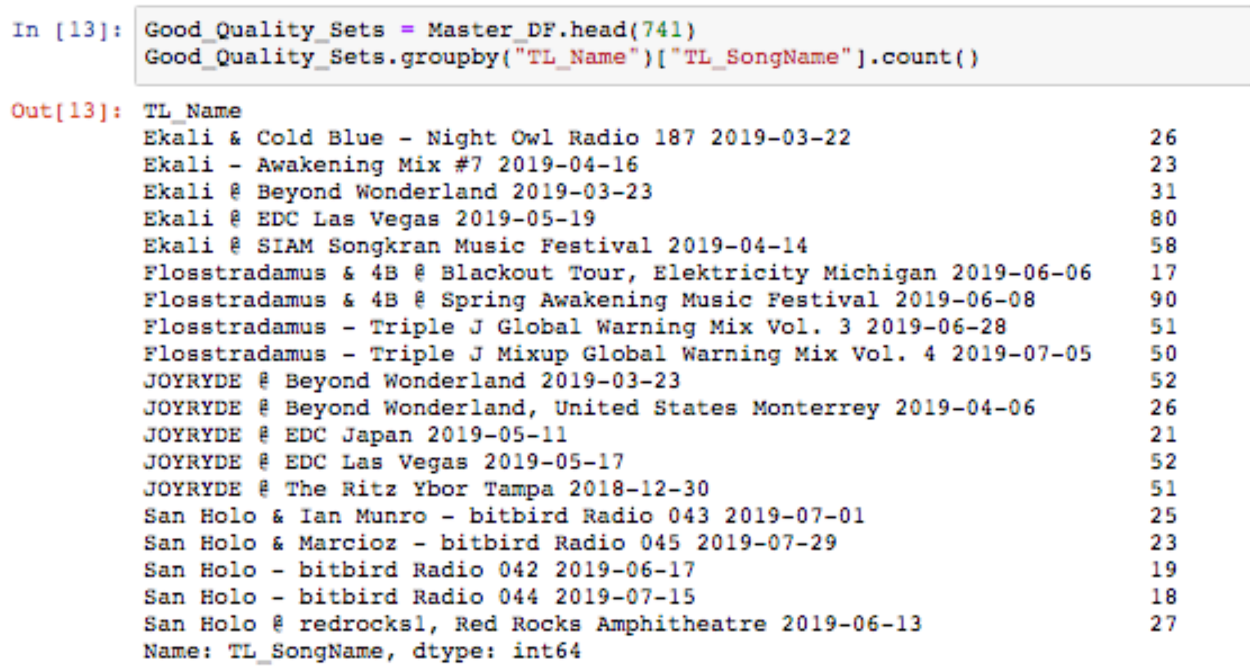
2.0 Exploratory Data Analysis

To illustrate the need for recommendation systems for sets/mixes, I try to highlight how significantly different sets/mixes can vary by artist through exploratory data analysis.

Due to data quality issues resulting from inconsistencies between 1001tracklists.com webpages and the time it would be needed to create a robust scraper that collected the datasets I needed, I went ahead with the data I was able to collect. With more time, I hope to expand the scraper to account for more nuances in each webpage to create a more robust dataset.

2.1 Genre Differences in Sets/Mixes

I selected 4-5 sets from each of my top edm artists that I thought to be of good quality so that I can conduct some high level exploratory data analysis:

