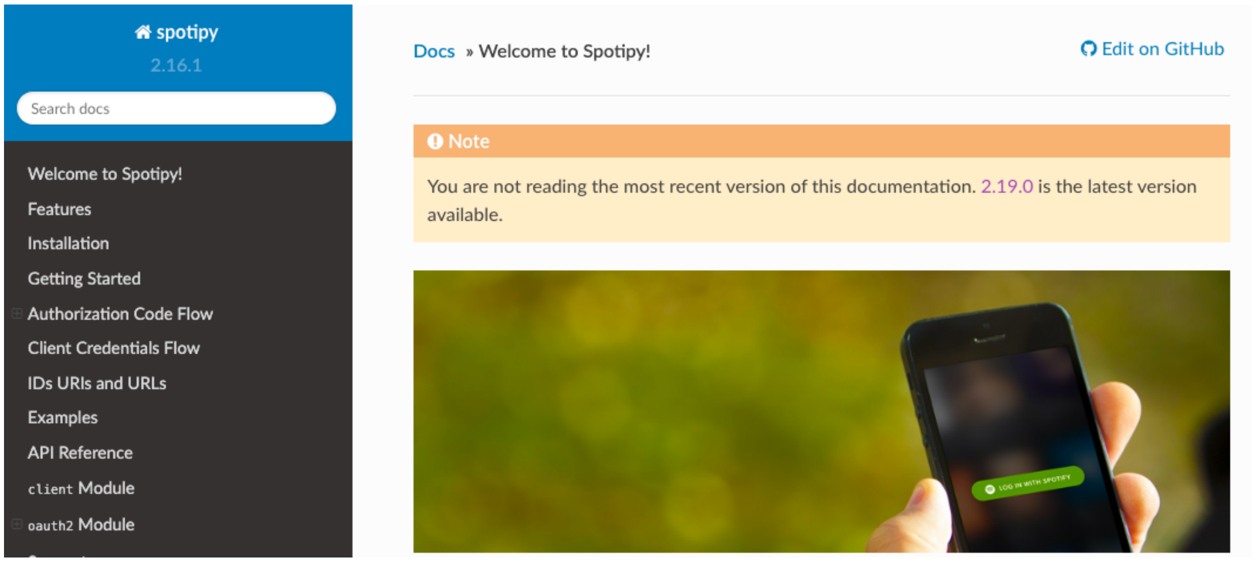
Data Access & Description

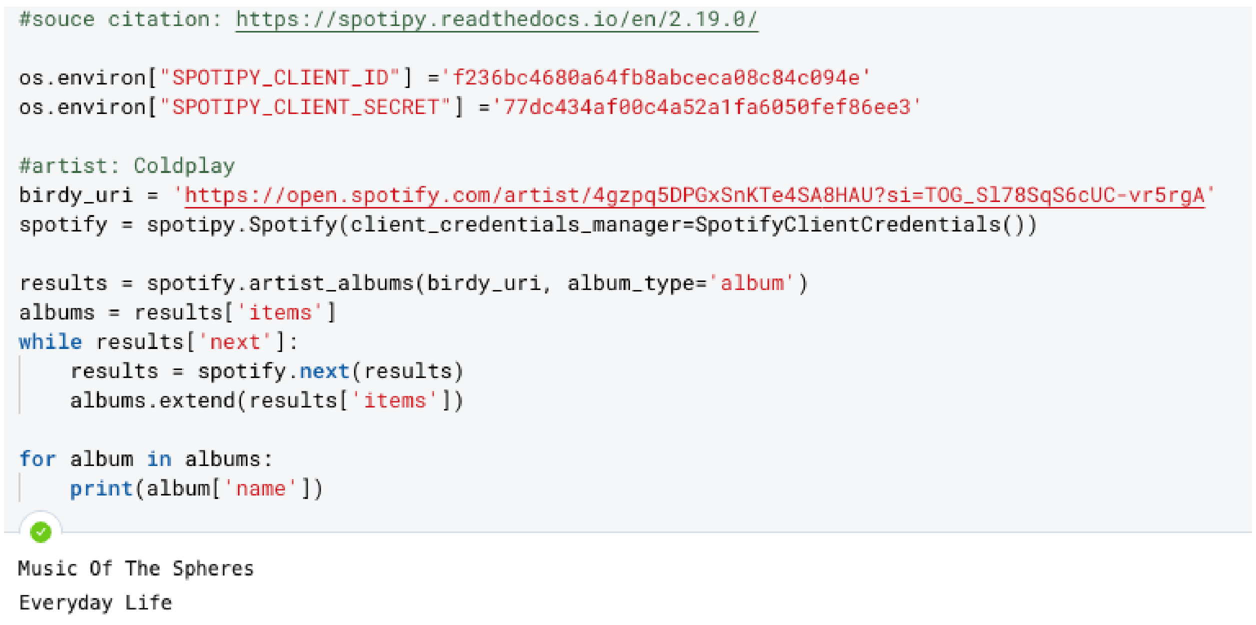


Note

You are not reading the most recent version of this documentation. 2.19.0 is the latest version available.

Spotipy is the "lightweight python library" for Spotify's Web API that gives developers full access to all of Spotify's music data. For the purposes of this project, the dataset consists of the following features: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence (positiveness), tempo, song_type, id, uri, track-href, analysis_url, duration_ms, time_signature, genre, album, popularity (popularity score), artist_name, and track_name. They contain raw data on the sound and musicality of each soundtrack, reducing any kind of bias and censoring of data. Note that popularity scores may contain bias based on the user who gave the rating.

In order to access Spotify's API or use its python library, we must create a Spotify Developer account, where we can generate the client ID and client secret. From there, the following python libraries must be downloaded: numpy, pandas, matplotlib, spotipy, and sklearn (cosine similarity and minimax scaler). A number of different ID types are