Semester Project Fall 2019

“Data analysis on Million Song Dataset”

Project supervisors:

SPYROPOULOS Thrasyvoulos

GIANNAKAS Theodoros

Student:

IAKIMOVA Iuliia

TABLE OF CONTENTS:

1. INTRODUCTION

Recommendation systems are one of the most developing approaches to keep users in the use of the application. The field of recommendation systems has gone through a lot of innovations and developments. In that research, project gives a study of a statistical analysis based on the large collection of music data set.

In overall the given project will contain information about the structure and meaning of collected data, followed by the measured relationships among categories of data and the analysis of its features.

For the correct placement of elements in the playlist, we need to clearly understand the relationship between the popularity of the song and the conditions of their location. Some conditions dictate such rules as a popular element tend to be in the first part of the playlist in order to keep the listener or certain order of popular elements is more common than others. The following chapters will demonstrate how song's position changes from its popularity.

Each item in the data set has its own value. The main analysis of this project is aimed at arranging the most popular elements and, to be more precise, the first ten, one hundred and one thousand elements most often appearing in the playlist. These extracted elements will be checked for distribution, percentage of appearance, their position in a specific place in the playlist and relation with each other. After this analysis, it will be understandable how the most trending elements in the recommendation systems should behave, which is important for future developments.

2. DATA ANALYSIS AND ITS INTERPRETATION

2.1 Pre-processing

Defining and structuring data. From the outset, it is crucial to obtain a clear understanding of the meaning of data *(Table 1)* and summarize the initial content of the data set for the further qualitative research.

The entire dataset contains a collection of:

- 1,000,000 playlists
- 66,346,428 songs/elements
- 2,262,292 unique songs/elements

By looking at the number of elements and their uniqueness, this research should take into account the relationship between these elements in every playlist. In considering the future steps, one needs to remember is that each set consists of intersections and dependencies.

Playlist	Song
['spotify:track:0UaMYEvWZi0ZqiDOoHU3YI', 'spotify:track:6I9VzXrHxO9rA9A5euc8Ak', 'spotify:track:0WqIKmW4BTrj3eJFmnCKMv', 'spotify:track:1AWQoqb9bSvzTjaLralEkT', 'spotify:track:1lzr43nnXAijIGYnCT8M8H', 'spotify:track:0XUfyU2QviPAs6bxSpXYG4']	'spotify:track:0UaMYEvWZi0ZqiDOoHU3YI'

Table 1. Example of independent playlist and its song tacking as a part of the dataset

2.2 Song position in the playlist

The first method aims at establishing the relationship between the song's popularity and its position in the playlist. This study allows to determine: if the song has a high percentage of popularity, will it be placed at the beginning of most playlists or it will not have any effect on the playlist.

The following implementation steps were taken to reach the solution:

Step 1. From the data set of unique elements, the frequency of occurrence of these elements over the entire data set is calculated. Thus, for each of those unique elements, we get mapping with the number of times it appeared. After obtaining the above relation, a complete analysis is built on the basis of the first 10 / 100 / 1000 popular elements.

Consequently, in the case when we consider the first 10 popular songs, we analyze only those playlists where any of the popular 10 songs appeared. Following table represents that for the top-10 songs we can analyze '175068' playlists where the combination of songs can locate in the different variances [['1', '7', '9'], ['3'], ['5, '8', '10'], ...]. In a similar way the process works for the top-100 and top-1000 elements *(Table 2)*.

Number of popular songs	Number of playlists
10	175068
100	403964
1000	508074

Table2.

Step 2. At this stage, each playlist has been divided into three segments (start, middle, end) and these segments were checked sequentially for the presence of a song from a popular list in it. In other words, establish the position of the song in the playlist.

Step 3. Going through each song individually. In the case when a song resides in a certain part of the playlist, its value increased by 1. Thus, at the end of the algorithm, we can get a set of data reveals how many times a certain song appeared in which part of the playlist. *(Table 3)*.

For each song from the popular list, now we have values indicating the number of times the selected song is found at the beginning part, the middle part or the end of the playlist.