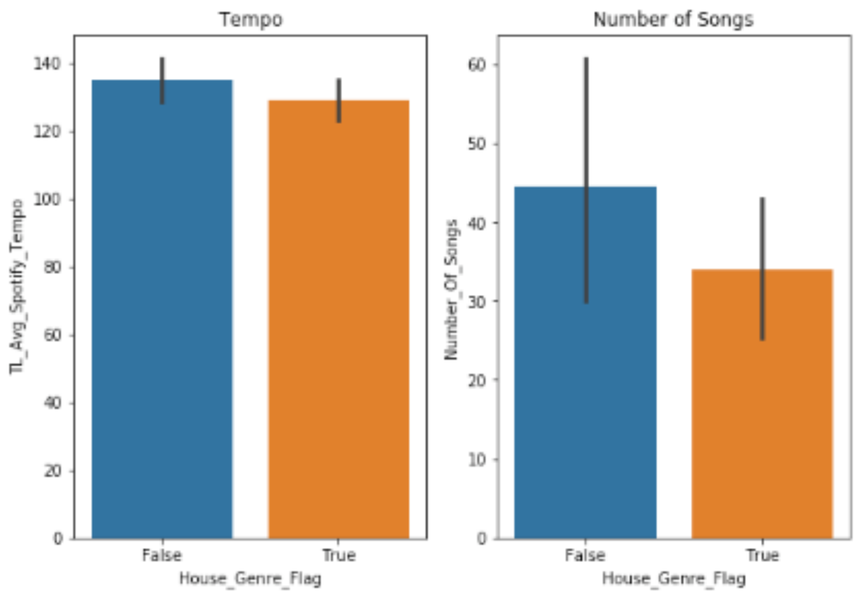


```
In [13]: Good_Quality_Sets = Master_DF.head(741)
         Good_Quality_Sets.groupby("TL_Name")["TL_SongName"].count()

Out[13]: TL_Name
         Ekali & Cold Blue - Night Owl Radio 187 2019-03-22                      26
         Ekali - Awakening Mix #7 2019-04-16                                     23
         Ekali @ Beyond Wonderland 2019-03-23                                    31
         Ekali @ EDC Las Vegas 2019-05-19                                        80
         Ekali @ SIAM Songkran Music Festival 2019-04-14                         58
         Flosstradamus & 4B @ Blackout Tour, Elektricity Michigan 2019-06-06    17
         Flosstradamus & 4B @ Spring Awakening Music Festival 2019-06-08        90
         Flosstradamus - Triple J Global Warning Mix Vol. 3 2019-06-28          51
         Flosstradamus - Triple J Mixup Global Warning Mix Vol. 4 2019-07-05    50
         JOYRYDE @ Beyond Wonderland 2019-03-23                                  52
         JOYRYDE @ Beyond Wonderland, United States Monterrey 2019-04-06        26
         JOYRYDE @ EDC Japan 2019-05-11                                          21
         JOYRYDE @ EDC Las Vegas 2019-05-17                                      52
         JOYRYDE @ The Ritz Ybor Tampa 2018-12-30                                51
         San Holo & Ian Munro - bitbird Radio 043 2019-07-01                     25
         San Holo & Marcioz - bitbird Radio 045 2019-07-29                       23
         San Holo - bitbird Radio 042 2019-06-17                                 19
         San Holo - bitbird Radio 044 2019-07-15                                 18
         San Holo @ redrocks1, Red Rocks Amphitheatre 2019-06-13                 27
         Name: TL_SongName, dtype: int64
```

Already we can see that there was a significant difference in number of tracks played, which is unusual given that the length of each set was generally similar. Looking at the differences between the individual tracks of each set might shed some light as to why there is this discrepancy.

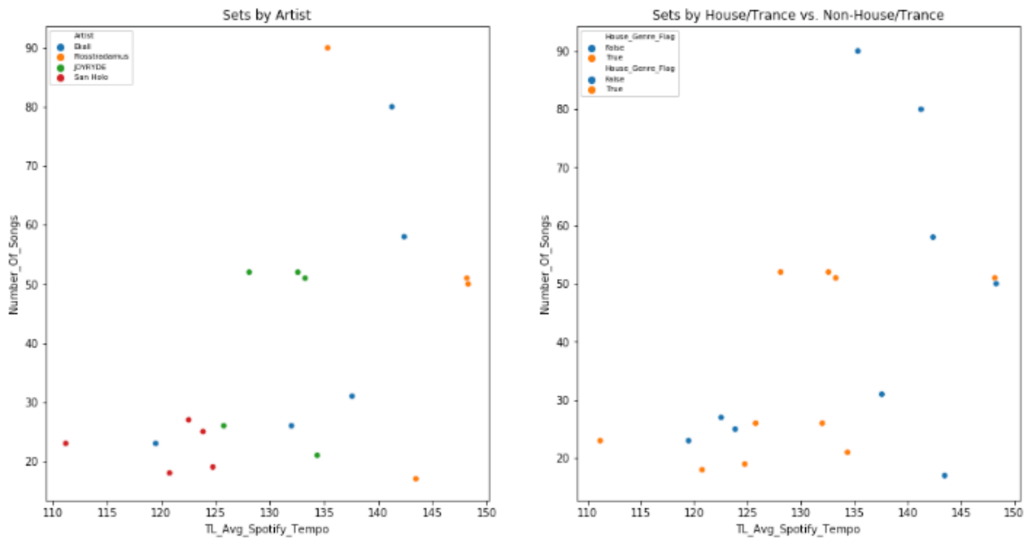
Using Spotify's genre metadata for each individual track in each set, I then flagged each set as having a house or trance track to see if there were any differences between tempo and number of songs played

