Ms in Data Science

ITC 6001-INTRODUCTION TO BIG DATA  
Final Project

**Instructor**

Dr. Vogiatzis



**Submitted to**

Deree – The American College of Greece

**Submitted by**Kapsalis C. , Stavrogiannis C**.**

**Date:**

13/12/2023

**Contents**

[Introduction 1](#_Toc153122777)

[Q1: Understanding the Data – Exploration 2](#_Toc153122778)

[Q1: Data Description 2](#_Toc153122779)

[Purpose 2](#_Toc153122780)

[Brief Description of Data Files Used 2](#_Toc153122781)

[Requested Frequency Plots 4](#_Toc153122782)

[Q1: Outlier Detection 5](#_Toc153122783)

[Purpose 5](#_Toc153122784)

[Z-Score Method 5](#_Toc153122785)

[IQR Method 7](#_Toc153122786)

[Q2: Similar Users 10](#_Toc153122787)

[Purpose 10](#_Toc153122788)

[User Similarity Calculation 10](#_Toc153122789)

[K-Nearest Neighbors 10](#_Toc153122790)

[Observations 11](#_Toc153122791)

[Additional Observations 12](#_Toc153122792)

[Top Ten Users Based on Average Similarity 12](#_Toc153122793)

[Analysis and Comparison 13](#_Toc153122794)

[Q3: Dynamic of Listening and Tagging 15](#_Toc153122795)

[Q3-a: Displaying the Activity per Interval 15](#_Toc153122796)

[Q3-b: Detecting most Active Items per Interval 17](#_Toc153122797)

[Q4: Comparing Prolific User Detect Methods 20](#_Toc153122798)

[Correlation Between Artists & Friends 20](#_Toc153122799)

[Correlation Between Listening Time & Friends 21](#_Toc153122800)

[Observations 22](#_Toc153122801)

[Conclusion 23](#_Toc153122802)

[References 24](#_Toc153122803)

[ANNEX 25](#_Toc153122804)

[Part A 25](#_Toc153122805)

[Part B 26](#_Toc153122806)

**Figures**  
[Figure 1: Total Listening Count per Artist 4](#_Toc153301707)

[Figure 2: Total Usage per Tag 5](#_Toc153301708)

[Figure 3: Violin plot on the times that artists have been listened to 6](#_Toc153301709)

[Figure 4: Violin plot on the total listening time for users 6](#_Toc153301710)

[Figure 5: Violin plot on the number of times tags have been used. 7](#_Toc153301711)

[Figure 6: Boxplot on the times that artists have been listened to. 8](#_Toc153301712)

[Figure 7: Box plot on the total listening time for users 9](#_Toc153301713)

[Figure 8: Box plot on the total usage count for tags 9](#_Toc153301714)

[Figure 12: k-nearest neighbors sample as appear in the resulted “neighbors-k-users.data” for k=3 and k=10. 11](#_Toc153301715)

[Figure 9: Number of (unique) Users Active per Quarter 15](#_Toc153301716)

[Figure 10: Number of (unique) Tags Used per Quarter 16](#_Toc153301717)

[Figure 11: Number of (unique) Artists Listened to per Quarter 16](#_Toc153301718)

[Figure 13: Scatterplot displaying correlation between count of friends and count of artists, including best fit line. 20](#_Toc153301719)

[Figure 14: Correlation matrix between count of artists and count of friends. 21](#_Toc153301720)

[Figure 15: Scatterplot displaying correlation between count of friends and sum of listening, including best fit line. 21](#_Toc153301721)

[Figure 16: Correlation matrix between sum of listening and count of friends. 22](#_Toc153301722)

**Tables**  
[Table 1: First 5 rows of 'artists.dat' in DataFrame format 2](#_Toc153368031)

[Table 2: First 5 rows of 'user\_artists.dat' in DataFrame format. 3](#_Toc153368032)

[Table 3: First 5 rows of 'tags.dat' in DataFrame format. 3](#_Toc153368033)

[Table 4: First 5 rows of 'user\_taggedartists-timestamps.dat' in DataFrame format. 3](#_Toc153368034)

[Table 5: Top-10 users with highest average cosine similarity for k=3. 12](#_Toc153368035)

[Table 6:Top-10 users with highest average cosine similarity for k=10. 12](#_Toc153368036)

[Table 7: Top-10 users regarding similarity for all users. 13](#_Toc153368037)