Song	Number of times it appears
'spotify:track:7KXjTSCq5nL1LoYtL7XAwS'	[13943, 13079, 18814]
'spotify:track:1xznGGDReH1oQq0xzbwXa3'	[17797, 12653, 12141]
'spotify:track:7yyRTcZmCiyzzJlNzGC9Ol'	[16702, 12149, 12177]

Table3.Number of times songs appears in the certain part of the playlist

Step 4. Since the main object of study is the playlist, we summarize the values of each of the columns throughout the top-10, top-100, and top-1000 songs and normalize them in accordance with their popularity.

$$P(i) = \frac{Frequency of song(i)}{Total number of songs} \sum \quad P(1) * List(1) + .. + P(n) * List(n)$$

Step 5. The following table lists the results for the implemented algorithm:

Thus, at the output for each set of popular songs, we get the ratio in which part of the playlist it is likely to appear. The outcomes indicate that the more popular the song, the more likely it is to be at the beginning of the playlist *(Table 4)*. Although the ratio between the three parts is not much different – the first part is several percentages higher than the rest.

Number of popular songs	Normalized Sum of songs indicating their position
10	[0.35734357078898976, 0.29621361972430393, 0.34644280948670625] [0.36, 0.29, 0.35]
100	[0.3636421486676663, 0.3061659976768844, 0.3301918536554493] [0.36, 0.31, 0.33]
1000	[0.3625561783456898, 0.3168605696712594, 0.3140135981397527] [0.36, 0.32, 0.31]

Table4.

2.3. Measurement of the song's weight sum in the playlist.

Using the output of the step described in part 2.2 (Song position in the playlist), it was further decided to view the individual parts of the split data set. Idea is to calculate the sum of the song's popularity for a particular part of the playlist divided into a selected number of parts. Where a song's popularity is its normalized number representing its weight relative to other songs.

The main difference between the two approaches is that in case 2.2, the probability of a song is considered, while in part 2.3, we look at the weight of the song and the sum of these weights. However, the probability of a song being also calculated on the basis of a normalized number and should represent conceptually the same result, which once again confirms a certain dependence in the playlist.

Step 1. With the help of the "collections.Counter()" function. The number of times each song appeared has been counted and retrieved.

Step 2. In order to calculate the amount for the full playlist, it is necessary to go through each playlist in the entire dataset and summarize the "key: value" results for each song that appeared along the way.

Step 3. As mentioned above, the main goal is to compare parts of the playlist and find out if the first part has a higher result than the second part. Therefore, each playlist was divided into two parts, after which the sum is calculated for each part individually and compared with each other at the final stage (*Table 5).*

Step 4. The same operations and comparisons have proceeded on the playlists divided into 3 parts. *(Table 6)*

Number of songs	First part	Second part	Normalized comparison
100	13539455	12245361	(0,53 - 0,47)
1000	163389308	150034564	(0,52 – 0.48)
10000001	154557784886	141707718160	(0,52 – 0,48)

Table5. The sum of popularities in a playlist divided into 2 parts

Number of songs	First part	Second part	Third part	Normalized comparison
100	9431386	7887498	8465932	(0,37 – 0,31 - 0,33)
1000	111050868	101226653	101146351	(0,36 – 0,32 - 0,32)
10000001	105125375142	95329741773	95810386131	(0,36 – 0,32 - 0,32)

Table6. The sum of popularities in a playlist divided into 3 parts

2.4 Probability mass function for the first 10/100/1000 popular songs.

The major part of statistical analysis is the frequency function for a discrete random variable that termed as probability mass function (PMF). It integrates exact probability for a particular value in a given dataset f(x) = n(x)/n.

In our case, the natural frequency of songs is a large number, calculated on the basis of a complete set of data. Therefore, the normalized number has been calculated for every element in a list in order to build a clear understandable graph that will represent the outcomes. Algorithm performed under a number of different parameters, including the value of the song's weights and the position number of the measured elements.

To visualize data, it requires us to extract appropriate elements and calculate the final value for them. More precisely, one approach is to extract the first 10/100/1000 elements from the entire dataset and determine their frequency in the case of the natural values and their normalized number. Looking at the outcomes *(Table7, Table8, Table9)* we will be able to visualize and represent them using the histogram below.

