Capstone Project - Music Data Analysis - Milestone Report

By
Harshavardhan Nilakantan

Guided By
AJ Sanchez

1. The Problem:

Many different genres of music exist, differing over numerous aspects, such as speed, time signatures, complexities around constructs of musical theory etc. Musical tastes also vary greatly from person to person and often the descriptions of these forms of music are highly subjective. But still within this subjective sphere we still know and can discern enough differences that we can categorize these forms of music into various genres, based on stylistic or structural differences between them. This is one way of looking at it.

We could also examine this in a different manner. There are certain quantifiable characteristics that exist for each song, and they may potentially share some kind of grouping/classification based on some or a combination of these characteristics. These include popularity, time signatures, genre, instrumentalness, danceability etc. to name a few. This is exactly what we will examine in our analyses. In addition to exploratory analysis, we will apply a number of clustering algorithms. We aim to find underlying structures in the data, and see if they cluster in any meaningful pattern using only these characteristics.

We have a dataset in hand, obtained from Spotify, with these characteristics quantified for each song. We move forward with the assumption that these measurements, made by Spotify, are accurate. The popularity measure is one based on the number of “listens” on that track over a period of time. We will move forward with this knowledge as well - that our analyses using these measurements apply to that frozen period of time for which this popularity measure is based.

Then we would potentially use a small test data set of a specific genre of music to see if it follows any of the previously found clusters.

2. The Client:

This project targets clients interested in understanding what kind of properties tunes need to possess in order to be considered popular. These clients might also be interested in questions about why some forms of music have low popularity or are not considered to be “mainstream”. For example, the more popular genres could have a simpler time signature or be more danceable, less instrumental etc.

This could also potentially helpful to someone looking to start a career in music. They may need to parallely pursue more lucrative trades if they are interested in creating music that does not have the combination of characteristics that would make it more popular.

3. Data Wrangling:

Our primary dataset is from Kaggle. It contains 131580 rows, one for each track, and 20 columns per track. The columns are:

1. Name
2. Danceability
3. Energy
4. Key
5. Loudness
6. Mode
7. Speechness
8. Acousticness
9. Instrumentalness
10. Liveness
11. Valence
12. Tempo
13. Type
14. ID
15. URI
16. Track Reference
17. URL Features
18. Duration (in milliseconds)
19. Time Signature
20. Genre

Columns 14-17 are Spotify internal data, such as the internal track ID, URI etc. Apart from the ID, we do not need the other columns. We need the ID (14) to pull more data from Spotify using the web API library - “SpotiPy”. This will be described in detail in the next section (3a).

Notice that this dataset does not contain the “popularity” metric that we plan to use. This metric and the artist name data are available in a different track object, that separately needs to be downloaded and appended to our dataset.

For this we would need to connect to Spotify and query the appropriate API endpoint using the track ID mentioned above, for the object that we need, and extract the relevant data from it. Once this is done, we’ll need to append it to each track.

So we first proceed to clean up the dataset, so that we only download for the tracks that remain, and avoid unnecessary extra downloads, for tracks that will be eliminated.

Descriptions of both - the clean up and the Spotify connection/download can also be found in the Jupyter notebook itself.

3a. Data Clean Up:

After some of the initial imports, the Kaggle dataset was read and imported into a data frame, **df**. This was followed by some preliminary examinations of the data such as the **df.head()**, **df.shape()**, **df.columns** etc. The next step was to set the index to be the track IDs, and assign this to the data frame **df_cleaned**.

I then proceeded to remove the unnecessary columns using **df_cleaned.drop()** function on the columns "**Type**", "**Uri**", "**Ref_track**", "**URL_features**", specifying the **axis**=**1,** to ensure columns are removed, and not rows. We assign this back to **df_cleaned**.

I looked for nulls using the chained function **.isnull()**. Indexing the **df_cleaned.columns** with these nulls showed us in which columns the nulls existed. There were **2 nulls** in the **Name** column, and **26 nulls** in the **genre** column. These obviously have to be dropped since they cannot be backfilled or interpolated. These were then dropped using the **df_cleaned.dropna()** function. Then I verified if the nulls were dropped or if any were missed.