Tempo by Genre & Number of Songs by Genre



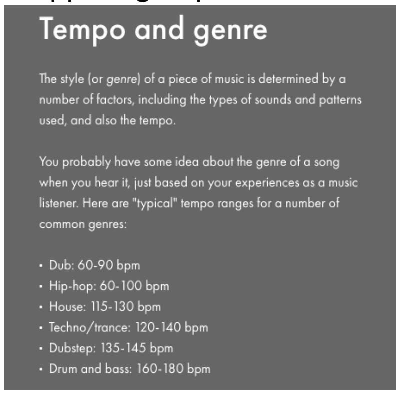
House/Trance sets on average have less songs and they on average contain more tracks with slower BPMs. We'll dig into this by artist to see if this trend remains true

Relationship between Tempo and Number of Songs by Genre



We can see that the earlier observed trend remains true by artist: House/Trance sets on average have less songs than non-House/Trance sets and the tracks are on average slower (as shown in the chart on the top right)

The chart on the top left also shows that there seems to be a lot of variation by artist. Taking Ekali as an example, he has sets that fall within the lower left and upper right quadrants.



Tempo and genre

The style (or genre) of a piece of music is determined by a number of factors, including the types of sounds and patterns used, and also the tempo.

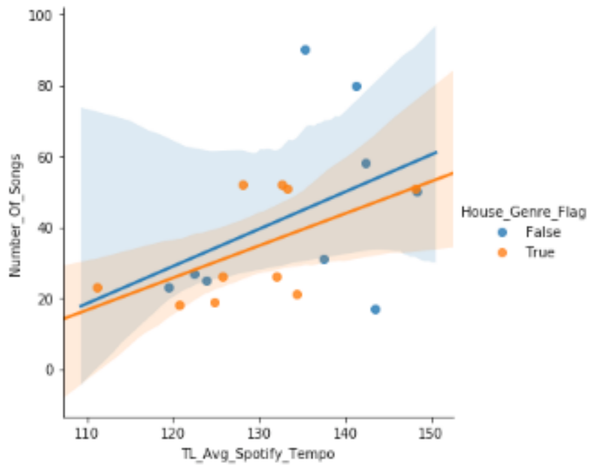
You probably have some idea about the genre of a song when you hear it, just based on your experiences as a music listener. Here are "typical" tempo ranges for a number of common genres:

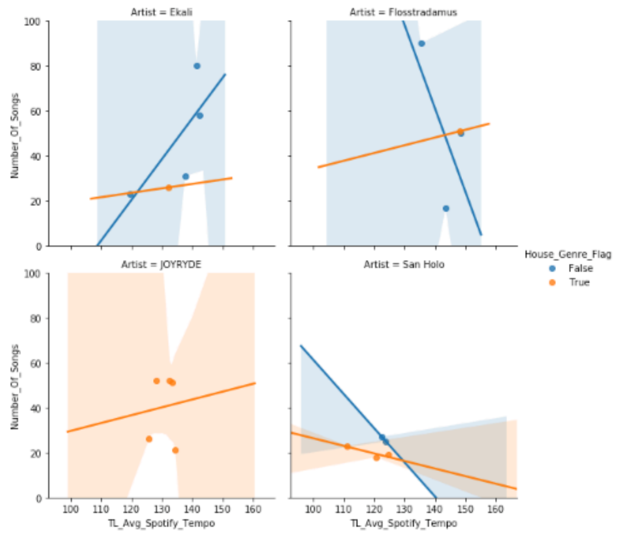
- Dub: 60-90 bpm
- Hip-hop: 60-100 bpm
- House: 115-130 bpm
- Techno/trance: 120-140 bpm
- Dubstep: 135-145 bpm
- Drum and bass: 160-180 bpm

With a quick google search on typical BPMs/Tempos for different genres (see below), I was able to contextualize the differences in average tempo of each set we obtained from Spotify with the differences in scraped genres(s) of each set we obtained from 1001tracklist.com.

At a quick glance, there seems to be a potential relationship between average tempo of a set and number of songs played. More data and further statistical testing would need to be conducted in order to establish a correlation between the two. Additionally, more data would allow for the testing of this relationship by artist.