[Table 8: Lowest, highest and mean "average cosine similarity" for all users. 13](#_Toc153368038)

[Table 9: Tagging Activity Data with Broken Down Timestamps 16](#_Toc153368039)

[Table 10: Problematic Rows with Wrong Timestamp Values 17](#_Toc153368040)

[Table 11: Count of Times that Artists were in the First Position of Quarterly Rankings 17](#_Toc153368041)

[Table 12: Number of Times that Tags were at the First Position of Quarterly Rankings 18](#_Toc153368042)

# **Introduction**

The MS in Data Science project for the course "Introduction to Big Data" focuses on a comprehensive exploration of the Last.fm dataset. This exploration aims at showcasing prolific user activity and observations, studying the interrelations between its items and describing their dynamics. The dataset, sourced from last.fm's online music system and containing social networking, tagging, and music artist listening information from a set of 1,892 users, serves as the foundation for the project.

The project is structured into six distinct tasks, each contributing to a holistic understanding of the dataset and revealing patterns and correlations within the data.

These tasks involve data description, outlier detection, similarity analysis between users, dynamics of listening and tagging over time, correlation analysis between user(s) behaviors.

# **Q1: Understanding the Data – Exploration**

## **Q1: Data Description**

### **Purpose**

The dataset[[1]](#footnote-1) we worked on contains information on user activity on the online music streaming platform called “last.fm”. The platform contains not only music streaming features; It also allows users to assign personalized tags on artists and follow other users to track their activity.  
The dataset files that we needed to use in our analysis were ‘artists.dat’, ‘tags.dat’, ‘user\_artists.dat’, ‘user\_friends.dat’, ‘user\_taggedartists-timestamps.dat’.

### **Brief Description of Data Files Used**

To briefly describe the data we worked on, we proceed to show a glimpse of the corresponding dataframe structures we created after reading those text files:

**artists.dat:** each record corresponds to one unique artist, and 4 attributes are displayed – artist id, artist name, url of their profile, and url of the picture in their profile.

**Table 1:** First 5 rows of 'artists.dat' in DataFrame format

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **Name** | **url** | **pictureURL** |
| 1 | MALICE MIZER | <http://www.last.fm/music/MALICE+MIZER> | <http://userserve-ak.last.fm/serve/252/10808.jpg> |
| 2 | Diary of Dreams | <http://www.last.fm/music/Diary+of+Dreams> | <http://userserve-ak.last.fm/serve/252/3052066.jpg> |
| 3 | Carpathian Forest | <http://www.last.fm/music/Carpathian+Forest> | <http://userserve-ak.last.fm/serve/252/40222717...> |
| 4 | Moi dix Mois | <http://www.last.fm/music/Moi+dix+Mois> | <http://userserve-ak.last.fm/serve/252/54697835...> |
| 5 | Bella Morte | <http://www.last.fm/music/Bella+Morte> | <http://userserve-ak.last.fm/serve/252/14789013...> |

**user\_artists.dat:** each record corresponds to one unique user-artist pair, showing 3 attributes of each pair – user id, artist id, weight of their interaction (i.e. the number of times the user has streamed songs of the corresponding artist). A user can be present in various rows, and the same holds true for artists, since a user might listen to many different artists, and each artist’s songs may be listened to by various users. We check in our code and it is validated that there are no duplicate rows, based on the [‘userID’, ‘artistID’] variables pair. The sum of ‘weight’ values in all the rows an artist shows up in is equal to the total count of streams for their music in the time period under study.

**Table 2:** First 5 rows of 'user\_artists.dat' in DataFrame format.

|  |  |  |
| --- | --- | --- |
| **userID** | **artistID** | **weight** |
| 2 | 51 | 13883 |
| 2 | 52 | 11690 |
| 2 | 53 | 11351 |
| 2 | 54 | 10300 |
| 2 | 55 | 8983 |

**tags.dat:** each record corresponds to a unique tag, naming its id and value.

**Table 3:** First 5 rows of 'tags.dat' in DataFrame format.

|  |  |
| --- | --- |
| **tagID** | **tagValue** |
| 1 | metal |
| 2 | alternative metal |
| 3 | goth rock |
| 4 | black metal |
| 5 | death metal |

**user\_taggedartists-timestamps.dat:** each record corresponds to one unique user-artist-tag triplet, showing the id of each object and the timestamp representing the time at which the specific tag was applied by the specific user on the specific artist. A user applies various tags to a single artist. We check in our code and it is validated that there are no duplicate rows, based on the [‘userID’, ‘artistID’, ‘tagID’] variables triplet.

