**Final Project Report**

**IST652 - Scripting for Data Analysis**

**Spotify Data Analysis**

**using Python**

**By:**

**Ankita Vartak**

**Dhrumil Parikh**

**Jay Maru**

**Nivesh Vaze**

**Content**

|  |  |  |
| --- | --- | --- |
| **1.** | **Introduction** |  |
| **2.** | **Data Description** |  |
| **3.** | **Part A: Spotify Playlist Data Analysis** |  |
|  | **3.1 Business Questions** |  |
| **4.** | **Part B: Streaming History Analysis** |  |
|  | **4.1 Data Pre-Processing** |  |
|  | **4.2 Merging the Datasets** |  |
|  | **4.3 Cleaning and Formatting** |  |
|  | **4.4 Exploratory Analysis & Visualization** |  |
|  | **4.5 Business Questions** |  |
| **5.** | **Conclusion** |  |
| **6.** | **Team Log** |  |

1. **Introduction**

In this project, we are going to explore my Spotify usage. Spotify is one of the most famous and popular music streaming platforms. The dataset used here showcases my personal usage of this platform. We downloaded this dataset from Spotify's Privacy Setting section, which allows anyone to download their personal usage data. With this dataset, we are trying to analyze users’ streaming history to gain some useful insights.

For this EDA Project, we are going to use several tools such as Jupyter Notebook, Python, and Python libraries such as NumPy, pandas, matplotlib, seaborn, and word cloud, Jovian.

Downloading the dataset:

### Instructions for downloading the dataset

1. Go to the Privacy Setting Page of your Spotify Account.
2. Scroll to the bottom and you'll see a section called Download Your Data.
3. You'll see a three-step process with instructions to download the data.
4. You must Request your data you'll get a confirmation email from Spotify to Confirm the request.
5. After collecting the required information, Spotify will create a Zip File and send you an email with the link to download it.
6. If you can't find the email, you can request it again from your Privacy Setting Page.
7. **Data Description**

|  |  |
| --- | --- |
| Title | Description |
| Name | Name of the song |
| Album | Name of the album |
| Artist | Name of the artist |
| Release Date | Released date of the song |
| Length | Count of the number of words in the song |
| Popularity | Popularity score of the song |
| Danceability | Danceability score of the song which lies between 0 & 1 |
| Acousticness | Acousticness score of the song which lies between 0 & 1 |
| Energy | Energy score of the song |
| Instrumentalness | This variable represents the number of vocals in the song. |
| Liveness | This variable determines the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. |
| Loudness | This variable determines the quality of the sound which is the psychological similarity of amplitude. In general, loudness is measured in dB. It has a range from -60 dB to 0 dB. |
| Speechiness | This variable detects the presence of spoken words in a track. |
| Tempo | This variable is used to determine the songs with the same beats per minute. Imagine you are running; Spotify’s application will detect our running stride in steps per minute and accordingly will suggest a song. |
| Time Signature | This variable is a notational convention used in Western musical notation to specify how many beats are to be contained in each bar and which note value is to be given one each beat. |

**Data Source**

We have collected the data from Spotify’s original website for Developers using Spotify API. Below is the Python code to extract the data:

Graphical user interface, text, application, Word

Description automatically generated

This is the snapshot for our Spotify account’s credentials which we collected and used it to extract the playlist data from the account.

1. **Part A: Spotify Playlist Data Analysis**

**Business Questions Answered:**

1. **Find the most popular song in the playlist?**

This question aimed at ranking the songs based on their popularity. The columns which we used here were the name of the song, artist name & popularity. Function idxmax() was used to find the song with the maximum popularity score.

A screenshot of a computer

Description automatically generated with medium confidence

1. **Determine the songs played which were released Pre-2010 & Post-2010 Era?**

The question helped us understand what the song preferences of the user were. So, based on the songs the user listens to, we were able to determine the likeability of the song. The column's release date was used to determine the question. Below is the snapshot:

The snapshot is for songs pre-2010 era

Background pattern

Description automatically generated

The below snapshot is for songs Post – the 2010 Era

Graphical user interface

Description automatically generated

1. **Determine the audio features which are causing the songs to be least played?**

The main aim was to determine the factors causing the songs to be least played based on their popularity of the song. we determined acoustics, danceability, tempo & loudness score of the song.

Graphical user interface, application

Description automatically generated

1. **Determine the most played artist’s song in the playlist?**

This question helped us understand the user's interest in a particular artist. count\_values() function was used on the artist column to determine the most played artist.

Graphical user interface, text, application, email

Description automatically generated

**Overall Description of the Python Program**

We divided the Python program into two parts based on the dataset.