Song number (1.2.3)	Normalized weight	Frequency
Song #1	0,1202	46574
Song #2	0,1121	43447
Song #3	0,1066	41309

Table7. Song's weight relative to the range of first 10 popular songs

Song number (10,20,30)	Normalized weight	Frequency
Song #10	0.0175	46574
Song #20	0.0127	33699
Song #30	0.0114	30353

Table8. Song's weight relative to the range of first 100 popular songs

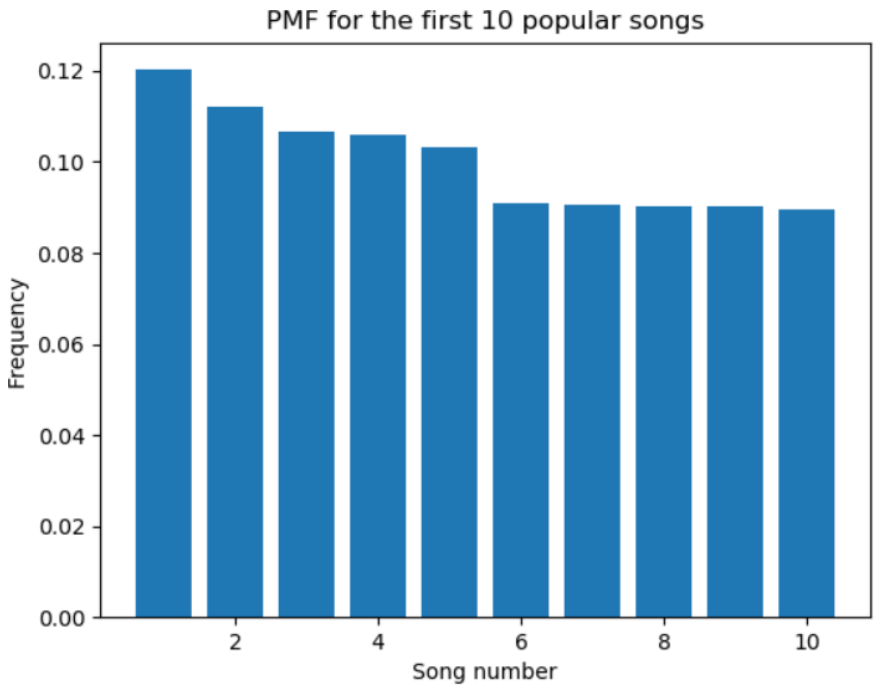
Song number (100,200,300)	Normalized weight	Frequency
Song #100	0.0035	46574
Song #200	0.0015	20202
Song #300	0.0012	16316

Table9. Song's weight relative to the range of first 1000 popular songs

PMF for popular songs plotted on a bar chart for easy interpretation of the given data.

The probability graphs *(Table 10)* are formed from:

- Vertical axis: Normalized frequency of songs according to the fixed amount of popular elements.
- Horizontal axis: Ordered elements in a range of fixed length.



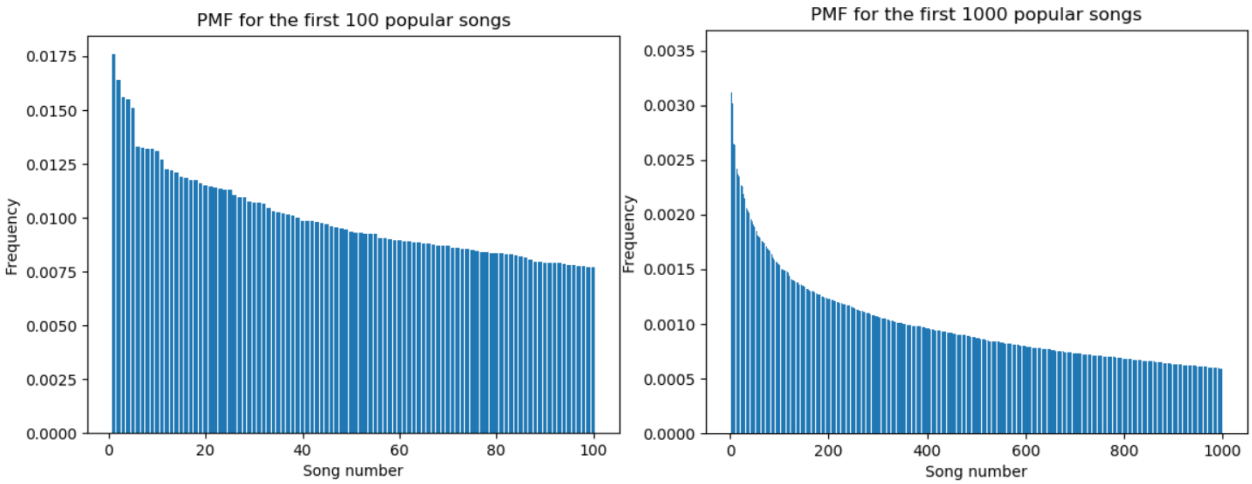


Table 10 (PMF graphs)

2.5 Kolmogorov-Smirnov test.

Kolmogorov–Smirnov test a very efficient way to determine if two samples are significantly different from each other. In our study, the main goal for using the KS test was to check the presence of the uniform distribution in the case of song's popularities.