**Table 4:** First 5 rows of 'user\_taggedartists-timestamps.dat' in DataFrame format.

|  |  |  |  |
| --- | --- | --- | --- |
| **userID** | **artistID** | **tagID** | **timestamp** |
| 2 | 52 | 13 | 1238536800000 |
| 2 | 52 | 15 | 1238536800000 |
| 2 | 52 | 18 | 1238536800000 |
| 2 | 52 | 21 | 1238536800000 |
| 2 | 52 | 41 | 1238536800000 |

### **Requested Frequency Plots**

##### **Listening Frequency of Artists by Users**

For the requested frequency plots we began by merging the dataframes formed using the ‘artists.dat’ and ‘user\_artists.dat’ files, to match each artist id to their name and make the resulting plot more meaningful to observers. For all plots we display the top-10 values.

To acquire the total listening count for each artist, we create a pivot table in which we match each artist name to the sum of ‘weight’ values for all users that have listened to them. Due to the vast number of artists in our dataframe (17,632 according to the dataset’s readme.txt[[2]](#endnote-1) file).

A graph with blue and white bars

Description automatically generated

**Figure 1:** Total Listening Count per Artist

##### **Frequency of Tag Usage**

We pivoted the tag values (which function as their names) to the count of their occurrence in the ‘user\_tags.dat’ data file.

A graph of blue and white bars

Description automatically generated

**Figure 2:** Total Usage per Tag

## **Q1: Outlier Detection**

### **Purpose**

We applied the two outlier detection methods on our data in order to recognize extreme values in 3 different distributions. The main factors behind our method selection were method popularity and differences in underlying distribution assumptions. Our goal was to showcase two methods that are popular, yet they can be applied in diverse sets of data.

### **Z-Score Method**

A Z-Score describes how far a data point is located from the mean of the random variable under study; , where X: a value of the random variable, μ: the (true-population) mean, and σ: the standard deviation of the variable’s distribution. It is expressed in terms of units of the underlying distribution’s standard deviation. To classify a data point as an outlier, it works based on researcher-defined thresholds. Those thresholds rely on the empirical rule, and this stems from the fact that this method assumes that the underlying distribution studied is (approximately) normal.

Adopting the assumption of the method, the threshold value we set in our analysis was 3. According to the empirical rule, 99.7% of the values of a normally distributed random variable will fall between 3 standard deviations above or below the mean. This is a very reasonable threshold to adopt in order to be sure that we only classify as outliers those values that are actually extreme and unusual.

As for the results in each of the distributions we were requested to study:

* **Total listens for the artists:** 
  + 85 outlier values disclosed in the ‘Z-outliers\_artists.csv’ file. This makes for 85 out of 17,632 artists in the dataset, i.e. 0.48% of data points.

A graph of a graph

Description automatically generated

**Figure 3:** Violin plot on the times that artists have been listened to

* + A violin plot indicates non-normal distribution in our data, with noticeable positive asymmetry. Few but highly extreme values significantly impact the mean, rendering this method irrelevant for our example.
* **Total listening time for users:** 
  + 43 outlier values out of 1892 users, or 2.27% of our data points; results showcased in the ‘Z-outliers\_users.csv’ file.
  + Even higher positive asymmetry than in the previous case, the same remarks are true.

A graph of a weight variable

Description automatically generated

**Figure 4:** Violin plot on the total listening time for users

* **Total times tags were used:** 
  + 68 outlier values out of 11,946 tags, or 0.57% of our data points; results showcased in the ‘Z-outliers\_tags.csv’ file.
  + The same holds true for the normality distribution as before.

**A graph of a graph

Description automatically generated**

**Figure 5:** Violin plot on the number of times tags have been used.

### **IQR Method**

The 'boxplot' method, an alternative to the Z-Score approach, relies on analyzing the boxplot representation of variable distributions. Unlike Z-Score, it's robust, suitable for non-normal distributions, as it doesn't depend on the mean. Instead, it utilizes outlier-robust metrics based on the distribution (Schwertman et al., 2004). However, it assumes approximate symmetry. In asymmetric distributions, a single metric may not suffice for outlier classification due to varying extremity on each side of the median.

