STA 141C Final Report
Spotify Music Trends Across Asia

Isabelle Berkowitz, Keon Sadeghi, Aryan Punjani, Vinay Singal, Carly Schwartzberg

Abstract:

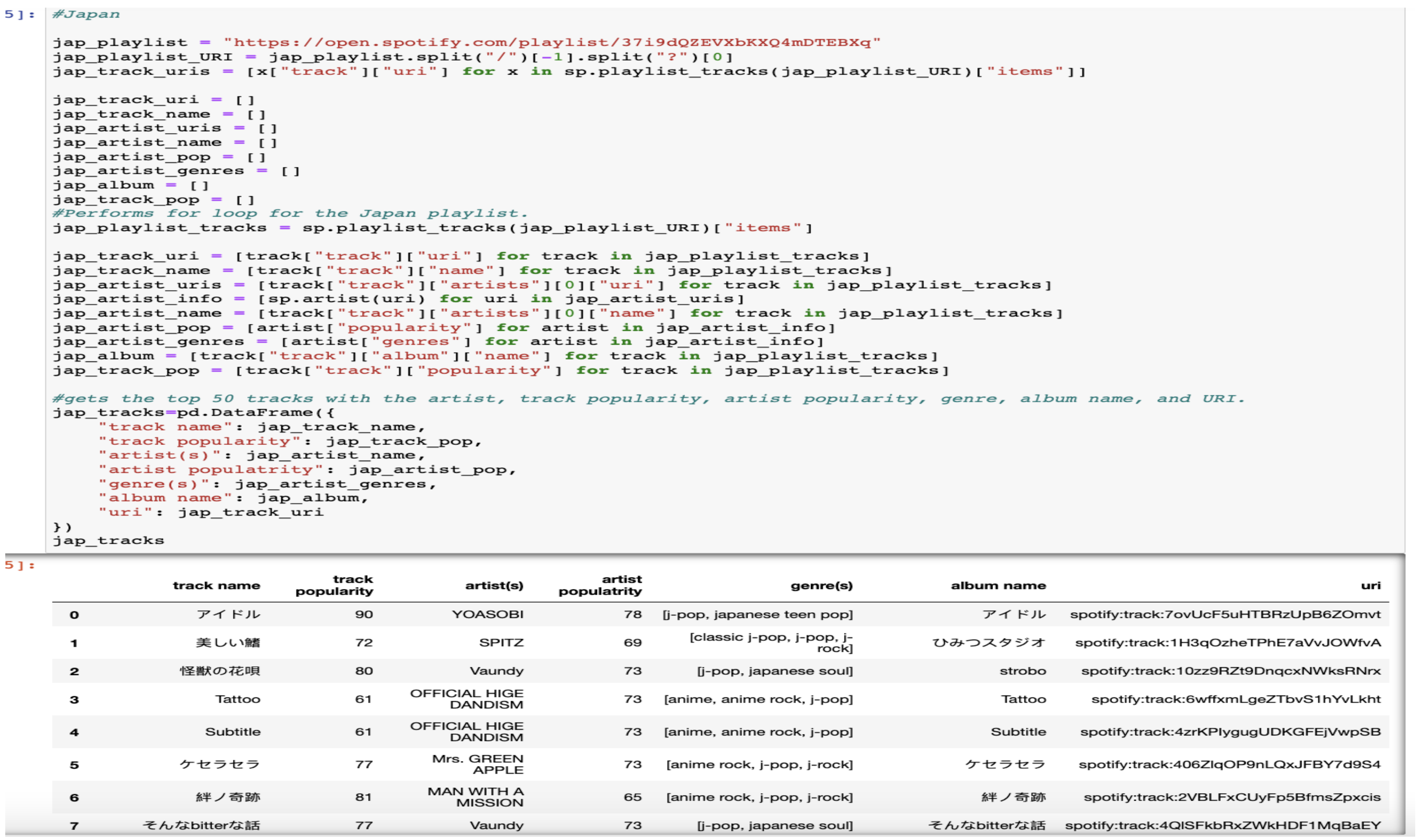
Around the world, music plays a significant role in expanding culture, sharing love, and allowing individuals to come together to appreciate the art of making. Music allows people to connect both on physical and emotional support as we find ways to relieve emotions, relax and find peace. It allows us to improve our current state of mind through the frequency within the beats, the tone of voice, and the speed of a song. Through understanding the impacts music has on individuals we can better understand the music and artists they listen to by recognizing mood fluctuations, tracking stress levels, and correlating certain genres with grade performances. Spotify is one of the most popular music services used globally by individuals that 'gives access to millions of songs and information about artists all around'. For this project, we extracted data through Spotify.com from three different levels of development throughout Asia: Japan, Vietnam, and India. Using the data we collected, we were able to find interesting listening trends within each country. For example, artists at the top of the charts in Japan and India are artists that are from those countries and make music in their native languages. Conversely, three of the top four artists are American, and through further research, we were able to conclude that this is most likely a result of American involvement in the Vietnam War. We applied many techniques and methods we learned from this course such as QR Decomposition to find the most telling audio features that affect people's listening patterns the most. We also used correlation analysis, sparsity patterns, and popularity distributions to give us a better understanding of the data we collected.

Introduction:

Music is an essential part of human culture that has the power to emotionally and physically change us as a society. It allows us to connect on many levels with those around us through the vibrations and feelings each word and beat causes us to hear. Many of us depend on music to get through the most basic tasks to studying for the biggest exam of our life. Without access to many of the biggest streaming services, many people would not be able to express their love for music. With Spotify being the biggest streaming service globally, our project analysis aims to look at three different countries that are in different stages of development within the same region. We want to understand what genres, artists, and song characteristics are most popular within these different countries. For this project, we decided to focus on Japan as our developed country, Vietnam as a developing country, and India as an underdeveloped country. Through this analysis of the top 50 songs in these countries, our goal is to determine if there are differences in listening patterns of the local music scene for our countries in different stages of development. Our project includes diving deep into the key factors that define a song's popularity and seeing its influence. We will analyze factors such as danceability, tempo, energy, etc. aiming to understand how they shape the music in each country. We will see if we observe any notable variations in these features based on the countries' developmental progress. By comparing these features across Japan, Vietnam, and India, the goal of our comparative analysis is to gain insights into the interplay between development and musical traits in these three countries. And although Spotify is not available in every country they are working towards that goal so individuals can appreciate and enjoy the content the app has to offer.

Description of Data Set:

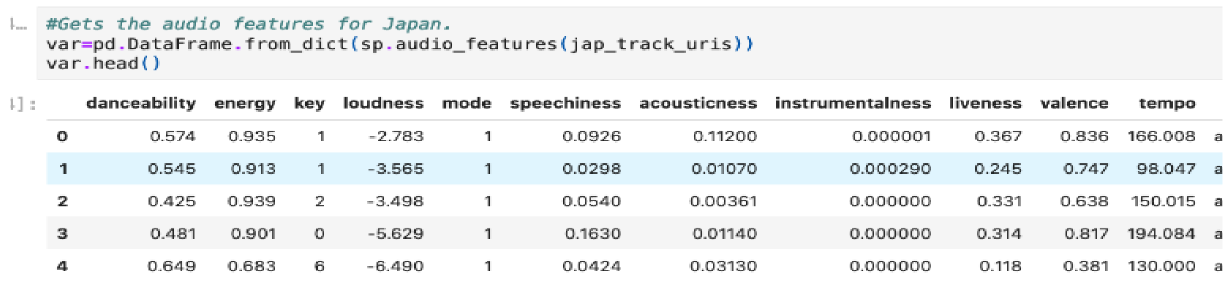
For this project, we extracted the data through the Spotify website for the three different countries we are looking at: Japan, Vietnam, and India. Each data set includes the different column names, ‘track name’, ‘track popularity’, ‘artist(s)’, ‘artist popularity’, ‘genre(s)’, ‘album name’, and ‘uri’. According to a blog discussing the leverage algorithm for the popularity ranking on Spotify our columns ‘track popularity’ and ‘artist popularity’ represent a score ‘from 0-100 that ranks the popularity of an artist in comparison to other artists on Spotify.’(Spotify Popularity Index) The higher the number represents the more popular a track or artist is in comparison to others. In addition, we also created new datasets for each country that include the column names: ‘danceability’, ‘energy’, ‘key’, ‘loudness’, ‘mode’, ‘speechiness’, ‘acousticness’, ‘instrumentals’, ‘liveness’, ‘valence’, and ‘tempo’ to help us identify influences countries and music variation have on each other.



5]:
```
#Japan

jap_playlist = "https://open.spotify.com/playlist/37i9dQZEVXbKXQ4mDTEBXq"
jap_playlist_URI = jap_playlist.split("/")[-1].split("?")[0]
jap_track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(jap_playlist_URI)["items"]]

jap_track_uri = []
jap_track_name = []
jap_artist_uris = []
jap_artist_name = []
jap_artist_pop = []
jap_artist_genres = []
jap_album = []
jap_track_pop = []
#Performs for loop for the Japan playlist.
jap_playlist_tracks = sp.playlist_tracks(jap_playlist_URI)["items"]

jap_track_uri = [track["track"]["uri"] for track in jap_playlist_tracks]
jap_track_name = [track["track"]["name"] for track in jap_playlist_tracks]
jap_artist_uris = [track["track"]["artists"][0]["uri"] for track in jap_playlist_tracks]
jap_artist_info = [sp.artist(uri) for uri in jap_artist_uris]
jap_artist_name = [track["track"]["artists"][0]["name"] for track in jap_playlist_tracks]
jap_artist_pop = [artist["popularity"] for artist in jap_artist_info]
jap_artist_genres = [artist["genres"] for artist in jap_artist_info]
jap_album = [track["track"]["album"]["name"] for track in jap_playlist_tracks]
jap_track_pop = [track["track"]["popularity"] for track in jap_playlist_tracks]

#gets the top 50 tracks with the artist, track popularity, artist popularity, genre, album name, and URI.
jap_tracks=pd.DataFrame({
    "track name": jap_track_name,
    "track popularity": jap_track_pop,
    "artist(s)": jap_artist_name,
    "artist populatrity": jap_artist_pop,
    "genre(s)": jap_artist_genres,
    "album name": jap_album,
    "uri": jap_track_uri
})
jap_tracks
```

5]:

	track name	track popularity	artist(s)	artist populatrity	genre(s)	album name	uri
0	アイドル	90	YOASOBI	78	[j-pop, japanese teen pop]	アイドル	spotify:track:7ovUcF5uHTBRzUpB6ZOmvt
1	美しい鰭	72	SPITZ	69	[classic j-pop, j-pop, j-rock]	ひみつスタジオ	spotify:track:1H3qOzheTPhE7aVvJOWfvA
2	怪獣の花唄	80	Vaundy	73	[j-pop, japanese soul]	strobo	spotify:track:10zz9RZt9DnqcxNWksRNrx
3	Tattoo	61	OFFICIAL HIGE DANDISM	73	[anime, anime rock, j-pop]	Tattoo	spotify:track:6wffxmLgeZTbvS1hYvLkht
4	Subtitle	61	OFFICIAL HIGE DANDISM	73	[anime, anime rock, j-pop]	Subtitle	spotify:track:4zrKPlygugUDKGFEjVwpSB
5	ケセラセラ	77	Mrs. GREEN APPLE	73	[anime rock, j-pop, j-rock]	ケセラセラ	spotify:track:406ZlqOP9nLQxJFBY7d9S4
6	絆ノ奇跡	81	MAN WITH A MISSION	65	[anime rock, j-pop, j-rock]	絆ノ奇跡	spotify:track:2VBLFxCUyFp5BfmsZpxcis
7	そんなbitterな話	77	Vaundy	73	[j-pop, japanese soul]	そんなbitterな話	spotify:track:4QISFkbRxZWkHDF1MqBaEY

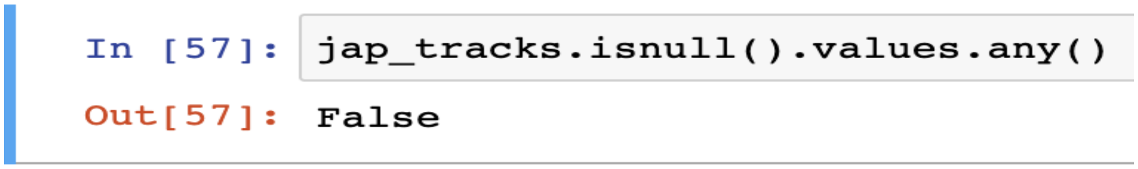


```
#Gets the audio features for Japan.
var=pd.DataFrame.from_dict(sp.audio_features(jap_track_uris))
var.head()
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	
0	0.574	0.935	1	-2.783	1	0.0926	0.11200	0.000001	0.367	0.836	166.008	a
1	0.545	0.913	1	-3.565	1	0.0298	0.01070	0.000290	0.245	0.747	98.047	a
2	0.425	0.939	2	-3.498	1	0.0540	0.00361	0.000000	0.331	0.638	150.015	a
3	0.481	0.901	0	-5.629	1	0.1630	0.01140	0.000000	0.314	0.817	194.084	a
4	0.649	0.683	6	-6.490	1	0.0424	0.03130	0.000000	0.118	0.381	130.000	a

Data Processing:

We worked with six different datasets for this project each that have been extracted from the Spotify Web API. We did not combine any of them together since we compared the individual trends from the three countries selected in the Asia region. By looking at each of them separately we were able to look at the correlation between the top artists, songs, and genres. In addition, we did not remove any columns as they all serve a purpose in the patterns and trends we found. Finally, we checked to make sure that each of our six data frames did not have any values of NAN by running the code: jap_tracks.isnull().values.any(), which returns 'False' because there were no NAN values found present.



