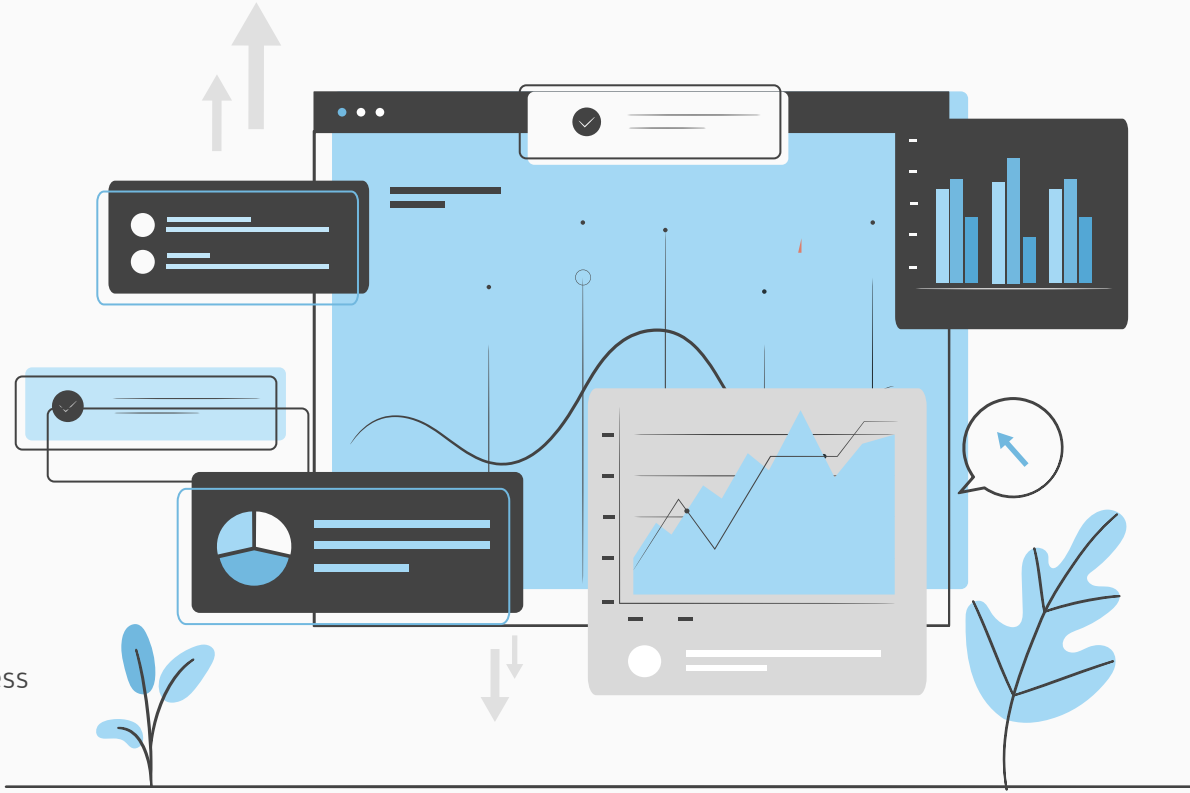
SPOTIFY PLAYLIST PROJECT

Team BLLLT 🥪 (B3LT for short)

BY: Bella Chang, Leah Hong, Logan Liu, Tess U-Vongcharoen, and Leslie Vasquez
(Mentor: Vaibhav Agrawal)



OUR JOURNEY

01 OVERVIEW

- Inspiration
- Project Description

02 CODING PROCESS

- Data Cleaning & Collection
- Code Overview
- Steps

03 CONCLUSIONS

- Final Product
- Future Steps

01. INSPIRATION

- Intrigued by Discover Weekly on Spotify & Spotify collection techniques
- Interested in personal recommendations (data sorting and classifying)

01. PROJECT DESCRIPTION

Our Spotify Playlist Project provides song recommendations based off 2 individual playlists using the NumPy, Pandas, and Spotipy libraries. The number of songs outputted depends on the lengths of the playlists. We focused these recommendations around certain attributes of different songs (danceability, acousticness, energy, etc.), and weighted these attributes accordingly in our overall classifying system.