The thresholds set in this method are the following:

* Lower bound: Q1 - 1.5 \* IQR, where Q1: the first quartile, Q3: the third quartile and IQR: the interquartile range (IQR = Q3 – Q1)
* Upper bound: Q3 + 1.5 \* IQR

Thus, if a value falls outside these boundaries, it is deemed as an outlier.

Due to the previously uncovered asymmetry of the distributions of the random variables under study, we expect that this method’s results are not reliable as well, but due to the less strict nature of its assumptions, we expect that the results will be more accurate.

As for the results in each of the distributions we were requested to study:

* How many times the artists have been listened to:
  + 2493 out of 17,632 artists, or 14.14% of our observations; this method gives many more outliers so it's much more effective. The specific values are included in the ‘IQR-outliers\_artists.csv’ file found in our deliverables folder.
  + The box plot on the data shows that the underlying distribution is far from symmetric; positive asymmetry is detected, with many relatively low values (50% of the observations lie in the range [0,350=median], and many extremely high ones, higher than 2916.125 = upper bound). Thus, the use of this method in our example gives irrelevant results. Asymmetry is shown in the plot through the fact that the median is much closer to q1 than q3, and the left-side whisker is much shorter than the right-side whisker. This shows that most values are concentrated at the left-side end of the distribution, and extremely high values influence the mean upwards.

**A graph with a line and a blue rectangle

Description automatically generated**

**Figure 6:** Boxplot on the times that artists have been listened to.

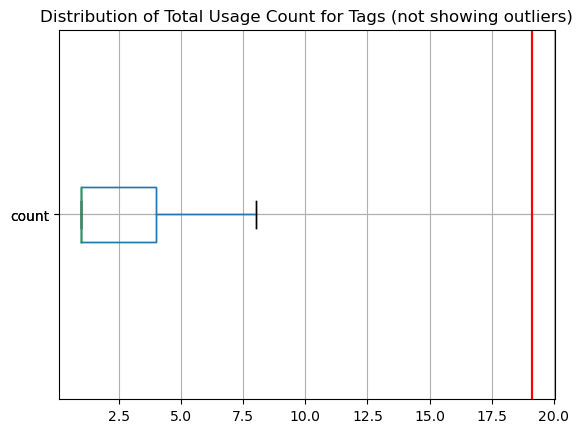
* Total listening time for users:
  + 161 outlier values out of 1892 users, or 8.51% of our data points; results showcased in the ‘IQR-outliers\_users.csv’ file.

**A graph with a line and a rectangle

Description automatically generated with medium confidence**

**Figure 7:** Box plot on the total listening time for users

* Total times tags were used:
  + 1,592 outlier values out of 11,946 tags, or 13.33% of our data points; results showcased in the ‘IQR-outliers\_tags.csv’ file.

****

**Figure 8:** Box plot on the total usage count for tags

# **Q2: Similar Users**

## **Purpose**

We analyze user interactions with artists, providing insights on user similarity and nearest neighbors. We utilize the cosine similarity measure to quantify user similarities based on the artists they have heard and the associated 'weight' parameter. Additionally, we identify the k-nearest neighbors for each user and store the results for further analysis.

## **User Similarity Calculation**

We base our analysis on the calculation of cosine similarity. This method measures the cosine of the angle between two vectors, representing users in the high-dimensional artist space. The dot product and normalization of these vectors yield a similarity measure.

* Create dataframe from a CSV file, containing information about users, artists, and the 'weight' of interactions.
* We create a pivot table from the 'user\_artists.dat' data.
* In the above interaction matrix rows represent users, columns represent artists, and values correspond to the 'weight' of interactions. Missing values are filled with zeros.
* Cosine similarity is calculated using the dot product and np.linalg.norm.

In the submitted code, the cosine similarity was calculated using two approaches:  
An “algebraic” approach using dot product and np.linalg.norm

The second method utilized the *cosine\_similarity()* function offered by python and *sklear* library.

This approach, beyond a more complete solution offers a manner to validate our results by comparing the resulted csv file derived from both methods discussed above.

For the final calculations we used the first “algebraic” approach, nevertheless the second method is included in the code but it has been commented out.

## **K-Nearest Neighbors**

In the below section, we focused on identifying the k-nearest neighbors and their average similarity using both cosine similarity and k-means clustering for k=3 and k=10.