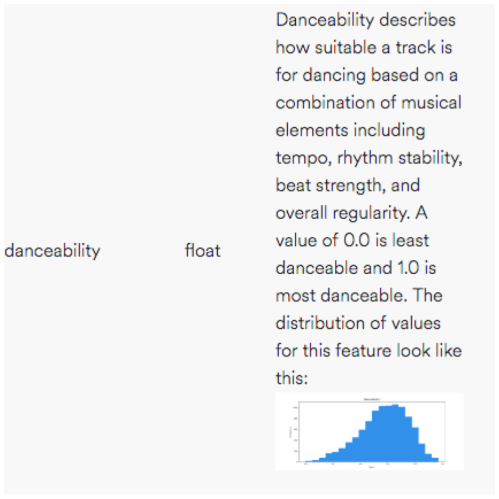
Relationship between Tempo and Number of Songs Played by Artist



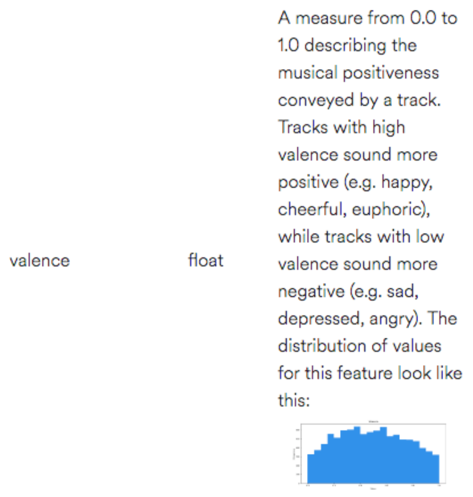


2.2 Relationship between Spotify Metrics

Some other interesting metrics Spotify captures is the danceability or valence of an individual song. Each song is measured on a scale from 0 to 1 (more details on this in the diagrams below)



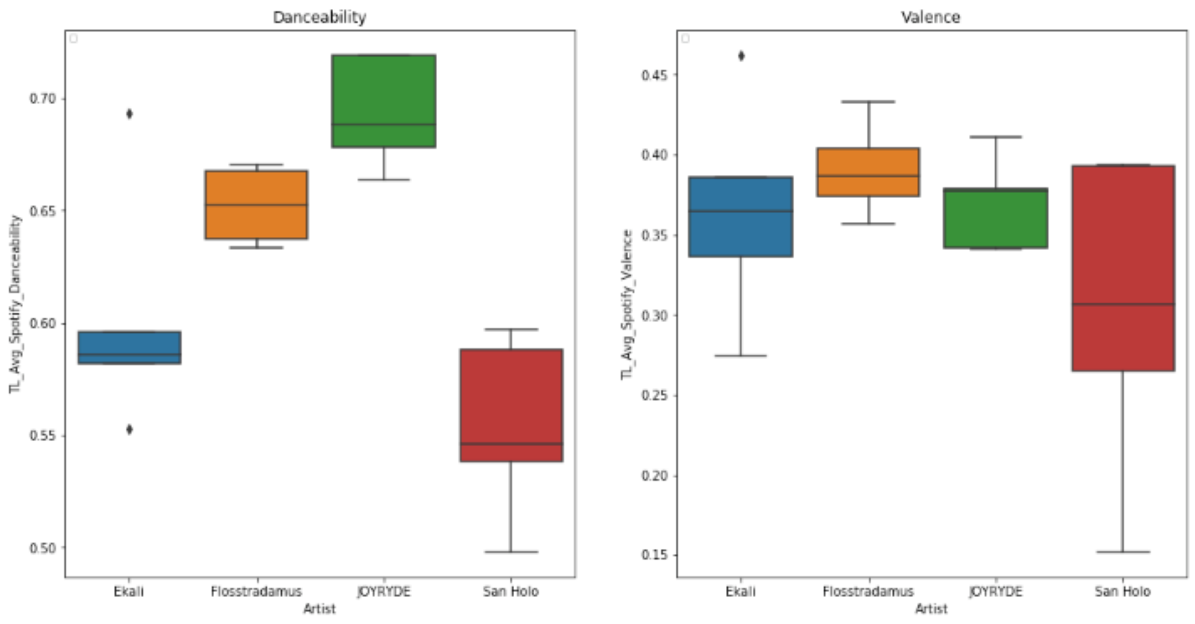
danceability	float	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. The distribution of values for this feature look like this:



valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). The distribution of values for this feature look like this:

