ITC6001A1 - INTRODUCTION TO BIG DATA - FALL TERM 2023

Final Project - Last.FM dataset

Alkiviadis Kariotis 241735
Konstantinos Margaritis 122451

Prof.: Dr. Dimitrios Vogiatzis

Presentation Overview

Data Processing

Similarity Assessment

Correlation Analysis

Dynamics of Listening and Tagging

Libraries Used for the entire project

- pandas: Data manipulation
- numpy: Numerical computing
- matplotlib.pyplot: Data visualization
- seaborn: better plotting
- scipy.stats.zscore: Standardization
- sklearn.metrics.pairwise.cosine_similarity: Similarity measurement
- scipy.sparse.csr_matrix: Sparse matrix representation
- json: Data serialization
- plotly.express: Interactive plotting
- plotly.graph_objs: Plotly graph objects
- ipywidgets: Interactive widgets

Data Processing Overview

Number of unique artists	17632
Number of unique tags	11946
Number of unique users	1892

- Data Loading: Process of loading data using pandas.
  - Inspecting the files to see what the separator is
  - Load the 6 .data files through the use of pandas with tab (\t) delimiter
  - For encoding we used 'ISO-8859-1' (Latin-1 WE)
- Data Overview: Description of the Last.fm dataset.
  - For all of the 6 files checked key statistics and information, through the functions
    - .info()
    - .describe()

Datasets: artists, tags, user_artists, user_friends, user_taggedartists & user_taggedartists-timestamps

user_artists data fame

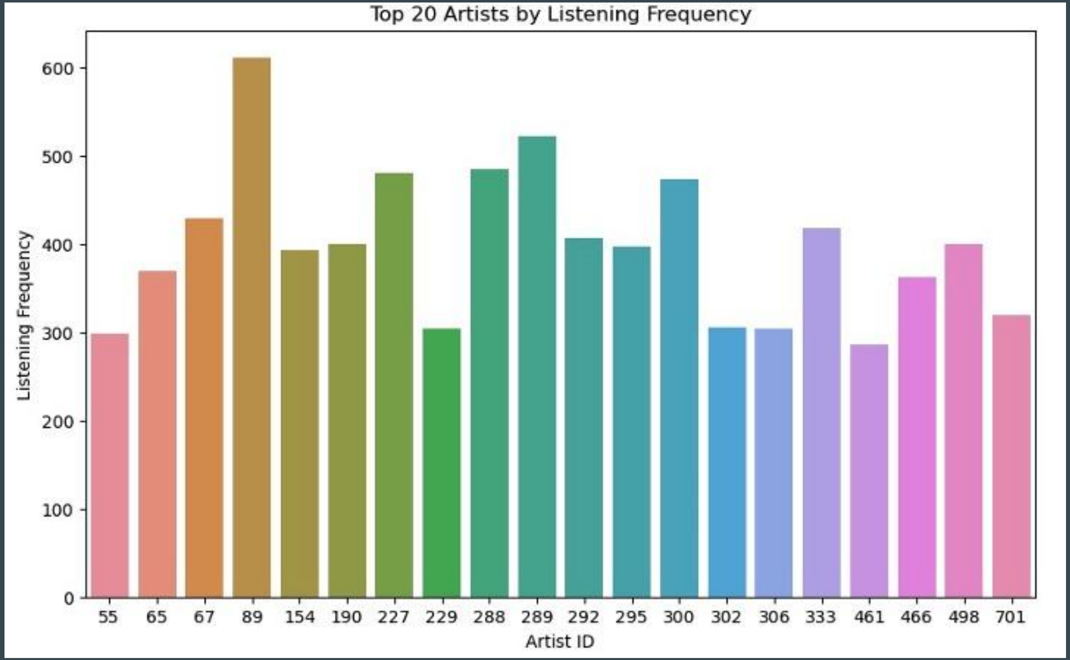
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92834 entries, 0 to 92833
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   userID    92834 non-null  int64
 1   artistID  92834 non-null  int64
 2   weight    92834 non-null  int64
dtypes: int64(3)
memory usage: 2.1 MB
None


             userID       artistID        weight
count  92834.000000   92834.000000   92834.00000
mean    1037.010481    3331.123145     745.24393
std      610.870436    4383.590502    3751.32208
min        2.000000       1.000000       1.00000
25%      502.000000     436.000000     107.00000
50%     1029.000000    1246.000000     260.00000
75%     1568.000000    4350.000000     614.00000
max     2100.000000   18745.000000  352698.00000
```

Visualization of Top20 artists by Listening Frequency

In the user_artists data frame we proceeded with:

- Slicing and keeping only the artistID column
- Counting the values (corresponds to frequency)
- Sorting the values
- Keeping the biggest 20 frequency values

