Examining how taste in music differs in different regions and how we can use that to predict popularity of songs and artists over a specific time frame

Gleb Sokolovski

April 11, 2022

1 Overview

Spotify is a $27 billion platform that's famous for listening and discovering music. It records what their customers' preferences are, tracks positions of popular songs and analyzes attributes of each song to provide the best possible music suggestions by using a lot of algorithms.

This presents data scientists with a lot of opportunities. There is a lot that can be done with this data, but this report focuses mainly on tracking popularity of songs over a 374 day period in 54 different regions. By the use of visualisations and statistical modeling the paper reveals which genres are popular in which regions over that period of time.

The report concludes that there is a clear divide in genre preference and as extra motivation for application in real world - this could help song producers to increase their listeners by focusing on marketing in certain regions depending which category their song falls under.

2 Introduction

Context and motivation What is the area of this data science study, and why is it interesting to investigate?

Spotify is a world-wide music streaming platform. It was launched in 2008 but wasn't very popular until 2013 [1]. Since then, it grew to being the world's most popular audio streaming subscription service - having 82 million tracks and 406 million users [2]. But it's not just music anymore - there's podcasts, audio books, adverts for upcoming shows, etc. So no wonder every music producer and any sort of entertainment creator is trying to get their content in there. That's their easy way to success - being recommended to users on Spotify.

Having that in mind, the motivation for this report is to analyse the big set of data provided to us and make our own conclusions.

Previous work Brief description of any previous work in this area (e.g., in the media, or scientific literature or blogs).

Due to the magnitude of the data, Spotify attracts a lot of data analysis projects.

The article by Dea Bardhoshi from 2021 [3] looks at how different attributes of songs compare and correlate, tying it all to the popularity of songs. Bardhoshi concluded that there is a lot of differences between popular and non-popular songs.

Another interesting work by Cristóbal Veas conducted in 2020 [4] uses clustering to seperate songs into different playlists according to their "mood".

Objectives What questions are you setting out to answer?

The main objective of this project is to find which regions share the same taste in music and therefore make predictions how popular tracks will be in certain regions given their popularity in other regions over time.

3 Data

Data provenance Who created the dataset(s)? How you have obtained it (e.g., file or web scraping), and do the T&Cs allow you to use obtain the data for the project?

There were two main data sets used in this project.

The first one (reffered to as dataset A from now on) was created by Eduardo [5] 4 years ago, named "Spotify's Worldwide Daily Song Ranking". There was no need to download the data (even though that was an option). Instead it's hosted on the university website and python reads directly from there.

The second data set (reffered to as dataset B from now on) was supplied by Spotify themselves. **Spotipy** is a Python library for the Spotify Web API. I needed to create a Spotify developer account in order to get the client id and the client secret id which were necessary to be authorised to use the data from the database. All I had to do after that was run a terminal command, which downloaded the library which in turn uses web scraping to access the data.

Data description Description of the data, e.g. variables in each table, number of records.

Dataset A consists of top 200 ranked songs for every day ranging from 01/01/2017 to 09/01/2018 (374 days) in 53 countries (also "global" is added to the region list making that 54). So that makes over 4 million entries. Each entry contains the three columns just mentioned (position, date and region) and also the track name, the track's artist, number of streams, and the URL to the song - in total 7 columns.

Dataset B is a bit different. **Spotipy** in general contains so many different functions which return different datasets. The one I used was **audio_features()**, which takes in the track's URL and returns a lot of different values (as a dictionary) but I focused on the following (all are a float number): dancability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, and tempo. Figure 1 helps to visualise the data when each attribute is compared to the other ones.

Data processing How you have processed the dataset, e.g., cleaning, removing missing values, joining tables.

Dataset A was cleaned and all the rows with missing values were removed in data_cleaning.py. It removed 657 rows, which - out of over 4 million rows - was not deemed statistically significant.

Dataset B did not require much cleaning but while parsing, I did have to throw a AttributeError exception. I recorded the URLs for which exception was caught and fortunately there were only 3:

- https://open.spotify.com/track/2DEYFawpGha5Zn54Fx6dX5
- https://open.spotify.com/track/4IjwENtDPy4KqpnU9iBBca
- https://open.spotify.com/track/3uMIfxV8q5tPTWzJh8s6Mk

Once again, 3 out of 21,735 values is not statistically significant.