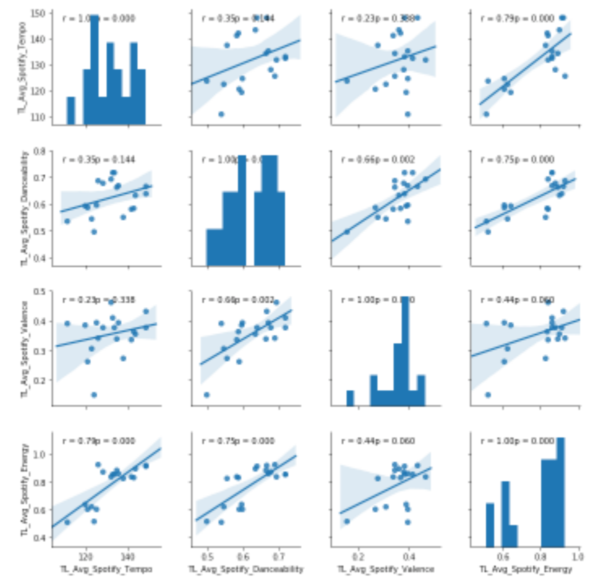
We have already established that there could be a difference between genres/tempos between sets from the same artist (as shown by Ekali). Let's see if this translates into differences in danceability and valence (i.e. "mood").

Differences between Valence/Danceability by Artist



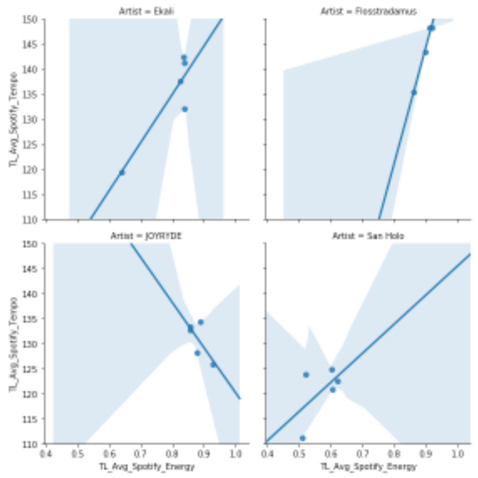
Although limited, the data illustrates that there is a significant difference in danceability and valence between artists.

Relationship between Spotify Metrics and Tempo



Furthermore, there seems to be a strong relationship between danceability and energy as well as tempo and energy. This intuitively makes sense as one would expect “fast-paced songs” to be of “higher energy”.

Relationship between Spotify Metrics and Tempo by Artist



With further data collection and statistical testing, I would like to verify if this strong correlation remains true by artist

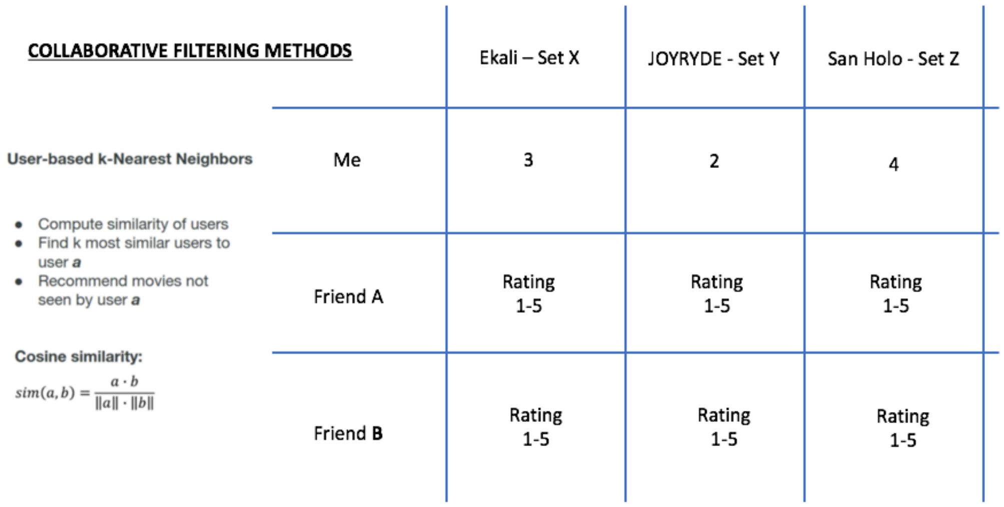
Summary of Exploratory Data Analysis:

- **Differences in Number of Songs per Set and Tempo for House/Trance Sets vs. Non-House/Trance Sets:** On average, House/Trance Sets seem to have fewer songs per set as well as slower BPMs, yet more data is needed to verify this trend by artist
- **Established initial hypothesis that there are differences between sets by the same artist:** Both the danceability and mood the most recent sets played by Ekali differed significantly. This will be useful in building out our recommendation engine using a Spotify User's most recently listened to songs, saved songs, and top artists
- **Differences in Danceability and Valence by Artist:** Spotify metrics varied significantly by artist, especially for valence
- **Established a potential correlation between Energy and Tempo as well as Danceability and Energy:** More data and further statistical testing can be conducted to see if this remains true across multiple genres/artists
-

3.0 Recommendation Engine Building

I hoped to build a recommendation engine using both content-based and collaborative filtering methods.