supported on this platform. The Spotify URI (e.g. spotify:track:6rqhFgbbKwnb9MLmUQDhG6) enables access to locate any artist, album or track and the Spotify URL takes in a HTML link which allows us to open any soundtrack, album, playlist or any other resource under a Spotify client. Soundtracks are distinguished using their Spotify ID, which is a base-62 number found at the end of the Spotify URI. Let's take a look at the following example [Figure 3.1.1] where we extract all albums under artist *Coldplay*.



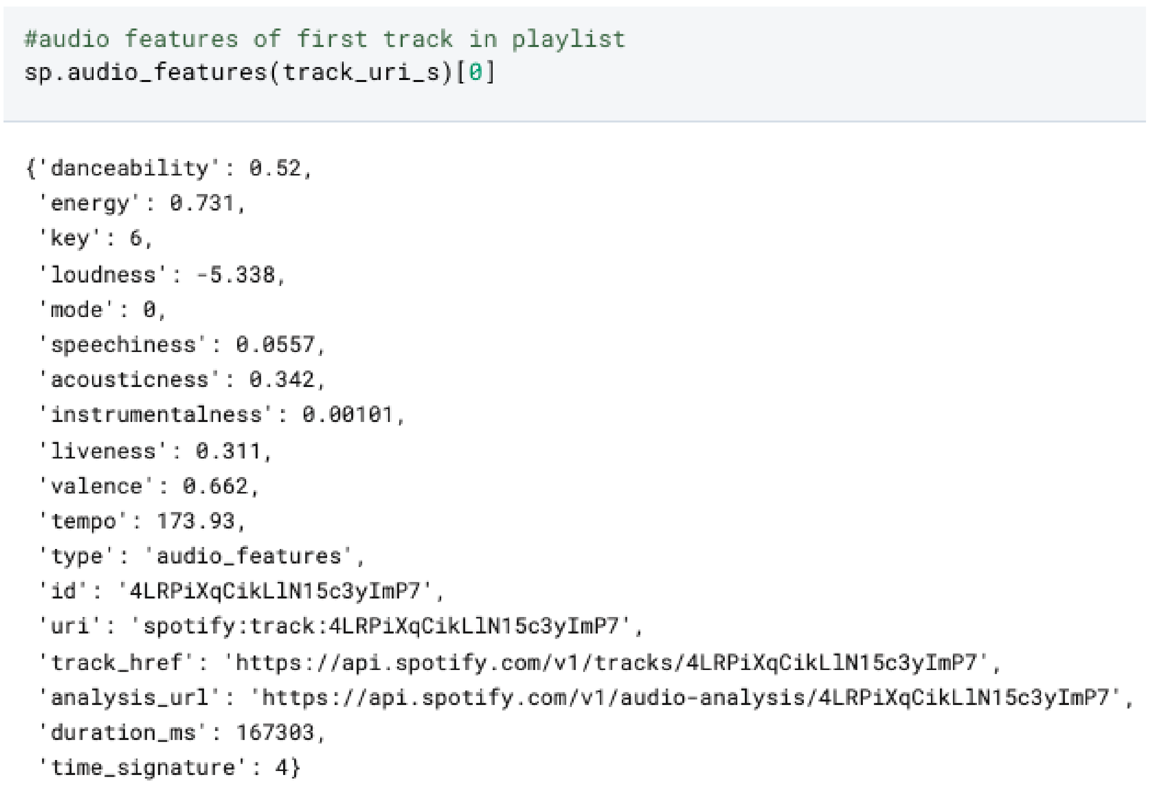
```
#souce citation: https://spotipy.readthedocs.io/en/2.19.0/

os.environ["SPOTIPY_CLIENT_ID"] ='f236bc4680a64fb8abceca08c84c094e'
os.environ["SPOTIPY_CLIENT_SECRET"] ='77dc434af00c4a52a1fa6050fef86ee3'

#artist: Coldplay
birdy_uri = 'https://open.spotify.com/artist/4gzpq5DPGxSnKTe4SA8HAU?si=TOG_S178SqS6cUC-vr5rgA'
spotify = spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials())

results = spotify.artist_albums(birdy_uri, album_type='album')
albums = results['items']
while results['next']:
    results = spotify.next(results)
    albums.extend(results['items'])

for album in albums:
    print(album['name'])
```

```
Music Of The Spheres
Everyday Life
```