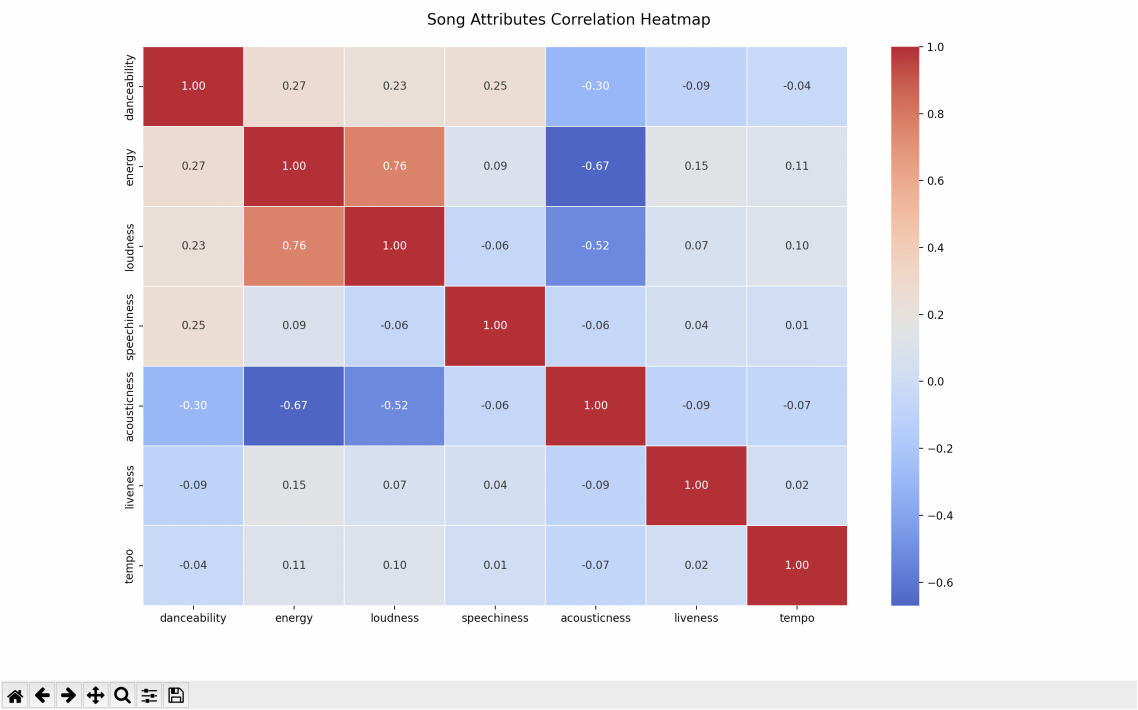

Figure 1: Heatmap of the song attributes from Dataset B in correlation to each other

The datasets were merged on URL, however not straight away - only after URLs have been Hierarchically clustered with the data from Dataset B - and the cluster to which each URL was added - was recorded as a seperate column in the Dataset B (I talk about this more extensively in the next section). Only after that were Dataset A and Dataset B merged.

4 Exploration and analysis

Data samples to run tests on As I've said before, there is a lot of data used in this project so I couldn't just sit back and wait for all the data to be recalculated every time I ran my code, as that took about 36 minutes for 21,735 unique values of URLs. So to make it more efficient, yet still fair, I used random samples to represent the entire population. I've realised after a while that it doesn't matter how big the sample is, the shapes of graphs will be very similar (sufficient tests were ran to prove the validity of this statement). So for Figure 2 I used half the population, but for Figure 3, I used 10,000 URLs, and for Figure 4, I used 1000.

Data processing and Analysis I started off by plotting histograms for my data to see the shape of it and decide how I'm going to go about dividing it according to attributes. The data was plotted in Figure 2. After deciding my data is suitable for it, I've decided to use Hierarchical clustering for unsupervised machine learning. I chose that over any other method, because I didn't know how many clusters I had to divide my data into.