CODING PROCESS:

COMPARING

PULLING DATA
SETUP
ORGANIZING
WEIGHTS & CATEGORIES

RECOMMENDING

GROUPING
EXTRACTING
CHOOSING

02. COMPARING (PULLING DATA)

- We imported the Spotipy, NumPy, and Pandas libraries
- Spotipy is a more comprehensive library with more updated songs than any dataset on Kaggle
  - Obviously, we must keep the songs for the purposes of the project
  - We also needed access to as many attributes as possible to test out accuracy in our creation of the weighting portion
  - Thus, we did not do much data cleaning for this project

```
from datascience import *
from scipy import stats
from scipy import special

#linear algebra
import numpy as np
import pandas as pd
import math
import random

#import spotify stuff
!pip install spotipy
import spotipy
import spotipy.oauth2 as oauth2
from spotipy.oauth2 import SpotifyClientCredentials
from spotipy.oauth2 import SpotifyOAuth
import time

cid = 'b8ef4ecc093c464191135d0ea204ea37'
secret = '9d03ba94da8a4fdeb9fc3d77a93d2552'

client_credentials_manager = SpotifyClientCredentials(client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
```

02. COMPARING (SETUP)

- Add in your playlists and set the amount of outputs you want!
  - Our code will tell you the exact output of songs

```
#these following five variables are the only things that need to labeled / changed!!!!!!

new_playlist_name = 'Bananas' #what do you want to name the new playlist??
playlist_1_link = 'https://open.spotify.com/playlist/0Dh79mnt4dty1DGqeZ6eJh?si=3k8xlPe7RdufMFce5ZzH7Q' #A
playlist_2_link = 'https://open.spotify.com/playlist/5gWlxkWtKT61EhlIyIA1jG?si=y5qpF3_6QwGiJZsGdZGfPA' #A
total_output = 0.8 #A value between 0 to 1 that represents the percentage of songs from the added playlist
percent_new = 0.5 #A value between 0 and 0.5 that represents percent of output returned that should be new
```

```
The number of tracks in both playlists combined is 77
The length of the shortest playlist, Playlist 2, is 25
The number of tracks to be outputted is 80.0% of the shorter playlist, which is 20 tracks
Of the songs outputted, 50.0%, or 10 tracks, will be newly recommended songs, and 10 will be from the inputted playlists
```

02. COMPARING (ORGANIZING)

- This table is an example of the collection of songs given to INPUT
  - Transfers it all to Spotipy ID's (seen on the left) with their attributes

	danceability float64 0.342 – 0.902	energy float64 0.0859 – 0.923	key int64 0 – 11	loudness float64 -16.061 – -3.092	mode int64 0 – 1	speechiness float64 0.0266 – 0.462	acousticness float64 0.000177 – 0.969	instrumentalness float64 0.0 – 0.207	liveness fl 0.0548 – 0.9
5A1w94uzc1kO1Zhe8WWxC3	0.788	0.427	5	-9.918	0	0.308	0.0649	0.0000362	0.171
1AHf5FSofKcUw8tyKkccKF	0.819	0.519	10	-7.16	0	0.0768	0.384	0	0.0898
5pJPgy2jGvvNUNfHPvG3Zp	0.342	0.609	8	-5.68	1	0.0943	0.185	0.00052	0.0799
5dmPNuHmRRJuHmJTDa7NuJ	0.739	0.471	8	-8.456	1	0.0436	0.0661	0.0000978	0.116
2ulMreFZwpCzxoXpDfORCh	0.902	0.536	5	-3.188	0	0.115	0.339	0.0000046	0.0981
Expand rows 5 - 68									
5XzmZjXhMjDHr7ZfJ6DELQ	0.579	0.268	2	-5.65	1	0.0442	0.868	0	0.12