Uniform distribution

To use the test for checking the uniformity of random numbers, we use the PMF (Probability mass function) of P[0, 1].

F(x)=x for 0<=x<=1

Looking at the PMF graphs of any numbers it can be directly said that there is no clear uniform distribution there and the difference in comparison the probability data and classic uniform distribution will be quite large. However, looking at the graph in detail, we see that a sharp decline happens on the first x-elements, after which the difference equalizes and the values gradually turn into a uniform distribution.

The values in the Kolmogorov-Smirnov test suggest that the numbers are uniformly distributed, but the hypothesis is provided with a high probability of rejection. If we are able to reject the Null Hypothesis, this means that the numbers are not uniformly distributed between.

KstestResult 10 (statistic=0.8797513128882509, pvalue=1.264261428856554e-09)

KstestResult 100 (statistic=0.9824101633859925, pvalue=6.717544343443967e-176)

KstestResult 1000 (statistic=0.9964855498099024, pvalue=0.0)

2.6 Playlist length and popularity measurement.

The main features of our dataset are the playlist and song. As defined before, the playlists have a different length what means they consist of the different number of elements in it. The main idea for the next measurement is to find dependencies between playlist length and the number of popular songs that appeared in them.

The entire dataset contains:

- 1,000,000 playlists
- 66,346,428 songs/elements

Here, we consider all playlist in a dataset and measure the number of elements appeared in every playlist individually. In this way, at the end of the calculation, we will see the exact length for each list and it will be easy to extract the average mean value and the median value in order to process the next implementations.

The results of such efforts were as follows*(Table9)*:

Playlist number (1.2.3)	Number of elements	Average number of elements	Median number
Playlist #1	52		
Playlist #2	39		
Playlist #3	64		
Playlist #4	126		
… … …			
All playlists		66,34	49

Table9. Playlist length

Next, all playlists were separated into two categories, one of them represents the lists where the number of elements less than the mean value while the other category has the playlists with the length more than the mean value. Therefore, now, we can look at the particular songs in the playlist part and, more specifically, to the 100/1000 list of the most popular songs appeared in the chosen categories.

The key concept for the listed algorithm is to find the percentage persistence of popular songs in the playlists which are more or less in the length than the average and median value among them. In this splitting approach, we get the following separation in two parts which are a number of playlists less than a certain number on the left and more than a certain number on the right:

- 495048 / 504953 (Median value)
- 627443 / 372558 (Average value)

Processing each playlist and its elements in the selected category. The elements existing in the list of popular songs are recorded in a new list while the rest are recorded in another list. In the final analysis, it will be clear to compare a number of popular elements in a short-length playlist with the number of popular elements in a long-length playlist. (Table10, Table11, Table 12):

Playlist length	Number of elements (popular – not popular)	Percentage number (popular – not popular)
Less than average number (66)	[940799 - 19615304]	[0,05 – 0,95]
More than average number (66)	[1706980 - 44083345]	[0,04 – 0,96]
Less than median (49)	[608529 - 12405120]	[0,05 – 0,95]
More than median (49)	[2039250 - 51293529]	[0,04 – 0,96]

Table10. Number of times first 100 popular songs appeared in the playlists

Playlist length	Number of elements (popular – not popular)	Percentage number (popular – not popular)
Less than average number (66)	[4426527 - 16129576]	[0,22 – 0,78]
More than average number (66)	[8825617 - 36964708]	[0,19 – 0,81]
Less than median (49)	[2806545 - 10207104]	[0,22 – 0,78]
More than median (49)	[10445599 - 42887180]	[0,20 – 0,80]

Table11. Number of times first 1000 popular songs appeared in the playlists

Measure the number of popular items in playlists where the number of items is very small or high.

Based on the fact that the median number is (49). It was decided to check the availability of popular elements on extreme sets of playlists, namely those playlists whose length is less than (19) and those playlists where the length of the set is more than (79).

By measuring popularity in this way, we can clearly check whether the length of the playlist affects the number of popular songs in it or not.