```
In [57]: jap_tracks.isnull().values.any()

Out[57]: False
```

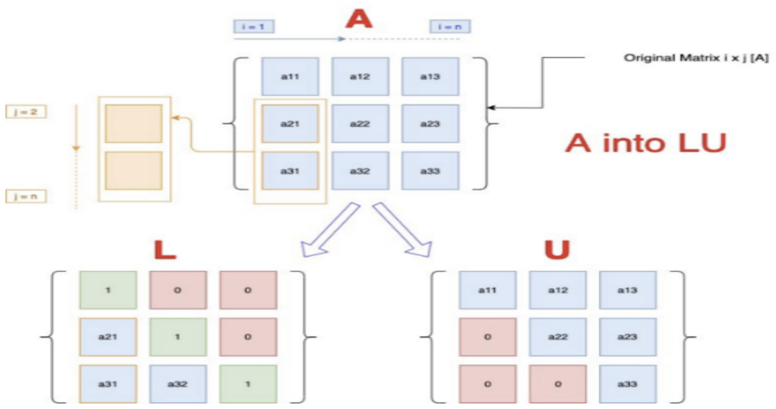
Proposed Methods and Algorithms:

Throughout this course, we learned a series of different methods and algorithms that can be applied to different data sets of different sizes. In the homework, we applied the methods taught in lectures such as LU decomposition, QR Factorization, Jacobi, Gauss-Seidel, SVD, and others. For our project, we demonstrated our understanding of the methods we learned both in this class and in previous classes in the 141 series by discussing the purpose of each method and later specifically applying some of them to our project.

1. LU-Decomposition (LU)

LU - Decomposition is used as a method to factorize some matrix that is the product of both the upper and lower triangular matrix. By using the linear equation Ax = b where A is a dense matrix, the goal is to "..use a series of elementary operations called Gaussian elimination, to turn A into a triangular system and then apply forward and backward substitutions to solve x" (Ning, Week 3-2) This method can be expressed through the equation A = LU. **A** represents the original matrix expressed as m x n, **L** is the lower triangular matrix following an
m x m matrix, and U, the upper triangular matrix following an m x n matrix. The image below explains the process of LU Decomposition demonstrating the ending goal is to obtain zeros in the upper right corner for our lower matrix and zeros in the lower left corner for our upper matrix.

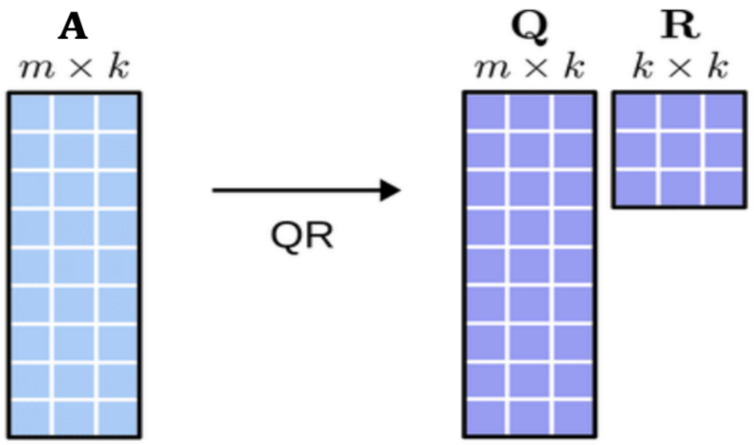
Figure 1:



2. **QR Decomposition(QR)**

QR decomposition is used as a matrix factorization tool that breaks down a matrix into the product of two matrices. These matrices are an upper triangular matrix and an orthogonal matrix. Generally, QR Decomposition is used on linear systems of equations, finding answers to matrix problems, and finding solutions to problems regarding least squares solutions. In our case, this method is useful to find the most important features of our dataset, allowing us to focus on what affects listening patterns. Below is a representation of how QR decomposition works in matrices.

Figure 2:



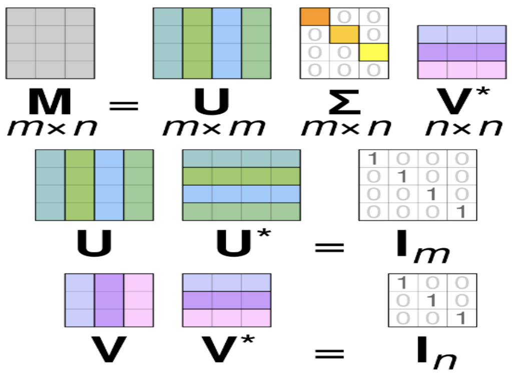
3. **AIC(Akaike Information Criterion) and BIC(Bayesian Information Criterion)**

Both AIC and BIC are statistical measures that help with model selection. Depending on the data you are evaluating and comparing you are able to best fit your data per model. You must first determine the models you want to use to analyze the specific relationship your project entitles to. In our project, we would compare the relationship between stages of development with the music features found in the different datasets. There are several models you can conduct that are relevant to your topic which include linear regression or logistic regression models. Once you have selected the model and applied that to your dataset consider the factors each column represents and calculate AIC and BIC for the models you chose. Once those values are calculated you can determine if your model represents a good fit and less complexity if the values are lower, compared to if they are high.

4. **SVD(Singular Value Decomposition)**

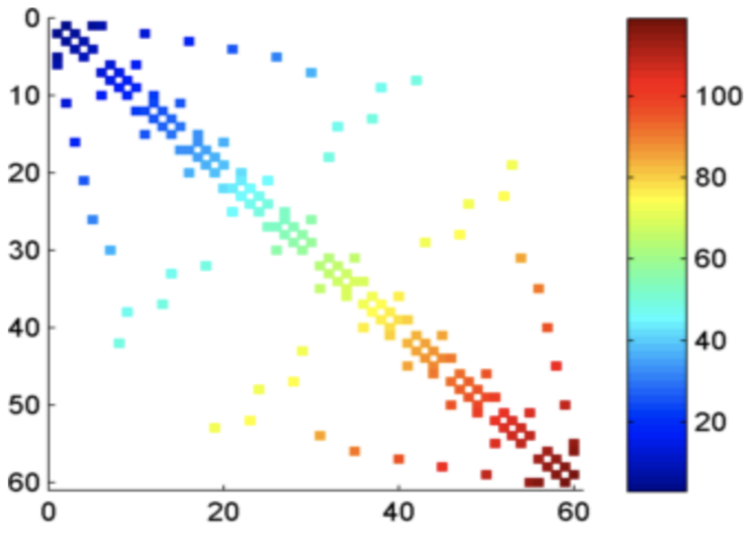
When using SVD we are looking at how to decompose a matrix. To do this we look at U, Σ, and V. If we set up a rectangular matrix A∈ R^ m×n we have the SVD X= UΣV'. When we look at this metric Σ represents a rectangular diagonal matrix and V and U are both orthogonal matrices. There can also be another SVD when m>n: X = UnΣnV' = **Σ** σiuiv'i. The matrices for V and U inverses are equivalent to their transposed matrices. The inverse of Σ can be found if we take the reciprocal of the nonzero element. For example this can be shown: Σ = diag(σ1, σ2, σ3, σ4…) => Σ^(-1) = diag(1/σ1, 1/σ2, 1/σ3, 1/σ4…). Using SVD helps to simplify the process of computation. This is because it removes the need for more matrix inversions. With SVD we can also find the eigenvalues and eigenvectors. One method for computing SVD is the power method. The power method finds the dominant eigenvalue and eigenvector of a matrix. The most basic algorithm for SVD is the power method.

Figure 3:


$\mathbf{M}_{m\times n} = \mathbf{U}_{m\times m}\ \Sigma_{m\times n}\ \mathbf{V}^*_{n\times n}$
$\mathbf{U}\ \mathbf{U}^* = \mathbf{I}_m$
$\mathbf{V}\ \mathbf{V}^* = \mathbf{I}_n$

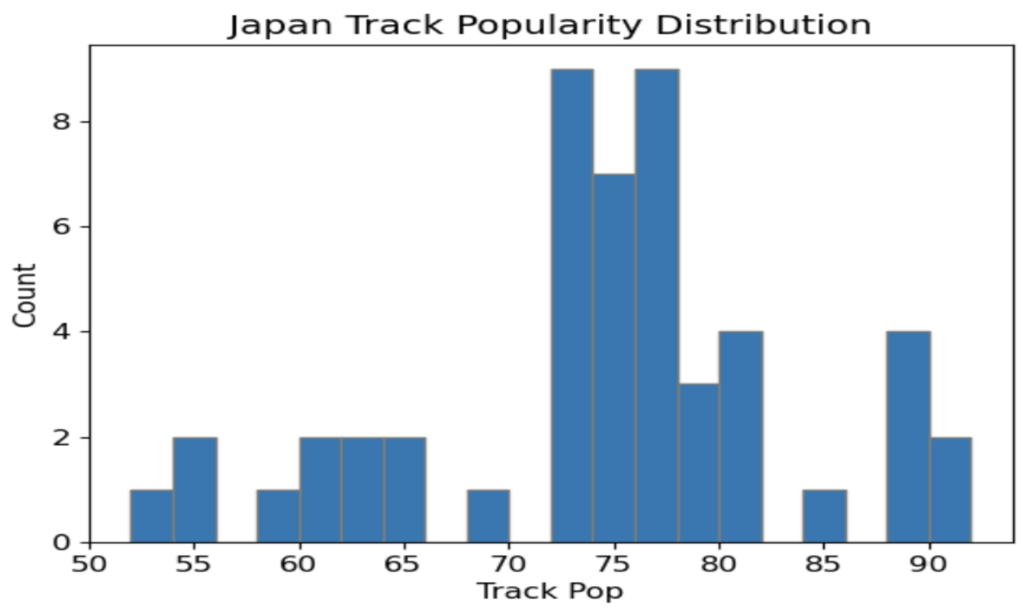
5. **Sparsity Pattern Significance**
   Sparsity pattern plots are graphs that represent coefficients in our model that hold nonzero values and compare different variables within your model. By creating these plots you are able to understand sparsity trends within your model and decipher which variables are important and which do not hold as much significance. One key reason these plots are important is due to model complexity. Through examination of the patterns produced, “ you can determine the degree of sparsity or the proportion of variables with zero coefficients.”(*Sparsity Pattern - an Overview)* Essentially helping us decide if a simpler model is preferred. In our project using sparsity pattern plots allows us to visually interpret the relationship between artists and genres.

Figure 4: Visualize sparsity pattern with intensity using Matlab spy function

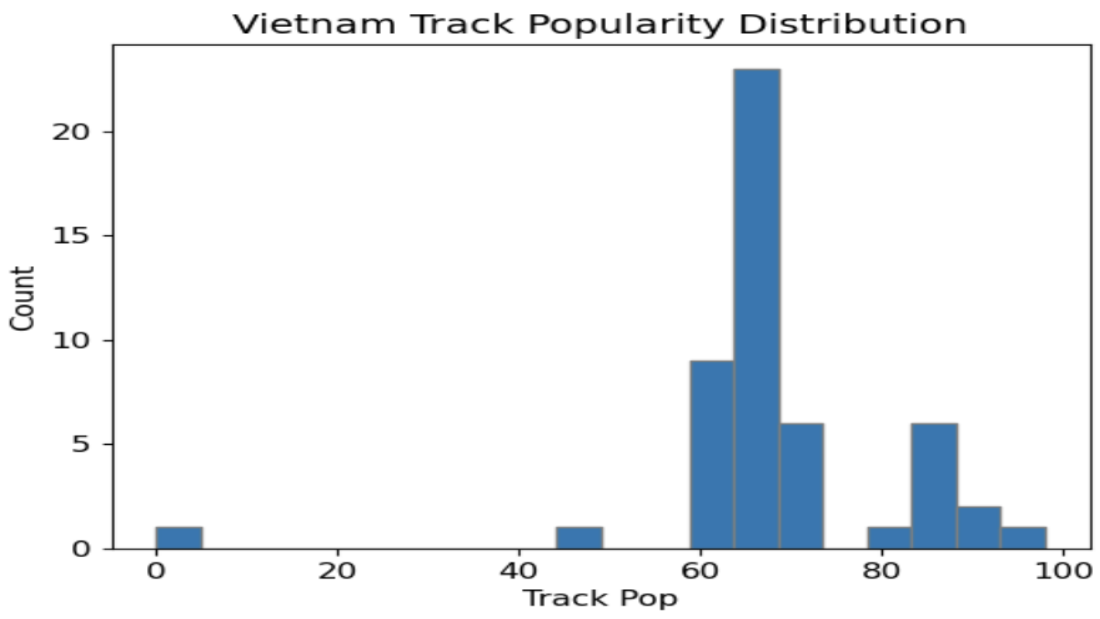


Simulation Study:

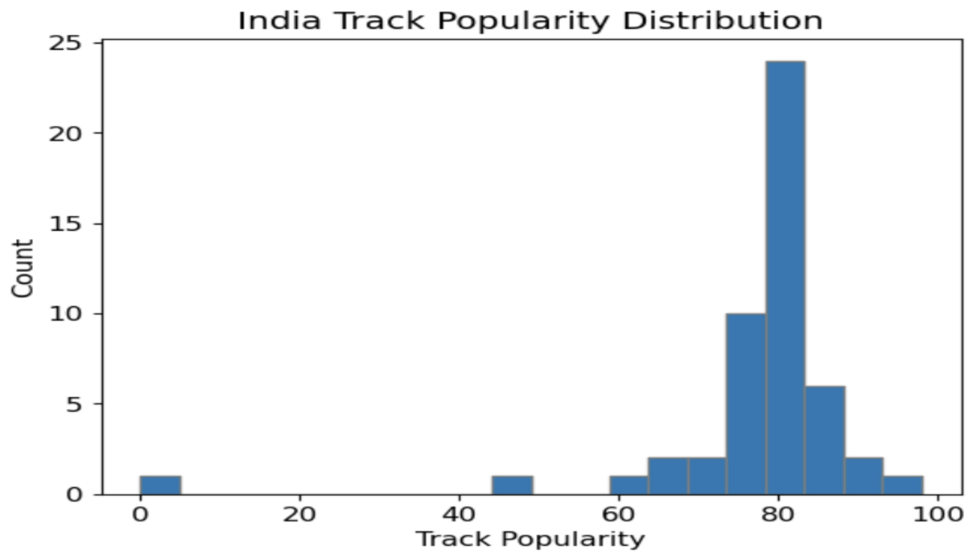
We want to make sure that all countries we are examining have an ample amount of listeners. To do this, we ran a simulation that determined the mean popularity scores for tracks from each country. Spotify's popularity scores consist of the total streams of a song, how recently a song has been played, and the frequency that a track has been played (*What is the Spotify Popularity Index*). What we found was, on a scale of 0-100, Japan has an average popularity score of 74, Vietnam has an average of 68 and India has an average of 78. Since all of these countries have tracks with popularity scores over 65, we deemed that all countries have enough listeners to proceed with our analysis.