```
artist_listening_frequency =
user_artists_df['artistID'].value_counts().
sort_values(ascending=False).head(20)
```

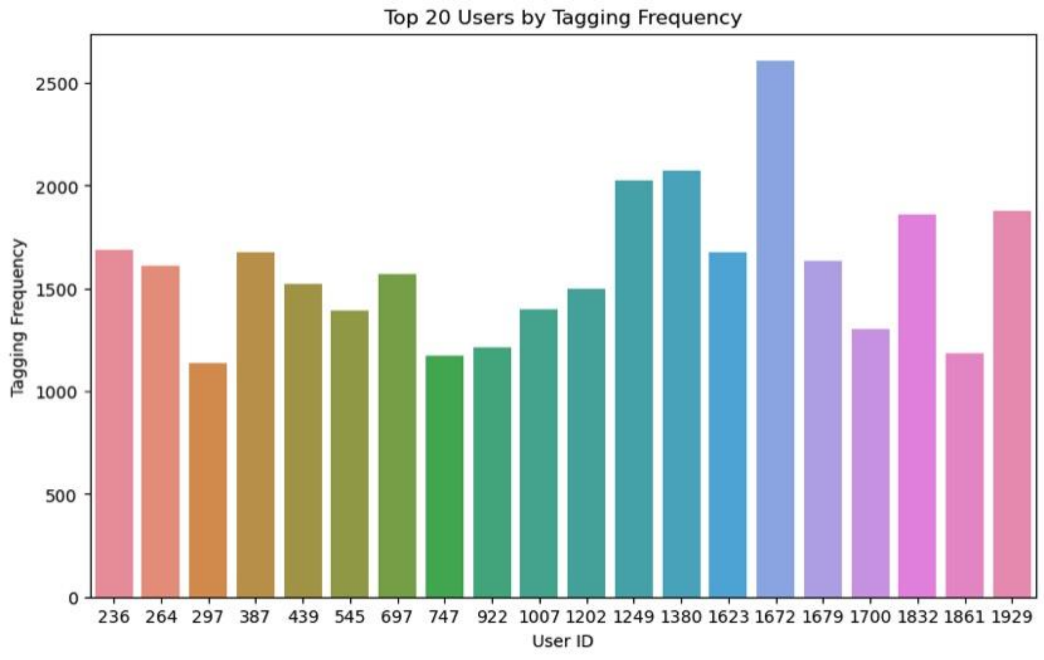


Visualization of Top20 users by Tagging Frequency

In the user_taggedartists data frame we proceeded with:

- Slicing and keeping only the userID column
- Counting the values (corresponds to frequency)
- Sorting the values
- Keeping the biggest 20 frequency values

tag_frequency = user_taggedartists_df['userID'].value_counts().sort_values(ascending=False).head(20)



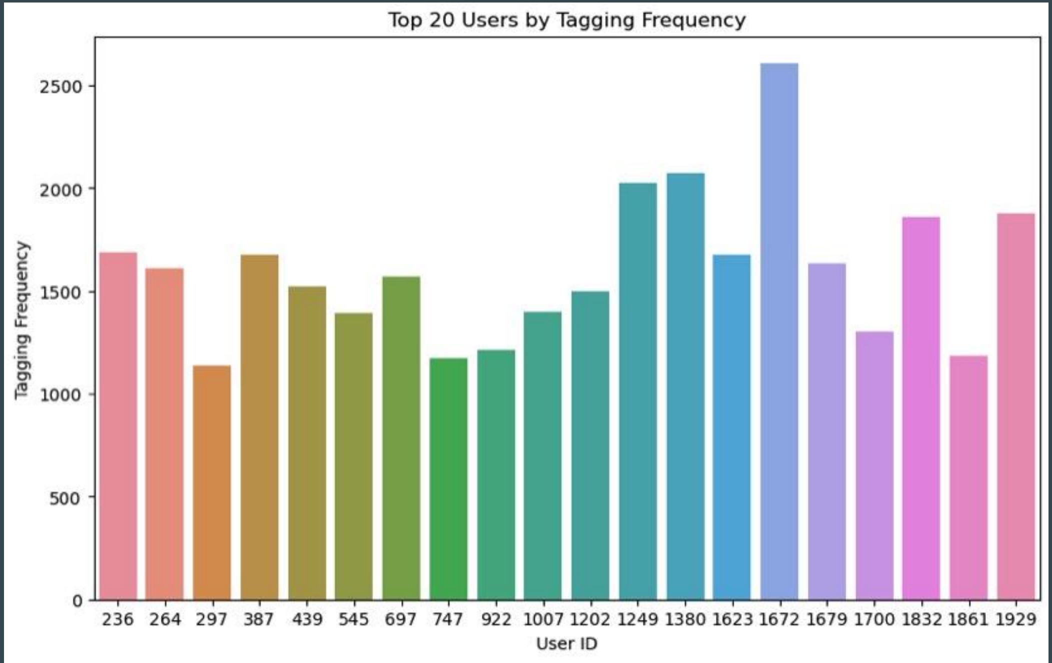
Visualization of Top20 users by Number of Friends

In the user_friends data frame we proceeded with:

- Slicing and keeping only the userID column
- Counting the values (corresponds to frequency)
- Sorting the values
- Keeping the 20 users with the most friends

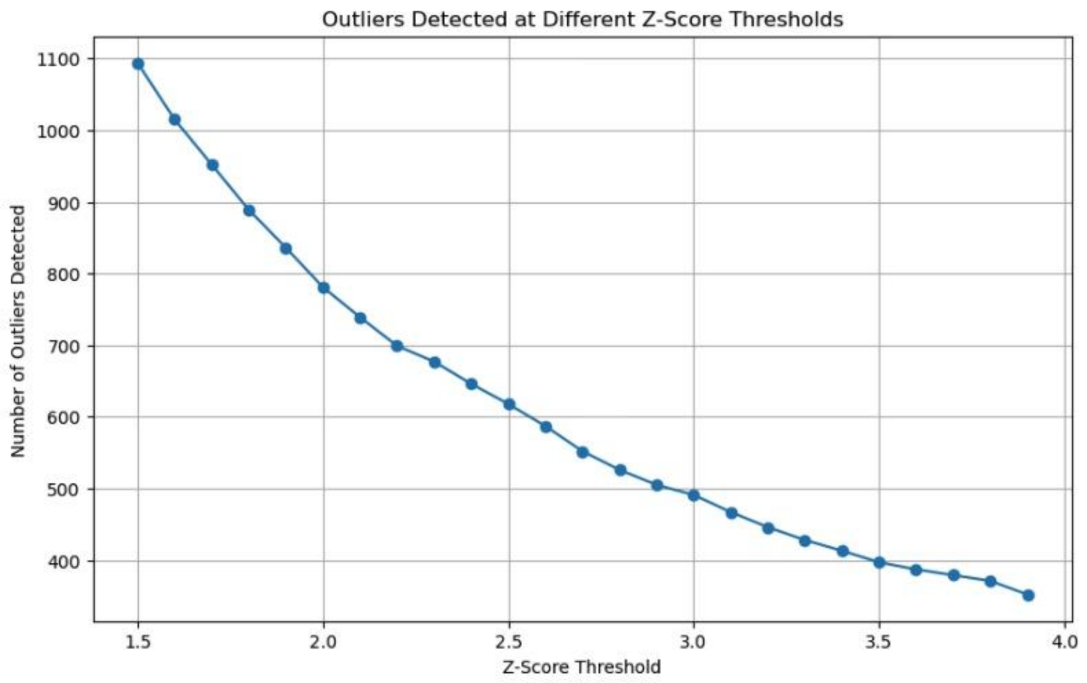
	userID	friendID
0	2	275
1	2	428
2	2	515
3	2	761
4	2	831
...	...	...