Playlist length	Percentage number for Top 100 (popular – not popular)	Percentage number for Top-1000 (popular – not popular)
Playlist length < 19	[0.05 – 0.95]	[0.21 – 0,79]
Playlist length > 69	[0.03 – 0.97]	[0,19 – 0,81]

Table12. Number of times first 100 and 1000 popular songs appeared in the playlists

The *table 12* shows the values obtained. For example, value (0.21) in line 1, column 3 indicates that the percentage number of the 1000 popular elements to exist in playlists is 0.21 and the remaining part of 0.79 are all other elements not existing in the popular set of 1000 elements.

From the measurements made, it can be concluded that short playlists are more likely to have the most popular songs compared to the long playlists. This is also confirmed by the fact that the longer the playlist, the more songs we need to add to it and accordingly reduce its popularity since a long playlist can't consist only of the popular elements.

2.7 Order of the first 4 popular songs among all playlists

The playlist presents a sequence of the elements taken from the dataset in order to make the final recommendation. While generating relevant items to user preferences in the recommended list is a prime requirement, the order in which they appear in the list is important as well. As such, this part explores the order in which the particular items appear in the chosen playlist.

Analyzing the element's order:

Step 1. First of all, it requires to identify the most popular items, its names and frequency number. Considering the fact that the track name in our dataset is a set of characters. For a better understanding of the next steps, the symbolic names are renamed to the ordinal digits ['1', '2'. '3', '4'] in accordance to their sequence in popularity. *(Table 15)*

Song's name	Number of times it appears	Given name
'spotify:track:7KXjTSCq5nL1LoYtL7XAwS'	46574	'1'
'spotify:track:1xznGGDReH1oQq0xzbwXa3'	43447	'2'
'spotify:track:7yyRTcZmCiyzzJlNzGC9Ol'	41309	'3'
'spotify:track:7BKLCZ1jbUBVqRi2FVlTVw'	41079	'4'

Table 15.

Step 2. Since the playlist can contain any item from a set of songs contained 66,346,428 elements. The next step is to extract the popular elements obtained in the previous step in the same order in which they are located in the original data set.

In our case, the main analysis is aimed at the first 4 most popular elements. Thus, their sequence over entire data set can be displayed in various ways. For example, the list can be represented like: [['1', '2', '3', '4'], ['2', '1', '4'], ['1', '3', '4'], ['2', 4'], ['1'], [] …etc.]

Step 3. It is necessary to exclude the case when none of the considered popular elements appeared in the playlist and remove inappropriate playlists from the considered set.

2.7.1 Number of all playlists in the dataset: 10000001

2.7.2 Number of playlists with first 4 popular items found in them: 124681

Also it was unexpected to find that in the same playlist, the identical popular song appears more than once in a different sequence. For example, excluding other songs, the playlist might look like this: [['2', '3', '3], ['2', '3', '4', '3'], ['2', '2', '2', '2'], ['2', '3', '2', '4', '1'], ['1', '1'] …etc.]

This means that the algorithm of recommendations from time to time may return the user to the song, which is the most popular in his case

Step 4. The following step is counting the frequency of each unique set of popular elements.

Here below is the part of the table with represented data (Order in the playlist - number of times it appears):

(1)	28219	(3, 4, 2, 3, 1)	1
(4)	24002	(3, 2, 1, 4, 1)	1
(2)	20192	(1, 3, 2, 4, 2)	1
(3)	15308	(2, 3, 1, 4, 2)	1
(3, 1)	5393	(4, 3, 3, 3, 3)	1
...	....	Length: 366	

Step 5. In the interest of making visualization, the data obtained is designed on a bar chart. Where the x-axis is the obtained sequence and the y-axis is the number of times this sequence appears over the entire dataset.