After reading up on it from what we have covered and from the internet [6], I have realised it was very similar to K-Means clustering, which we have deeply covered in lectures.

First step was to plot the dendogram, which can be seen in Figure 3 on the left. After that we had to draw a horizontal line through the longest vertical distance with no horizontal line passing through it (in our case the blue one). This line (red) can be seen in Figure 3 on the right. Theoretically speaking the horizontal line we draw will define the minimum distance required to be a separate cluster. I.e. the

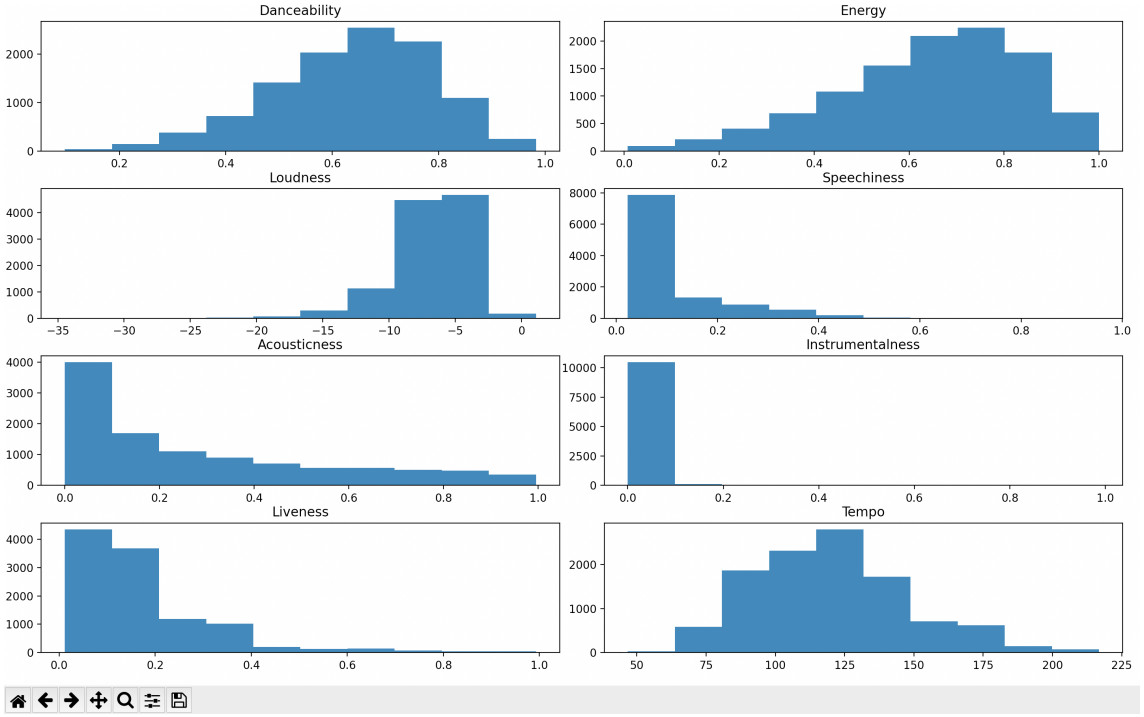

Figure 2: Histogram plot of attributes of tracks

amount of vertical lines this horizontal line passes is the amount of clusters - which we can clearly see is 3 in our case.

We run the clustering algorithm by running the **AgglomerativeClustering()** command, with input parameters: *n_clusters*=3, *affinity*='euclidean', and *linkage*='ward'. The function outputted the cluster labels. I wanted to make a visual of the clustered data, however as you can imagine it's hard to visualise 8-dimensional data, so the closest I've come to was to pair the attributes up and plot them against each other, as can be seen in 4. The only reasonable-looking one was probably the last one "liveness vs tempo".

As was mentioned in the "Data" section, I then added "cluster_group" as a separate column to Dataset B, which took 1 of 3 values: 0, 1, or 2 - representing which cluster the data point was placed.