25429	2099	1801
25430	2099	2006
25431	2099	2016
25432	2100	586
25433	2100	607

25434 rows × 2 columns



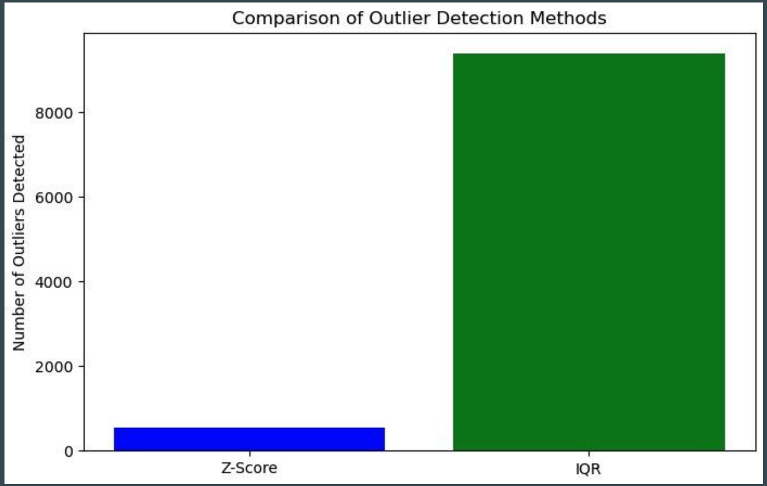
Outlier Detection | Z-score

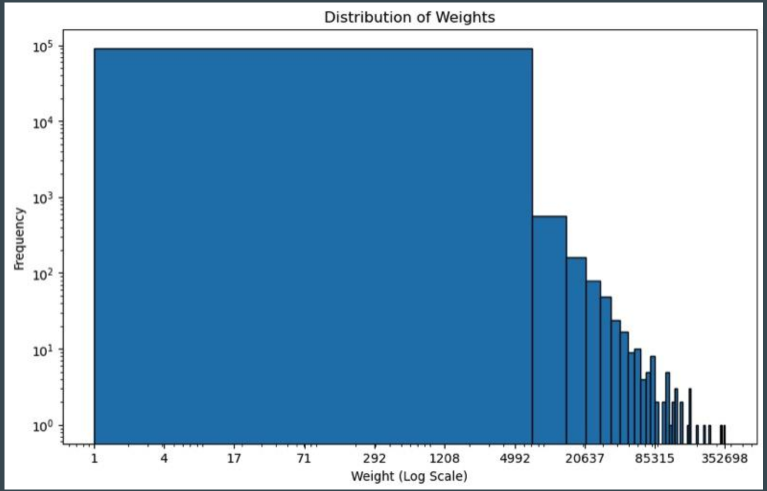
- Firstly, we looked at the listening frequencies (weight column of user_artists).
- We started by trying to estimate the optimal threshold for the z-score function.
- Picked values from 1.5-4 with step 0.1 and looped everything to get the plot based on the number of outliers detected.
- The area where the rate of decrease, of the number of outliers started to significantly slow down, should show the optimal threshold point.
- This is between 2.5 and 3, based on the plot.

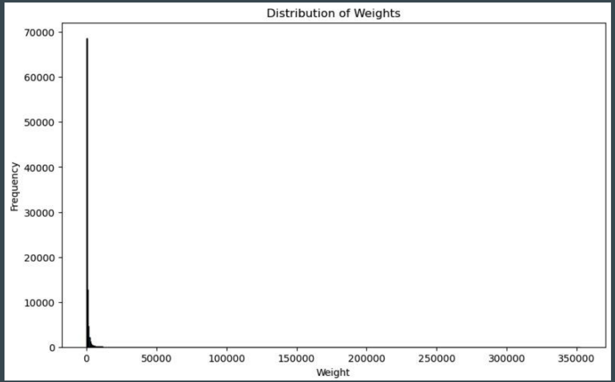


Outlier Detection | IQR

- The IQR (Interquartile Range) method for outliers involves calculating the range between the first quartile (25th percentile) and the third quartile (75th percentile) of a