Final Project

Motivation surrounding project topic

TikTok is a social media platform that has seen significant usage in such a short period of time. A main feature of the application is being able to use music in the videos created. With over 1 billion users in 2021 according to *Business of Apps*, I was curious whether there is a correlation between the popular tracks used in TikTok and the hot 100 billboard music chart for a given time period. So my motivation is to explore the features of the popular tracks and compare it with the billboard chart, finding out similar songs between the datasets, exploring the release date of songs in comparison to the hot 100 chart and trying to find out what makes an artist popular.

Brief description of data sources

The three datasets that I will be using are the hot 100 billboard music chart, the Spotify API and a dataset from Kaggle of TikTok trending tracks. The links to the source of the data is below:

1. https://www.billboard.com/charts/hot-100/2021-06-06/
2. https://developer.spotify.com/
3. https://www.kaggle.com/datasets/yamqwe/tiktok-trending-tracks
    - https://www.kaggle.com/code/eharian1/top-tiktok-tracks

The hot 100 billboard music chart dataset, like the name implies shows the hot 100 billboard music for a certain week. It consists of the artist name, track name and positions on the chart. For the billboard dataset, I am specifically using data from the week of 06/06/2021. This is because the TikTok trending tracks dataset was uploaded within that time frame. Since the billboard chart is constantly changing every week, it will be an inaccurate representation if I use the current week's chart as it does not encompass the other dataset.

The Spotify dataset consists of all the audio features provided from the API on the songs in the hot 100 billboard music chart.

The TikTok dataset constains the top trending tracks used in TikTok, it contains the track information and audio features. For the TikTok dataset, I added a kaggle notebook I made because currently if you go to the original link to download the dataset, it will lead to a page not found. Thus, I created a notebook to read the data and download it through the notebook.

Analysis performed

Final Project

May 11, 2022

[1]:
```
# load all necessary libraries
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
pd.options.mode.chained_assignment = None  # default='warn'
```

[2]:
```
# load data from saved datasets
billboard_df = pd.read_csv('./datasets/billboard_hot_100.csv')
spotify_df = pd.read_csv('./datasets/spotify.csv')
tiktok_df = pd.read_csv('./datasets/tiktok.csv')
```

[3]:
```
# remove duplicates from the tiktok dataset
tiktok_df = tiktok_df.drop_duplicates(subset=['track_id']).copy()

# remove the first column as it is a proxy id
tiktok_df = tiktok_df.iloc[:, 1:]

# some songs have different ids like if a song is released as both a single or␣
↪in an album.
# so we want to remove it as they are essentialy the same song
tiktok_df = tiktok_df.drop_duplicates(subset=['track_name', 'artist_name'],␣
↪keep='first')

# remove the duration in minutes as the duration in ms is already available
del tiktok_df['duration_mins']
```

[4]:
```
# look at data types of data
tiktok_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3366 entries, 0 to 6743
Data columns (total 22 columns):
```

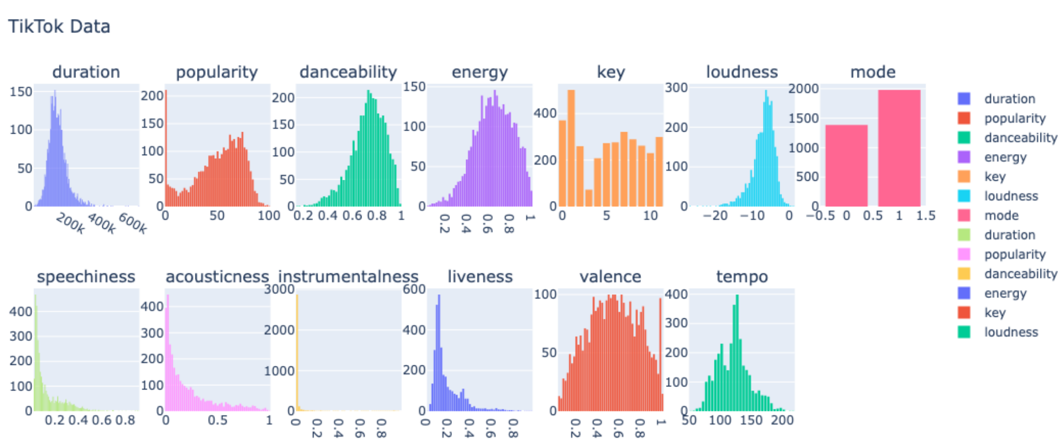
```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   track_id          3366 non-null   object
 1   track_name        3366 non-null   object
 2   artist_id         3366 non-null   object
 3   artist_name       3366 non-null   object
 4   album_id          3366 non-null   object
 5   duration          3366 non-null   int64
 6   release_date      3366 non-null   object
 7   popularity        3366 non-null   int64
 8   danceability      3366 non-null   float64
 9   energy            3366 non-null   float64
 10  key               3366 non-null   int64
 11  loudness          3366 non-null   float64
 12  mode              3366 non-null   int64
 13  speechiness       3366 non-null   float64
 14  acousticness      3366 non-null   float64
 15  instrumentalness  3366 non-null   float64
 16  liveness          3366 non-null   float64
 17  valence           3366 non-null   float64
 18  tempo             3366 non-null   float64
 19  playlist_id       3366 non-null   object
 20  playlist_name     3366 non-null   object
 21  genre             3366 non-null   object
dtypes: float64(9), int64(4), object(9)
memory usage: 604.8+ KB
```