This dataset was combined off of multiple playlists and appended together, as a result it has many songs that were repeated through the dataset. The next step was to check for and eliminate duplicates using the **df_cleaned.drop_duplicates()** function. This eliminated close to 10,000 rows. Our data now stood at **121815 rows** with **15 columns**, cleaned of duplicates and nulls. We could now proceed with the download of the supplemental data.

3b. Spotify Web Connection & Download:

Using Spotify's Web API entails the following:

1. Set up Test app in Spotify web API
2. Set up credentials in local machine
3. Set up a test connections to download data
4. Authentication issue - token expiry
5. Steps to overcome authentication issue
6. Download supplemental data for all tracks
7. Append to dataset

To connect with Spotify, I used the SpotiPy library, which provides easy methodic access to Spotify and the various track objects that it supplies. There are two parts to this. The first involves work to be done from the Spotify side and changes to be made on your local machine. The second involves the code and usage of the SpotiPy library to build a client capable of authenticating seamlessly and handling the download.

Spotify and System Work:
1. Create an account and register your test application.
2. Set up your Client ID and Secret, and set them up as environment variables in your local machine, from where you will run your application. There are other ways to

dynamically use your Client ID and Secret without setting them as environment variables. Read the documentation of the Spotify Web API and the Spotipy client to explore other options.
3. Cue the specific imports necessary to build a client, connect to your test app and authenticate. Then call the desired APIs to download the specific data that you need.

Code Work with Method 1 (Implicit):
1. Get username/login from os object.
2. Use the SpotiPy util to use credentials set in environment variables for the current user. This prompt will return a token. **The received token cannot be refreshed manually**.
3. Pass this token to the SpotiPy object while creating a Spotify object. The authentication should work and allow your client to pull data artifacts from Spotify.

Code Work with Method 2 (Explicit):
1. Get username/login from os object.
2. Explicitly specify the keys for the Client ID, Secret and URI, set as environment variables and retrieve them.
3. Set up a Spotify OAuth object explicitly specifying the ID, Secret and the URI. This will return a wrapper Token Info object. The actual token can be extracted from this object. **This token can be manually refreshed** by refreshing the wrapper Token Info object.
4. Pass this token to the SpotiPy object while creating a Spotify object. The authentication should work and allow your client to pull data artifacts from Spotify.

The received token in method 1 is only valid for a limited duration and will expire after that, in which case the connection gets terminated. So this approach is not good for prolonged use (as in our case, downloading 121K records worth of data, which will take at least a few hours). So, I wrote a function to refresh the token manually, which I would call every N downloads, controlled by a counter.

This was followed by iterating over the track IDs and downloading the track-specific data. And refreshing the token every time the counter reached its max, which in our case was every 500 downloads.

So, I proceeded with method 2 and the data was downloaded without incident. The token was refreshed periodically, to prevent expiry.

4. Other Potential Data Sets:

There are numerous similar datasets available on Kaggle. Any one of them could be used. They would also probably require supplemental data for which the connection protocols, described above, would be useful.

We could also manually create and extract such a dataset by creating playlists. This would be a highly manual-intensive and time-consuming process, but it is possible.

If the analysis with our current dataset shows promise, we could explore further with a limited dataset belonging to a specific genre which we could examine to understand how it clusters, and if it behaves as one would expect, given the genre specificity.

5. EDA I - Data Storytelling:

The idea here is to start exploring the dataset and answer some simple questions to see if we can find any initial patterns or trends, and if it is possible to draw some observations that could potentially be further explored.

We start with reimporting the data to see if there are any exceptions or nulls coming in. Surprisingly, there were a few nulls in the "Artists" column, either possibly due to a corruption during the file write in the previous notebook, or an error during the import. There were a total of 7 records, and they were eliminated.

The data was explored w.r.t many different characteristics such as **"Genre"**, **"Danceability"**, **"Instrumentalness"**, **"time_signature"**, **"Tempo"** and **"Popularity"**, and a concerted effect of some of these characteristics together.