To gain a deeper understanding of user communities, we performed k-means clustering based on the similarity of the artists preferences of users for k=3, k=10.

For each user, the k-nearest neighbors are determined based on the cosine similarity calculated in the previous step.

* We define a function to find k-nearest neighbors for a user based on cosine similarity.
* A dictionary, 'neighbors\_dict,' is created to store neighbors for each user for k=3 and k=10.
* The results are stored in a JSON file ('neighbors-k-users.data'), where the user IDs serve as keys, and the values are lists of the IDs of their k-nearest neighbors.
* k-nearest neighbors for both k=3 and k=10 is stored in the above JSON file and have the below structure as depicted in figure 12 below.

A close-up of numbers

Description automatically generated A black text on a white background

Description automatically generated A black text on a white background

Description automatically generated

**Figure 12:** k-nearest neighbors sample as appear in the resulted “neighbors-k-users.data” for k=3 and k=10.

## **Observations**

The code employs filtering techniques to analyze user interactions and provide meaningful information about user similarity and relationships. This user-similarity is expressed with cosine similarity matrix which facilitates personalized content recommendations ('user-pairs-similarity.data'). The k-nearest neighbors data ('neighbors-k-users.data') further refines this analysis by identifying users with the highest similarity for each user adding a layer of interpretation and revealing user “communities” based on artist preferences.

## **Additional Observations**

### **Top Ten Users Based on Average Similarity**

We calculate the average similarity for each user with their k=3 , k=10 neighbors as well as all users focusing on the top ten users.

**Table 5:** Top-10 users with highest average cosine similarity for k=3.

|  |  |
| --- | --- |
| **User ID** | **Value** |
| 1135 | 0.9980933951348255 |
| 13 | 0.998080734531007 |
| 1307 | 0.9980049803761313 |
| 542 | 0.9971899386731602 |
| 1693 | 0.9970053186450509 |
| 819 | 0.9967635204172088 |
| 816 | 0.9954902299027047 |
| 54 | 0.9933947271439837 |
| 1471 | 0.992792088974439 |
| 1642 | 0.9906409491499284 |

**Table 6:**Top-10 users with highest average cosine similarity for k=10.

|  |  |
| --- | --- |
| **User ID** | **Value** |
| 819 | 0.988218882824939 |
| 1693 | 0.9873817591055735 |
| 542 | 0.986990837275509 |
| 1471 | 0.9854549840394174 |
| 717 | 0.984850764833246 |
| 54 | 0.9841137315914026 |
| 1726 | 0.9818474661057548 |
| 414 | 0.9804837611522028 |
| 911 | 0.9790216072133591 |
| 503 | 0.9752955865090858 |

### **Analysis and Comparison**

1. **Top Ten Users Comparison (k=3 vs. k=10):**
   * k=3 yields a user (1135) with extremely high average similarity, suggesting a tight-knit community around that user.
   * k=10 provides a broader view of similarity, capturing users with consistently high similarity across a larger group. The highest average similarity is for user 819.
2. **Highest, Lowest, and Mean Analysis:**
   * The highest average similarity user (1135) remains consistent for k=3.
   * The highest average similarity user (819) remains consistent for k=10.
   * We can conclude then that the community formed around users (1135, 819) is consisted of users with high alignment regarding their taste in music.
   * The difference between mean average similarity and highest average similarity indicates varying levels of user interaction in the dataset.

User with Highest Average Cosine Similarity (All Users):

**Table 7:** Top-10 users regarding similarity for all users.

|  |  |
| --- | --- |
| **User ID** | **Average Cosine Similarity** |
| **1132** | **0.11570471841209973** |
| **339** | **0.11371739585394361** |
| **470** | **0.11350962406058603** |
| **1606** | **0.11035185709443213** |
| **2033** | **0.10766300041294584** |
| **632** | **0.10667613722605586** |
| **1572** | **0.10538932440780718** |
| **415** | **0.10465541419681036** |
| **965** | **0.10446988120693881** |
| **1394** | **0.1042068119118706** |

**Table 8:** Lowest, highest and mean "average cosine similarity" for all users.

|  |  |  |
| --- | --- | --- |
| **Average Cosine Similarity** | **User ID** | **Value** |
| **Highest** | 1132 | 0.1157 |
| **Lowest** | 112 | 0.0005 |
| **Mean** | - | 0.0334 |

A comparison with graphical representation of the above “Additional Observations” is included in [ANNEX](#_ANNEX).