02. COMPARING (WEIGHTS / CATEGORIES)

- While making the weights class, we sorted all the attributes by what we thought were most important and gave each value an arbitrary number based off of our instincts
- The final list of attributes that we believed would be the most important to consider were **danceability**, **valence**, **energy**, **liveness**, and **speechiness**

```
#attributes to focus on:

weights_dictionary = {
    'genre_weight':17,   #need to come up with a way to translate genre differences into numer
    'artist_weight':12,   #need to come up with a way to determine if two artists are similar,
    'mode_weight':8,   #also a binary value
    'valence_weight':15,
    'tempo_weight':6,   #simplify this into like 4 buckets or something
    'danceability_weight':13,
    'energy_weight':11,
    'acousticness_weight':5,
    'instrumentalness_weight':3,
    'loudness_weight':0,
    'liveness_weight':5,
    'speechiness_weight':5
                        }

weights_total = sum(list(weights_dictionary.values()))
assert sum(list(weights_dictionary.values())) == 100, f'Weights do not add up to 100, adds to

focus_attributes = ['danceability', 'valence', 'energy', 'liveness', 'speechiness'] #attribute

print('This cell has a dictionary of set weights for key song attribute values')
print(f'Current attributes considered are: {focus_attributes}')
```

```
This cell has a dictionary of set weights for key song attribute values
Current attributes considered are: ['danceability', 'valence', 'energy', 'liveness', 'speechiness']
```

02. COMPARING (SIMILARITIES)

- Our function calculates the similarity between two songs based off of our chosen weights
  - In order to see how similar the songs are specifically in weights, we multiplied percent similarity in focus attributes by the weights of those respective attributes
  - Then we summed the two values together in order to get the similarity score

```
def similarity(song_a, song_b, attributes):
    """
    Takes in two songs and calculates a similarity score by multiplying percent similarity
    in focus attributes by the weights of those respective attributes and summing together
    """
    similarity_sum = 0
    for elem in attributes:
        a_value = tbl.loc[f'{song_a}'][f'{elem}']
        b_value = tbl.loc[f'{song_b}'][f'{elem}']
        att_weight = weights_dictionary[f'{elem}_weight']
        value_diff = abs(a_value - b_value)
        percent_similar = value_diff / ((a_value + b_value) / 2) #comparing difference value to the mean of the two numbers
        weighted_similarity = percent_similar * att_weight
        similarity_sum = similarity_sum + weighted_similarity
    return similarity_sum
```

02. COMPARING (SIMILARITIES)

- We also created a song matrix that compares each song to every other song in the playlist by our weights and ultimately outputs the averages (shown on the far right)

This song matrix has a column that averages the scores for each song at the very right

	5A1w94uzc1kO1Zhe8W… float… 1.0 – 53.896813196636906	1AHf5FSofKcUw8tyKk… float… 1.0 – 48.757757693474474	5pJPgy2jGvvNUNfHPv… float… 1.0 – 44.54881291007505	5dmPNu… 1.0 – …
5A1w94uzc1kO1Zhe8WWxC3	1	16.10538223173137	29.153195681872905	21.020
1AHf5FSofKcUw8tyKkccKF	16.10538223173137	1	15.83733522118313	12.022
5pJPgy2jGvvNUNfHPvG3Zp	29.153195681872905	15.83733522118313	1	21.718
5dmPNuHmRRJuHmJTDa7NuJ	21.020683095478596	12.022818521057934	21.718693857709475	1
2ulMreFZwpCzxoXpDfORCh	15.419325430156526	12.13550632181429	24.848695856160017	22.369
Expand rows 5 - 68				
5XzmZjXhMjDHr7ZfJ6DELQ	27.451725257718508	20.697806740932304	24.192146905897804	9.9604
6pm3SR1vvrV54AOJWsN7y7	10.3821870032874	16.59729754918807	27.737949842606696	21.332
18uwL0vNUanqZH0ro2QcOP	21.79464599725608	24.490945325894707	38.7207260195468	28.846
31qCy5ZaophVA81wtlwLc4	14.887267840613589	11.460050391253692	21.99457871701465	12.918
4aVes2FpiAqzFrpTE0Zj9C	23.089970239925588	22.04312119503104	26.619868561846392	20.883

…

average float64 15.366197428268817 – 45.48…
22.0778301148894
17.913418367439384
22.45383724234433
19.078261848103114
21.27346256664395
22.670716537161482
20.56069796015068
28.95954428259887
15.714863947059973

02. RECOMMENDATION (GROUPING)

- We grouped the songs that were the most similar to one another in the aspects of valence, danceability, and energy to try and identify the “most common similarity attributes” amongst our inputs…

```
Valence groups:
[['5aXgz1oKK8Q9z9xvTmSnrO'], ['34xTFwjPQ1dC6uJmleno7x'], ['5zsHmE2gO3RefVsPyw2e3T'], ['30QNjcM3Q1GnLFIIJjWQL1', '3mSoxi4aC7oiTGJjsLLkaM'],
```