First, we used Spotipy, pandas & time libraries to extract the data from Spotify API. We created a function and passed on the client\_id, client\_secret\_code to extract certain playlist’s data of the user.

Once the data was extracted and stored in a variable, we converted it into Pandas data frame. We dropped duplicate columns as a part of data validation checks. Once the data was cleaned and ready to use, we performed an analysis & exploration of the dataset to answer several business questions.

1. **Part B: Streaming History Data Analysis**

Once Spotify’s playlist data was analyzed, we worked on the search history dataset for further analysis & exploration. The data was in an unstructured format which we converted into a structured one using the pandas library.

* We have used a streaming data source that is in JSON format.
* Download the Streaming History from Spotify App by following below steps listed steps: -

1. Go to the Privacy Setting Page of your Spotify Account.
2. Scroll to the bottom and you'll see a section called Download Your Data.
3. You'll see a three-step process with instructions to download the data.
4. You must Request your data you'll get a confirmation email from Spotify to Confirm the request.
5. After collecting the required information, Spotify will create a Zip File and send you an email with the link to download it.
6. If you can't find the email, you can request it again from your Privacy Settings Page.

We are reading the files with the read\_json method and using it for analysis.

First, loading the data and then converting it into a CSV file.

* 1. **Data Pre-Processing**:

Steps to do:

1. To merge the two different Streaming History JSON Files.
2. To convert the endTime column into the proper Date-Time format column
3. To convert the msPlayed into the proper time column (Hours, Minutes)
4. To add more columns that can be useful for further analysis

Import Data:

We have used the read\_json function from the Pandas module to read the JSON file.

Table

Description automatically generated

Table

Description automatically generated

* 1. **Merging the Datasets:**

Here we merged both the datasets to make a bigger data set by using concatenating function.

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Here, we are checking the data features and types by using the “info” method. With “nunique” function we can count the number of unique elements we have in the different columns of the dataset

Text, table

Description automatically generated

As we added the two different data frames with the concat method, we have also given an additional attribute called ignore\_index as True.

This helps to reset the index and make a new one, as both of the data frames have their own individual indexes.

### 4.3 Cleaning and Formatting:

### We'll use the pd.to\_datetime method from the Pandas module to convert an object column into a proper date-time column.

### We'll also use pd.DatetimeIndex method from Pandas module to obtain several unique information from date-time columns like a year, month, day, weekday, time, hour, day\_name.

### Graphical user interface, text, application, email Description automatically generated

### Here we used the to\_datetime module to obtain the proper date-time format and then we added that into a new column called Play-Time.

### Here we used the DatetimeIndex module to obtain several unique information. We also added them to a new column as per their representation and information.

### Information contained in different columns:

### year has information about the year when we played that song

### month has information about the month when we played that song

### weekday has information about the weekday when we played that song

### time has information about the time when we played that song

### hours have information about the hour when we played that song

### day-name has information about the day with the name when we played that song. Here we used to apply the method for pandas data frame with a lambda function.

### We also added a new column called count to keep track of the count of the songs.

### Here we used the to\_timedelta module to obtain the information regarding the time frame of the song played in milli-seconds.

### We also converted the information given in milliseconds into the proper format and added it into another column called Time-Played (hh-mm-ss)

### Here we made two separate functions called hours and minutes which will help us to convert the time we got in Time-Played (hh-mm-ss) to convert in approximate minutes and hours.

### We used the apply method to apply those two functions along with the round method here to do the calculation and approximation.

### We can see that now we have a lot of columns, some of which are not useful anymore, so we'll drop a few of them.

### We are dropping the following columns:

### endTime as we got a new column called Playtime which has all information with proper format.

### msPlayed as we got the Time-Played (hh-mm-ss) in a proper format.

### Time-Played (hh-mm-ss) as we got the approx time in minutes and hours in Listening Time(Hours) and Listening Time(Minutes).

### Table Description automatically generated

### 4.4 Exploratory Analysis & Visualization:

Chart, pie chart

Description automatically generated

1. The unique artist percentage comes around 21.83 %.
2. We can also confirm this through the pie chart we made.

Chart, bar chart, histogram

Description automatically generated

Here we made a bar chart to show the same list in better visualization. We gave two-axis x & y as the attributes and color for better visual. We used the set method to give title, xlabel, and ylabel. By This chart, we can also see how close these artists are in terms of playtime in hours.

Chart, bar chart

Description automatically generated

Here we made a bar chart to show the same list in better visualization. By this chart, we can also see how close these artists are in terms of playtime in counts.

This pie chart shows that generally, I streamed my Spotify more on Friday.