dataset and identifying any data points that fall below the first quartile minus 1.5 times the IQR or above the third quartile plus 1.5 times the IQR as potential outliers.
- On the listening frequency we applied this method, and the difference of outliers in terms of number was significant.
- Useful for identifying outliers in skewed distributions and less influenced by extreme values → robust application when dealing with non-normal distributions datasets → identifies more outlies than the z-score method.





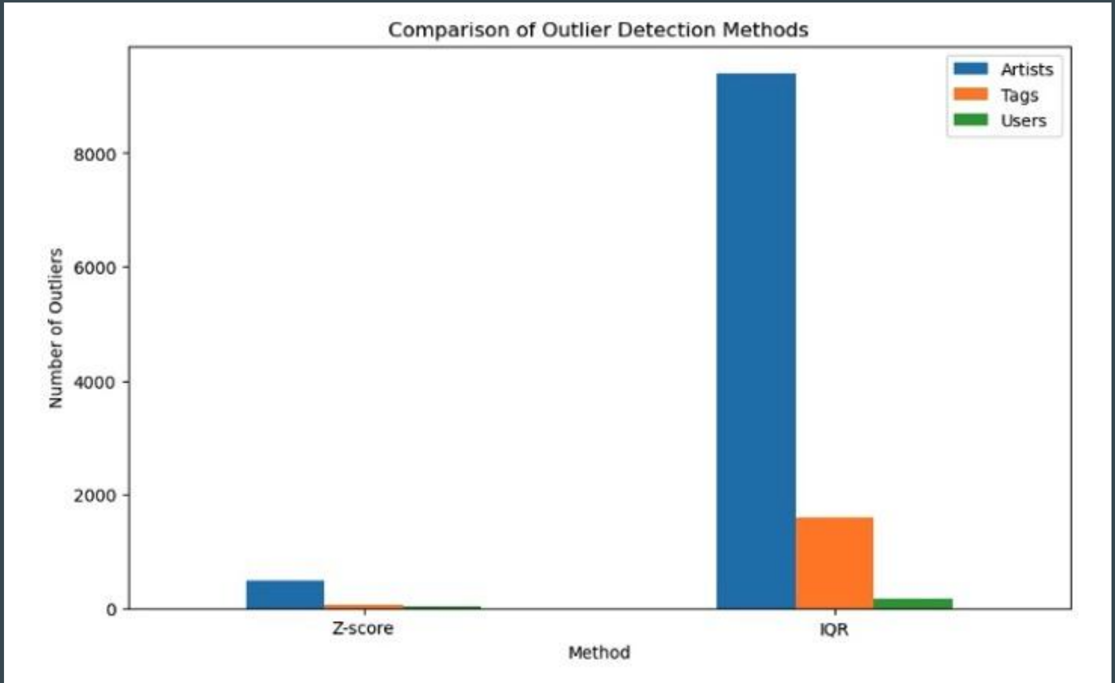


Outlier Detection | Both methods for artists, tags and users

- Then we calculated based on z-score and IQR for all three cases, by firstly finding the respective frequencies, and then applying the following formula: (Frequency - mean) / std

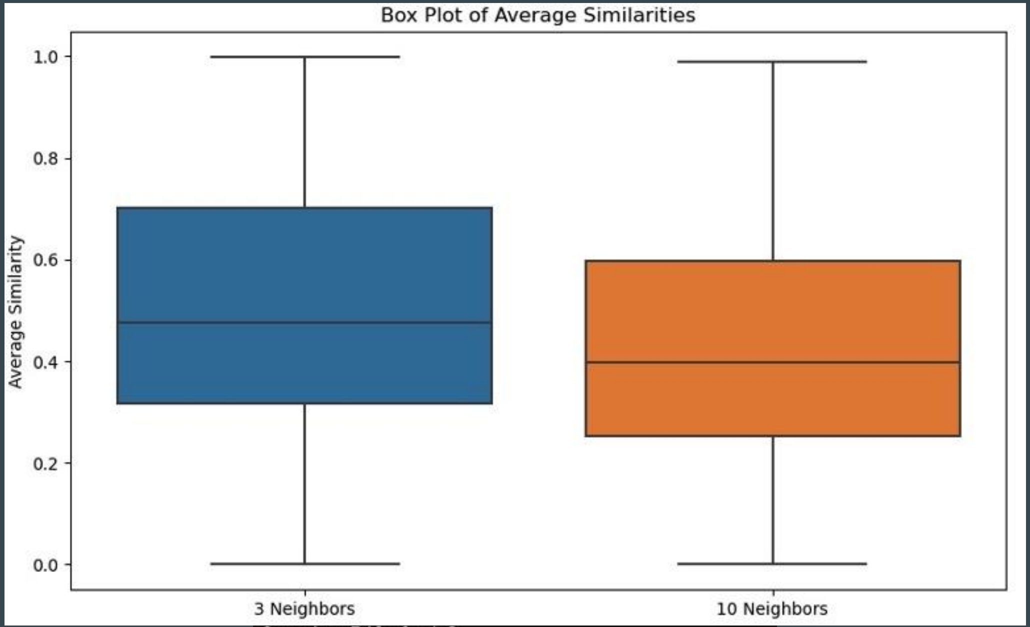
- Artists → Listening Frequency
- Tags → Tag usage Frequency
- Users → Listening time (weight_sum)