```
mean: 74.06
median: 74.5
standard deviation: 9.177190001657946
```



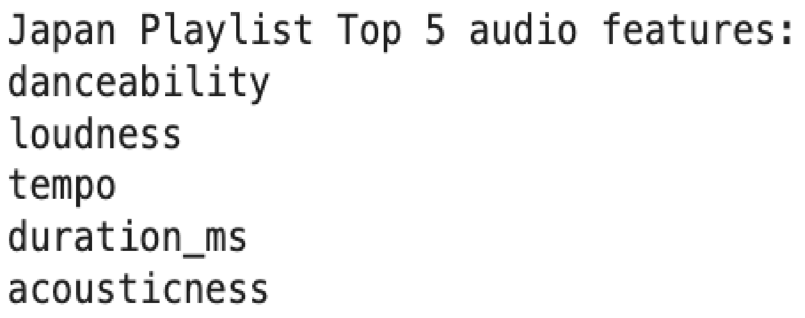
```
mean: 68.1
median: 65.0
standard deviation: 14.234551239136128
```



```
mean: 77.64
median: 80.0
standard deviation: 13.579336042198042
```

Analyzation of LU Decomposition: LU decomposition was the first method we analyzed for our project. We attempted to apply the linear equation we learned in class for creating a matrix on the different variables. However, we found that for the data in our model, LU is possible to perform, although the information gathered would not be beneficial for our analysis.

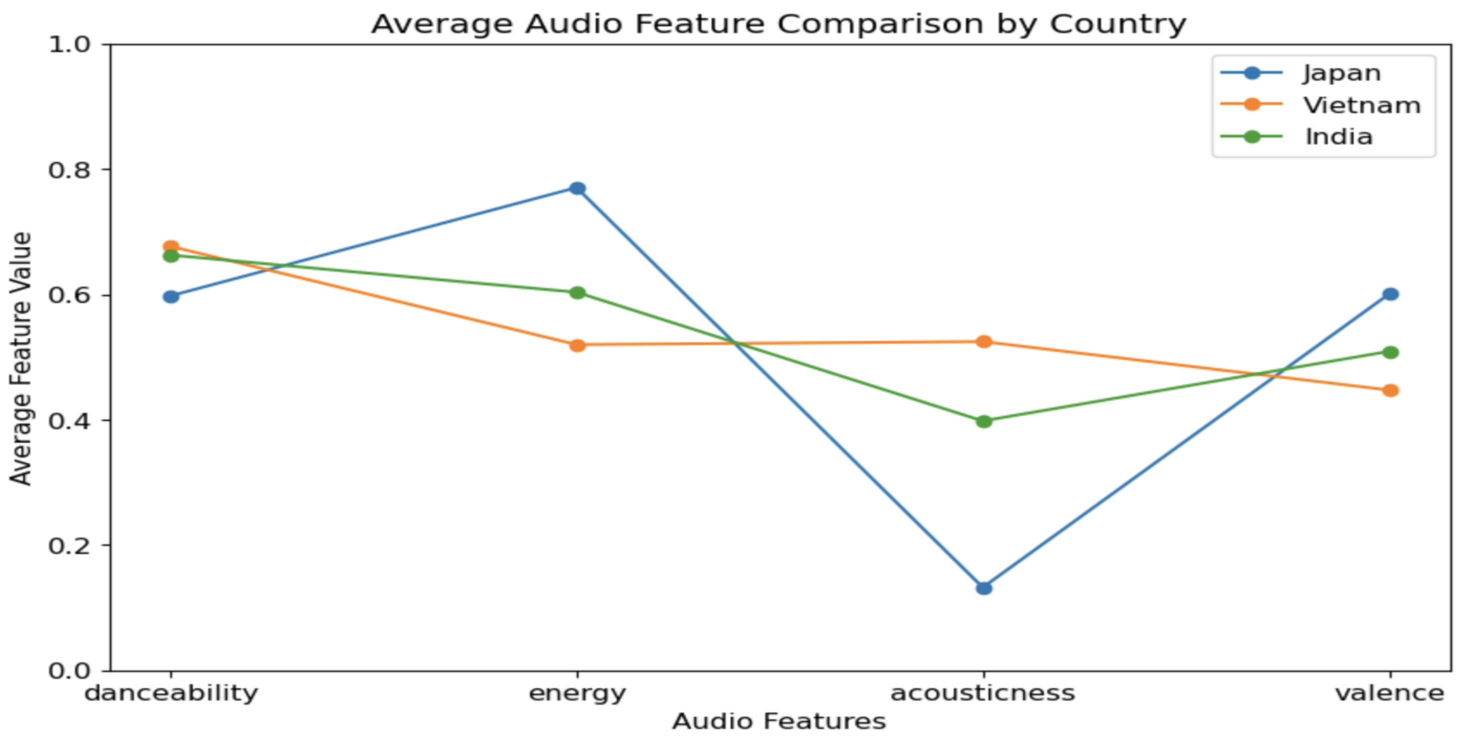
QR Decomposition: After researching alternative methods, we found that QR Decomposition would be applicable. We used this to help us identify the five most important audio features in regard to people's listening patterns. Finding which features are most prevalent tells us what types of songs are successful in the three countries we are studying. Using QR Decomposition, we found that the top audio features for all three countries are as follows (in order from most to least important): ***Japan*** — danceability, loudness, tempo, song duration, and acousticness — ***Vietnam*** — danceability, valence, key, liveness, and acousticness — ***India*** — danceability, acousticness, loudness, energy, and tempo.



```
Japan Playlist Top 5 audio features:
danceability
loudness
tempo
duration_ms
acousticness
```

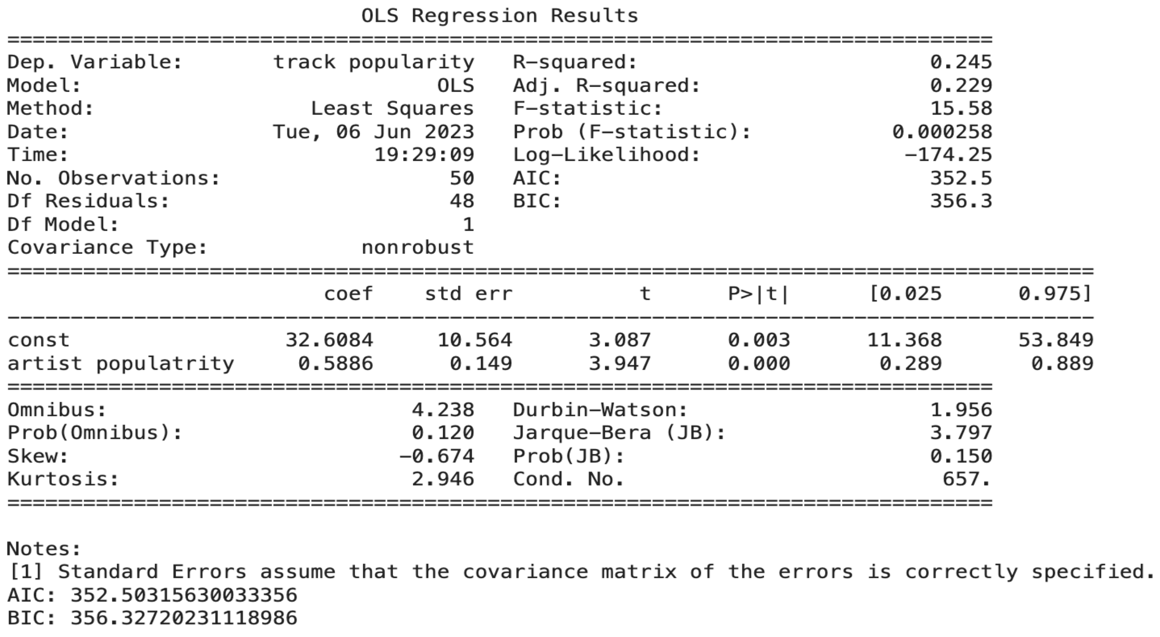
```
Vietnam Playlist Top 5 audio features:
danceability
valence
key
liveness
acousticness
```

