

Master of Science (MS) in Data Science  
Module: ITC6001A1 – Introduction to Big Data

Instructor: Dr. Dimitrios Vogiatzis

Students’ names:

Alkiviadis Kariotis 241735

Konstantinos Margaritis 122451

Term: Fall Term 2023  
Type: Final Group Project

Submission Date: Wednesday, 13th of December 2023

Words: 3254

Table of Contents

[Introduction 3](#_Toc152849098)

[Main Body 4](#_Toc152849099)

[Libraries and their roles 4](#_Toc152849100)

[Data Processing 5](#_Toc152849101)

[Data Overview 5](#_Toc152849102)

[Data Loading Process 5](#_Toc152849103)

[Initial Data Exploration 6](#_Toc152849104)

[Data Visualization 7](#_Toc152849105)

[Outlier Detection 9](#_Toc152849106)

[Similarity Assessment 12](#_Toc152849107)

[Cosine Similarity 12](#_Toc152849108)

[K-Nearest Neighbors (KNN) 12](#_Toc152849109)

[Dynamics of Listening and Tagging 13](#_Toc152849110)

[Users, tags, and artists per interval 13](#_Toc152849111)

[Top5 artists and tags per interval 13](#_Toc152849112)

[Correlation Analysis 15](#_Toc152849113)

[Artists Listened and number of friends 16](#_Toc152849114)

[Listening time and number of friends 17](#_Toc152849115)

[Conclusion and further analysis 17](#_Toc152849116)

[References 19](#_Toc152849117)

# Introduction

Our project is a comprehensive and multistep analysis of the Last.fm dataset obtained from <https://grouplens.org/datasets/hetrec-2011/>. The dataset consists of information based on user behavior in digital music platforms, having many different aspects such as their music listening habits, their friends, or their social networking, and also tagging activities.

Our analysis aims at identifying and understanding the user’s listening patterns, by exploring how they engage with all the different artists that the music digital platform has in order to reveal their preferences across the user base. This can be crucial when it comes to personal recommendations but also towards understanding the popularity of the existing artists in the platform. Then we proceeded at researching the tagging behavior of the users of the dataset, in order to identify how they tend to categorize and perceive the different artists and music genres that are offered by the platform. Furthermore, we performed an outlier detection analysis, using two different methods, which could be found very useful towards seeing unusual behaviors of niche users or potential data entry errors that may be included. Then, we also performed a cosine similarity assessment, comparing our dataset users with their preferences in terms of music, which could aid towards segmenting the users into cohorts with similar interests that could end up in effective recommendations from the aforementioned techniques. Finally, we proceeded at analyzing user activity based on time-based intervals. We used by month and also by trimester in order to discover if there are any patterns on the listening habits of users from time to time. From there we proceeded researching if there was a correlation between the friends of a user and their listening habits and also the relationship between their listening behavior and the social interaction that they had done on the platform.

# Main Body

## Libraries and their roles

In the first cell of our code snippet, we can find all the libraries that are used throughout the entire project. This of course was done and update as we were progressing through all the necessary steps. Below is a brief overview of the libraries that we used and how what each one was accomplishing:

* pandas

Pandas is a library that we use almost in every cell of our entire project, because it provides the data frames, which are robust data structures. On top of this the library provides various efficient functions optimal for loading, cleaning, and transforming our data (McKinney, 2011). Finally, aid us to deal especially with CSV files and data aggregation techniques.

* numpy

We used numpy because of the computation capabilities that it can offer when it comes to numerical data. It aided us in our outlier detection part where we needed to calculate the z-scores. In general, the library is known for its ability to handle large sized datasets and perform fast complex mathematic operations on large arrays (Harris et al., 2020).

* matplotlib.pyplot and seaborn

These two libraries were used in order to produce the visualization through our project. They provided a framework in order to create a static, an animated or an interactive visualization in python programming language. Seaborn is built on top of matplotlib, and it offers some more options when it comes to the visual appearance of the visualization. They provide various choices, like some of them that we used, like heatmaps and plotting bar graphs.