Figure 3.1.1: Extracting albums under artist, *Coldplay*

First, we access the OS Environment using the client ID (i.e. `'f236bc4680a64fb8abceca08c84c094e'`) and client secret (i.e. `'77dc434af00c4a52a1fa6050fef86ee3'`) . In order to do so, we must create a Spotify Developer account to access both keys. Then, Spotify URI (i.e. `'https://open.spotify.com/artist/4gzpq5DPGxSnKTe4SA8HAU?si=TOG_S178SqS6cUC-vr5rgA'`), which can be extracted from the artist page on Spotify is used to access the artist page and return all albums under that artist, as shown above. Next, we iterate through the artist profile page to get all the album titles under that artist.

The first step for constructing the content-based recommender is to access Spotify playlist data for the top 50 songs streamed in the USA. This subset of data was chosen since it is constantly up-to-date with the current trends and popularity in the relevant geographic area. In order to extract all song attributes and features under a specific playlist, we must access the client ID, client secret, client credentials manager, and playlist URI from Spotify's Web API. While looping through the playlist, we get the title of each track, the album it belongs to and audio features are extracted along with genre and artist. Each track should have information on the following audio features: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, type of feature, id, uri, track href, analysis url, duration in milliseconds and the time signature. Each audio feature can be categorized into three categories: mood (i.e. danceability, valence, energy, tempo), property (i.e. loudness, speechiness, instrumentalness), and context (i.e. acouticness). Data on the popularity of the song is also extracted, where we expect all popularity scores to be quite high since our data is based on the top 50 most popular songs in the USA. Down below, Figure 3.1.2 shows example code to retrieve the attributes of the first soundtrack under the playlist, *Top 50 - USA*.



```python
#audio features of first track in playlist
sp.audio_features(track_uri_s)[0]
```

```
{'danceability': 0.52,
 'energy': 0.731,
 'key': 6,
 'loudness': -5.338,
 'mode': 0,
 'speechiness': 0.0557,
 'acousticness': 0.342,
 'instrumentalness': 0.00101,
 'liveness': 0.311,
 'valence': 0.662,
 'tempo': 173.93,
 'type': 'audio_features',
 'id': '4LRPiXqCikLlN15c3yImP7',
 'uri': 'spotify:track:4LRPiXqCikLlN15c3yImP7',
 'track_href': 'https://api.spotify.com/v1/tracks/4LRPiXqCikLlN15c3yImP7',
 'analysis_url': 'https://api.spotify.com/v1/audio-analysis/4LRPiXqCikLlN15c3yImP7',
 'duration_ms': 167303,
 'time_signature': 4}
```

Figure 3.1.2: Audio features of the first soundtrack under playlist *Top 50 - USA*

After extracting sound features for each soundtrack, all metrics are converted into a data frame for exploratory data analysis, data cleaning, and ultimately, building the recommender. Each track's URI, trackname, artist name, album name, genre, and popularity score are then appended to the dataframe to include other features that could be factored into the model as well.