This code has been hidden. Show it.

```
Danceability groups:
[['5pJPgy2jGvvNUNfHPvG3Zp', '7pcANiSH8mEKLUIPAxiSDr'], ['30QNjcM3Q1GnLFIIJjWQL1'], ['3mSoxi4aC7oiTGJjsLLkaM'], ['1NfqGbaWcZLor12cITE5Fv',
```

This code has been hidden. Show it.

```
Energy groups:
[['30QNjcM3Q1GnLFIIJjWQL1', '34xTFwjPQ1dC6uJmleno7x'], ['18uwL0vNUanqZH0ro2QcOP'], ['4aVes2FpiAqzFrpTE0Zj9C'], ['5XzmZjXhMjDHr7ZfJ6DELQ'],
```

02. RECOMMENDATION (EXTRACTING)

- We sort the groups by size
- Large groups imply a highly common attribute amongst all the songs in the input
- Then, we take these largest groups and input into a Spotipy recommendation function...

```
This cell groups songs through valence, danceability, and energy
```

```
[['2hwOoMtWPtTSSn6WHV7Vp5',
  '07ZQLYn9x4x3L3vxStc1zr',
  '3NxuezMdSLgt4OwHzBoUhL',
  '3NxuezMdSLgt4OwHzBoUhL',
  '2ZwIO3ufWLFYxtEoam9ydu',
  '6lY38FkInSA0QVHRb1PiEy'],
 ['6SzMMd1rNtyfj8bAgm2BLw',
  '6SzMMd1rNtyfj8bAgm2BLw',
  '1ITJTMrS4cx8zdlI7DdSoo',
  '2m1EQGGjWt8z4rqcjyzFoR',
  '2hwOoMtWPtTSSn6WHV7Vp5',
  '3usbnvDFtOhY09cRNar8Zg'],
 ['15k2nBQJ0teDmPZHrOXL2N',
  '1444RlH5FNC5fJmBcLB9xI',
  '3RoycW4yhd2HCsWmLR7xIi',
  '75ZKw8JLaFsYr51J44fQ4N',
  '4qjLvvBh5ZeKEPyShKRf06'],
```

02. RECOMMENDATION (CHOOSING)

- Finally, this cell finds the songs in the recommended groups that rank the highest by our algorithm and ultimately adds them to our playlist
- Then your recommended songs will be outputted! (SO COOL!😎😎🥶🥶❄️❄️ (just like Vaibhav))

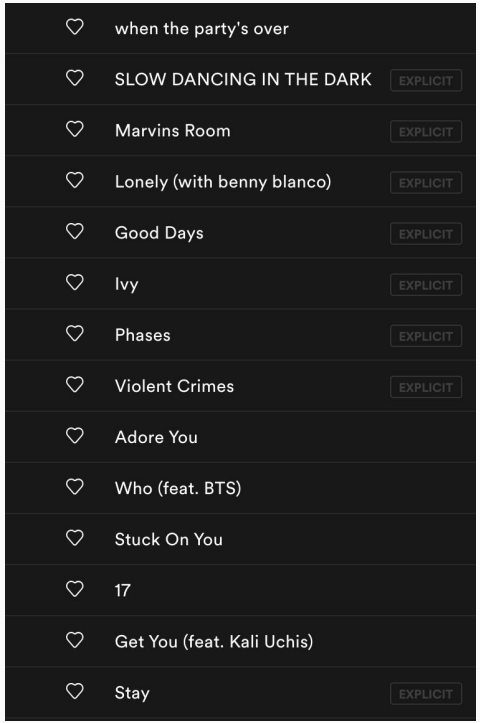
```
for elem in interest_groups:
    if len(elem) > 5:  #this code edits the groups, as you can only have 5 song ids passed into the recommendation function
        edited = []
        limit = 5
        while limit != 0:
            edited.append(elem[5 - limit])
            limit -= 1
        elem = edited
    song_added = False
    raw_rec = sp.recommendations(seed_tracks=elem, limit=10)
    possible_tracks = raw_rec['tracks']
    list_of_possible = []
    for item in possible_tracks:
        item_id = item['id']
        list_of_possible.append(item_id)
    while song_added == False:
        random_track = random.choice(list_of_possible)
        if random_track not in final_song_ids:
            if random_track not in og_song_ids:
                song_added = True
                new_songs_ids.append(random_track)
```