For collaborative filtering, I hoped to use user's evaluations of recommended sets to recommend additional sets based off the user's similarity between other users. Due to the scope of this project, I was unable to collect evaluations of recommended sets to use for collaborative filtering methods and this is something I hope to revisit in the future. Below is a diagram of what I would have conducted with more time for data collection:



COLLABORATIVE FILTERING METHODS

User-based k-Nearest Neighbors

- Compute similarity of users
- Find k most similar users to user ***a***
- Recommend movies not seen by user ***a***

Cosine similarity:

$$sim(a, b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

	Ekali – Set X	JOYRYDE - Set Y	San Holo - Set Z
Me	3	2	4
Friend A	Rating 1-5	Rating 1-5	Rating 1-5
Friend **B**	Rating 1-5	Rating 1-5	Rating 1-5

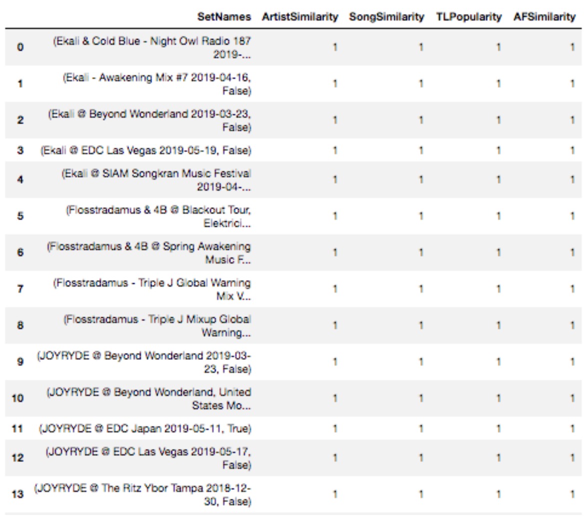
For now, I focused on content-based filtering methods and tried to develop a score per set based on the data collected on each set.

Since my goal is to recommend a set/mix that a Spotify User will most likely like, I will try to score each set/mix based on 4 factors:

- Set's Artist Similarity to current top Artists
- Set's Song Similarity to current top Songs and saved Songs
- Set's Overall popularity based on 1001tracklists.com
- Audio Feature Similarities between current top Songs and saved Songs with the Set's Average Audio Feature Metrics (Calculating difference in means between each set and Spotify user's current average danceability and mood)

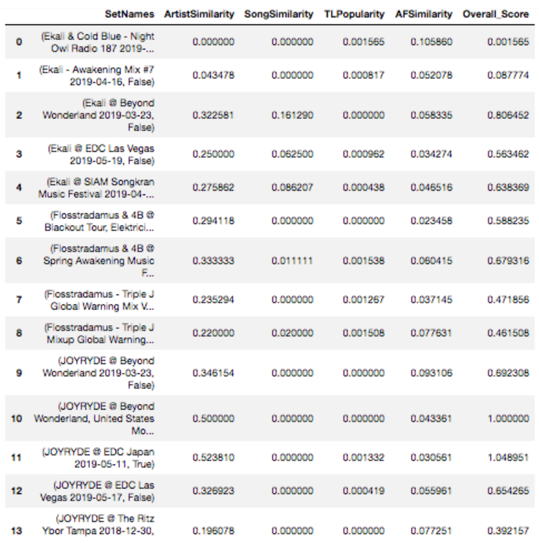
(See Github for detailed calculations on each of the 4 factors)