```
India Playlist Top 5 audio features:
danceability
acousticness
loudness
energy
tempo
```



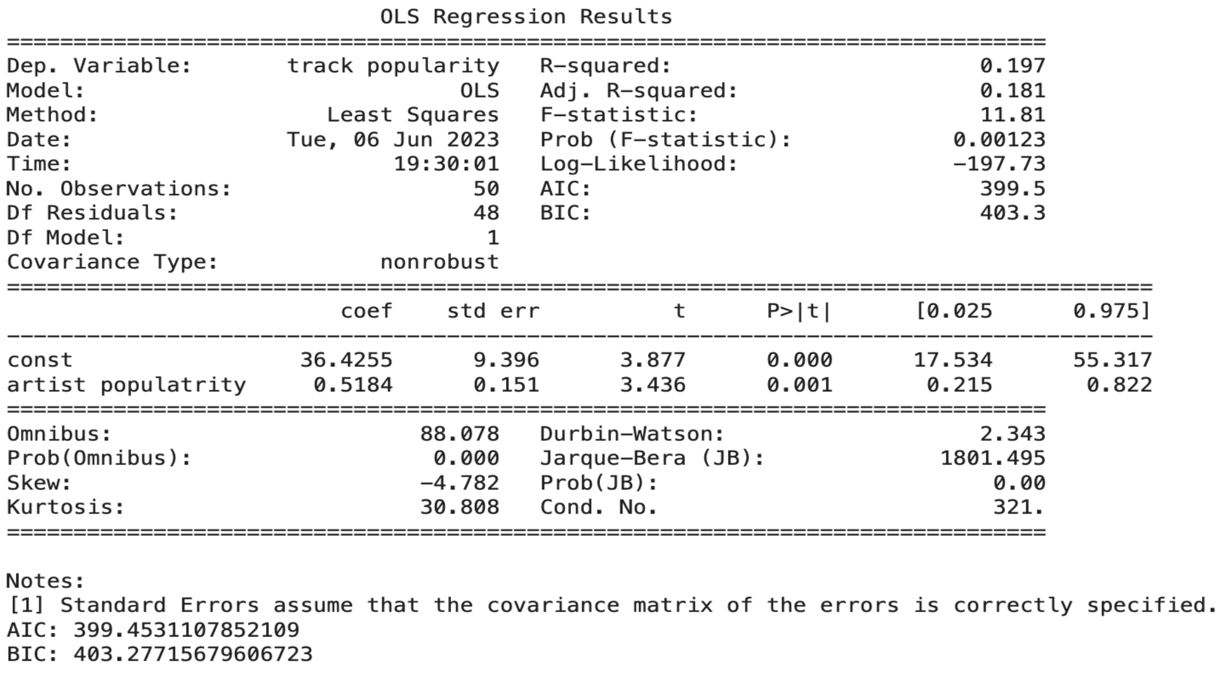
AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion): We decided to calculate AIC and BIC for a linear regression model with our x variable being artist popularity and the y value being track popularity. We did this to see how good of a fit these two predictor variables are in the linear regression model we outlined. For Japan, using AIC, we got a value of 352.5 and using BIC, we got 356.3. For Vietnam, using AIC, we got a value of 399.5 and using BIC, we got 403.3. For India, using AIC, we got a value of 393.6, and using BIC, we got 397.5. With these relatively small values, we can say that these variables are both a good fit in the linear regression model. With this being said, in order from best to worst fit by country, Japan is the best fit with India coming in second and Vietnam being the worst fit model.

Regression results for Japan



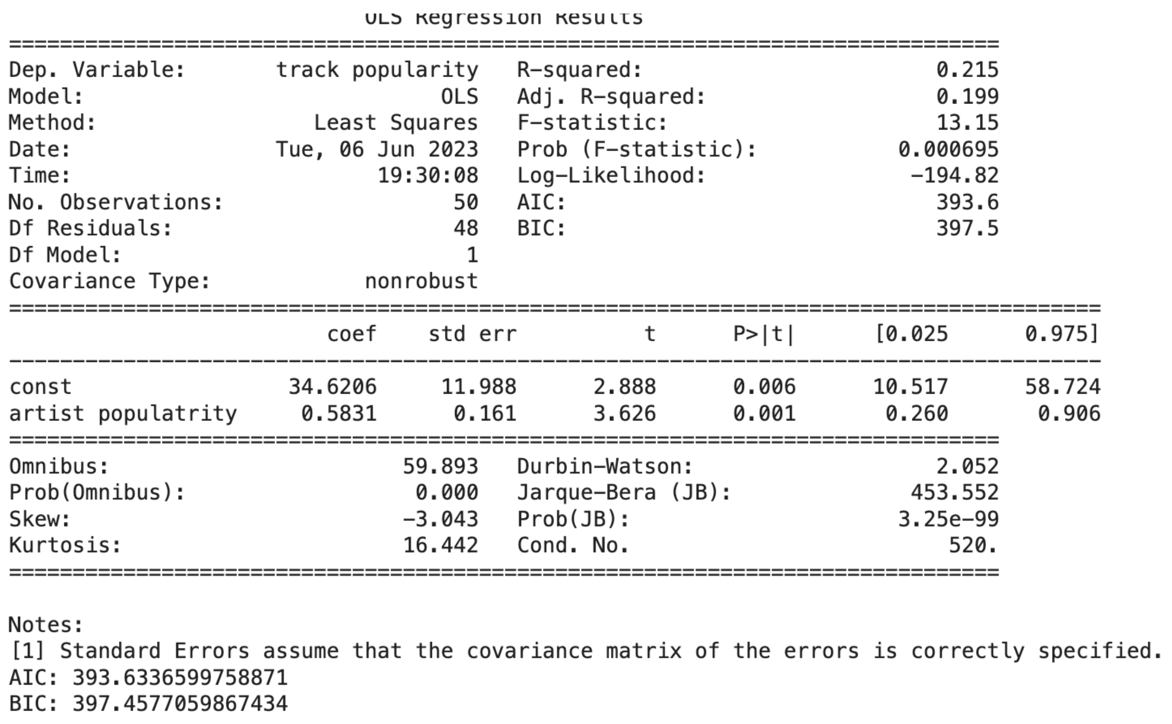
```
                            OLS Regression Results
==============================================================================
Dep. Variable:       track popularity   R-squared:                       0.245
Model:                            OLS   Adj. R-squared:                  0.229
Method:                 Least Squares   F-statistic:                     15.58
Date:                Tue, 06 Jun 2023   Prob (F-statistic):           0.000258
Time:                        19:29:09   Log-Likelihood:                -174.25
No. Observations:                  50   AIC:                             352.5
Df Residuals:                      48   BIC:                             356.3
Df Model:                           1
Covariance Type:            nonrobust
======================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
const                32.6084     10.564      3.087      0.003      11.368      53.849
artist populatrity    0.5886      0.149      3.947      0.000       0.289       0.889
==============================================================================
Omnibus:                        4.238   Durbin-Watson:                   1.956
Prob(Omnibus):                  0.120   Jarque-Bera (JB):                3.797
Skew:                          -0.674   Prob(JB):                        0.150
Kurtosis:                       2.946   Cond. No.                         657.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
AIC: 352.50315630033356
BIC: 356.32720231118986
```

Regression results for Vietnam



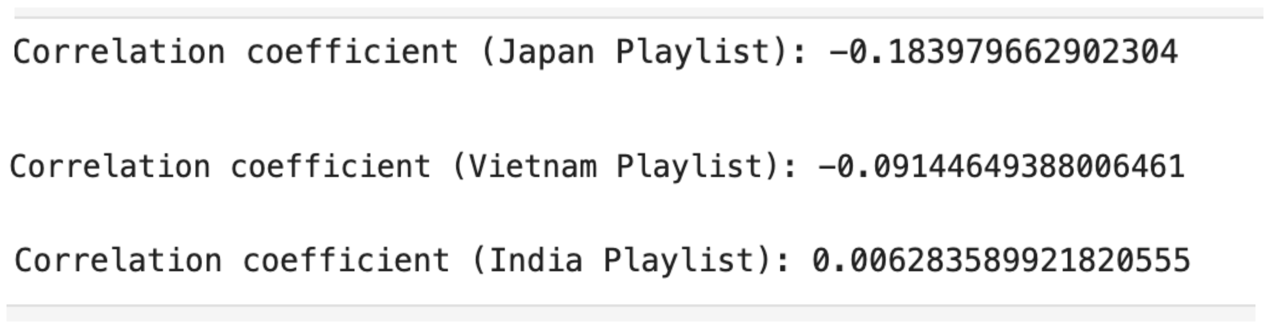
```
                            OLS Regression Results
==============================================================================
Dep. Variable:       track popularity   R-squared:                       0.197
Model:                            OLS   Adj. R-squared:                  0.181
Method:                 Least Squares   F-statistic:                     11.81
Date:                Tue, 06 Jun 2023   Prob (F-statistic):            0.00123
Time:                        19:30:01   Log-Likelihood:                -197.73
No. Observations:                  50   AIC:                             399.5
Df Residuals:                      48   BIC:                             403.3
Df Model:                           1
Covariance Type:            nonrobust
======================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
const                36.4255      9.396      3.877      0.000      17.534      55.317
artist populatrity    0.5184      0.151      3.436      0.001       0.215       0.822
==============================================================================
Omnibus:                       88.078   Durbin-Watson:                   2.343
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1801.495
Skew:                          -4.782   Prob(JB):                         0.00
Kurtosis:                      30.808   Cond. No.                         321.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
AIC: 399.4531107852109
BIC: 403.27715679606723
```

Regression results for India



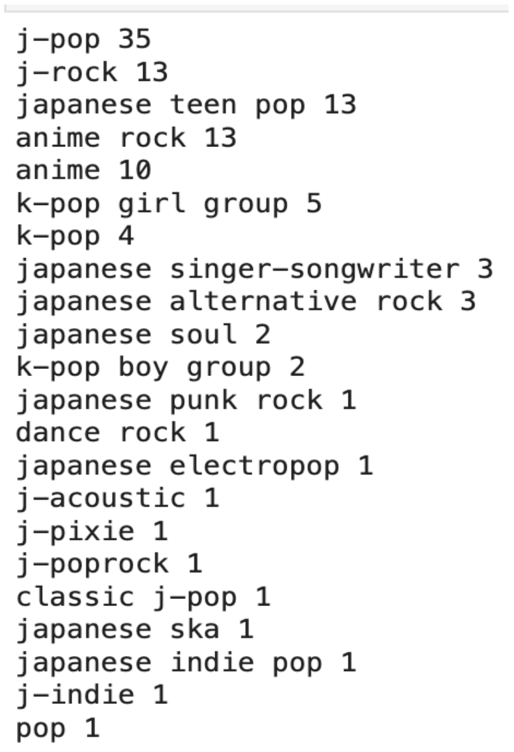
```
                            OLS Regression Results
==============================================================================
Dep. Variable:       track popularity   R-squared:                       0.215
Model:                            OLS   Adj. R-squared:                  0.199
Method:                 Least Squares   F-statistic:                     13.15
Date:                Tue, 06 Jun 2023   Prob (F-statistic):           0.000695
Time:                        19:30:08   Log-Likelihood:                -194.82
No. Observations:                  50   AIC:                             393.6
Df Residuals:                      48   BIC:                             397.5
Df Model:                           1
Covariance Type:            nonrobust
========================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                 34.6206     11.988      2.888      0.006      10.517      58.724
artist populatrity     0.5831      0.161      3.626      0.001       0.260       0.906
==============================================================================
Omnibus:                       59.893   Durbin-Watson:                   2.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              453.552
Skew:                          -3.043   Prob(JB):                     3.25e-99
Kurtosis:                      16.442   Cond. No.                         520.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
AIC: 393.6336599758871
BIC: 397.4577059867434
```

Correlation between song name length and popularity:
Since we already compare all the audio features and how they affect popularity, we wanted to check if the length of the song name had any effect on the popularity score. To do this, we plot the correlation coefficient of popularity against the length of the track name. This results in the following values. We see that there is barely any real correlation between the two. Thus we can conclude that the popularity of the songs is mainly dependent on the audio features and are not affected by the length of the song name.

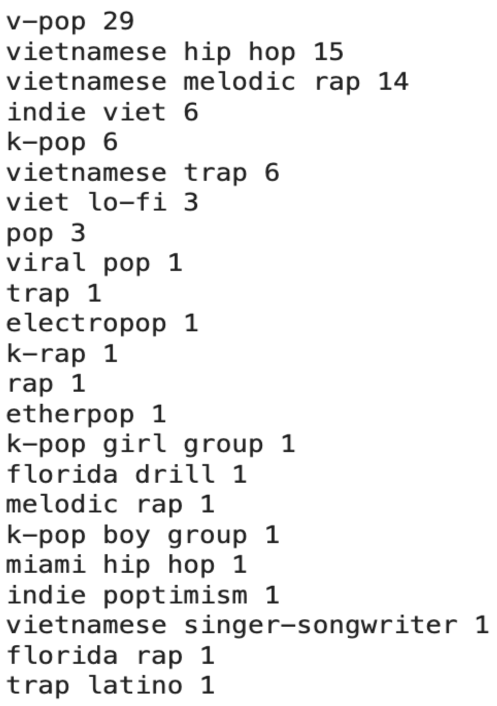


```
Correlation coefficient (Japan Playlist): -0.183979662902304

Correlation coefficient (Vietnam Playlist): -0.09144649388006461

Correlation coefficient (India Playlist): 0.006283589921820555
```

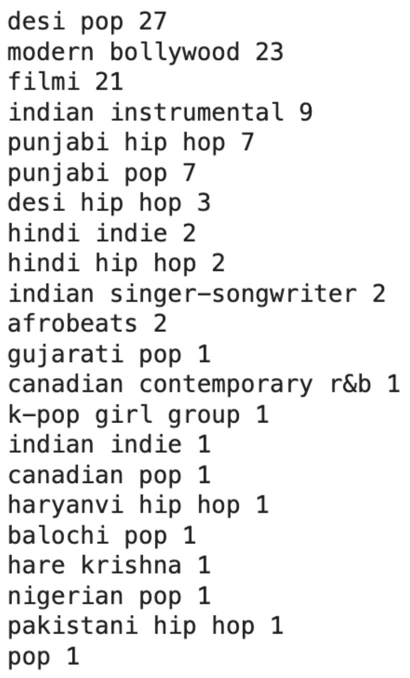
Genre Breakdown:
These outputs indicate that each country contains a different version of pop that is the most popular genre in the country's playlist. In Japan, we almost exclusively see genres that are prevalent in Japan while for Vietnam we see some American genres such as Florida rap and trap Latino. Also for India, we see Pakistani hip-hop and Nigerian pop as some of the categories included in the list. This can indicate that other countries have more of an influence on Vietnam and India than Japan in terms of the culture and music their people listen to.