```
(       userID  artistID  weight     z_score
 0           2        51   13883    3.502167
 50          3       101   13176    3.313700
 254         7       288   43864   11.494283
 506        12       333   23961    6.188686
 508        12       377   59695   15.714395
 ...       ...       ...     ...         ...
 91717    2072      3489   12842    3.224665
 91986    2080        72   80721   21.319352
 92138    2084       207   12498    3.132964
 92551    2094      1459   26090    6.756220
 92649    2096       436   14690    3.717291

 [491 rows x 4 columns],
```



Similarity Assessment

- To identify the listening habits of the users, we performed a cosine similarity analysis.
- We created a pivot table, using as index the userID, as column the artistID and as values the weight.
- Then used the function of cosine similarity to find the angle between the non-zero vectors from the pivot table.
- Finally stored that to a csv file in order to be used with the KNN method.
- Here we identified the nearest neighbors for each of the two cases (3&10) based on the similarity scores of each user, stored in the csv file from above.




```
Comparison Table for k=3:
      User Nearest Neighbors (k=3)
0        2          428, 1210, 1866
1        3          1740, 255, 1891
2        4           1642, 62, 1163
3        5           567, 277, 1785
4        6          1533, 924, 1248
...    ...                      ...
1887  2095            1976, 601, 33
1888  2096          856, 1414, 1402
1889  2097           1642, 2080, 31
1890  2099          842, 1060, 1874
1891  2100             586, 108, 33

[1892 rows x 2 columns]