Two bar charts are presented in order to clearly show the full data with the frequency moving down from '28219' to '1' (Table 16) and the beginning part of the graph showing the first 20 lists. (Table 17)

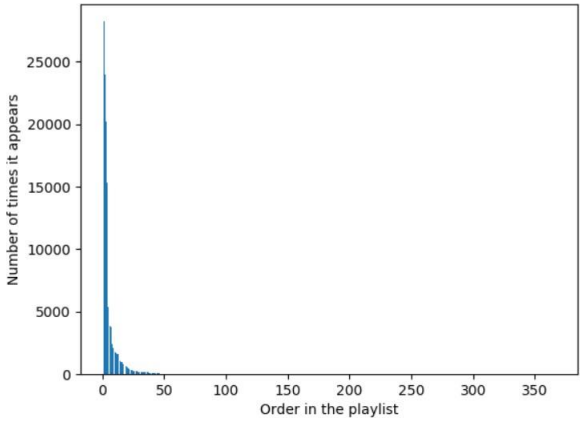


Table 16. (Full data range)

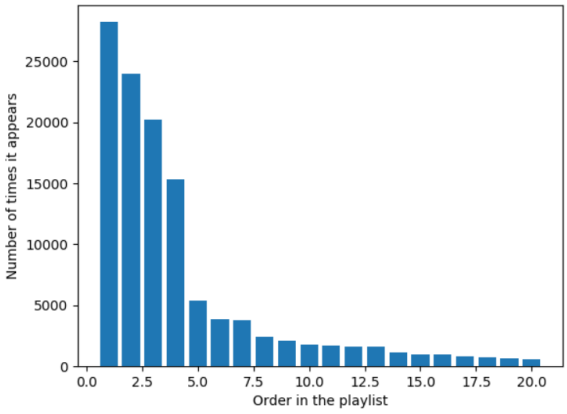


Table 17. (First 20 playlists)

In order to examine in detail, the resulting table, pairs consisting of two elements are selected.

Over the entire dataset, 16 complete combinations of pairs were found. The graph below shows the probability mass function of a particular pair appearing *(Table 18).*

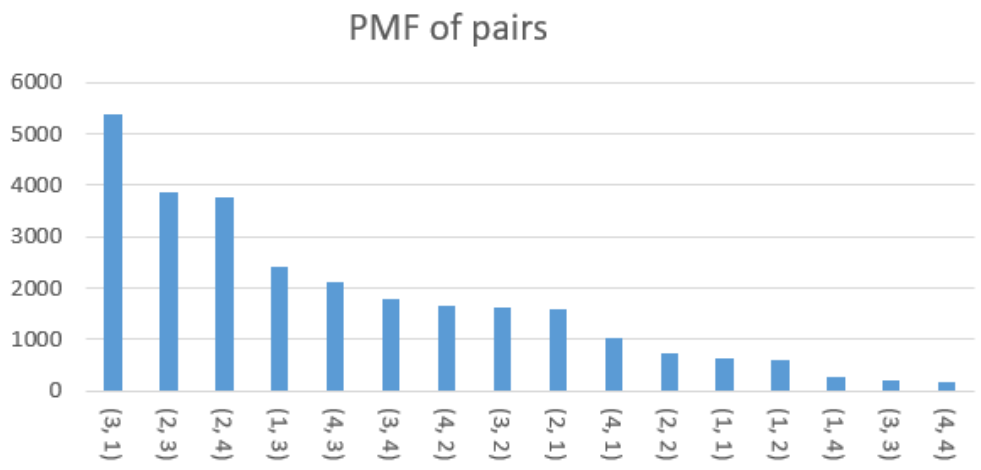


Table 18 (PMF of pairs)

One of the analysis options that we can perform on the basis of the received data is to calculate if the order in the songs in the playlist is prescribed or not. Thus, from 16 pairs we find that 10 elements follow each other and can be located as [[1,2], [3,4], [1,3], [1,1]] while the other part is located in a distorted version [[3,2], [4.1]]. In the graph below *(Table 19),* we can see that the prescribed order of the songs provides the probability of '0.52' and the distorted order is '0.48'