Example Scoring Matrix used to recommend a Set



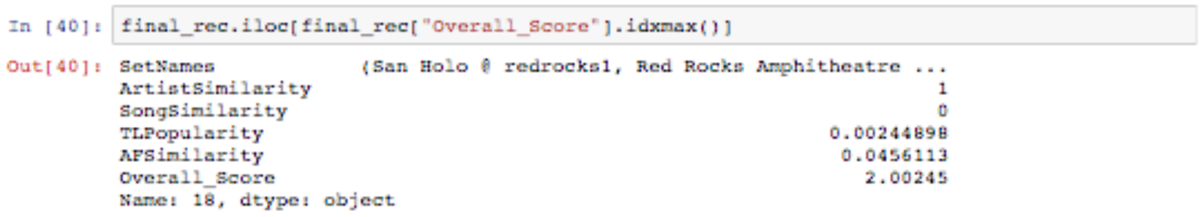
	SetNames	ArtistSimilarity	SongSimilarity	TLPopularity	AFSimilarity
0	(Ekali & Cold Blue - Night Owl Radio 187 2019-...	1	1	1	1
1	(Ekali - Awakening Mix #7 2019-04-16, False)	1	1	1	1
2	(Ekali @ Beyond Wonderland 2019-03-23, False)	1	1	1	1
3	(Ekali @ EDC Las Vegas 2019-05-19, False)	1	1	1	1
4	(Ekali @ SIAM Songkran Music Festival 2019-04-...	1	1	1	1
5	(Flosstradamus & 4B @ Blackout Tour, Elektrici...	1	1	1	1
6	(Flosstradamus & 4B @ Spring Awakening Music F...	1	1	1	1
7	(Flosstradamus - Triple J Global Warning Mix V...	1	1	1	1
8	(Flosstradamus - Triple J Mixup Global Warning...	1	1	1	1
9	(JOYRYDE @ Beyond Wonderland 2019-03-23, False)	1	1	1	1
10	(JOYRYDE @ Beyond Wonderland, United States Mo...	1	1	1	1
11	(JOYRYDE @ EDC Japan 2019-05-11, True)	1	1	1	1
12	(JOYRYDE @ EDC Las Vegas 2019-05-17, False)	1	1	1	1
13	(JOYRYDE @ The Ritz Ybor Tampa 2018-12-30, False)	1	1	1	1

->



	SetNames	ArtistSimilarity	SongSimilarity	TLPopularity	AFSimilarity	Overall_Score
0	(Ekali & Cold Blue - Night Owl Radio 187 2019-...	0.000000	0.000000	0.001565	0.105860	0.001565
1	(Ekali - Awakening Mix #7 2019-04-16, False)	0.043478	0.000000	0.000817	0.052078	0.087774
2	(Ekali @ Beyond Wonderland 2019-03-23, False)	0.322581	0.161290	0.000000	0.058335	0.806452
3	(Ekali @ EDC Las Vegas 2019-05-19, False)	0.250000	0.062500	0.000962	0.034274	0.563462
4	(Ekali @ SIAM Songkran Music Festival 2019-04-...	0.275862	0.086207	0.000438	0.046516	0.638369
5	(Flosstradamus & 4B @ Blackout Tour, Elektrici...	0.294118	0.000000	0.000000	0.023458	0.588235
6	(Flosstradamus & 4B @ Spring Awakening Music F...	0.333333	0.011111	0.001538	0.060415	0.679316
7	(Flosstradamus - Triple J Global Warning Mix V...	0.235294	0.000000	0.001267	0.037145	0.471856
8	(Flosstradamus - Triple J Mixup Global Warning...	0.220000	0.020000	0.001508	0.077631	0.461508
9	(JOYRYDE @ Beyond Wonderland 2019-03-23, False)	0.346154	0.000000	0.000000	0.093106	0.692308
10	(JOYRYDE @ Beyond Wonderland, United States Mo...	0.500000	0.000000	0.000000	0.043361	1.000000
11	(JOYRYDE @ EDC Japan 2019-05-11, True)	0.523810	0.000000	0.001332	0.030561	1.048951
12	(JOYRYDE @ EDC Las Vegas 2019-05-17, False)	0.326923	0.000000	0.000419	0.055961	0.654265
13	(JOYRYDE @ The Ritz Ybor Tampa 2018-12-30,	0.196078	0.000000	0.000000	0.077251	0.392157

Final Mix Recommendation based off my current Spotify listening habits:



```
In [40]: final_rec.iloc[final_rec["Overall_Score"].idxmax()]

Out[40]: SetNames           (San Holo @ redrocks1, Red Rocks Amphitheatre ...
         ArtistSimilarity                                                 1
         SongSimilarity                                                   0
         TLPopularity                                            0.00244898
         AFSimilarity                                             0.0456113
         Overall_Score                                              2.00245
         Name: 18, dtype: object
```

4.0 Conclusion

4.1 Results and Next Steps

Results

The recommendation system returned all sets with varying scores based off my current listening habits (taken from my personal Spotify data), with the highest-scoring set being returned as the recommended set to listen to. In this case, a set by San Holo at Red Rocks seems to be the most aligned with my current listening habits on Spotify.