[5]:
```python
# take the columns with numerial values
numerical_features=tiktok_df.select_dtypes(include=['int64','float64']).columns.
 ↪tolist()
```

[6]:
```python
numerical_features
```

[6]:
```
['duration',
 'popularity',
 'danceability',
 'energy',
 'key',
 'loudness',
 'mode',
 'speechiness',
 'acousticness',
 'instrumentalness',
 'liveness',
 'valence',
 'tempo']
```

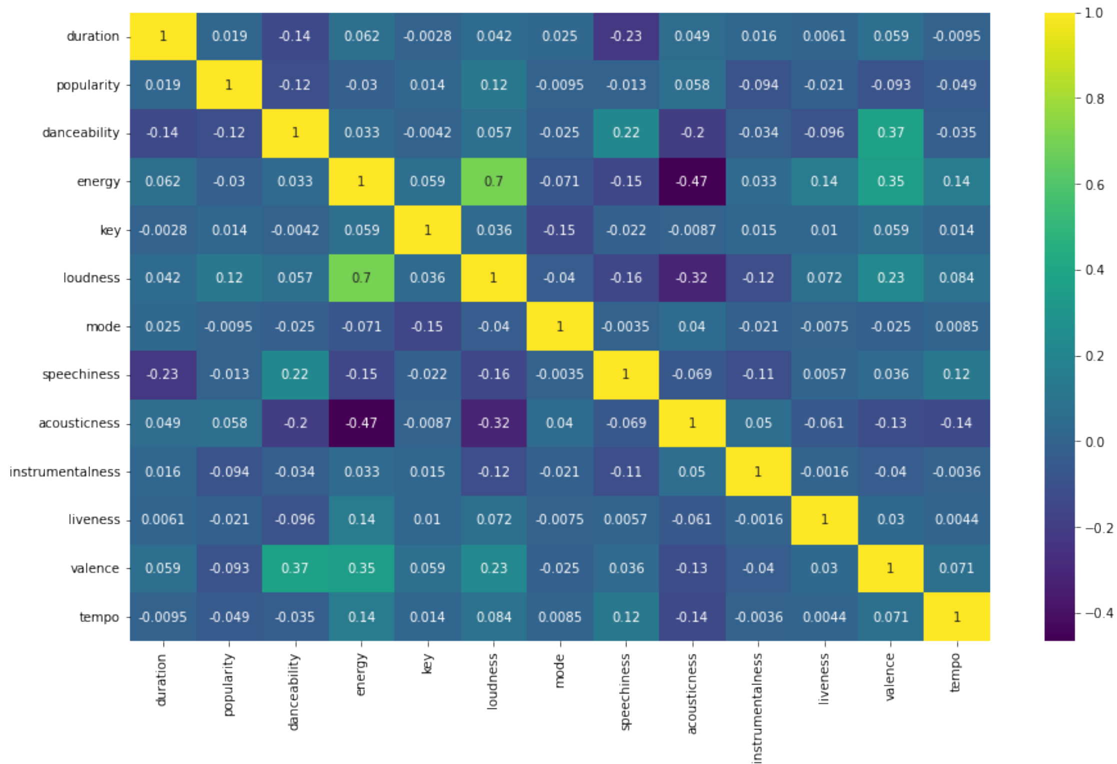
[7]:
```
# create a histogram for TikTok data
j=1
d=1

top=tiktok_df[numerical_features].columns[0:7]
bottom=tiktok_df[numerical_features].columns[7:13]
fig= make_subplots(rows=2, cols=7, start_cell ='top-left',␣
↪subplot_titles=numerical_features)


for idx, k in enumerate(bottom):
    for idx2, i in enumerate(top):
        if j<len(top)+1:
            fig.add_trace(go.Histogram(x=tiktok_df[i],␣
↪name=numerical_features[idx2]),row=1,col=j)
            j+=1
    if d<len(bottom)+1:
        fig.add_trace(go.Histogram(x=tiktok_df[k], name =␣
↪numerical_features[idx]),row=2,col=d)
        d+=1
fig.update_layout(bargap=.2,width=1100,height=500, title_text='TikTok Data')
fig.show()
```