```
j-pop 35
j-rock 13
japanese teen pop 13
anime rock 13
anime 10
k-pop girl group 5
k-pop 4
japanese singer-songwriter 3
japanese alternative rock 3
japanese soul 2
k-pop boy group 2
japanese punk rock 1
dance rock 1
japanese electropop 1
j-acoustic 1
j-pixie 1
j-poprock 1
classic j-pop 1
japanese ska 1
japanese indie pop 1
j-indie 1
pop 1
```



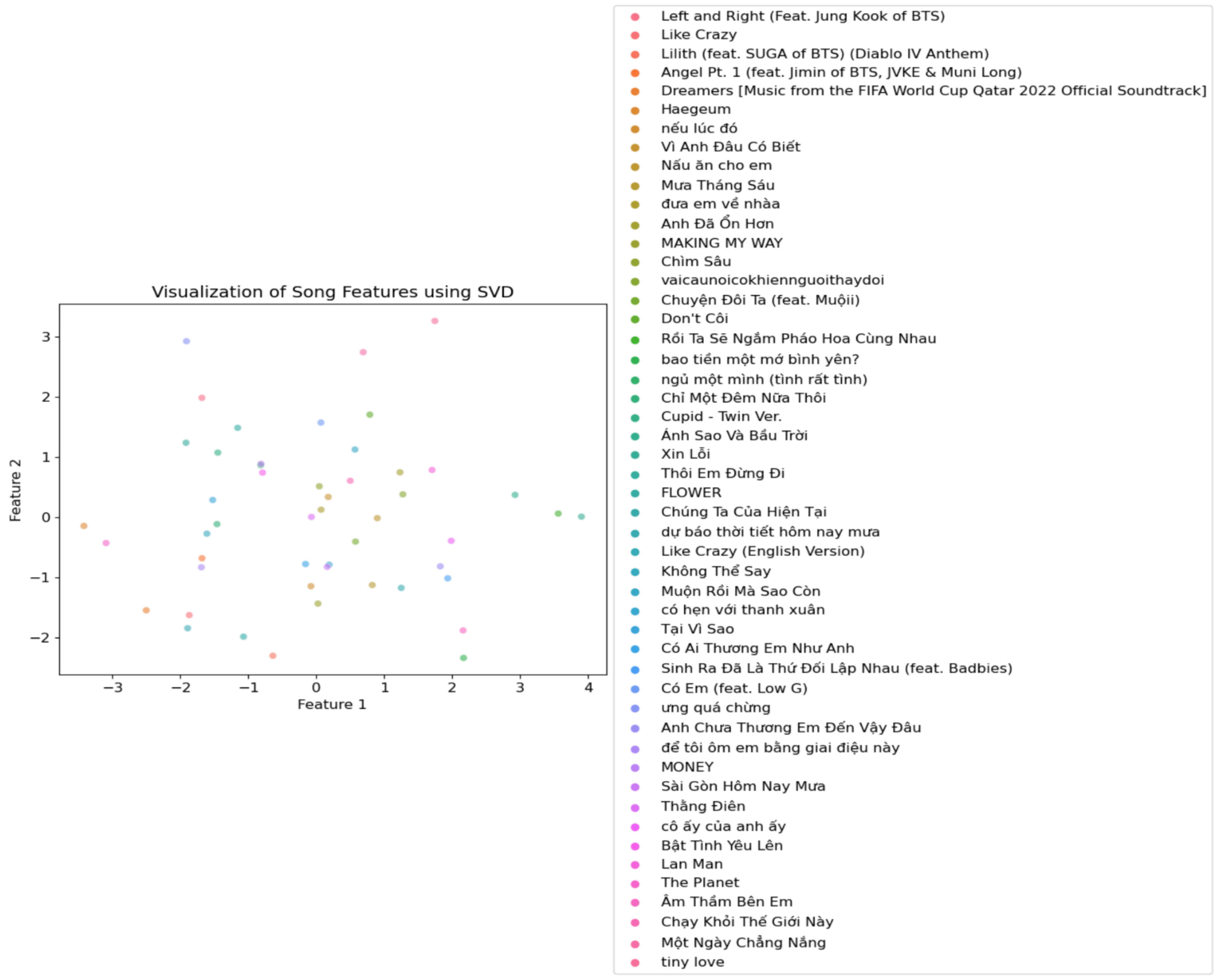
```
v-pop 29
vietnamese hip hop 15
vietnamese melodic rap 14
indie viet 6
k-pop 6
vietnamese trap 6
viet lo-fi 3
pop 3
viral pop 1
trap 1
electropop 1
k-rap 1
rap 1
etherpop 1
k-pop girl group 1
florida drill 1
melodic rap 1
k-pop boy group 1
miami hip hop 1
indie poptimism 1
vietnamese singer-songwriter 1
florida rap 1
trap latino 1
```



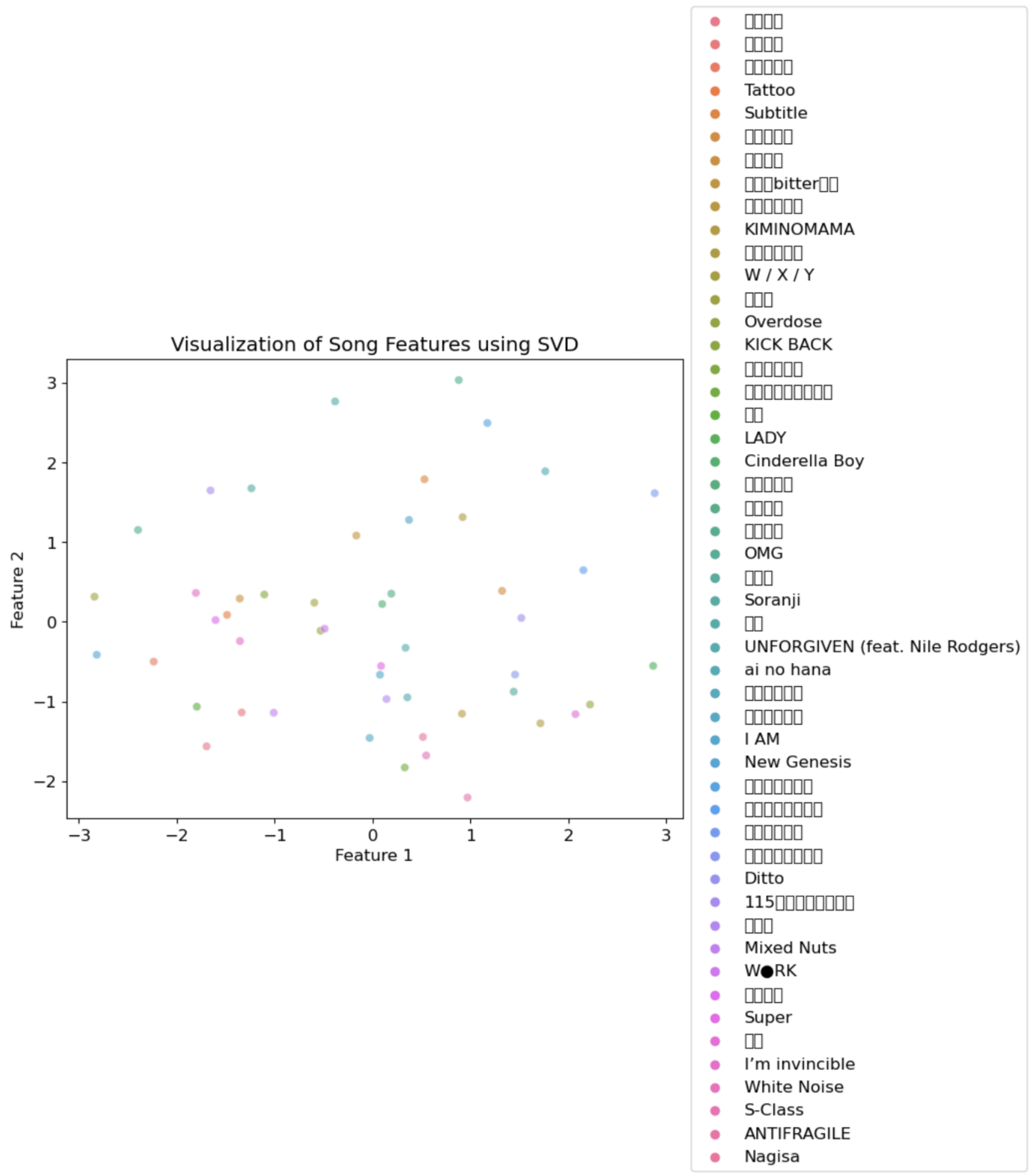
```
desi pop 27
modern bollywood 23
filmi 21
indian instrumental 9
punjabi hip hop 7
punjabi pop 7
desi hip hop 3
hindi indie 2
hindi hip hop 2
indian singer-songwriter 2
afrobeats 2
gujarati pop 1
canadian contemporary r&b 1
k-pop girl group 1
indian indie 1
canadian pop 1
haryanvi hip hop 1
balochi pop 1
hare krishna 1
nigerian pop 1
pakistani hip hop 1
pop 1
```

SVD (Singular Value Decomposition): For the scatter plots created using SVD, we found danceability and energy to be the most important features. These are labeled as Feature 1 and Feature 2. The scatter plot graphs the 50 songs in each playlist on a scatter plot, comparing the two aspects of each song to the same features of the other songs. Some of the names on the legend show blank letters. This is because those specific song names are in the native languages and the software does not recognize non-English characters.

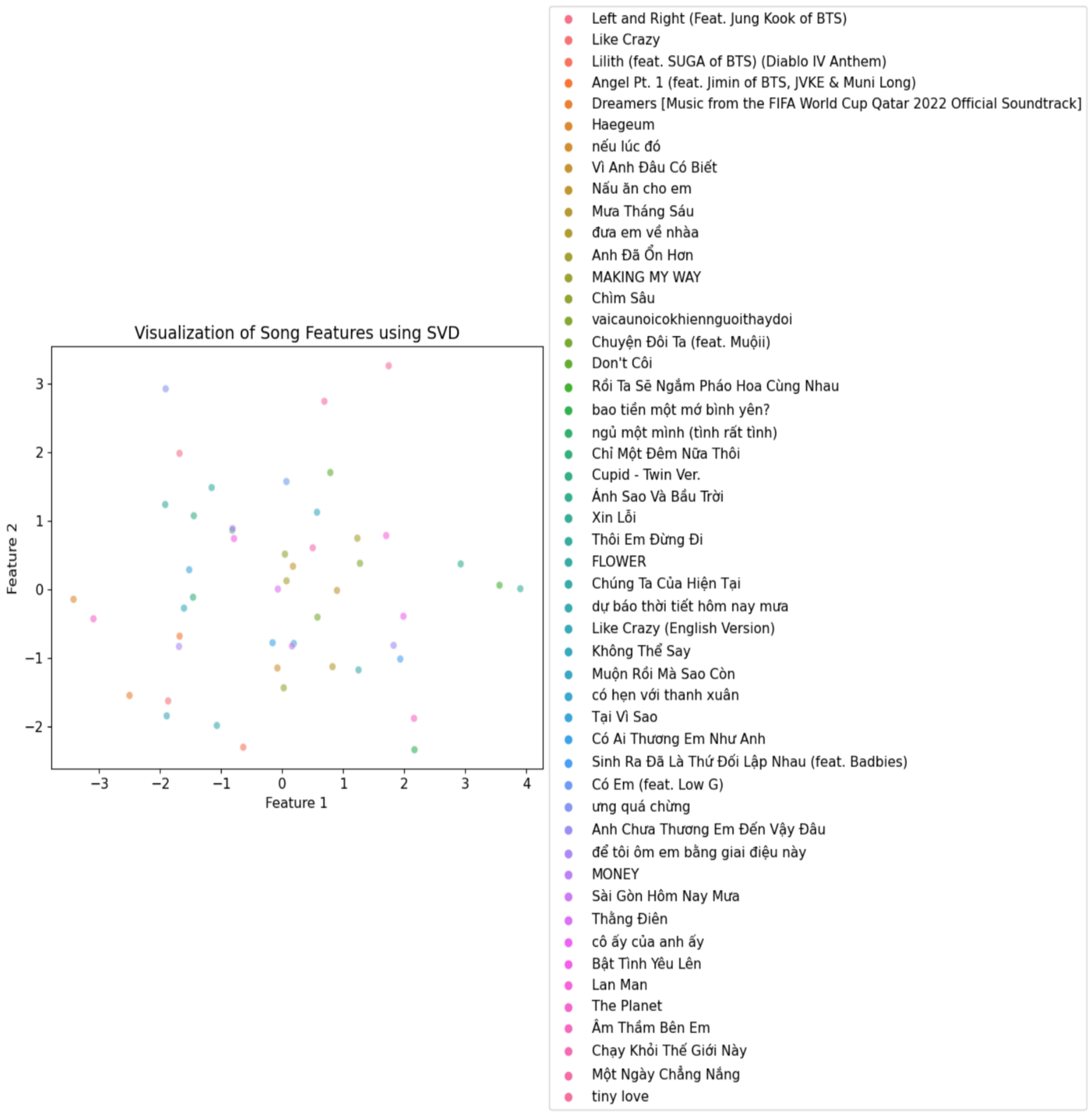
SVD PLOT FOR JAPAN:



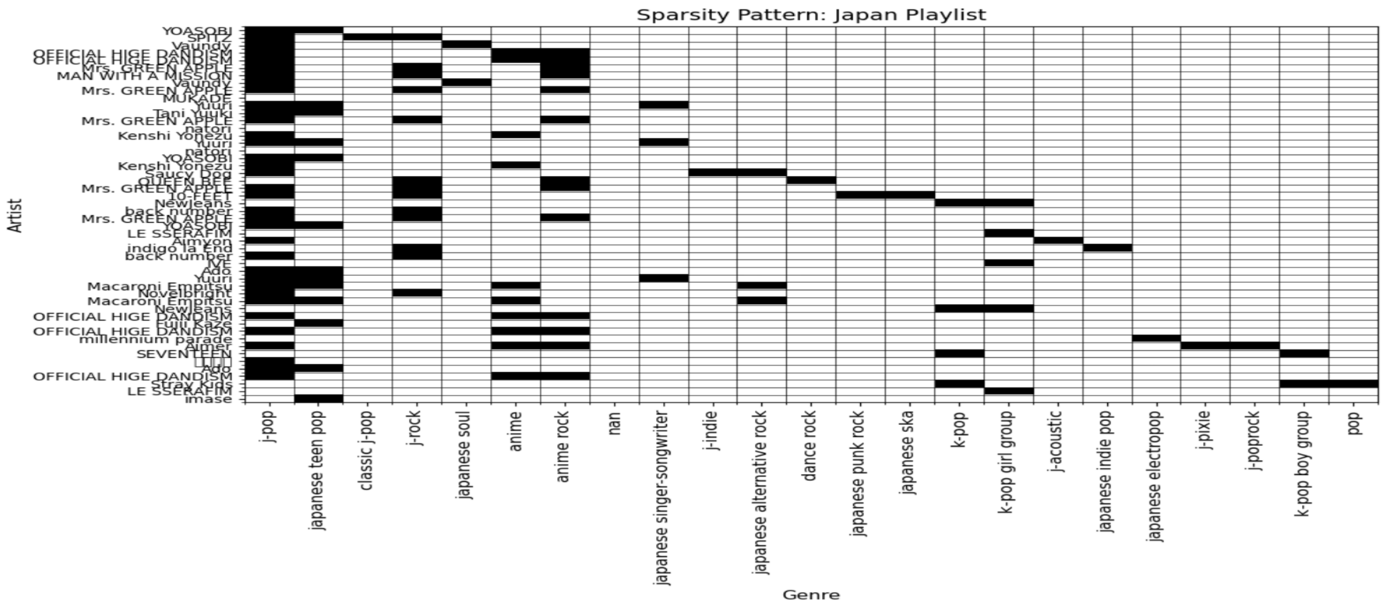
SVD PLOT FOR VIETNAM:

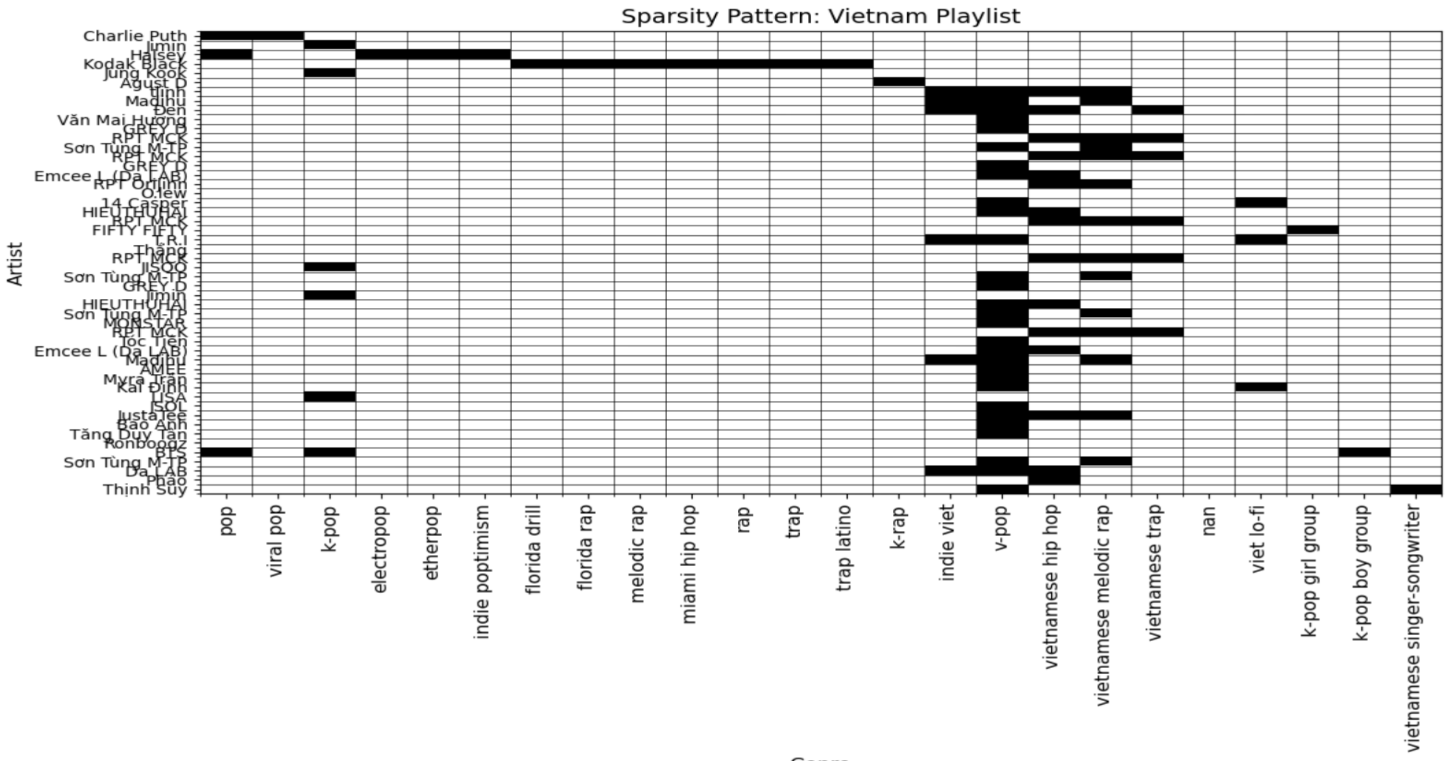


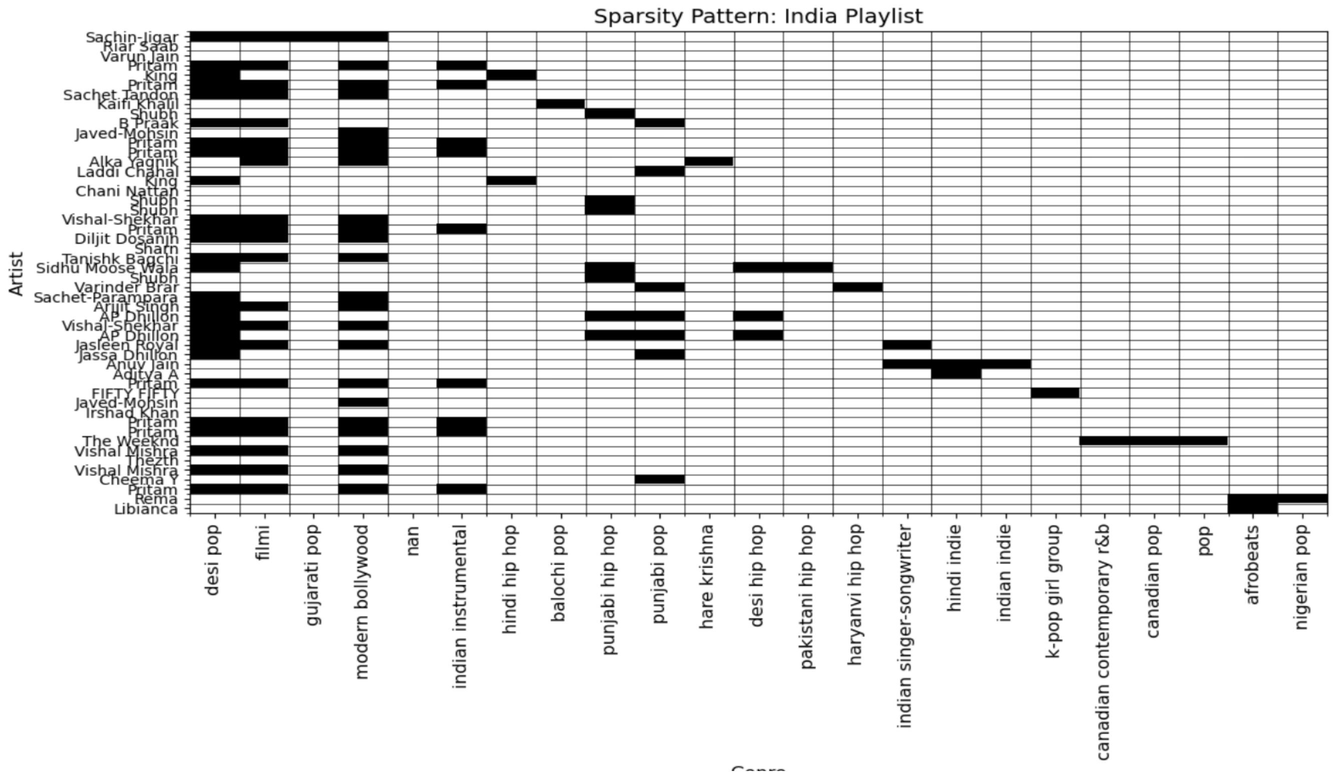
SVD PLOT FOR INDIA:



Sparsity Plot: Sparsity plots, also known as binary heatmaps, are visual representations that depict the sparsity pattern or the presence/absence of certain features or elements in a matrix. In the given code, a sparsity plot is created to analyze the distribution of genres across the tracks in the India playlist. The code snippet provided utilizes the matplotlib library in Python to create the sparsity plot. Here's a breakdown of how it works: First, the necessary libraries, numpy, and matplotlib.pyplot, are imported. The Japan playlist URI is assigned to the variable jap_playlist_URI. Similarly we use viet_playlist_URI and ind_playlist_URI for the tracks for Vietnam and India respectively. We use the code to retrieve track information from all three playlist using the Spotify API. It fetches the tracks for each country from the playlist and stores them in their respective variables. We then extract additional information like track URI, track name, artist name, artist popularity, and artist genres from each track in all three country playlists and store them in separate lists. Three DataFrames called jap_tracks,viet_tracks,and ind_tracks are created using the extracted information, combining all the lists for the given country A binary matrix called sparsity_matrix is initialized with zeros. The dimensions of the matrix are set to the number of tracks and the number of unique genres, respectively. The code then iterates over each track in the ind_tracks DataFrame and populates the sparsity_matrix with ones at the corresponding indices, representing the presence of a genre for a particular track. The figure and axes for the plot are created, specifying the size and font size. The sparsity pattern is plotted using imshow function, which visualizes the binary matrix as a heatmap with black and white cells. A binary colormap ('binary') is used. WE set the aspect ratio 'auto' for proper cell alignment. The tick labels for the x-axis and y-axis are customized to display the unique genres and artist names, respectively. The rotation of x-axis labels is set to 'vertical' for better readability. The font size for y-axis labels is also adjusted. We set the Y axis as the name of the artist for the given song because a lot of the song names have non-english characters and cannot be displayed by the compiler. This way, although there is a repetition on the Y-axis for an artist that has multiple songs in the top 50 charts, we are at least able to display some information relating to the origin of the song and not have a blank label. Finally, we display the graphs for each country using plt.show(). The sparsity_matrix and DataFrames provide a visual representation of the distribution of genres in each of the three top 50 playlists. The sparsity plot helps to identify patterns, clusters, or gaps in genre preferences among the artists and tracks in the playlist. By examining the plot, one can gain insights into the diversity or concentration of genres and understand the overall composition of the playlist. We then compare the sparsity plots for each country to see which genres are the most popular in each country.




Sparsity Pattern: Vietnam Playlist
Artist
Charlie Puth
Jimin
Halsey
Kodak Black
Jung Kook
Agust D
tlinh
Madihu
Đen
Văn Mai Hương
GREY D
RPT MCK
Sơn Tùng M-TP
RPT MCK
GREY D
Emcee L (Da LAB)
RPT Orijinn
O.lew
14 Casper
HIEUTHUHAI
RPT MCK
FIFTY FIFTY
T.R.I
Thắng
RPT MCK
JISOO
Sơn Tùng M-TP
GREY D
Jimin
HIEUTHUHAI
Sơn Tùng M-TP
MONSTAR
RPT MCK
Tóc Tiên
Emcee L (Da LAB)
Madihu
AMEE
Myra Trần
Kai Đinh
LISA
JSOL
JustaTee
Bảo Anh
Tăng Duy Tân
Ronboogz
BTS
Sơn Tùng M-TP
Da LAB
Pháo
Thịnh Suy
pop
viral pop
k-pop
electropop
etherpop
indie poptimism
florida drill
florida rap
melodic rap
miami hip hop
rap
trap
trap latino
k-rap
indie viet
v-pop
vietnamese hip hop
vietnamese melodic rap
vietnamese trap
nan
viet lo-fi