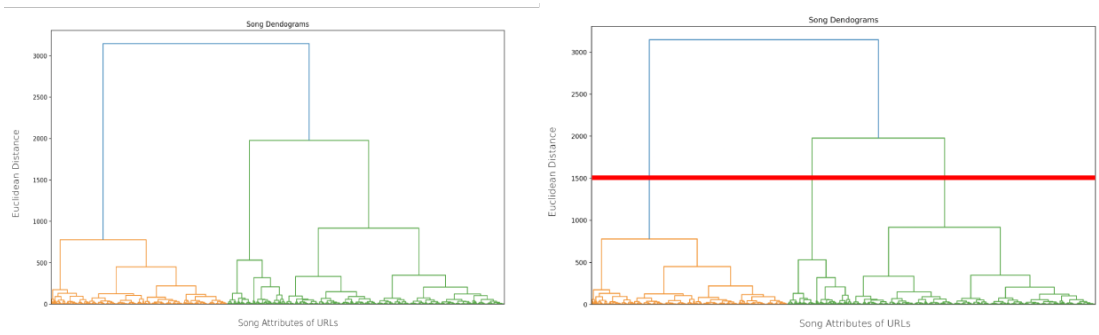

Figure 3: Dendogram plots. On the left: simple dendogram of all the points. On the right: dendogram with the threshold line at d=1500

Interpretation of the findings Here are the sizes of the 3 clusters:

- Cluster 0: 807,326

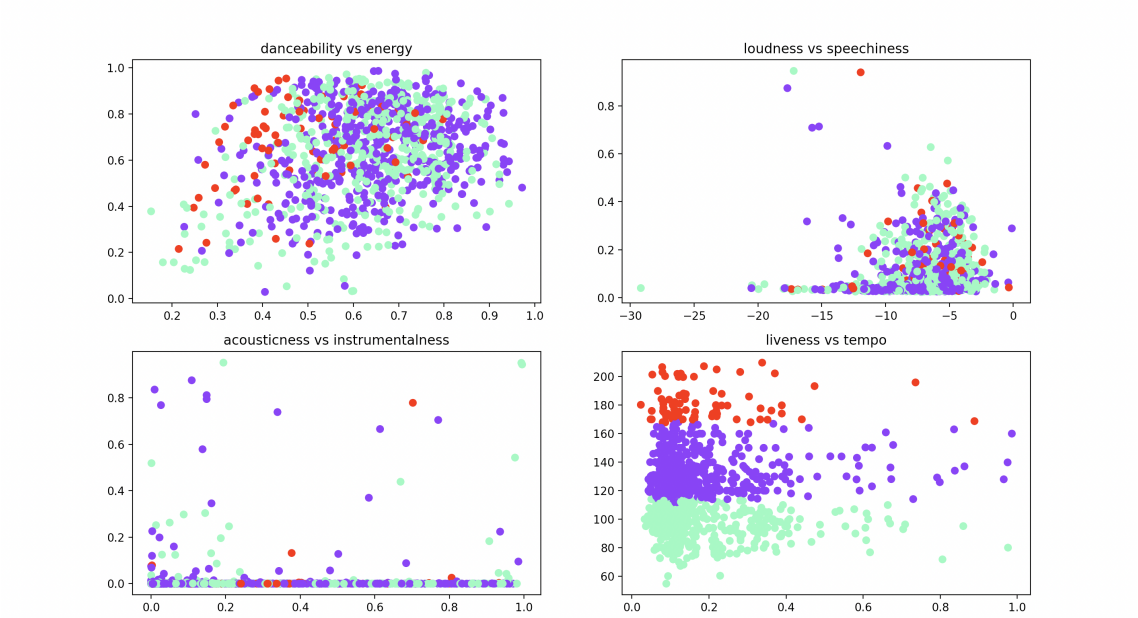

Figure 4: Data separated by clusters

Cluster 0:

	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	tempo
count	807326.000000	807326.000000	807326.000000	807326.000000	807326.000000	807326.000000	807326.000000	807326.000000
mean	0.618687	0.667292	-6.101766	0.134789	0.201869	0.167751	161.252399	
std	0.137082	0.156738	2.173414	0.107424	0.214182	0.123377	17.381527	
min	0.116000	0.035800	-31.042000	0.024000	0.000004	0.014100	136.506000	
25%	0.533000	0.560000	-7.028000	0.050600	0.046200	0.097300	144.945000	
50%	0.632000	0.691000	-5.879000	0.094100	0.137000	0.125000	159.797000	
75%	0.715000	0.785000	-4.757000	0.198000	0.256000	0.194000	175.914000	
max	0.954000	1.000000	1.107000	0.940000	0.994000	0.993000	229.921000	

Cluster 1:

	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	tempo
count	1.534656e+06	1.534656e+06	1.534656e+06	1.534656e+06	1.534656e+06	1.534656e+06	1.534656e+06	1.534656e+06
mean	6.861321e-01	6.761084e-01	-5.614721e+00	9.322262e-02	2.203697e-01	1.155166e-01	1.642191e-01	2.202691e+00
std	1.155166e-01	1.642191e-01	2.202691e+00	8.463428e-02	2.235229e-01	1.132828e-01	8.314459e+00	
min	6.450000e-02	6.910000e-03	-3.644300e+01	2.220000e-02	3.220000e-06	1.900000e-02	4.671800e+01	
25%	6.200000e-01	5.730000e-01	-6.647000e+00	4.320000e-02	5.040000e-02	9.160000e-02	9.083800e+01	
50%	7.030000e-01	6.980000e-01	-5.350000e+00	6.120000e-02	1.340000e-01	1.150000e-01	9.602200e+01	
75%	7.660000e-01	8.060000e-01	-4.097000e+00	1.090000e-01	3.380000e-01	1.860000e-01	1.018210e+02	
max	9.600000e-01	9.950000e-01	1.162000e+00	9.550000e-01	9.960000e-01	9.880000e-01	1.082760e+02	

Cluster 2:

	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	tempo
count	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06
mean	7.131472e-01	6.610454e-01	-5.926358e+00	8.454748e-02	2.116520e-01	2.219862e-01	1.628986e-01	2.146814e+00
std	1.860000e-01	2.620000e-03	-4.398800e+01	2.390000e-02	5.140000e-06	6.440000e-01	5.490000e-01	-7.611000e+00
min	7.320000e-01	6.730000e-01	-5.596000e+00	5.660000e-02	1.270000e-01	7.950000e-01	7.950000e-01	-4.424000e+00
25%	7.950000e-01	7.950000e-01	-4.424000e+00	9.810000e-02	3.680000e-01	9.830000e-01	9.990000e-01	1.585000e+00
50%	9.830000e-01	9.990000e-01	1.585000e+00	9.440000e-01	9.950000e-01	1.098539e+06	1.098539e+06	1.098539e+06
75%	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.650697e-01	1.221260e+02	
max	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.098539e+06	1.205453e-01	7.967503e+00	
						1.190000e-02	1.047780e+02	
						9.390000e-02	1.159520e+02	
						1.230000e-01	1.220470e+02	
						1.910000e-01	1.280280e+02	
						9.880000e-01	1.401690e+02	

Figure 5: Description of the data by cluster

- Cluster 1: 1,534,656
- Cluster 2: 1,098,539

Clusters 0 and 2 are fairly similar in size but Cluster 1 is a bit bigger. So let's review how hierarchical clustering has split our data - the results are Figure 5.

The results show us that Cluster 0 was more danceable, more upbeat and more "speechy" music, but it was also more spread out in these areas as indicated by higher standard deviation. However all clusters had very similar minimum and maximum values.

The more interesting part however is: which regions fall under the same categories of music.

Cluster 0 had the following regions with most in common: **'us', 'ca', 'br', 'mx', 'my', 'hk'**, and **'tw'**.

Cluster 1: **'co', 'ec', 'pe', 'es', 'cl', 'cr'**, and **'ar'**.

Finally, **Cluster 2**: **'fr', 'be', 'ie', 'de', 'at', 'pl'**, and **'nl'**.

Full results can be seen in "output.txt".

It would be interesting to further experiment on that and see why these regions are grouped together - look at how close to each other they are located and see if there are any similarities in the languages they speak. But that's a project for another time.