* scipy.stats

This is a sub model of the SciPy library, that is used to mainly to perform statistical calculations. In our project that took part in the computation of the z-scores for the outlier detection. That library provided us an efficient way to calculate the score of the outliers with the aforementioned method directly from our data.

* sklearn.metrics.pairwise

This is a sub model or category of the Scikit-learn library, and we used in to calculate the cosine similarity between our dataset users, based on their music preferences.

* scipy.sparse

This module gave us the opportunity to work with sparse matrices, meaning the matrices that have in most of their elements the zero value, enabling a more efficient storage and computational times when it comes to dense one’s (Abbasi, 2018). This was found in our project when we tried to handle the user-artist matrix, which was heavily sparse because of the nature of the user interaction with such a large number of artists that were included in the dataset.

* json

This library was used for the purpose of encoding and decoding data of JSON file types, like we did in the k-nearest neighbors analysis part. JSON files are famous for their lightweight, easily interchangeable format and also at the same time easily to parse programmatically and readable from the human eye (Cecconi et al., 2023).

* plotly.express and plotly.graph\_objs

This is a library that we used in order to create an interactive plot, because it is suitable for making complex plots helping us understand the data and their insights through the interactivity that they provide. We used it in the final part, enabling us to explore our data in more depth, by hovering, zooming and by applying filtering capabilities.

* ipywidgets

Ipywidgets is a library that it is used for creating interactive HTML widgets inside Jupyter Notebook and Jupyter Lab environment, which we used in combination with plotly in the final part enabling us to use a drop down lists between the time intervals that we had segmented our users, allowing us a more dynamic exploration of the dataset.

## Data Processing

### Data Overview

The Last.fm dataset that we used contained various information regarding the interaction of the users with the music artists the digital platform music has, listening frequencies of the users, tagging information and last but not least social networking information of our users. So, in other words it provides an analytical framework that we can analyze and try to gain insights about not only the listening habits of the users but also about their social’s behaviors on the digital music platform.

### Data Loading Process

All of the six different datasets (artists, tags, user\_artists, user\_friends, user\_taggedartists and user\_taggedartists-timestamps) contained in the Last.fm dataset, were loaded using pandas' read\_csv function. After inspecting each one separately we found out that the delimiter was common throughout all of them, and it was the tab, so we used the delimiter=’\t’ as a parameter in the reading function. Finally, we used the encoding='ISO-8859-1' parameter throughout our reading process, which is the character encoding for Western European languages, and means that a single byte is used for the representation of the characters, taking into account special symbols and accents, ensuring that way the accurate data representation (Martelli et al., 2005).

### Initial Data Exploration

So, after reading all the files, we proceeded with creating a loop for inspecting for each one of them some basic information. The pandas’ functions that we used here were the df.info() to get an idea of the data types that each one contained, the non-null counts alongside with the memory that each one takes (Pandas DataFrame info() method, n.d.). Also, inside the same aforementioned loop we placed also the df.describe() function, which generates a summary statistics of the numeric columns found in our data frame, such as count, mean, max and min values, standard deviation and quartiles that helped us gain an idea regarding the data distribution and central tendencies (pandas DataFrame describe() method, n.d.). By performing those steps, we obtain a better idea on what we should do in terms of data cleaning and transformation.

A computer screen shot of a black screen

Description automatically generated

Here is also an indicative result of the aforementioned loop for the user\_artists data fame.