k-pop girl group
k-pop boy group
vietnamese singer-songwriter


Sparsity Pattern: India Playlist
Artist
Sachin-Jigar
Riar Saab
Varun Jain
Pritam
King
Pritam
Sachet Tandon
Kaifi Khalil
Shubh
B Praak
Javed-Mohsin
Pritam
Pritam
Alka Yagnik
Laddi Chahal
King
Chani Nattan
Shubh
Shubh
Vishal-Shekhar
Pritam
Diljit Dosanjh
Sharn
Tanishk Bagchi
Sidhu Moose Wala
Shubh
Varinder Brar
Sachet-Parampara
Arijit Singh
AP Dhillon
Vishal-Shekhar
AP Dhillon
Jasleen Royal
Jassa Dhillon
Anuv Jain
Aditya A
Pritam
FIFTY FIFTY
Javed-Mohsin
Irshad Khan
Pritam
Pritam
The Weeknd
Vishal Mishra
Thezth
Vishal Mishra
Cheema Y
Pritam
Rema
Libianca
desi pop
filmi
gujarati pop
modern bollywood
nan
indian instrumental
hindi hip hop
balochi pop
punjabi hip hop
punjabi pop
hare krishna
desi hip hop
pakistani hip hop
haryanvi hip hop
indian singer-songwriter
hindi indie
indian indie
k-pop girl group
canadian contemporary r&b
canadian pop
pop
afrobeats
nigerian pop

Conclusion: We originally started off looking at the Spotify datasets of the three countries in different stages of development. We were working to look at the top 50 songs for each country and their features in order to see if there are any deviations. Since each of the three countries is at a different stage of development, we wanted to see if there is a correlation between listening patterns and the development stage of the country.

At first glance, the songs for each country were centered around their culture. For example, Japan's top genre was J-pop, Vietnam's top genre was V-pop, and India's top genre was Desi pop. When we took a closer look at the specific top songs, the artists matched the idea that the songs were centered around culture for Japan and India, but for Vietnam the top artists of those songs were American. We have researched and believe that this could be because of American involvement and influence in the country during the Vietnam war. After applying sparsity patterns to all three countries we have confirmed that the genres are relative to the area.

In the track popularity distribution, we found the mean popularity scores for tracks in each country. In our findings, we saw that Japan had a relatively even distribution of popular tracks. However, Vietnam and India were significantly less diverse with the popularity scores for tracks hovering around 80 for India and 65 for Vietnam, without much data in other popularity score values.

We then used QR decomposition to find the five most relevant features so that we could narrow our analysis scope. After finding these features, we created a line graph that looked at the averages of the feature scores for each country and compared them on the plot. By looking at this we can see that there is a wide range of values from country to country when it comes to acousticness, while other features like danceability and valence were more similar. This shows us that across all the countries there are some features that are quite similar, and we have concluded that this is the case because they are in the same region.

Overall, we can see that there are noticeable differences in listening patterns between these three countries, but we are unable to say that these differences correlate in any meaningful way to the stages of development of each country.

Works Cited

Alexis, By: Marco. “What Is the Spotify Popularity Index?” *Two Story Melody*, 3 Mar. 2023, www.twostorymelody.com/spotify-popularity-index/. Accessed 6 Jun 2023.

Ning, Bo. Gaussian elimination (Lecture 3.2). STA 141C: Big Data & High Performance Statistical
Computing. University of California, Davis. Accessed 3 Jun 2023.

Ning, Bo. Iterative methods and singular value decomposition(Lecture 6.2). STA 141C: Big Data & High-Performance Statistical Computing. University of California, Davis. Accessed 3 Jun 2023.

Ning, Bo. QR decomposition(Lecture 5.1). STA 141C: Big Data & High-Performance Statistical Computing. University of California, Davis. Accessed 3 Jun 2023.

(NNK), Naveen. “Pandas - Check Any Value Is Nan in Dataframe.” *Spark By {Examples}*, 31 Jan. 2023, www.sparkbyexamples.com/pandas/pandas-check-if-any-value-is-nan-in-a-dataframe/. Accessed 28 May 2023.

lightalchemistlightalchemist10k44 gold badges4646 silver badges5555 bronze badges, et al. “Visualize Sparsity Pattern with Intensity Using MATLAB Spy Function.” *Stack Overflow*, 1 Apr. 1960, www.stackoverflow.com/questions/18395117/visualize-sparsity-pattern-with-intensity-using-matlab-spy-function. Accessed 6 Jun 2023.

Oyedeji, Miracle. “Spotify Popularity Index: A Little Secret to Help You Leverage the Algorithm.” *Loudlab*, 29 Mar. 2022, www.loudlab.org/blog/spotify-popularity-leverage-algorithm/#:~:text=The%20Spotify%20Popularity%20Index%20is,on%20algorithmic%20playlists%20and%20recommendations.Accessed 3 Jun 2023.

Shahawy, Salma El. “Understanding Matrix Factorization for Recommender Systems.” *Medium*, 24 Jan. 2020, www.towardsdatascience.com/understanding-matrix-factorization-for-recommender-systems-4d3c5e67f2c9. Accessed 2 Jun 2023.

“Sparsity Pattern.” *Sparsity Pattern - an Overview | ScienceDirect Topics*, www.sciencedirect.com/topics/computer-science/sparsity-pattern.Accessed 6 June 2023.

“What Is Spotify?” *Spotify*, www.support.spotify.com/us/article/what-is-spotify/. Accessed 6 June 2023.

APPENDIX

For our appendix each cell is separated by a horizontal line

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.linalg as la
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
```

```
#importing packages and libraries

#Authentication details for spotify API
client_credentials_manager =
SpotifyClientCredentials(client_id='9b436bfad3ef429d924e3d07fdad8d74',
client_secret='3efb51b6ffb346889d452f04a5af36e0')
sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