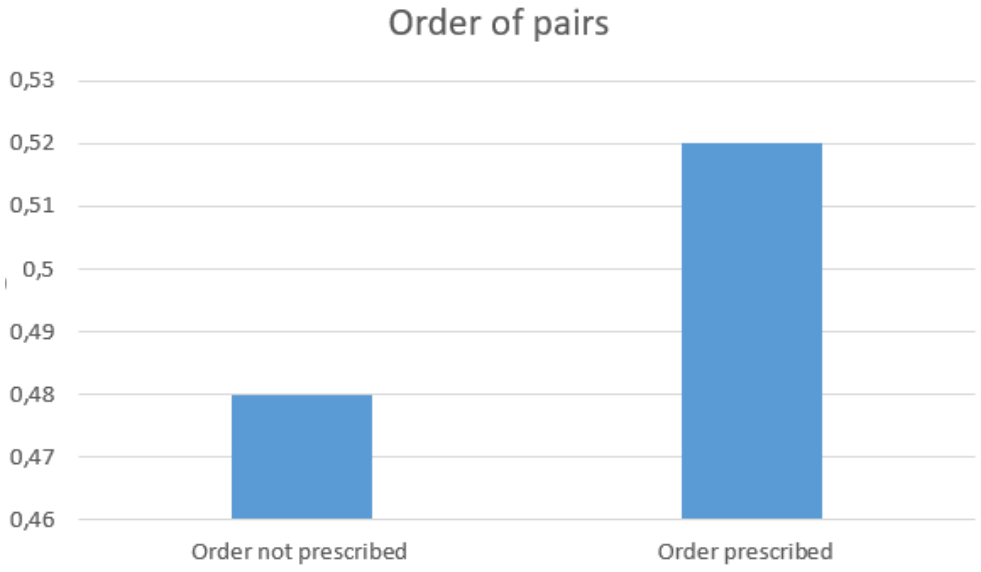


Table 19 (Order of pairs)

2.8 Measure the popularity of the playlist.

Each playlist in our dataset is a collection of the elements. And it is worth considering that all the elements in the playlists have their own level and weight of popularity. To be clear the popularity of the song is the normalized number indicating how many times this song appeared in the entire dataset among other songs.

By measuring the popularity of songs in the playlists we can thus measure the popularity of a playlist. To be more precise, if one of the playlists contains only mainstream elements, then the sum of these elements will show a high value among the summed song's weights and will mean that the higher the value the popular is the playlist.

The process is explained below:

Passing through each playlist individually, we summarize the popularity values of the songs found in it. Thus, if we have 100,000,000 playlists, then the loop will go through each playlist individually and at the output, we will get a million values.

For example, if our list consists of 10 elements (song1 .... song10), then specifically exploring this playlist, we summarize the popularity value of each of these elements (Sum (p(i)) where p(i) is the popularity of the song (i). And in the end, we divide the resulting sum to the length of the playlist in order to obtain equal conditions for all playlists of our dataset. Because every playlist has its own length and its own set of elements.

Visualization of the playlist popularity:

At the time when a million values have been received, we want to see how the popularity of the playlist changes throughout the entire data set. To do this, we can plot the received data in sorted order to a histogram.

A million values are combined into the 100 chunks and compared with their values. On the graph below (Table 20) we see that the popularity of playlist in an ordered way decreases, while the numbers belong to the beginning part a quite high which will mean that around 30% of all playlists consist of the popular items or of the high amount of the elements.

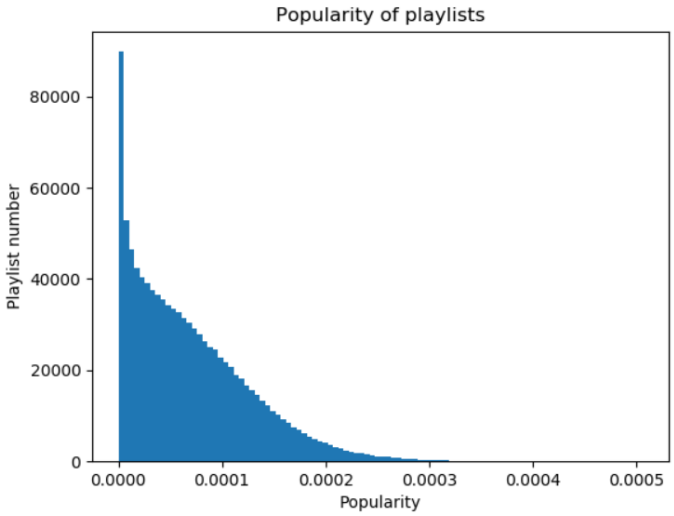


Table 20 (Popularity of playlists)

3. CONCLUSION

The project made openly shows that the relation between the elements in the recommendation systems exists and this should not be ignored.

Considering that the selected element is one of the most popular elements, most likely you need to carefully think about its location and analyze its nearest neighbors for building the correct relation. In addition, when compiling a playlist, be sure to pay attention to its length. Needs to note that the shorter the music list, the better and faster it should hook the listener so that the user returns to the playlist again and again.

The analysis of probability, distribution, and normalization that may be related in the future work can be used by recommendation systems for the better compilation of a playlist since this is of no small importance.