The list of histograms above show the audio features of all the tracks in the TikTok dataset. The duration histogram can be ignored because songs used in TikTok videos are not of full length, so it is irrelevant for our case. There seems to be a similarity in the graphs of danceability, energy and loudness. This inferrence makes sense because there are a lot of dance challenges in TikTok, so music with higher danceability should be preffered but this requires more analysis in comparison to the Spotify data.

What is interesting to note is that there are a lot of songs with low popularity. According to the Spotify API, the popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. This is actually a great indication as it implies that there are plenty of new songs being used in TikTok which may contribute to its positio in the hot 100 billboard chart. Later we will look at these songs and check if they are shared among the other dataset.

[8]:
```python
# correlation matrix of TikTok data
plt.figure(figsize=(15,9))
sns.heatmap(tiktok_df.corr(),annot=True,cmap='viridis');
```

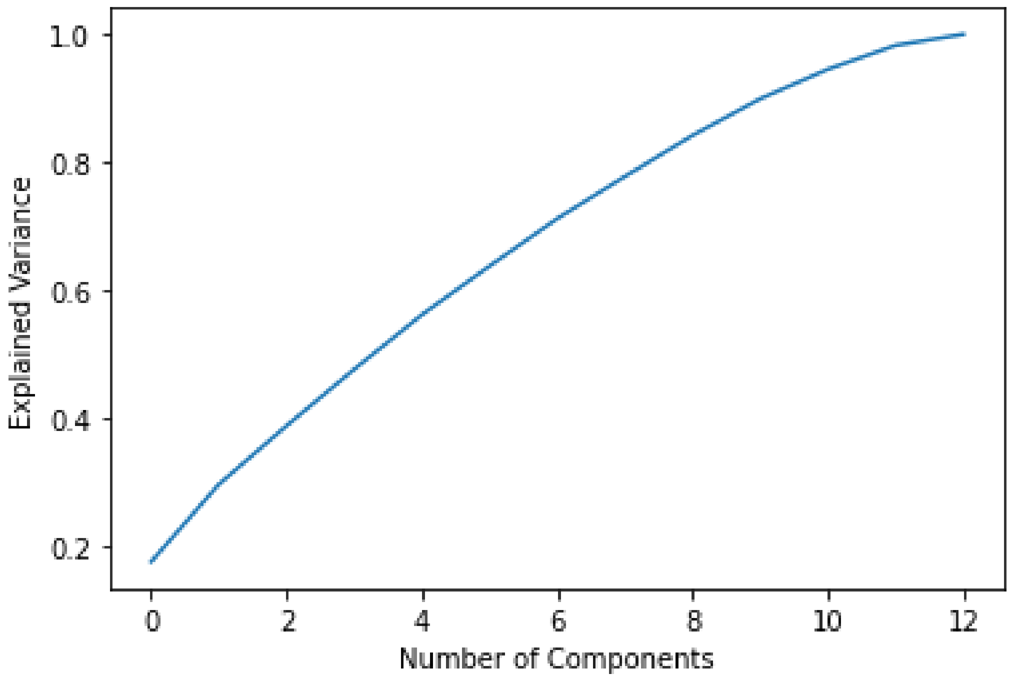


Above is the correlation matrix of the TikTok data, higher value numbers means a positive relationship and vice versa for lower values. There are a lot of variables in play that contribute to the entire dataset, but this can be unfeasible when analyzing as some variables might contribute very minimally. To figure out which components can encompass a majority of the information, we will perform a principal component analysis on the audio features, which reduces the dimensionality of large datasets

[9]:
```python
X = tiktok_df.loc[:, numerical_features].values
# Standardizing the features
X = StandardScaler().fit_transform(X)


pca = PCA(n_components=13)
pca.fit(X)
```