```
#Loading in data for Japan

jap_playlist = "https://open.spotify.com/playlist/37i9dQZEVXbKXQ4mDTEBXq"
jap_playlist_URI = jap_playlist.split("/")[-1].split("?")[0]
jap_track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(jap_playlist_URI)["items"]]

jap_track_uri = []
jap_track_name = []
jap_artist_uris = []
jap_artist_name = []
jap_artist_pop = []
jap_artist_genres = []
jap_album = []
jap_track_pop = []
#Performs for loop for the Japan playlist.
jap_playlist_tracks = sp.playlist_tracks(jap_playlist_URI)["items"]
```

```
jap_track_uri = [track["track"]["uri"] for track in jap_playlist_tracks]
jap_track_name = [track["track"]["name"] for track in jap_playlist_tracks]
jap_artist_uris = [track["track"]["artists"][0]["uri"] for track in jap_playlist_tracks]
jap_artist_info = [sp.artist(uri) for uri in jap_artist_uris]
jap_artist_name = [track["track"]["artists"][0]["name"] for track in jap_playlist_tracks]
jap_artist_pop = [artist["popularity"] for artist in jap_artist_info]
jap_artist_genres = [artist["genres"] for artist in jap_artist_info]
jap_album = [track["track"]["album"]["name"] for track in jap_playlist_tracks]
jap_track_pop = [track["track"]["popularity"] for track in jap_playlist_tracks]

#gets the top 50 tracks with the artist, track popularity, artist popularity, genre, album name, and
URI.
jap_tracks=pd.DataFrame({
    "track name": jap_track_name,
    "track popularity": jap_track_pop,
    "artist(s)": jap_artist_name,
    "artist populatrity": jap_artist_pop,
    "genre(s)": jap_artist_genres,
    "album name": jap_album,
    "uri": jap_track_uri
})
jap_tracks
```

```
# checking if data has any null values
jap_tracks.isnull().values.any()
```

```
#Represents a TfidfVectorizer object.
vectorizer = TfidfVectorizer()

#transforms the artist names into a document-term matrix.
dtm = vectorizer.fit_transform(jap_tracks['artist(s)'])

#Represents a TruncatedSVD object.
svd = TruncatedSVD(n_components=5)

#Transforms the document-term matrix into a latent semantic analysis matrix.
lsa = svd.fit_transform(dtm)
lsa
```

```
jap_genres = jap_tracks['genre(s)'].explode().unique()

#Makes a sparsity matrix.
sparsity_matrix = np.zeros((len(jap_tracks), len(jap_genres)), dtype=int)

#Populates the sparsity matrix by using for loops.
for i, track in jap_tracks.iterrows():
    genres = track['genre(s)']
    for genre in genres:
        genre_index = np.where(jap_genres == genre)[0][0]
        sparsity_matrix[i, genre_index] = 1

#Sets the plot features.
fig, ax = plt.subplots(figsize=(12, 8))
plt.rcParams.update({'font.size': 12})

#Plots the sparsity pattern.
im = ax.imshow(sparsity_matrix, cmap='binary', aspect='auto')

#Customizes the tick labels.
ax.set_xticks(np.arange(len(jap_genres)))
ax.set_xticklabels(jap_genres, rotation='vertical')
ax.set_yticks(np.arange(len(jap_tracks)))
ax.set_yticklabels(jap_tracks['artist(s)'], fontsize=10)

#Adds the grid lines.
ax.set_xticks(np.arange(-0.5, len(jap_genres)), minor=True)
ax.set_yticks(np.arange(-0.5, len(jap_tracks)), minor=True)
ax.grid(which='minor', color='black', linestyle='-', linewidth=0.5)

#Sets the title and axes.
plt.title('Sparsity Pattern: Japan Playlist', fontsize=14)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Artist', fontsize=12)
plt.tight_layout()

plt.show()
```

```
#Gets the audio features for Japan.
var=pd.DataFrame.from_dict(sp.audio_features(jap_track_uris))
var.head()

# checking if data has any null values
var.isnull().values.any()
```

```
#load in data for Vietnam

viet_playlist = "https://open.spotify.com/playlist/37i9dQZEVXbLdGSmz6xilI"
viet_playlist_URI = viet_playlist.split("/")[-1].split("?")[0]
viet_track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(viet_playlist_URI)["items"]]

viet_track_uri = []
viet_track_name = []
viet_artist_uris = []
viet_artist_name = []
viet_artist_pop = []
viet_artist_genres = []
viet_album = []
viet_track_pop = []
#Performs for loop for the Vietnam playlist.
viet_playlist_tracks = sp.playlist_tracks(viet_playlist_URI)["items"]

viet_track_uri = [track["track"]["uri"] for track in viet_playlist_tracks]
viet_track_name = [track["track"]["name"] for track in viet_playlist_tracks]
viet_artist_uris = [track["track"]["artists"][0]["uri"] for track in viet_playlist_tracks]
viet_artist_info = [sp.artist(uri) for uri in viet_artist_uris]
viet_artist_name = [track["track"]["artists"][0]["name"] for track in viet_playlist_tracks]
viet_artist_pop = [artist["popularity"] for artist in viet_artist_info]
viet_artist_genres = [artist["genres"] for artist in viet_artist_info]
viet_album = [track["track"]["album"]["name"] for track in viet_playlist_tracks]
viet_track_pop = [track["track"]["popularity"] for track in viet_playlist_tracks]


#Gets the top 50 tracks with the artist, track popularity, artist popularity, genre, album name, and
URI.
viet_tracks=pd.DataFrame({
   "track name": viet_track_name,
   "track popularity": viet_track_pop,
   "artist(s)": viet_artist_name,
   "artist populatrity": viet_artist_pop,
   "genre(s)": viet_artist_genres,
   "album name": viet_album,
   "uri": viet_track_uri
})
viet_tracks
```

```
viet_tracks.isnull().values.any()

#Gets the genres in the Vietnam playlist.
viet_genres = viet_tracks['genre(s)'].explode().unique()

#Creates a sparsity matrix.
sparsity_matrix = np.zeros((len(viet_tracks), len(viet_genres)), dtype=int)

#Populates the sparsity matrix.
for i, track in viet_tracks.iterrows():
    genres = track['genre(s)']
    for genre in genres:
        genre_index = np.where(viet_genres == genre)[0][0]
        sparsity_matrix[i, genre_index] = 1

#Sets the plot details.
fig, ax = plt.subplots(figsize=(12, 8))
plt.rcParams.update({'font.size': 12})

#Plots the sparsity pattern.
im = ax.imshow(sparsity_matrix, cmap='binary', aspect='auto')

#Customizes the tick labels.
ax.set_xticks(np.arange(len(viet_genres)))
ax.set_xticklabels(viet_genres, rotation='vertical')
ax.set_yticks(np.arange(len(viet_tracks)))
ax.set_yticklabels(viet_tracks['artist(s)'], fontsize=10)

#Adds grid lines.
ax.set_xticks(np.arange(-0.5, len(viet_genres)), minor=True)
ax.set_yticks(np.arange(-0.5, len(viet_tracks)), minor=True)
ax.grid(which='minor', color='black', linestyle='-', linewidth=0.5)

#Sets the axes and title.
plt.title('Sparsity Pattern: Vietnam Playlist', fontsize=14)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Artist', fontsize=12)
plt.tight_layout()

plt.show()
```

```
#Gets the audio features for Vietnam.
var=pd.DataFrame.from_dict(sp.audio_features(viet_track_uris))
```

```
var.head()


var.isnull().values.any()
```

```
#loading in data for India

ind_playlist = "https://open.spotify.com/playlist/37i9dQZEVXbLZ52XmnySJg"
ind_playlist_URI = ind_playlist.split("/")[-1].split("?")[0]
ind_track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(ind_playlist_URI)["items"]]

ind_track_uri = []
ind_track_name = []
ind_artist_uris = []
ind_artist_name = []
ind_artist_pop = []
ind_artist_genres = []
ind_album = []
ind_track_pop = []
ind_playlist_tracks = sp.playlist_tracks(ind_playlist_URI)["items"]

ind_track_uri = [track["track"]["uri"] for track in ind_playlist_tracks]
ind_track_name = [track["track"]["name"] for track in ind_playlist_tracks]
ind_artist_uris = [track["track"]["artists"][0]["uri"] for track in ind_playlist_tracks]
ind_artist_info = [sp.artist(uri) for uri in ind_artist_uris]
ind_artist_name = [track["track"]["artists"][0]["name"] for track in ind_playlist_tracks]
ind_artist_pop = [artist["popularity"] for artist in ind_artist_info]
ind_artist_genres = [artist["genres"] for artist in ind_artist_info]
ind_album = [track["track"]["album"]["name"] for track in ind_playlist_tracks]
ind_track_pop = [track["track"]["popularity"] for track in ind_playlist_tracks]

#Gets the top 50 tracks globally with the artist, track popularity, artist popularity, genre,
#album name, and URI
ind_tracks=pd.DataFrame({
    "track name": ind_track_name,
    "track popularity": ind_track_pop,
    "artist(s)": ind_artist_name,
    "artist populatrity": ind_artist_pop,
    "genre(s)": ind_artist_genres,
    "album name": ind_album,
    "uri": ind_track_uri
})
ind_tracks

ind_tracks.isnull().values.any()
```

```
#getting the genres in the Ind playlist
ind_genres = ind_tracks['genre(s)'].explode().unique()

# Create a binary matrix representing the sparsity pattern
sparsity_matrix = np.zeros((len(ind_tracks), len(ind_genres)), dtype=int)

# Iterate over the tracks and populate the sparsity matrix
for i, track in ind_tracks.iterrows():
    genres = track['genre(s)']
    for genre in genres:
        genre_index = np.where(ind_genres == genre)[0][0]
        sparsity_matrix[i, genre_index] = 1

#seting plot details
fig, ax = plt.subplots(figsize=(12, 8))
plt.rcParams.update({'font.size': 12})

#Plotting the sparsity pattern
im = ax.imshow(sparsity_matrix, cmap='binary', aspect='auto')

#Customizing tick labels
ax.set_xticks(np.arange(len(ind_genres)))
ax.set_xticklabels(ind_genres, rotation='vertical')
ax.set_yticks(np.arange(len(ind_tracks)))
ax.set_yticklabels(ind_tracks['artist(s)'], fontsize=10)

#Add the grid lines
ax.set_xticks(np.arange(-0.5, len(ind_genres)), minor=True)
ax.set_yticks(np.arange(-0.5, len(ind_tracks)), minor=True)
ax.grid(which='minor', color='black', linestyle='-', linewidth=0.5)

#Sets the title, axis labels, and adjusts the layout.
plt.title('Sparsity Pattern: India Playlist', fontsize=14)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Artist', fontsize=12)
plt.tight_layout()

plt.show()
```

```
var=pd.DataFrame.from_dict(sp.audio_features(ind_track_uri))
var.head()

var..isnull().values.any()
```

```
#Gets the audio features for the top 50 songs using spotipy audio features
jap_var = pd.DataFrame.from_dict(sp.audio_features(jap_track_uris)).T
jap_var.columns = jap_track_uris
jap_var.head()
```

```
#Creates an empty list to store song titles.
jap_track_titles = []

#Iterates over the track URIs and fetch the song titles.
for track_uri in jap_track_uris:
    track_info = sp.track(track_uri)
    jap_track_titles.append(track_info["name"])

#Gets the audio features for the top 50 songs using spotipy audio features.
jap_var = pd.DataFrame.from_dict(sp.audio_features(jap_track_uris)).T
jap_var.columns = jap_track_titles
jap_var
```

```
#gets audio features for the top 50 songs using spotipy audio features
viet_var = pd.DataFrame.from_dict(sp.audio_features(viet_track_uris)).T
viet_var.columns = viet_track_uris
viet_var.head()
```

```
#Creates an empty list to store song titles.
viet_track_titles = []

#Iterates over the track URIs and gets the song titles.
for track_uri in viet_track_uris:
    track_info = sp.track(track_uri)
    viet_track_titles.append(track_info["name"])

#Gets audio features for the top 50 songs using spotipy audio features.
viet_var = pd.DataFrame.from_dict(sp.audio_features(viet_track_uris)).T
viet_var.columns = viet_track_titles
viet_var
```

```
#Creates an empty list to store song titles.
ind_track_titles = []
```

```
#Iterates over the track URIs and fetch the song titles
for track_uri in ind_track_uris:
    track_info = sp.track(track_uri)
    ind_track_titles.append(track_info["name"])

#Gets audio features for the top 50 songs using spotipy audio features.
ind_var = pd.DataFrame.from_dict(sp.audio_features(ind_track_uris)).T
ind_var.columns = ind_track_titles
ind_var
```

```
jap_var = pd.DataFrame.from_dict(sp.audio_features(jap_track_uris))
viet_var = pd.DataFrame.from_dict(sp.audio_features(viet_track_uris))
ind_var = pd.DataFrame.from_dict(sp.audio_features(ind_track_uris))
jap_var['country'] = 'Japan'
viet_var['country'] = 'Vietnam'
ind_var['country'] = 'India'


merged_df = pd.concat([jap_var, viet_var, ind_var])
merged_df

import numpy as np

#Calculates the average audio feature values for each country.
avg_features = merged_df.groupby('country').mean()

#Gets the audio feature column names.
feature_columns = ['danceability', 'energy', 'acousticness', 'valence']

#Plots the line graph.
plt.figure(figsize=(10, 6))
for country in merged_df['country'].unique():
    country_values = avg_features.loc[country, feature_columns].values
    plt.plot(np.arange(len(feature_columns)), country_values, 'o-', label=country)

plt.xlabel('Audio Features')
plt.ylabel('Average Feature Value')
plt.title('Average Audio Feature Comparison by Country')
plt.xticks(np.arange(len(feature_columns)), feature_columns)
plt.ylim(0, 1)
plt.legend()
plt.show()

merged_df = pd.concat([jap_var, viet_var, ind_var])
```

```
#Divides the value of tempo by 500.
merged_df['tempo'] = merged_df['tempo'] / 500