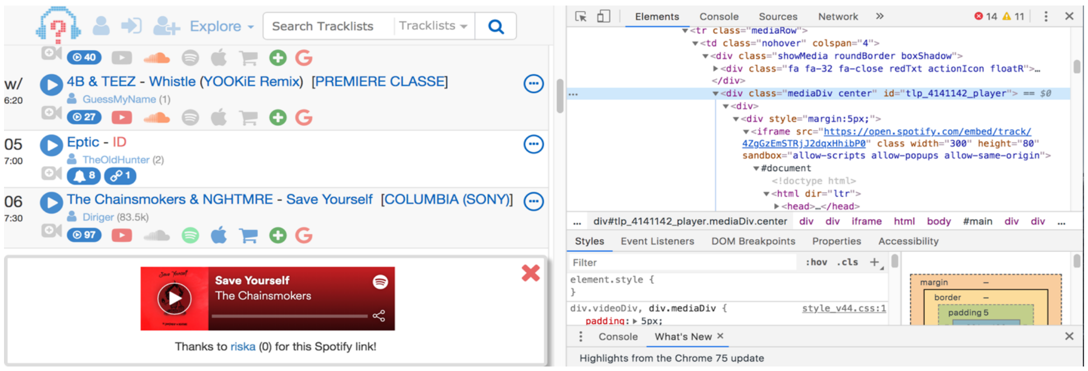
Ideally, I would have wanted to come up with an evaluation system for for users to test out my recommendation system. The evaluation metrics could then be used to fine-tune my scoring algorithm or be used as inputs for a social collaborative filtering method.

For the scope of this project, I was unable to collect enough data due to the amount of time and effort it would have required to understand the nuances of the website in which the data was stored. With more time, I hope to implement additional features to my web scraper to more efficiently and more accurately collect data on each set/mix.

Next Steps

With more time, I hope to implement the following:

- **Web Scraper Optimization - Broken Spotify Links**: The spotify objects on each page were not always 100% complete. As 1001tracklists.com is a website updated manually by users, it could have been the case that some individual songs had no Spotify links at all despite the same song having a working Spotify link on a different webpage. With more time, I hope to locally cache working Spotify links with their corresponding song and artist name as a key - value pair that I could then use to fill some of the gaps on some of the webpages
- **Web Scraper Optimization - Unstandardized HTML**: Another pitfall that I didn't initially account for that proved to be too time-consuming to overcome was the lack of uniformity amongst each webpage. As you can see in the screenshot below, the Spotify objects are located in a class called "mediaDiv center" with the id "tlp_4141142_player". Sometimes, these classes were labelled differently on different webpages. Even more challenging was trying to use Selenium to automate the clicking of these objects. Spotify trackid is only available upon "clicking" the Spotify object which then loads an iframe. Sometimes, these iframes wouldn't even load at all despite the Spotify Object showing up as green. More time would have been needed to improve my automated webscraper to overcome these challeneges and I hope to revisit this outside of the Capstone 2 Project to create a more robust dataset to feed into my recommendation system



w/ 6:20 4B & TEEZ - Whistle (YOOKiE Remix) [PREMIERE CLASSE]
GuessMyName (1)

05 7:00 Eptic - ID
TheOldHunter (2)

06 7:30 The Chainsmokers & NGHTMRE - Save Yourself [COLUMBIA (SONY)]
Diriger (83.5k)

Save Yourself
The Chainsmokers

Thanks to riska (0) for this Spotify link!

```
▼<tr class="mediaRow">
  ▼<td class="nohover" colspan="4">
    ▼<div class="showMedia roundBorder boxShadow">
      ▶<div class="fa fa-32 fa-close redTxt actionIcon floatR">…
      </div>
      ▼<div class="mediaDiv center" id="tlp_4141142_player"> == $0
        ▼<div>
          ▼<div style="margin:5px;">
            ▼<iframe src="https://open.spotify.com/embed/track/
            4ZgGzEmSTRjJ2dqxHhibP0" class width="300" height="80"
            sandbox="allow-scripts allow-popups allow-same-origin">
              ▼#document
                <!doctype html>
                ▼<html dir="ltr">
                  ▶<head>…</head>
```

… div#tlp_4141142_player.mediaDiv.center div div iframe html body #main div div …

```
element.style {
}
div.videoDiv, div.mediaDiv {                style_v44.css:1
  padding: ▶ 5px;
```

Highlights from the Chrome 75 update

- **Web Scraper Optimization - Cache Artist Data**: Most challenging of all was trying to build my automated web scraper in a way so that I wouldn't get banned each time I ran my scripts. Trying to find the optimal wait time between each click proved to be frustrating yet with more time I could implement wait conditions that other people have documented on various resources online to avoid being banned and collect a good amount of data