[9]: PCA(n_components=13)

[10]:
```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel("Number of Components")
plt.ylabel("Explained Variance")
plt.show()
```

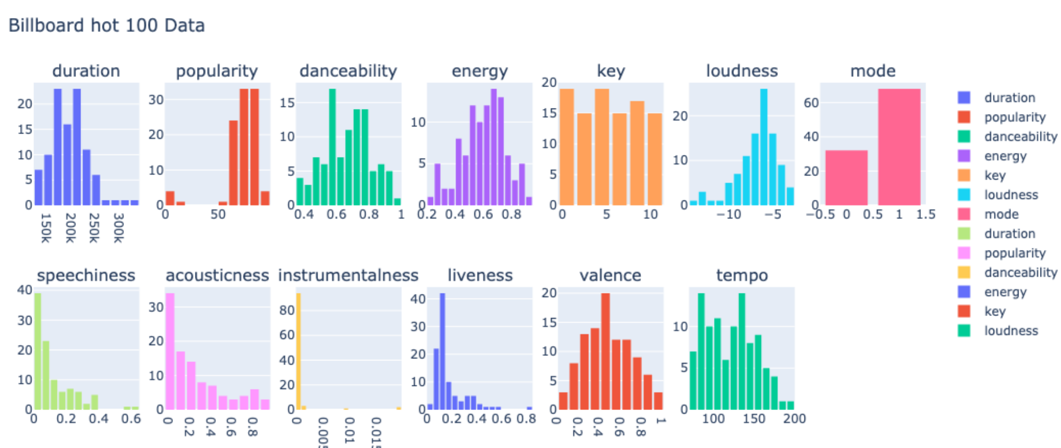


[11]:
```
for i in range(13):
    print(f'Variance explained by the {i+1}th component: {np.cumsum(pca.
 ↪explained_variance_ratio_*100)[i]}' )
```

```
Variance explained by the 1th component: 17.649268888864086
Variance explained by the 2th component: 29.706756356628844
Variance explained by the 3th component: 38.84652956290471
Variance explained by the 4th component: 47.67016508026491
Variance explained by the 5th component: 56.20568079777212
Variance explained by the 6th component: 63.81216632556783
Variance explained by the 7th component: 71.22066911712781
Variance explained by the 8th component: 77.79989382265346
Variance explained by the 9th component: 84.2061092758767
Variance explained by the 10th component: 89.9267655293408
Variance explained by the 11th component: 94.53535033142292
Variance explained by the 12th component: 98.27500511604164
Variance explained by the 13th component: 99.99999999999999
```

Based on the correlation matrix, I assumed that only a few components would be needed to capture the data, but after performing the PCA, to get most of the data, about 8 or 9 components are needed. It is still reduced from the original 13 though. Next, we will explore the audio features of the hot 100 billboard tracks then compare the data with that of the TikTok dataset

[12]:
```python
# create a histogram for hot 100 data
j=1
d=1

top=spotify_df[numerical_features].columns[0:7]
bottom=spotify_df[numerical_features].columns[7:13]
fig= make_subplots(rows=2, cols=7, start_cell ='top-left', subplot_titles=numerical_features)


for idx, k in enumerate(bottom):
    for idx2, i in enumerate(top):
        if j<len(top)+1:
            fig.add_trace(go.Histogram(x=spotify_df[i], name=numerical_features[idx2]),row=1,col=j)
            j+=1
    if d<len(bottom)+1:
        fig.add_trace(go.Histogram(x=spotify_df[k], name = numerical_features[idx]),row=2,col=d)
        d+=1
fig.update_layout(bargap=.2,width=1100,height=500, title_text='Billboard hot 100 Data')
fig.show()
```



On first glance, it is instantly noticeable that the popularity is skewed to the right which makes sense since songs in the hot 100 billboard should be popular in streaming services. An interesting thing to note is that the there are a lot of songs which have lower danceability, which is counter intuitive of the assessment made with the TikTok data. Lower danceability with lower valence seems to go in pair whereby if a song is less "positive", it would be likely be less danceable. But in terms of the other histograms, the trend is very similar.