Our analyses yielded the following salient observations:

1. Over 121808 songs, there are 625 unique genres in this data set and they separate quite well in the number of songs that each genre has.
2. We explored a number of other characteristics such as ***Danceability***, ***Instrumentalness***, ***Time Signature***, ***Tempo*** etc. and also further explored how some of these vary with ***Genre*** and ***Popularity***.
3. For ***Danceability***, we saw that 75% of the data was contained under 0.7 and while working with the mean of this metric across *Genre*, we saw that a good 70 genres were "highly danceable" (>0.7)
4. For ***Instrumentalness***, after we worked with the genre means we saw that there is a population that varied in popularity from 0-60 but was overall low in instrumentalness. And we had 101 highly instrumental genres. We also varied this with popularity. We saw that the most instrumental genre had very low popularity - ***earlyromanticera***. And the most popular of the highly instrumental genres was ***meditation***.
5. For ***Time Signatures***, we saw that there were a good ***13101 songs*** with non-standard time signatures (not 4.0). These did not separate well on genres, and included ***585*** out of the ***625 genres*** in the **13101 songs**. Of these songs, ***3.0*** and ***5.0*** were the most frequently occurring and also had ***similar distributions, medians and IQR***, as evidenced by the violin plot.
6. For ***Tempo***, the data was spread roughly between 40 and 200, with some outliers. It did not vary distinctly with popularity.
7. ***Danceability*** and ***Energy*** did not vary distinctly.
8. The scatter of ***Speechness*** and ***Instrumentalness***, shows a distinct population which ranges from ***0.0 to 0.4*** in ***Speechness*** while varying from ***0.0 through 1.0*** in ***Instrumentalness***. This seems to be the majority of the data. So it doesn't vary distinctly with instrumentalness.
9. The metric of ***Popularity*** by itself is spread across from 0-100, 75% of the data falls under 43. And it doesn't seem to be spread across any specific genre distinctly. Some

of the lowest genres include ***celticmetal***, which shows values of ***0***, implying that it was not listened to at all during that week.

6. EDA II - Inferential Statistics:

This portion builds off of where the data storytelling work ends. We'll carry forward from the salient observations in the storytelling using statistical analyses techniques such as hypothesis testing, t-tests, test for correlations etc.

Looking through the summary of our observations above, it seems that points 3-5 need further analysis. For all our analyses, the threshold/$\boldsymbol{\alpha}$ = 0.05.
The analyses were broken down in the following ways:

Understanding the Relationship between Highly Danceable Tracks and Popularity:

In the initial EDA, it was seen that **75%** of the data was contained under the **0.7** value of our **Danceability** metric, and this allowed to filter out **70 genres** that went above this value, that we could consider **"highly danceable"**. So, the natural next step was to see if these highly danceable tunes were correlated with popularity overall.

Approach: Hypothesis testing using Pearson R Correlation.

Our Hypotheses:

H_0 (Null): Danceability and **Popularity** are independent for highly danceable music.

H_A (Alternative): A dependency exists between **Danceability** and **Popularity** for highly danceable music.

First, the "highly danceable" tracks (Danceability >= 0.7) were isolated and then grouped by the genre, grouping to the **mean** of each genre, and finally sorted by danceability.
A scatter plot with a line fit, showed a loose trend that as Danceability increases, Popularity also increases.

The **Danceability** and **Popularity** means for these highly danceable genres was then subject to the **pearsonr test**, which produced a **very low p-value**, allowing to **reject H_0 and accept H_A**. Through these means we have now established that for highly danceable music - Danceability and Popularity are indeed correlated.

Understanding the Relationship Between Track Instrumentalness and Popularity:

As per the initial EDA, it was found that the **"highly instrumental"** genres had an **Instrumentalness** >= 0.54, with 101 genres falling in this category with reasonable to high popularity. So, our next step was to understand if highly instrumental music seemed more popular or if there is any correlation at all.

Approach: Hypothesis testing using Pearson R Correlation.

Our Hypotheses:

H_0 (Null): Instrumentalness and **Popularity** are independent for highly instrumental music.