Comparison Table for k=10:
      User                           Nearest Neighbors (k=10)
0        2  428, 1210, 1866, 374, 1643, 1209, 1585, 1202, ...
1        3  1740, 255, 1891, 78, 60, 1054, 1638, 426, 951,...
2        4  1642, 62, 1163, 767, 2080, 124, 446, 103, 566, 31
3        5  567, 277, 1785, 913, 1797, 919, 1411, 171, 42,...
4        6  1533, 924, 1248, 498, 1697, 1663, 1791, 1435, ...
```

Dynamics of Listening and Tagging

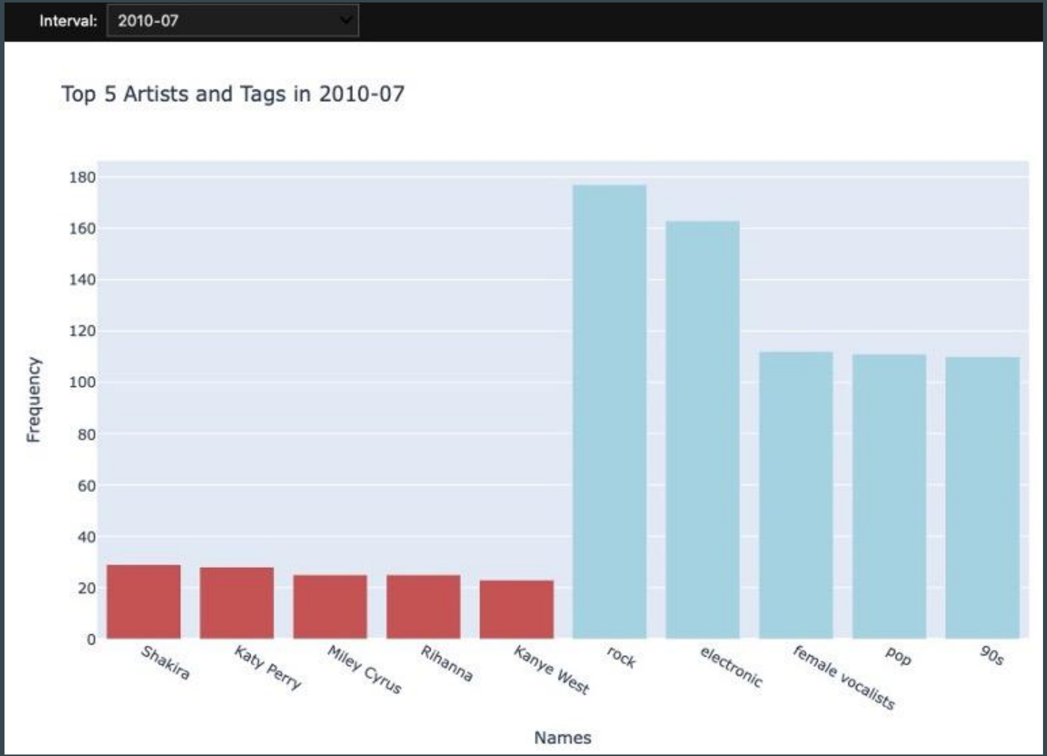
- We started by making the necessary transformation to the timestamp column, assigning also a safety function to turn milliseconds to seconds.
- We created a dynamically changing code for the choice of the time interval changing from monthly to trimester with just changing M to Q
- Then we grouped the data by interval and started to assigning to a dictionary the following values for each one of them:

```
#store results in a dictionary
results[interval] = {
    'unique_users': unique_users,
    'unique_tags': unique_tags,
    'unique_artists': unique_artists,
    'top_artists': top_artists,
    'top_tags': top_tags
```

```
Interval: 2005-07
  Number of unique users: 11
  Number of unique tags: 78
  Number of unique artists: 465
  Top 5 artists (by ID): [599, 59, 2861, 1083, 1013]
  Top 5 tags (by ID): [2191, 17, 370, 3496, 642]

Interval: 2005-08
  Number of unique users: 13
  Number of unique tags: 58
  Number of unique artists: 111
  Top 5 artists (by ID): [11892, 227, 234, 550, 14468]
  Top 5 tags (by ID): [73, 828, 371, 735, 1]

Interval: 2005-09
  Number of unique users: 8
  Number of unique tags: 81
  Number of unique artists: 178
  Top 5 artists (by ID): [11892, 1377, 5555, 6852, 51]
  Top 5 tags (by ID): [2726, 824, 839, 424, 76]