This is only looking at the datasets from a birds eye view, now we will extract the songs that appear in both the billboard hot 100 and the TikTok dataset and make inferences from it

[13]:
```
# find the tracks that exist in the billboard hot 100 and TikTok dataset
j = 0
shared_tracks = pd.DataFrame(columns = spotify_df.columns.values)
shared_tracks['status'] = 0
artist_count = {}
for i, track in enumerate(billboard_df['track_name']):
    # need to make lower case because the formatting is inconsistent from the␣
↪TikTok dataset
    billboard_track = track.lower().strip()
    artist = billboard_df['artist_name'][i]

    # find the index of the data from the dataframe
    idx = tiktok_df.index[billboard_track == tiktok_df['track_name'].str.
↪lower().values].tolist()
    if idx:
        shared_tracks = shared_tracks.append(spotify_df.loc[i],␣
↪ignore_index=True)
        shared_tracks['status'][j] = billboard_df['status'][i]
        j += 1

        artist_count[artist] = artist_count.get(artist, 0) + 1
```

[14]:
```
shared_tracks.head()
```

[14]:
```
                 track_id             track_name  \
0  1mWdTewIgB3gtBM3TOSFhB                 Butter
1  4ZtFanR9U6ndgddUvNcjcG               Good 4 U
2  02VBYrHfVwfEWXk5DXyf0T    Leave The Door Open
3  5Q079kh1waicV47BqGRL3g        Save Your Tears
4  2bIYS7iV3IjUixYZsVXGvQ                Peaches

                                         artist_name duration release_date  \
0                                                BTS   164442   2021-06-04
1                                     Olivia Rodrigo   178147   2021-05-21
2        Silk Sonic (Bruno Mars & Anderson .Paak)   242096   2021-11-11
3                       The Weeknd & Ariana Grande   215627   2020-03-20
4  Justin Bieber Featuring Daniel Caesar & Giveon   198082   2022-04-19

  popularity  danceability  energy key  loudness mode  speechiness  \
0         88         0.759   0.459   8    -5.187    1       0.0948
1         92         0.563   0.664   9    -5.044    1       0.1540
2         84         0.586   0.616   5    -7.964    1       0.0324
```

```
3          89          0.680    0.826    0     -5.487     1       0.0309
4           0          0.677    0.696    0     -6.181     1       0.1190

   acousticness  instrumentalness  liveness  valence    tempo     status
0       0.00323          0.000000    0.0906    0.695  109.997  no-change
1       0.33500          0.000000    0.0849    0.688  166.928  no-change
2       0.18200          0.000000    0.0927    0.719  148.088  no-change
3       0.02120          0.000012    0.5430    0.644  118.051  no-change
4       0.32100          0.000000    0.4200    0.464   90.030  no-change
```

[15]:

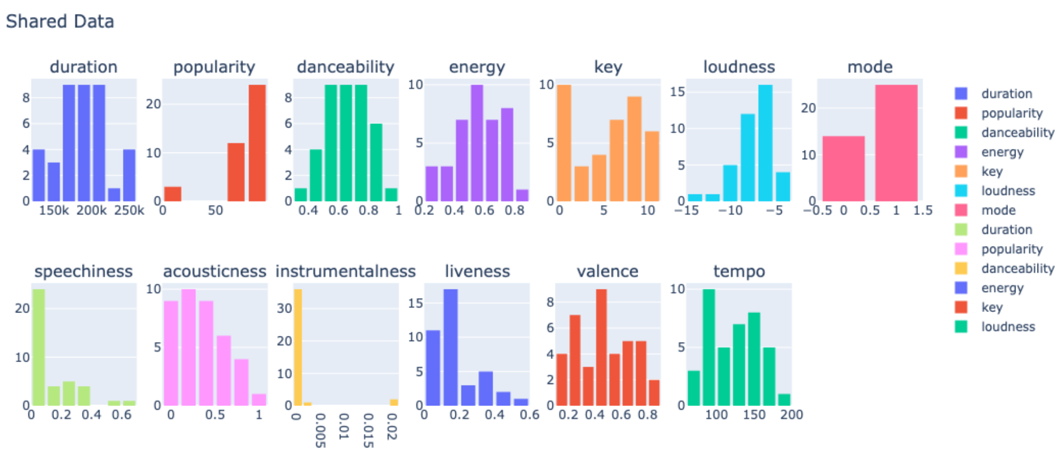
```
# create a histogram for shared data
j=1
d=1

top=shared_tracks[numerical_features].columns[0:7]
bottom=shared_tracks[numerical_features].columns[7:13]
fig= make_subplots(rows=2, cols=7, start_cell ='top-left',␣
 ↪subplot_titles=numerical_features)


for idx, k in enumerate(bottom):
    for idx2, i in enumerate(top):
        if j<len(top)+1:
            fig.add_trace(go.Histogram(x=shared_tracks[i],␣
 ↪name=numerical_features[idx2]),row=1,col=j)
            j+=1
    if d<len(bottom)+1:
        fig.add_trace(go.Histogram(x=shared_tracks[k], name =␣
 ↪numerical_features[idx]),row=2,col=d)
        d+=1
fig.update_layout(bargap=.2,width=1100,height=500, title_text='Shared Data')
fig.show()
```