#Calculates the average audio feature values per country.
avg_features = merged_df.groupby('country').mean()

# Get the audio feature column names.
feature_columns = ['danceability', 'energy', 'acousticness', 'valence', 'tempo']
```

```
#Converts the non-numeric columns to NaN.
jap_var_numeric = jap_var.apply(pd.to_numeric, errors='coerce')

#Replaces the missing values with zeros.
jap_var_numeric = jap_var_numeric.fillna(0)

#Drops the non-numeric columns.
jap_var_numeric = jap_var_numeric.select_dtypes(include=[np.number])

#Converts the DataFrame to a matrix.
jap_var_matrix = jap_var_numeric.values

#Performs the QR decomposition.
Q, R = la.qr(jap_var_matrix)

#Reconstructs the data matrix.
reconstructed_matrix = np.dot(Q, R)

#Calculates the residuals.
residuals = jap_var_matrix - reconstructed_matrix

#Calculates the sum of the squared residuals.
sum_squared_residuals = np.sum(residuals**2)

#Prints the sum of squared residuals.
print("Sum of Squared Residuals:", sum_squared_residuals)
```

```
#Converts the non-numeric columns to NaN.
jap_var_numeric = jap_var.apply(pd.to_numeric, errors='coerce')

#Replaces the missing values with zeros.
jap_var_numeric = jap_var_numeric.fillna(0)

#Drops the non-numeric columns.
jap_var_numeric = jap_var_numeric.select_dtypes(include=[np.number])
```

```
#Converts the DataFrame to a matrix.
jap_var_matrix = jap_var_numeric.values

#Performs the QR decomposition.
Q, R = la.qr(jap_var_matrix)

#Computes the magnitudes of the Q matrix rows.
magnitudes = np.abs(Q)

#Calculates the importance scores for each audio feature.
importance_scores = np.sum(magnitudes, axis=1)

#Sorts the audio features based on their importance scores.
sorted_indices = np.argsort(importance_scores)[::-1]

#Gets the names of the top k audio features.
k = 5
top_feature_names = jap_var.columns[sorted_indices[:k]]

# Print the top feature names
print("Japan Playlist Top", k, "audio features:")
for feature_name in top_feature_names:
    print(feature_name)
```

```
#Converts the non-numeric columns to NaN.
viet_var_numeric = viet_var.apply(pd.to_numeric, errors='coerce')

#Replaces the missing values with zeros.
viet_var_numeric = viet_var_numeric.fillna(0)

#Drops the non-numeric columns.
viet_var_numeric = viet_var_numeric.select_dtypes(include=[np.number])

#Converts the DataFrame to a matrix.
viet_var_matrix = viet_var_numeric.values

#Performs the QR decomposition.
Q, R = la.qr(viet_var_matrix)

#Computes the magnitudes of the Q matrix rows.
magnitudes = np.abs(Q)

#Calculates the importance scores for each audio feature.
importance_scores = np.sum(magnitudes, axis=1)
```

```
#Sorts the audio features based on their importance scores.
sorted_indices = np.argsort(importance_scores)[::-1]

#Gets the names of the top k audio features.
k = 5
top_feature_names = viet_var.columns[sorted_indices[:k]]

#Prints the top feature names.
print("Vietnam Playlist Top", k, "audio features:")
for feature_name in top_feature_names:
    print(feature_name)
```

```
#Converts the non-numeric columns to NaN.
ind_var_numeric = ind_var.apply(pd.to_numeric, errors='coerce')

#Replaces the missing values with zeros.
ind_var_numeric = ind_var_numeric.fillna(0)

#Drops the non-numeric columns.
ind_var_numeric = ind_var_numeric.select_dtypes(include=[np.number])

#Converts the DataFrame to a matrix.
ind_var_matrix = ind_var_numeric.values

#Performs the QR decomposition.
Q, R = la.qr(ind_var_matrix)

#Computes the magnitudes of the Q matrix rows.
magnitudes = np.abs(Q)

#Calculates the importance scores for each audio feature.
importance_scores = np.sum(magnitudes, axis=1)

#Sorts the audio features based on their importance scores.
sorted_indices = np.argsort(importance_scores)[::-1]

#Gets the names of the top k audio features.
k = 5
top_feature_names = ind_var.columns[sorted_indices[:k]]

#Prints the top feature names.
print("India Playlist Top", k, "audio features:")
for feature_name in top_feature_names:
    print(feature_name)
```

```
viet_var=viet_var.T

#Selects the features you want to analyze.
features = viet_var[['danceability', 'energy', 'acousticness', 'liveness', 'valence', 'loudness',
'duration_ms']]

#Scales the features.
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

#Applies the Singular Value Decomposition (SVD) for dimensionality reduction.
svd = TruncatedSVD(n_components=2)
features_svd = svd.fit_transform(features_scaled)

#Converts the SVD transformed features array to a dataframe.
features_svd_df = pd.DataFrame(features_svd, columns=['Feature 1', 'Feature 2'])

#Adds the track titles as a column in the dataframe.
features_svd_df['Track Title'] = viet_track_titles

#Creates a scatter plot.
plt.figure(figsize=(8, 6))
scatterplot = sns.scatterplot(data=features_svd_df, x='Feature 1', y='Feature 2', hue='Track Title',
alpha=0.6)
plt.legend( loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Visualization of Song Features using SVD')

# Shows both figures.
plt.show()
```

```
#Selects the features you want to analyze.
features = jap_var[['danceability', 'energy', 'acousticness', 'liveness', 'valence', 'loudness',
'duration_ms']]

#Scales the features.
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

#Applies Singular Value Decomposition (SVD) for dimensionality reduction.
```

```
svd = TruncatedSVD(n_components=2)
features_svd = svd.fit_transform(features_scaled)

#Converts the SVD transformed features array to a dataframe.
features_svd_df = pd.DataFrame(features_svd, columns=['Feature 1', 'Feature 2'])

#Adds the track titles as a column in the dataframe.
features_svd_df['Track Title'] = jap_track_titles

#Creates a scatter plot.
plt.figure(figsize=(8, 6))
scatterplot = sns.scatterplot(data=features_svd_df, x='Feature 1', y='Feature 2', hue='Track Title',
alpha=0.6)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Visualization of Song Features using SVD')

#Shows the plot.
plt.show()
```

```
ind_var=ind_var.T

#Selects the features you want to analyze.
features = ind_var[['danceability', 'energy', 'acousticness', 'liveness', 'valence', 'loudness',
'duration_ms']]

#Scales the features.
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

#Applies Singular Value Decomposition (SVD) for dimensionality reduction.
svd = TruncatedSVD(n_components=2)
features_svd = svd.fit_transform(features_scaled)

#Converts the SVD transformed features array to a dataframe.
features_svd_df = pd.DataFrame(features_svd, columns=['Feature 1', 'Feature 2'])

#Adds track titles as a column in the dataframe.
features_svd_df['Track Title'] = ind_track_titles

#Creates a scatter plot.
plt.figure(figsize=(8, 6))
scatterplot = sns.scatterplot(data=features_svd_df, x='Feature 1', y='Feature 2', hue='Track Title',
alpha=0.6)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Visualization of Song Features using SVD')

#Show the plot.
plt.show()
```

```
#here we are creating X using artist populatrity
X = jap_tracks["artist populatrity"]
#here we are creating X using track populatrity
y = jap_tracks["track popularity"]

#here we are adding a constant so that we can have an intercept
X = sm.add_constant(X)

#here we are making a linear regression model
model = sm.OLS(y, X)
#here we are fitting the linear regression model
results = model.fit()
print(results.summary())
#here we find the AIC and BIC
AIC = results.aic
BIC = results.bic

print("AIC:", AIC)
print("BIC:", BIC)
```

```
#here we are creating X using artist populatrity
X = viet_tracks["artist populatrity"]
#here we are creating X using track populatrity
y = viet_tracks["track popularity"]

#here we are adding a constant so that we can have an intercept
X = sm.add_constant(X)

#here we are making a linear regression model
model = sm.OLS(y, X)
#here we are fitting the linear regression model
results = model.fit()
print(results.summary())
#here we find the AIC and BIC
AIC = results.aic
BIC = results.bic
```

```
print("AIC:", AIC)
print("BIC:", BIC)
```

```
#here we are creating X using artist populatrity
X = ind_tracks["artist populatrity"]
#here we are creating X using track populatrity
y = ind_tracks["track popularity"]

#here we are adding a constant so that we can have an intercept
X = sm.add_constant(X)

#here we are making a linear regression model
model = sm.OLS(y, X)
#here we are fitting the linear regression model
results = model.fit()
print(results.summary())
#here we find the AIC and BIC
AIC = results.aic
BIC = results.bic

print("AIC:", AIC)
print("BIC:", BIC)
```

```
#create ind_popularity using the track popularity column
jap_popularity = jap_tracks['track popularity']
#here we are calculating the length
jap_track_name_length = jap_tracks['track name'].apply(len)

#here we are finding the correlation coefficient
jap_correlation = jap_popularity.corr(jap_track_name_length)

print("Correlation coefficient (Japan Playlist):", jap_correlation)
```

```
#create ind_popularity using the track popularity column
viet_popularity = viet_tracks['track popularity']
#here we are calculating the length
viet_track_name_length = viet_tracks['track name'].apply(len)

#here we are finding the correlation coefficient
viet_correlation = viet_popularity.corr(viet_track_name_length)
```

```
print("Correlation coefficient (Vietnam Playlist):", viet_correlation)
```

```
#create ind_popularity using the track popularity column
ind_popularity = ind_tracks['track popularity']
#here we are calculating the length
ind_track_name_length = ind_tracks['track name'].apply(len)
#here we are finding the correlation coefficient
ind_correlation = ind_popularity.corr(ind_track_name_length)

print("Correlation coefficient (India Playlist):", ind_correlation)
```

```
#here we make an empty set jap_genres
jap_genres = set()
#here we add each genre into the set
for genres in jap_tracks['genre(s)']:
   jap_genres.update(genres)

#here we make an empty dictionary jap_genres
genre_counts = {}
#here we are summing up the number of each genre
for genre in jap_genres:
   genre_counts[genre] = sum(genre in genres for genres in jap_tracks['genre(s)'])

#here we are creating sorted_genres looking at the descending order
sorted_genres = sorted(genre_counts.items(), key=lambda x: x[1], reverse=True)

#here we are going over each genre and add in the sorted_genres
for genre, count in sorted_genres:
   print(genre, count)
```

```
#here we make an empty set viet_genres
viet_genres = set()
#here we add each genre into the set
for genres in viet_tracks['genre(s)']:
   viet_genres.update(genres)

#here we make an empty dictionary viet_genres
genre_counts = {}
#here we are summing up the number of each genre
for genre in viet_genres:
   genre_counts[genre] = sum(genre in genres for genres in viet_tracks['genre(s)'])
```

```
#here we are creating sorted_genres looking at the descending order
sorted_genres = sorted(genre_counts.items(), key=lambda x: x[1], reverse=True)

#here we are going over each genre and add in the sorted_genres
for genre, count in sorted_genres:
    print(genre, count)
```

```
#here we make an empty set ind_genres
ind_genres = set()
#here we add each genre into the set
for genres in ind_tracks['genre(s)']:
    ind_genres.update(genres)

#here we make an empty dictionary ind_genres
genre_counts = {}
#here we are summing up the number of each genre
for genre in ind_genres:
    genre_counts[genre] = sum(genre in genres for genres in ind_tracks['genre(s)'])

#here we are creating sorted_genres looking at the descending order
sorted_genres = sorted(genre_counts.items(), key=lambda x: x[1], reverse=True)

#here we are going over each genre and add in the sorted_genres
for genre, count in sorted_genres:
    print(genre, count)
```

```
#here we are plotting the histogram using the values track popularity
#20 bins, color grey
plt.hist(jap_tracks['track popularity'], bins=20, edgecolor='grey')
#label x axis: Track Pop
plt.xlabel('song Pop')
#label y axis:Count
plt.ylabel('frequency')
#label title: Japan Track Popularity Distribution
plt.title('Track Popularity Distribution for Japan')
plt.show()

#solve for mean
popularity_mean = jap_tracks['track popularity'].mean()
#solve for median
popularity_median = jap_tracks['track popularity'].median()
#solve for standard deviation
popularity_std = jap_tracks['track popularity'].std()
```

```
#print statistics
print('mean:', popularity_mean)
print('median:', popularity_median)
print('standard deviation:', popularity_std)
```

```
#here we are plotting the histogram using the values track popularity
#20 bins, color grey
plt.hist(viet_tracks['track popularity'], bins=20, edgecolor='grey')
#label x axis: Track Pop
plt.xlabel('song Pop')
#label y axis:Count
plt.ylabel('frequency')
#label title: Vietnam Track Popularity Distribution
plt.title('Track Popularity Distribution for Vietnam')
plt.show()

#solve for mean
popularity_mean = viet_tracks['track popularity'].mean()
#solve for median
popularity_median = viet_tracks['track popularity'].median()
#solve for standard deviation
popularity_std = viet_tracks['track popularity'].std()

#print statistics
print('mean:', popularity_mean)
print('median:', popularity_median)
print('standard deviation:', popularity_std)
```

```
#here we are plotting the histogram using the values track popularity
#20 bins, color grey
plt.hist(ind_tracks['track popularity'], bins=20, edgecolor='grey')
#label x axis: Track Pop
plt.xlabel('song Popularity')
#label y axis:Count
plt.ylabel('frequency')
#label title: India Track Popularity Distribution
plt.title('Track Popularity Distribution for India')
plt.show()

#solve for mean
popularity_mean = ind_tracks['track popularity'].mean()
#solve for median
popularity_median = ind_tracks['track popularity'].median()
```

```
#solve for standard deviation
popularity_std = ind_tracks['track popularity'].std()

#print statistics
print('mean:', popularity_mean)
print('median:', popularity_median)
print('standard deviation:', popularity_std)
```