# **Q3: Dynamic of Listening and Tagging**

## **Q3-a: Displaying the Activity per Interval**

We worked only on data from the ‘user\_taggedartists-timestamps.dat’ data file and the corresponding dataframe we created after importing the data into python.

We computed the distinct count of occurrence of each item (user/artist/tag) per interval – we selected this interval to be **a quarter of the year**.

In doing this, we used the ‘groupby’ method of pandas dataframes, so as to group the various user IDs, artistIDs, and tagIDs, by the year and next the quarter in the year that they belong to. We then proceed to count the unique items (users/artists/tags) per time period considered.

The results of this analysis are contained in the corresponding data files: ‘Q3a\_artists.csv’, ‘Q3a\_tags.csv’, ‘Q3a\_users.csv’.

A graph showing the growth of a quarter

Description automatically generated

**Figure 9:** Number of (unique) Users Active per Quarter

A graph showing the number of different tags

Description automatically generated

**Figure 10:** Number of (unique) Tags Used per Quarter

A graph with a line going up

Description automatically generated

**Figure 11:** Number of (unique) Artists Listened to per Quarter

There was a clear uptrend from the beginning of the platform up to the third quarter of 2010, which coincided with the great expansion of internet usage in general, and specifically social media. Then there was a sudden drop, which coincided with the removal of many features central to the platform's offering, and there also was a redesign of its look that was welcomed with hostility by many users. This reduction of the platform’s user base lead to general drop-in activity on it, as measured by the tagging and music streaming activity on last.fm[[3]](#endnote-2) [[4]](#endnote-3).

## **Q3-b: Detecting most Active Items per Interval**

We translated the given timestamps in a column showing the year and quarter an observation relates to. The first rows of the resulting dataframe have as follows:

**Table 9:** Tagging Activity Data with Broken Down Timestamps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **userID** | **artistID** | **tagID** | **timestamp** | **date\_** | **quarters** |
| 2 | 52 | 13 | 1238536800 | 2009-04-01 1:00:00 | (2009, 2) |
| 2 | 52 | 15 | 1238536800 | 2009-04-01 1:00:00 | (2009, 2) |
| 2 | 52 | 18 | 1238536800 | 2009-04-01 1:00:00 | (2009, 2) |
| 2 | 52 | 21 | 1238536800 | 2009-04-01 1:00:00 | (2009, 2) |
| 2 | 52 | 41 | 1238536800 | 2009-04-01 1:00:00 | (2009, 2) |

To address the problem of finding the 5 most listened to artists per quarter (we worked in the same way to find the most popular tags per interval), we do the following:

* We consider each quarter on its own, based on the unique values of the ‘quarters’ column above.
* For a given quarter, we find all corresponding records in the dataframe; in those rows, we compute the number of occurrences of each artist ID (i.e. the total number of times a tag was assigned to them) using the ‘groupby’ method of pandas dataframes, complete with the ‘count’ aggregating function.
* We sort the results in descending order by the count of occurrence, and keep only the top 5 artists, keeping the resulting data in a numpy series.
* In some quarters there could be less than 5 unique artists that got assigned a tag; we fill in these potential blanks with NaN values. We checked this at the end and saw that no NaN values needed to be added.
* Then, from the resulting object we want to extract lists; the quarters (5 times each), the ids of the top 5 artists, and the frequencies of occurrence of those artists.
* We built a dataframe using those lists and dropped rows with NaN values (which correspond to quarters in which we didn't have tags assigned to as many as 5 unique artists).
* The final result is a dataframe showing each quarter, and then, in descending order (by their frequency) it shows the top artistID values and their corresponding frequencies.

We observed that there are some problematic timestamp values in our dataset, corresponding to dates prior to 2006 (i.e. prior to the date the platform under study first went online). We consider those rows to be added by mistake or to comprise errors in measurement; there is no way of correcting them, so we go forth with dropping them.