[16]:

```
print(f'Number of shared tracks between datasets: {len(shared_tracks)}')
print('Statuses of songs in hot 100 billboard chart:')
print(shared_tracks['status'].value_counts())
print(f"The artist with the most songs in both datasets is: {max(artist_count,␣
 ↪key=artist_count.get)}")

from datetime import date
cum_days = 0
billboard_date = date(2021, 6, 6)
for dates in shared_tracks['release_date']:
    split_date = dates.split('-')
    track_date = date(int(split_date[0]), int(split_date[1]), int(split_date[2]))
    delta = billboard_date - track_date
    cum_days += delta.days
```

```python
avg_days = (cum_days / len(shared_tracks['release_date']))
print(f'Average number of days song is released before billboard of the week: {avg_days}')
```

```
Number of shared tracks between datasets: 39
Statuses of songs in hot 100 billboard chart:
no-change    34
new           3
re-entry      2
Name: status, dtype: int64
The artist with the most songs in both datasets is: Olivia Rodrigo
Average number of days song is released before billboard of the week:
163.7948717948718
```



From the small function, we were able to determine how many tracks were shared between the TikTok and the hot 100 dataset. There were 2 re-entries and 3 new tracks that made it into the chart for that week. I was pleasantly surprised when finding out 39/100 songs were featured in both datasets. Even though it is less than 50%, considering there are a plethora of songs and remixes used in TikTok, these songs were popular in the app. Due to some re-entries and no changes for songs that have been on the charts for months, it is hard to infer the importance of release date.

From the graph above, I was intrigued when analyzing the histograms, I expected that the tracks would have higher danceability and energy as dance challenges is a main trend in TikTok. The songs are also more acoustic which is also opposite of what I thought the popular tracks would be.

Conclusions drawn

A conclusion drawn is that even though TikTok tracks have higher audio feature values for some variables like danceability, energy, loudness which makes sense given the nature of TikTok, the hot 100 songs have lower values. I learned from this project that not everything can be analyzed with just data. External influence plays a role in determining the popularity of a song or artist. An example of this is the artist with the most songs that match in both datasets which is Olivia Rodrigo. She was an up and coming artist at that time who just released her first album with tracks ranging in different genres. Hence this skewed the billboard chart as her first single garnered a lot of success.

I do think however that TikTok still plays a role in the ability to make a song reach the hot 100 billboard. This is evident by the fact that songs that have been released quite a while ago, made a re-entry to the hot 100. This is coupled by the fact that the song became a trending track in TikTok. Overall, I was able to gain insight to the music that is popular in both TikTok and the hot 100 billboard chart.

There were some complications during the analysis, mainly with the inconsistency of the data. Identical songs had a possibility of containing different ids, which complicated the process tremendously because now I had to compare titles instead. Even the titles were inconsistent, for example the song *Peaches* by *Justin Bieber Featuring Daniel Caesar & Giveon* from the Spotify API is titled *Peaches (feat. Daniel Caesar & Giveon)* by *Justin Bieber*. These minor differences really increased the difficulty in comparing data between the datasets.

A change I made to the sources is adding a status column for the hot 100 billboard data which indicates whether a track has not left the chart, is new to the chart or a re-entry to the chart. In my opinion, the maintainability of this project is minimal because the data is ever changing and trends change overtime. One would need to keep constantly scraping TikTok and the billboard to be up to date. Also, it is hard to determine a timeframe on when and and how much to scrape, as the billboard chart only holds 100 songs that updates weekly while TikTok trends change much rapidly.

For the future, I would rather focus on a certain country or location to find local trends instead. I believe this will be more fruitful because the taste of people in the same area should be more representative rather than comparing with every individual globaly. Further analysis would include possibly collecting data on whether or not a majority of people liked a specific song. Having labels is useful as it will allow for supervised learning such as decision trees with the audio features. This could lead to being able to predict whether a song will be liked by people.