API App Data Analysis

Group Members–Amina Karabek, Zuha Ansari, Danica Chen, Nathaniel Lee, Henry Tran, Panagioti Blanas, Isaiah Pacheco, Carlos Ayala

The Data Analysis Process:

1. Define the Objective–*What is the goal of the analysis?*

The goal of our project was to create a user app that gives you personalized movie recommendations based on your Spotify activity. Many adults listen to music nowadays, and there are plenty of apps that figure out what song to recommend, or what future playlist is similar to your current music taste. However, there isn't an app that connects the music on your playlist to a movie that could be of your potential liking.

BigMusic is an app that collects the data from your Spotify account, such as the singers you listen to or the songs that you play, and connect them with movies that other people who listen to the same music watch. The movies could be based on the mood of your playlist, the style, or simply the movies that other people who listen to your taste find enjoying as well in different films.

Music and movies are deeply related by a music's ability to enhance a film's storyline by setting the tone, shaping emotional responses, and creating memorable moments with the viewer and audience. The same way music can manipulate the audience's emotions, movies play a powerful role in this, and we want the audience to find movies that share the same memorable experiences as their music.

2. Collect the Data–*Gathering relevant information from various sources*

To build an app that recommends movies based on a user's Spotify listening activity, our team had to collect data from multiple APIs and combine them smoothly into one dataset. Because each of the APIs provide different types of information, our backend was designed in order to pull data from multiple different sources and merge them together into a consistent format.

From our Github structure, we can see that the backend/api/ folder contains all of our AP clients, which are the following:

- spotify_client.py – collects user music activity, like
    - Top artists
    - Top genres associated with the artists
    - Top tracks

    - Track the metadata (mood, energy. etc.)
- tastedive_client.py – collecting related movies that are based on music artist names and genres. TasteDive is used because it connects the similar interests, which is the foundation of our multiple–media recommendation cross service system.
- tmdb_client.py –pulls any details from the movies such as synopsis, popularity, genre list, release year, etc.) so that the recommendations are accurate and descriptive.
- omdb_client.py – provides additional movie data such as extra ratings or metadata when the TMDB information was missing.

How the data was collected:

1. Users will log into Spotify: The backend uses the Spotify API to retrieve their top artists and tracks.
2. The Spotify data is processed into usable attributes.
    a. Example: track name, artist name, genre(s), mood or audio features
3. The TasteDive API is used next.
   We pass the specific Spotify artist names into TasteDive, which will then return a list of recommended movies based on what similar users enjoy.
4. OMDB and TMDB API's fill in missing details/gaps.
   Once the list of potential movie titles are loaded,
    a. OMDB is used to cross check the data or fill in fields not provided by TMDB.
    b. TMDB provides the official title, poster, movie summary, rating, genre.

Final step 5: All the data collected is stored in a CSV.
From the GitHub repo, we can see that the recommend.csv file is located within the data/folder. This file contains the merged output of the APIs, making it easier to analyze, clean, and display them within our web templates.
The purpose of collecting multiple APIs is to ensure that we have a full picture of both the user's music taste and the movies that match those tastes accordingly.

3. Clean the Data–*Preparation of the data and the process of doing so*

After collecting all the raw data, the next step was to prepare and clean it so that the app could make accurate recommendations. From our GitHub structure, the data cleaning process mainly happens in the following files:

- models/movie_recs.py
- models/spotify_track.py
- services/csv_writer.py
- services/rec_service.py

Steps used to clean the data:

1. Removing any duplicate entries:

APIs such as TasteDive sometimes will return repeated movies–the backend filters duplicates before saving them into the CSV or before generating the complete final recommendation list.

2. Handling missing or incomplete data:
   Some APIs return partial results within the process, such as
   - Missing genre tags
   - Missing movie ratings
   - Empty descriptions
   - To fix this, our code will need to check the data and then call TMDB or OMDB to fill the gaps. If information is still missing, then the movie is skipped so that the users can see fully valid recommendations only.
3. Standardizing the data formatting:
   Each API returns different structures of JSON: for example, TMDB returns genres as ID numbers and Spotify genres come as lists.
   Our cleaning code converts these original formattings into a consistent structure before saving them into the CSV.
4. Converting API results into model objects:
   The files in /models (such as spotify_track.py and movie_recs.py) turn the raw API responses into clean objects with uniform attributes. This will make sure that every movie has the same data fields and every track has the same structure.
5. Writing the cleaned data into the CSV:
   The file csv_writer.py takes the cleaned objects and writes them into the recommend.csv file in a structured format.
   This will allow us to: debug problems, inspect the dataset, and resume the data for the recommendations page.
6. Filtering out the results for the final output:
   In the rec_service.py file, the data is filtered based on the following: genre or mood, accuracy of the API match, and relevance. This will remove any sort of irrelevant recommendations or low quality recommendations.

4. Analyze the Data–*Using statistical data or analytical data to explore the data. What data did we use?*

   We used the cleaned recommend.csv file generated by the API. I compared the song genres against the movie genres so we could see the basic correlation between what users listen to and the movies they get recommended.

5. Interpret the Results and Communicate the Findings–*Translate the analysis into meaningful insights, Infographic chart/image generated down below:*

This is a bar chart to visualize the relationship between the genre of the song and the genre of the movie recommendation. This chart shows how the music genres in our dataset match up with the types of movies the app recommends.