Here we also inspected the unique numbers of the main categories, which are:

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

### Data Visualization

In order to visually describe the dataset, we proceeded with two visualizations. First, we proceeded with a listening frequency plot with seaborn, to explore the listening frequencies of artists by the users of the user\_artists dataset. So, in order to get the most popular one’s we sliced the original data frame by taking only the artistID column, and because our dataset is user per row by counting the values of the specific column and sorting them, we were able to obtain the frequency that the users have heard of a specific artistID. Here is the code that achieved what was just described:



And then we continued with plotting it, putting by using seaborn’s barplot function, putting as x-axis the aristsID and in the y-axis the frequency or the value counts of their appearance. Here is the plot:

A graph of colored bars

Description automatically generated with medium confidence

Similarly, we proceeded with doing a frequency plot of tags per user. We performed again the same programmatic logic as the above plot, taking now from the user\_taggedartists dataset a sliced data frame with only the column of userID. Here we counted the values and that’s how we obtained the tagging frequency. Finally, we sorted the sliced data frame and kept the top 20 values in terms of frequency. Here is the respected command:



And then we continued with plotting it, putting by using seaborn’s barplot function, putting as x-axis the userID and in the y-axis the frequency or the value counts of the tagging. Here is the plot offering a clear visualization of the values:

A graph of different colored bars

Description automatically generated

Finally, we did another plot to see which the ‘best’ users in terms of how many friends they have, to see which one’s are more active users on the digital music platform is. Here is the plot:

A bar graph of different colored bars

Description automatically generated

### Outlier Detection

In exploratory data analysis, outlier detection has a significant role because through this process we are able to identify anomalous data points, which if not removed can skew the results of our analysis and lead us to incorrect conclusions. For this part of our project, we proceeded at comparing two different methods for the detection of outliers, the z-score one and the Interquartile Range (IQR) (Singh & Upadhyaya, 2012).

#### Z-Score

This is a method that measures how many standard deviations an element differs from the mean of the specific value that it is compared to, from the dataset. This method is quite sensitive to the values of the mean and standard deviation, and because of that is a more suitable method for distributions that are closer to normal and worse for the skewed one’s (Ghosh & Vogt, 2012).

In our script, through the help of numpy library we calculated the absolute z-scores of the column ‘weight’ in the data frame of user\_artists, which is the column that represents the listening frequency of each user in the digital music platform. So, this method uses a threshold to determine the difference of the standard deviation in order for an element to be characterized as outlier. We tried values from 1.5 to 4 and the results can be shown in the following plot:

A graph with a line graph

Description automatically generated

As long as we increase the threshold it is logical that the number of outliers detected is decreased as well, because as we assig higher thresholds values that means that more and more extreme features are indicated as outliers by the z-score method. In order to determine the optimal threshold, we looked at the above plot, and try to locate that area where the rate of the decrease of the number of outliers started to significantly slow down, meaning that in this particular point the optimal threshold should be around. The aforementioned point seems to be around 2.5 and 3, where between the specific range the plot starts to flatten a bit, suggesting that beyond that point the threshold will not reduce significantly the number of outliers, and that way giving us a better image regarding which ones are the true outliers of the listening frequency users.

#### IQR Method

This is a measure of statistical dispersion and in order to calculate the Interquartile Range we must find the difference between the 75th and the 25th percentiles. To be more precise this method turns back results based on the spread of the middle 50% of the data, and it identifies observations as outliers when they are below Q1-1.5IQR or then above Q3+1.5IQR (Rousseeuw & Hubert, 2011).

This method is particularly useful for identifying outliers in skewed distributions and is less influenced by extreme values. In general, it is a more robust application when dealing with non-normal distributions datasets, and typically identifies more outlies than the z-score method.

Below are two plots, left with the distribution of the weight’s column and to the right the same one in logarithmic scale, which justify the above mentioned:

A blue graph with numbers

Description automatically generated with medium confidenceA graph of weight distribution

Description automatically generated

And finally, this is an overall comparison of the outliers detected by each method:

A green and blue bar graph

Description automatically generated

All the above were done for the detection of outliers based on the weight column, which represents the frequency. On the final python code snippet cell for this task, we proceeded with calculating all the three different categories for outliers’ detection. We now did it by hand and without the z-score function. For each one of the three cases we calculated first the respective frequency and then we proceeded with calculating the standard deviation and the mean for all the three different cases, and then we assigned in new data frame with a respective name for each one the z-score with was the frequency deducted by the mean and then divided by the standard deviation. All the z-scores for all three cases used the standard threshold of ±3 by using the absolute value.