**Table 10:** Problematic Rows with Wrong Timestamp Values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **userID** | **artistID** | **tagID** | **timestamp** | **date\_** | **quarters** |
| 43 | 1395 | 39 | -428720400 | 1956-06-01 1:00:00 | (1956, 2) |
| 133 | 2984 | 1474 | -405133200 | 1957-03-01 1:00:00 | (1957, 1) |
| 1604 | 1583 | 103 | -420771600 | 1956-09-01 1:00:00 | (1956, 3) |
| 1604 | 1583 | 10021 | -420771600 | 1956-09-01 1:00:00 | (1956, 3) |
| 1929 | 250 | 311 | 294357600 | 1979-05-01 1:00:00 | (1979, 2) |

The top 5 artists and tags per interval are included in the ‘Q3b\_artists.csv’ and ‘Q3b\_tags.csv’ file, respectively.

We continued by searching for artists and tags that have systematically dominated on the quarterly rankings, being in the first position for more times than others. We used the dataframes that resulted from the previous step of our analysis (where we extracted the 5 most popular artists and tags per quarter). We extracted the artists and tags in the first position in each quarter, and counted the times that each appeared in the first position of the corresponding rankings. The resulting data files are included in our deliverables as ‘dominating\_artists.csv’ and ‘dominating\_tags.csv’.

Regarding dominating artists:

**Table 11:** Count of Times that Artists were in the First Position of Quarterly Rankings

|  |  |  |
| --- | --- | --- |
| **artistID** | **count** | **name** |
| 292 | 3 | Christina Aguilera |
| 11892 | 2 | Blood Ruby |
| 289 | 2 | Britney Spears |
| 72 | 2 | Depeche Mode |
| 972 | 1 | t.A.T.u. |
| 15675 | 1 | Modus |
| 67 | 1 | Madonna |
| 157 | 1 | Michael Jackson |
| 295 | 1 | Beyoncé |
| 51 | 1 | Duran Duran |
| 1412 | 1 | Led Zeppelin |
| 154 | 1 | Radiohead |
| 1768 | 1 | Pato Fu |
| 14057 | 1 | Herr M. |
| 4118 | 1 | The Postal Service |
| 316 | 1 | Alanis Morissette |
| 217 | 1 | Death Cab for Cutie |
| 55 | 1 | Kylie Minogue |
| 227 | 1 | The Beatles |

As for dominating tags:

**Table 12:** Number of Times that Tags were at the First Position of Quarterly Rankings

|  |  |  |
| --- | --- | --- |
| **tagID** | **count** | **tagValue** |
| 73 | 21 | rock |
| 2191 | 1 | college rock |
| 79 | 1 | alternative |
| 24 | 1 | pop |

We see that rock-related tags dominate in 22 out of the 24 intervals (i.e. 87.5% of them) under study. At the same time, there's no unique artist dominating in the quarterly rankings, and those that have appeared on the top of them are mixed among the musical genres; 2 of them are pop musicians (Christina Aguilera, Britney Spears) and the other 2 (Depeche Mode, Blood Ruby) venture mainly in the rock genre.

This tells us that, even though pop artists have extremely loyal fans that stream their songs much more than fans of other music genres (as Figure 1 displays), the rock fans user cohort of last.fm is far more active, interacting with the profiles of their loved artists much more than pop fans do.

# **Q4: Comparing Prolific User Detect Methods**

We explore the correlation between the number of artists a user listens to and the number of friends they have, as well as the correlation between a user's total listening time and the number of friends they have. We focus on understanding potential relationships between user behavior and social interactions on the last.fm dataset.

## **Correlation Between Artists & Friends**

We compute the correlation between the user's count of friends and the count of artists. Utilizing the Pearson correlation coefficient to gauge the strength and direction of their linear relationship, we employ a scatterplot and correlation matrix for visual representation. Despite these visual insights, the correlation coefficient registers at a low 0.0224, indicating a weak positive correlation. This suggests scant evidence supporting a direct relationship between the number of friends a user possesses and the number of artists they listen to.

A graph of blue dots

Description automatically generated

**Figure 13**: Scatterplot displaying correlation between count of friends and count of artists, including best fit line.

A blue squares with white text

Description automatically generated

**Figure 14:** Correlation matrix between count of artists and count of friends.

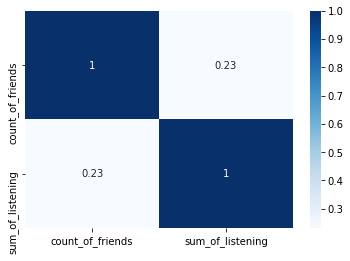
## **Correlation Between Listening Time & Friends**

We investigate the correlation between a user's total listening time (sum of weights) and the count of friends they have. The Pearson correlation coefficient is again utilized. The scatterplot and correlation matrix visually represent the relationship between these variables. Interestingly, the correlation coefficient is 0.2306, indicating a moderate positive correlation. This suggests that users with more friends tend to have a higher total listening time, potentially influenced by shared music interests or recommendations.