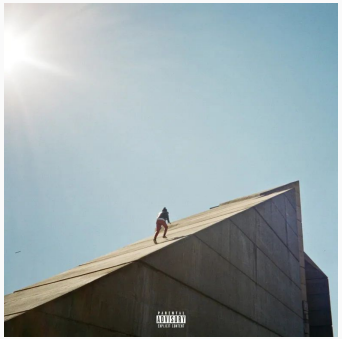
```
new_song_dict = {}
for elem in new_songs_ids:
    name = sp.track(elem)['name']
    new_song_dict[elem] = name

new_songs_names = []
for elem in new_songs_ids:
    name = new_song_dict[elem]
    new_songs_names.append(name)

new_songs_table = Table().with_column("Name", new_songs_names)
new_songs_table
# print(f'The new songs are: {new_songs_names}')
```

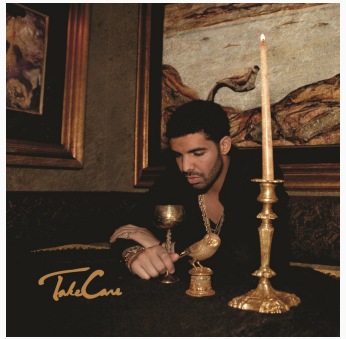
Name
You and I (Park Bom)
Cold Fire
Transient
Light Me Up
Boys Will Be Bugs
Wasting All My Time
Happy Man
Velvet Light
Agua Verde

03. FINAL PRODUCT

Our group put together **PLAYLISTS BASED ON SAD POP** (full playlists not pictured)

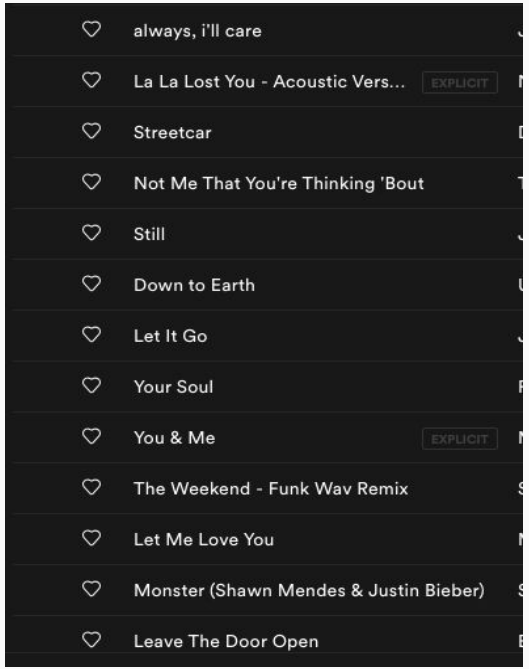






03. FINAL PRODUCT

Our group put together **PLAYLISTS BASED ON SAD POP** (full playlists not pictured)



03. FINAL PRODUCT

Our project outputted 8 songs based on the **MOST SIMILAR** songs between the two playlists.*
These songs matched the theme of either **minor**, **slow**, or **pop songs**.
**The 8 songs were a result of the lengths of both our playlists*

Name
Can I
By My Side
Lose
Hit Different
Renee's Song
What A Time (feat. Niall Horan)
The Few Things (With Charlotte Lawrence)
Try Again

DEMO!

03. FUTURE STEPS

WEIGHTS

So far, the weights for our attributes is mostly arbitrary and not based on any past experimentation. Our project can be even more accurate in recommendations if we were to experiment with the different weights for each attributes on different kinds of playlists.

ATTRIBUTES

In our code, we only considered the attributes energy, valence, and danceability. In the future, we would like to consider acousticness, liveness, etc. By considering more attributes, our recommendations will improve.

COMPLEXITY

Our project heavily simplifies the process of comparing songs to one another. “Better” recs would certainly use much more complicated methods of comparing. Additionally, defining “similarity” between music can be objective and nuanced, so more factors need to be considered.

THANKS!

Any Questions?

Bella "Kachow" Chang

Vaibhav "Cool Mentor" Agrawal

Tess "Strawberry Shortcake" U-Vongcharoen

Lo"gangsta" Liu

Leah "#2" Hong

Leslie "Denim Dame" Vasquez