Chart, pie chart

Description automatically generated

With this histogram graph we can see my average usage:

Maximum around 4-5 PM hour mark

Minimum around 12-1 AM hour mark

We used a seaborn-based plot module to make a histogram. Seaborn is based on Matplotlib Library only, but it offers better visualization options.

Chart, histogram

Description automatically generated

Chart, bar chart

Description automatically generated

We plotted this count plot which comes with a seaborn module to show my usage of Spotify over years.

We can see that there's a big gap between the 4th Month and 7th month. This is because I started using Spotify in July 2020 (the 7th Month) and this streaming history is up to April 2021 (the 4th Month) only.

Hence there are no data regarding the month which lies between the 4th and 7th. But we can also notice that my Spotify Usage in April is way higher than in other months. This is new information for me.

### 4.5 Business Questions:

**Q1A: How many hours did I spend on Spotify Streaming since the day I signed up for it?**

A picture containing graphical user interface

Description automatically generated

For this, we can simply do a summation of all the time I spent listening to all songs which come out to be around **369 Hours**

**Q1B: What is actual usage in percentage compared to the total possible?**

Graphical user interface, text

Description automatically generated

First, we subtracted the time, from the start time (when I logged in to use my Spotify for the first time) to the end time (The last time I used Spotify as per the dataset.

Then we divided that with np.timedelta64 function to convert and calculate the time into days.

We again multiplied it by 24 to convert it into hours which comes out to be around 6688.5 hours.

Graphical user interface, text, application, email

Description automatically generated

Chart, pie chart

Description automatically generated

**Q1C: What are the average numbers of songs I played daily?**

Graphical user interface, text, application, email

Description automatically generated

Here we can see that on an average I played 39 songs per day.

**Q1D: On which day do I play a maximum number of songs? Plot a scatter plot to show all the dates**

Graphical user interface, text, application, email

Description automatically generated

**Q2A: Who are my favorite 100 artists? Make a word cloud to show them.**

Table

Description automatically generated with medium confidence

Text

Description automatically generated

Here we used the word cloud function from word cloud module which allows us to make a word cloud of the given list or series. We can additionally give other attributes like:

1. max\_words to fix the number of words to display
2. width and height for the size.
3. normalize plurals for removing additional 's from the words and more

We also used the generate\_from\_frequencies function to generate the counted words with the given frequencies in the data frame.

**Q2B: Make a My Favorite Artist playlist for the user, based on his daily usage.**

Like before, here we simply made a data frame with our favorite artists and their songs. This was made with the help of group by function but this time we gave two columns to do the categorical grouping of the data. Then collectively we made a list based on the counts**.**

Graphical user interface, application, Word

Description automatically generated

**Q3: What is my favorite 100 Songs? Make a word cloud to show them**

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

**Q4: Most Usage (Active usage in a day over a week)?**

Table

Description automatically generated with low confidence

Here first we made a new data frame with a group like before but then we made a pivoted table with the help of the pivot function. This gave us a comprehensive chart with days of the week and time of day and the number of songs I played at that time/day.

**Q5A: How my usage varies over a week? Make a count plot to showQ5B: What is the percentage of usage distribution between weekdays and Weekends?**

Chart, bar chart

Description automatically generated

**Q5B: What is the percentage of usage distribution between Weekday and Weekend?**

Graphical user interface, text, application, email

Description automatically generated

**5. Conclusions:**

**Part A:**

1. The most popular song from the dataset we analyzed is **Believer**, which has the artist’s name “**Imagine Dragons**”, and a popularity score of **85.**
2. The number of songs which has a release date after 2010 is more than the number of songs released before 2010.
3. There are a total of 24 songs that are not at all popular and have a popularity score below 50.
4. The most played artist is “**Various Artists**” who has 18 songs under his belt. The most played artist’s song on an individual basis is **Pritam**

**Part B:**

We've got many useful insights from this dataset. Here's a few of them:

* I have played songs of 2359 Unique Artists which comprises 21.83 % of total artists in my streaming history.
* I have played 4068 Unique Songs which comprise 37.65 % of the total songs in my streaming history.
* Of all the days, I'm most active on Spotify on Fridays.
* My most active hours range from 8 AM to 7 PM. Maximum around 4 PM-5 PM hour period and Minimum at12AM-1 PM hour period.
* I played 39 songs on average per day.
* On 21st April 2021, I played 390 Songs which is my personal maximum.
* Since the day I created my Spotify Account, I played the most songs in April 2021.
* I have spent almost 369 hours streaming on Spotify till now. That's about 5.5% of the possible 6688.5 hours
* My favorite artist whom I've played most is Lavu. While my favorite song is **bitches broken hearts** by Billie Eilish.