After the above research we performed the two methods for outliers’ detection in all the three different categories that we are asked to do, following the same procedure as aforementioned, and coming up with a total comparison plot, in terms of the number (numbers differ so much because of the distributions as mentioned above):

A graph showing different colored bars

Description automatically generated

## Similarity Assessment

### Cosine Similarity

Then we proceeded with the cosine similarity assessment of our users between the artists that they have heard and the weight parameter. So, for that reason we created a pivot table with index the userID, columns the aritstsID and as values the weight. Then we proceeded with applying the cosine similarity, which is the measurement of the angle between two non-zero vectors in the context of a multiple dimensional space, which in our case represented the habits of listening of our users (Lahitani, Permanasari, & Setiawan, 2016). Finally, we stored the file into CSV type, which can be found useful for later analysis and the making of a recommendation system.

### K-Nearest Neighbors (KNN)

After the cosine similarity analysis, we proceeded as requested with the application of the KNN algorithm. This is a simple but nevertheless effective method towards finding similar entities into a dataset, which we used to identify the k-nearest neighbors for each user based on the similarity scores that we have performed in the previous step (Zhang et al., 2018). So, we performed the method for two different variables (3 and 10 neighbors), which could provide insights regarding user segmentation and categorization in broader groups, as well as further marketing targeting techniques. Below is a short preview:

A screenshot of a computer

Description automatically generatedA diagram of a box plot

Description automatically generated

## Dynamics of Listening and Tagging

Splitting the data into time intervals can provide meaning insights on how our users perform between the different intervals of time and also it can reveal if there are any seasonality trends across the entire timestamp range of our dataset, or even how a new release determines a specific change towards user behavior.

So, firstly we proceeded with a transformation process of our timestamp column of the user\_taggedartist data frame, in order to facilitate our time-based analysis. We created a safety function in order to convert timestamp from milliseconds to seconds and then proceeded with the two intervals, monthly and trimester (the code snippet is running for the monthly interval but can very easily be adjusted for trimester with a change of M to Q), here is the respected code that achieves that process:



### Users, tags, and artists per interval

After the aforementioned steps, we proceeded at saving and printing the number of the unique users, unique tags, unique artist, top5 artists and top5 tags to a dictionary and printed out to see the results. This was quite extensive, so we proceeded with creating an interactive plot with plotly plotting every interval separation that we have made above. Here is a static picture of the mentioned plot, in the code it offers some dynamic interaction:

A graph showing the growth of a number of individuals

Description automatically generated with medium confidence

### Top5 artists and tags per interval

As for the top5 artists and tags per interval (and specifically the month which have as default value), we continued on making another interactive visualization with plotly and ipywidgets library, in order to be able to change between the different intervals and see the top5 results in terms of frequency from both of the 2 categories mentioned. Here is a static representation of the plot with the dynamic widget of choosing list interval:

A screenshot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

## Correlation Analysis

To perform our correlation analysis, first we started by aggregating the necessary data in order to calculate the total number of unique artists a user has listened to, every user’s total listening time and finally the number of friends that they have on the digital music platform. By using the power of panda’s library, we started merging different data frames that we have read in the beginning of our code snippet. So, now we are ready to perform the correlation analysis, which is a statistical measure that aims towards describing the extent to which two variables change together. This means that when we have a positive correlation, as one variable increases the other one’s following the same trend the when the correlation is negative, they go in the opposite directions (Santamaría, Pokharel, & Principe, 2006).