A graph with blue dots

Description automatically generated

**Figure 15:** Scatterplot displaying correlation between count of friends and sum of listening, including best fit line.



**Figure 16:** Correlation matrix between sum of listening and count of friends.

## **Observations**

The analysis provides insights into the potential correlations between user behavior and social interactions on last.fm. Part A reveals a weak positive correlation between the count of friends and the count of artists, while Part B shows a moderate positive correlation between total listening time and the count of friends. These findings suggest that users with more friends may have more diverse musical interests, impacting their listening habits. However, additional factors may influence these relationships, warranting further investigation.

The code submitted includes both an algebraic approach of calculation as well as with usage of the *corr()* function of python ([ANNEX Part B](#_Part_B)).

# **Conclusion**

In conclusion, this project-report encapsulates a rigorous exploration of the Last.fm dataset, emphasizing comprehensive data analysis across various dimensions.

Our efforts spanned critical tasks, from understanding and describing the dataset to detecting outliers, analyzing user similarities, and exploring temporal dynamics. By comparing methods for detecting prolific users, we sought to extract meaningful insights and correlations from the rich Last.fm dataset.

This project underscores the application of data science methodologies, programming expertise, and statistical analysis, providing a holistic learning experience in the realm of Big Data.

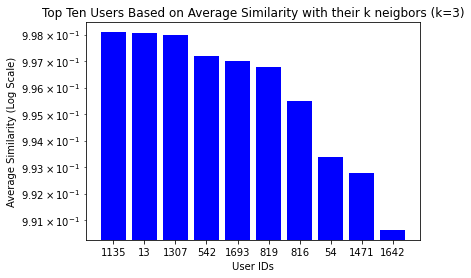
# **References**

Schwertman, N. C., Owens, M. A., & Adnan, R. (2004). A simple more general boxplot method for identifying outliers. Computational Statistics & Data Analysis, 47(1), 165-174. <https://doi.org/10.1016/j.csda.2003.10.012>

# **ANNEX**

## **Part A**

BarCharts showcasing Average Cosinile similarity for the top-10 users for a) k=3 b) k=10 c) all users.



A graph of blue bars

Description automatically generated A bar graph with numbers and a number of people

Description automatically generated with medium confidence

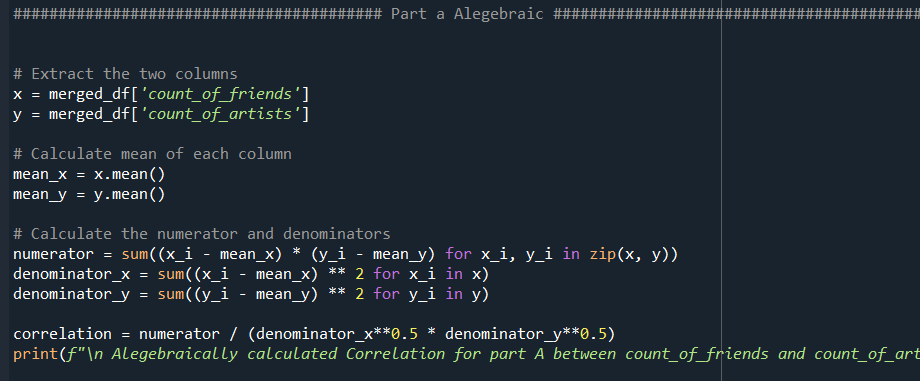
## **Part B**

Indicative calculation of correlation using corr() function

A computer screen shot of a code

Description automatically generated

Indicative manner of calculation of correlation using algebraic approach.



1. <https://grouplens.org/datasets/hetrec-2011/> [↑](#footnote-ref-1)
2. <https://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-readme.txt> [↑](#endnote-ref-1)
3. <https://web.archive.org/web/20080723042046/http://www.paidcontent.co.uk/entry/419-some-lastfm-users-revolt-over-new-look/> [↑](#endnote-ref-2)
4. <https://www.musicweek.com/news/read/last-fm-claims-controversial-re-design-a-success/038208> [↑](#endnote-ref-3)