H_A (Alternative): Instrumentalness and **Popularity** have a mutual dependency for highly instrumental music.

Similar to the previous approach for **Danceability**, the data was isolated for **Instrumentalness** and **Popularity** as well. A similar scatter plot with fit line, showed that as **Instrumentalness** increases, **Popularity** decreased.

This was then tested using the **pearsonr** test, using the means for **Instrumentalness** and **Popularity,** which also produced a very low **p-value**, thereby allowing us to reject H_0, and accept our H_A. We have now established that Instrumentalness and Popularity are anticorrelated.

Understanding Popularity Differences Between Non-Standard Time Signatures 3.0 and 5.0:

For the time signatures, from the initial EDA, we concluded that we found a portion of the data which had non-standard time signatures, i.e., not 4.0. The key among them being 1.0, 3.0 and 5.0 (ignoring the 0.0 - as this might likely have been a Spotify-introduced data error). Through the violin plots it was seen that the distributions showed increasing popularity from 1.0 to 5.0. And the distributions of the 3.0 and 5.0 were very similar, with a very large group of tracks belonging to 3.0. The non-standard time signatures were distributed among 585 genres, so they did not separate well on genres. The next step was to evaluate the differences in these subgroups.

First, we were to evaluate the 3.0 and 5.0 subgroups, which may be distributed very similar to each other. This was tested through the difference in the means of their popularities.

Approach: two-sample independent t-test, since we have no way of knowing the population means or standard deviations.

Our Hypotheses:

H_0 (Null): 3.0 and 5.0 time signature tracks do not differ in **Popularity** means, probably come from the same distribution.

H_A (Alternative): 3.0 and 5.0 time signature tracks do differ in **Popularity** means, and probably do not come from the same distribution.

The 3.0 and 5.0 time signature tracks were isolated and then, subject to the two sample independent t-test, which produced a **p-value larger than 0.05**, thereby allowing us to accept the **Null Hypothesis**. We can therefore, conclude that the two groups do not differ in their Popularity means and probably come from the same or a similar distribution.

Understanding Popularity Differences Between Non-Standard Time Signatures 1.0 and 5.0:

Following through from the previous analysis, we were to evaluate the 1.0 and 5.0 groups. This was tested again using a two sample independent t-test.

Approach: two-sample independent t-test, since we have no way of knowing the population means or standard deviations.

Our Hypotheses:

H_0 (Null): 1.0 and 5.0 time signature tracks do not differ in **Popularity** means, probably come from the same distribution.

H_A (Alternative): 1.0 and 5.0 time signature tracks do differ in **Popularity** means, and probably do not come from the same distribution.

The test concluded with a very low **p-value (< 0.05)**, thereby allowing us to **reject the Null Hypothesis** and **accept our Alternate Hypothesis**.

The two groups did indeed differ in the means of their Popularities and possibly came from different distributions, as predicted earlier by our plots.

Conclusions:
We found that the Danceability and Instrumentalness metrics were in some way correlated with Popularity measures. The time signatures for a small subgroup of the data (3.0 and 5.0) were found to be correlated to Popularity as well.
This is important as we build our model, because these might be important metrics that characterize our clusters.

7. The Code So Far:

All the code and notebooks for the analyses and steps carried out above can be found in GitHub. Additionally they contain detailed notes and comments to guide along the descriptions given in each section above. They have been reviewed and accepted by my mentor.

All the Capstone work can be found in this repository. All notebooks are within the same directory.

The Data Wrangling notebook is titled - Capstone Project - Data Wrangling.ipynb.

The EDA I - Data Storytelling notebook is titled - Capstone Project - Data Storytelling.ipynb.

The EDA II - Inferential Statistics notebook is titled - Capstone Project - Inferential Statistics.ipynb.

8. Next Steps:

We will build our unsupervised learning model. To do so, we will do the following:

1. We will use K-Means Clustering
2. We will find an optimal K to use
3. Perform the clustering with optimal K
4. Visualize the cluster
5. Analyze and characterize the clusters