Interval: 2005-10
  Number of unique users: 5
  Number of unique tags: 73
  Number of unique artists: 142
  Top 5 artists (by ID): [227, 51, 59, 225, 235]
  Top 5 tags (by ID): [73, 79, 84, 134, 296]
```

Dynamics of Listening and Tagging | Interactive Plots

- Then we proceeded with two interactive plots for visualization by using plotly and ipywidgets.



Correlation Analysis

Gathering Artist and Listening Data for Each User:

- Music Habits: for each user, count how many different artists they've listened to and calculate the total time they've spent listening.

Finding Out How Social Each User Is:

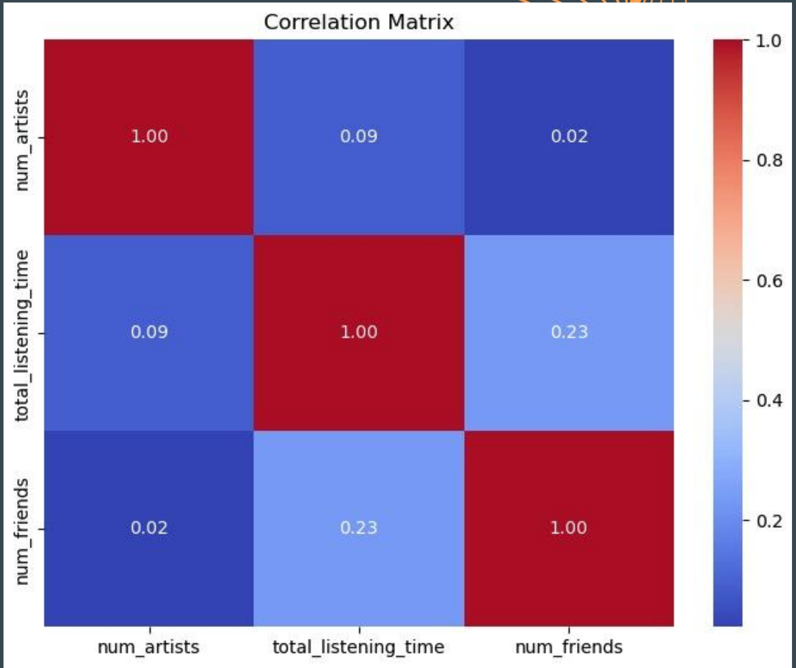
- How many friends each user has. This is done by counting the number of connections each user has in our friends' data.

Bringing It All Together:

- Combine the music and social data. We merged the artist and listening information with the friends' data → make sure to account for users who might not have any friends listed in our data by assigning them a friend count of zero→ ensures that no user is left out.

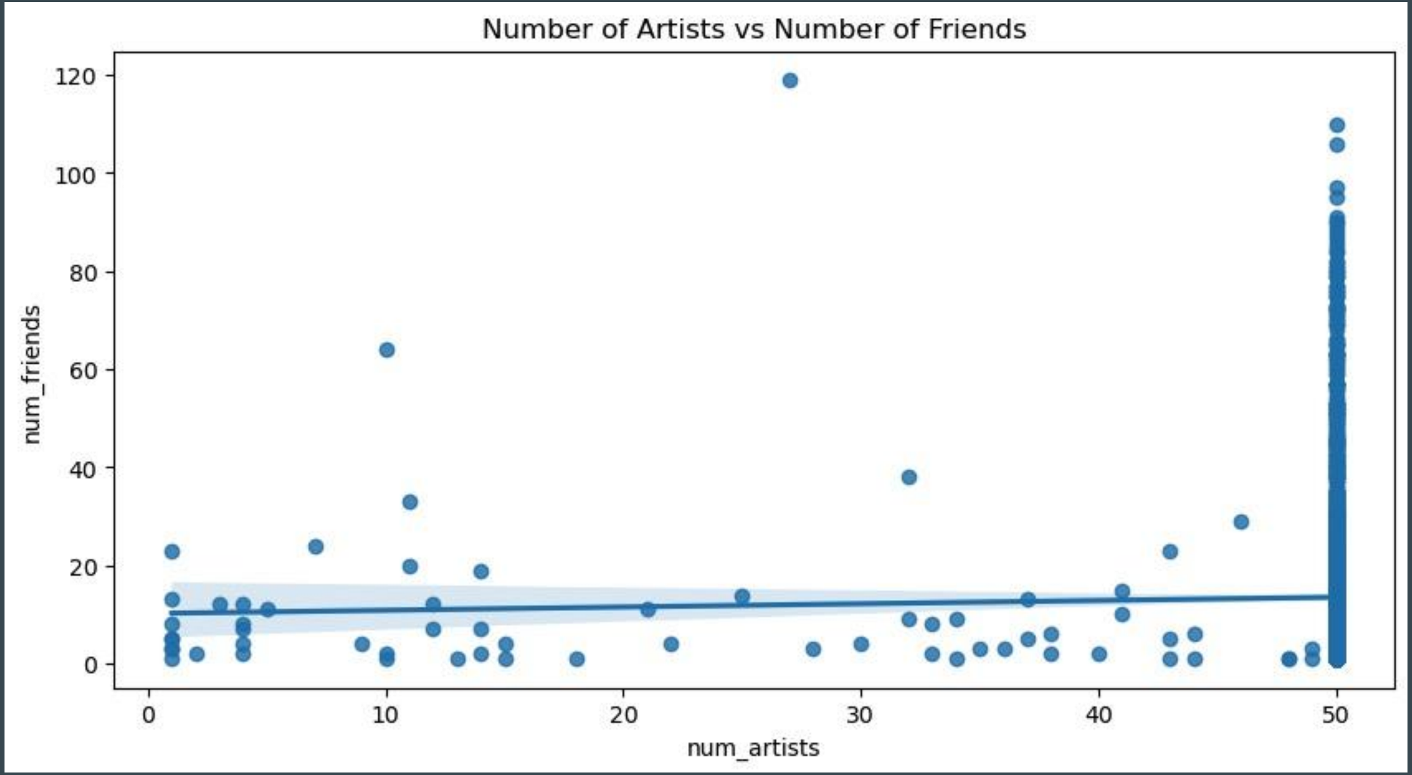
Uncovering Relationships in the Data:

- All the data compiled together → explore the relationships between how many artists a user listens to, their total listening time, and the number of friends they have →correlation matrix to see if and how these different aspects are related to each other.



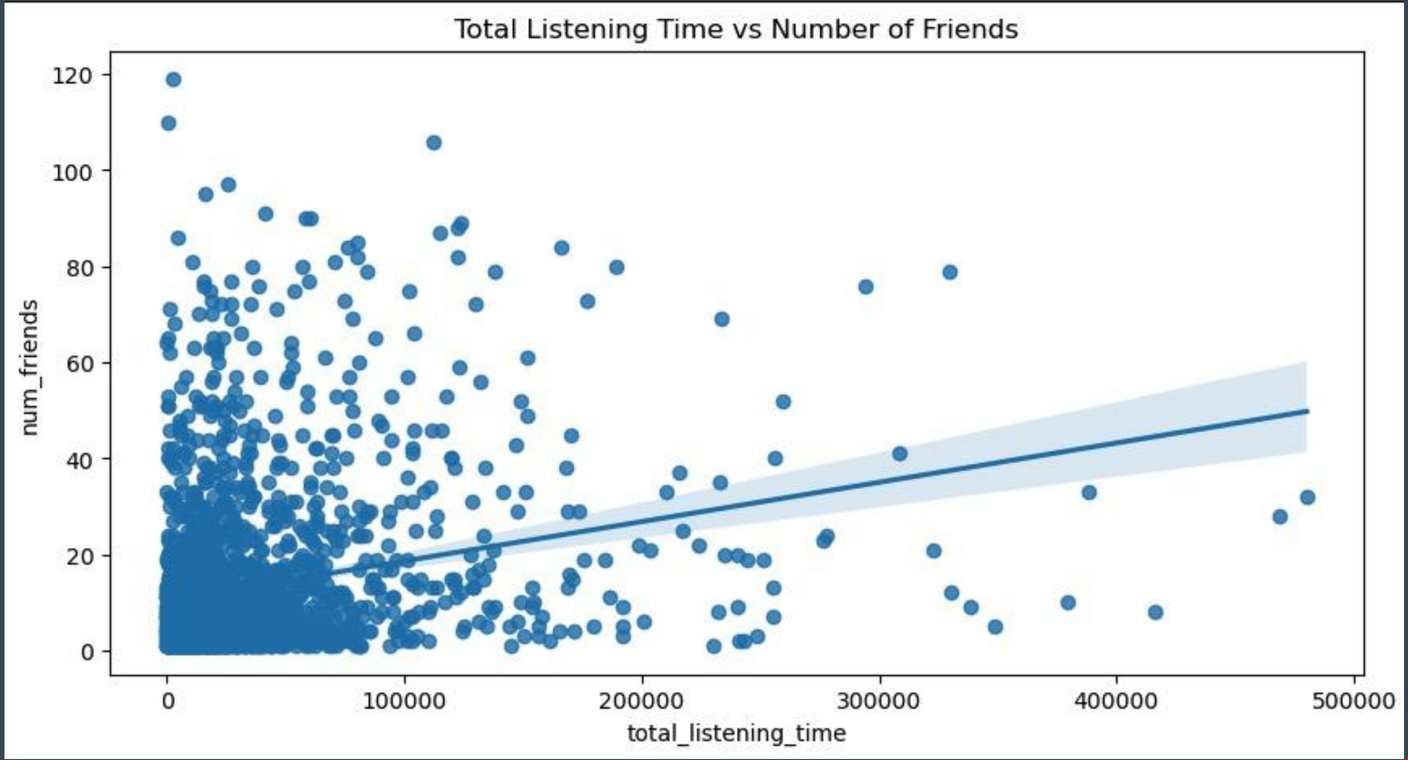
Number of Artists vs Number of Friends

- Scatter plot with Regression Line



Total Listening Time vs Number of Friends

- Scatter plot with Regression Line



Thank you for your time 😊

Github link:

https://github.com/alkisKariotis/Data_Science_Master/blob/main/ITC6001_Final_Project_Code_241735%26122451.ipynb