So, we proceeded at using the .corr() function of Pandas to perform the correlation analysis, and then we plotted a heatmap to visualize all the relationships between the three different variables. Through the heatmap is quite easy to identify which variable has a positive correlation with which one, and the opposite. The plot was performed with seaborn library for an enhanced interpretability:

A screenshot of a graph

Description automatically generated

### Artists Listened and number of friends

So, from the table we can see that there almost no correlation between the number of artists that a user has listened to and the number of friends that it has. In order to be completely sure, we proceeded with making a scatter plot with a regression line (to aid us towards understanding the linear regression correlation) in it, using in the y-axis the number of friends and in the x-axis the number of artists. Here is the plot:

A graph showing the number of artists

Description automatically generated

### Listening time and number of friends

From the correlation matrix, we can observe that there is a correlation between the number of friends that a user has and their total listening time, so in order to test that we proceeded with plotting again a scatter plot with seaborn library, with a regression line in it, to visualize even better the situation that we are dealing with. Here is the plot:

A diagram of a number of people listening

Description automatically generated

# Conclusion and further analysis

Through the analysis that we have performed in the Last.fm dataset, we are able to spot and analyze behaviors of user in the music digital platform. This varies from understanding their preferences and the patterns that they show when it comes to engagement with the platform, to identifying similarities between users, and even how the perform during the entire time that we have recordings on separated by intervals.

The results can provide benefits to multiple stakeholders of the company, because of understanding their user’s behavior and that way adjusting their recommendation systems accordingly, for the artists themselves, how their releases were performing and shaping the foreground of the digital music platform, and finally even to providing some guidelines to marketing and promotional acts because of the identification of the popular trends and insights.

Concluding, some future work that could be performed is a genre-based analysis, to understand the behavior and the trends between the two different genres with even the combination of a machine learning model that could aid towards predictions of trends and preferences. Also, another thing would be a user segmentation, by applying clustering algorithms in order to segment the users of the dataset into based on their behaviors and interactions.

# References

Abbasi, H. (2018, July). Sparse: A more modern sparse array library. *In Proceedings of the 17th Python in Science Conference (pp. 27-30).*

Cecconi, B., Louis, C. K., Bonnin, X., Loh, A., & Taylor, M. B. (2023). Time-frequency catalogue: JSON implementation and Python library. *Frontiers in Astronomy and Space Sciences, 9, 1049677.*

Ghosh, D., & Vogt, A. (2012, July). Outliers: An evaluation of methodologies. *Joint Statistical Meetings*, 12(1), 3455-3460.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Lahitani, A. R., Permanasari, A. E., & Setiawan, N. A. (2016, April). Cosine similarity to determine similarity measure: Study case in online essay assessment. *In 2016 4th International Conference on Cyber and IT Service Management* (pp. 1-6). IEEE.

Martelli, A., Ravenscroft, A., & Ascher, D. (2005). *Python Cookbook.* O'Reilly Media, Inc.

McKinney, W. (2011). pandas: A foundational Python library for data analysis and statistics. *Python for High-Performance and Scientific Computing, 14(9), 1-9.*

pandas DataFrame describe() method. (n.d.). Pandas documentation. Retrieved December 1, 2023, from <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html>

pandas DataFrame info() method. (n.d.). Pandas documentation. Retrieved December 1, 2023, from <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.info.html>

Rousseeuw, P. J., & Hubert, M. (2011). Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), 73-79.

Santamaría, I., Pokharel, P. P., & Principe, J. C. (2006). Generalized correlation function: Definition, properties, and application to blind equalization. *IEEE Transactions on Signal Processing, 54(6)*, 2187-2197.

Singh, K., & Upadhyaya, S. (2012). Outlier detection: Applications and techniques. *International Journal of Computer Science Issues (IJCSI),* 9(1), 307.

Zhang, S., Cheng, D., Deng, Z., Zong, M., & Deng, X. (2018). A novel kNN algorithm with data-driven k parameter computation. *Pattern Recognition Letters,* 109, 44-54.