5 Discussion and conclusions

Summary of findings Overall, a lot of findings can be drawn from this data and what we can say about these results. Mainly there are two conclusions that hold from this specific project:

a) We can confidently say: if a song is popular in one region, it's likely to be popuar in the other.

For example: if a song is popular in 'pe' region, it's very likely going to be popular in 'co' region - reference to Cluster 1.

b) We can take any song, categorise it into a cluster and then confidently say which regions it's going to be popular in.

For example (choosing numbers at random): if a song has dancability = 0.43, energy = 0.78, loudeness = -4.0, speechiness = 0.06, and tempo = 140, it's very likely going to be in cluster 3 and therefore very popular in 'fr' region.

Evaluation of own work: strengths and limitations One of the main strengths of this project is the amount of different analytical techniques tried and researched before picking one to stick to. Even though hierarchical clustering was the right choice here, there are other techniques I talk about later which - if I had more time to learn and understand them - some would have given a lot more accurate and more complex results which are more applicable for real world applications.

Another strength would be visualisation. The only problem there is that I was unable to fully visualise the clusters formed purely because of the number of dimensions of data. Had I split the features of the track into smaller groups (e.g. dancability, energy, and tempo - which had very similar shapes of histograms), it would have been a lot easier to visualise, but it would have given such accurate results.

I also wish I had more DJ-like knowledge about music because then I would be able to better tell genres of songs apart, simply by looking at some features.

I think it would also be a lot better if I performed the same analysis for songs in 2021 and see if the clusters would change, because if not - my results would be very definitive, as we would have proven that they stand the test of time.

Comparison with any other related work There are a lot of studies that have been conducted around the prediction of popularity of songs. Above the ones that I've mentioned before, there are ones that do support my hypothesis about popularity being due to certain features of tracks [7], and some say the opposite - that they are only popular because of the artist and how popular they are [8]

Unfortunately I didn't find anything to support my claim about how popular songs differ by regions.

Improvements and extensions My main limitation here was time and computing power. My first idea was to go through every region, calculate their **probability distribution function** using the **fitter** library. Then compare them to every other region and while doing that, computing their **joint probability distribution function**. Because then we'd be able to say: "the probability of this song being ranked 10th in this region and that region is..." (P(X=10, Y=10).

Then we could go even further and introduce conditional distributions of joint variables. With this we could say "the probability that this song is ranked 1st in this region **given that** it's ranked 10th in that region is..." (P(X=1 | Y=10)).

I've started running some code for that but then realised that it will take about 19 billion iterations, which I calculated to take around 200 days on my laptop so I gave up on that idea. There is definitely a workaround, but I'd need more time to find it.

6 Bibliography

[1] Fast Company magazine. The definitive timeline of Spotify's critic-defying journey to rule music. 2018. https://www.fastcompany.com/90205527/the-definitive-timeline-of-spotifys-critic-defying-journey-to-rule-music

[2] Spotify Website. About Spotify. 2022. https://newsroom.spotify.com/company-info/

[3] Dea Bardhoshi. Visualizing Spotify Songs with Python: an exploratory data analysis. 2022. https://towardsdatascience.com/visualizing-spotify-songs-with-python-an-exploratory-data-analysis-fc3fae3c2c09

[4] Cristóbal Veas. Clustering Music to Create your Playlists on Spotify Using Python and R. 2020. https://towardsdatascience.com/clustering-music-to-create-your-personal-playlists-on-spotify-using-python-and-k-means-a39c4158589a

[5] Kaggle Profile. Eduardo. 2017.https://www.kaggle.com/edumucelli

[6] Usman Malik. Hierarchical Clustering with Python and Scikit-Learn. 2019. https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/

[7] Rutger Nijkamp. Prediction of product success: explaining song popularity by audio features from Spotify data. https://essay.utwente.nl/75422/1/NIJKAMP_BA_IBA.pdf

[8] Philip Peker. Predicting Popularity on Spotify — When Data Needs Culture More than Culture Needs Data. 2021. https://towardsdatascience.com/predicting-popularity-on-spotify-when-data-needs-culture-more-than-culture-needs-data-2ed3661f75f1