Clustering Algorithms for Economic and Psychological Analysis of Human Behavior

Nataliya Boyko[0000-0002-6962-9363], Hanna Komarnytska[0000-0002-5533-6439]

Yurii Kryvenchuk[0000-0002-2504-5833], Yuriy Malynovskyy[0000-0002-7139-5623]

Lviv Polytechnic National University, Lviv79013, Ukraine

nataliya.i.boyko@lpnu.ua, lozjuk7@gmail.com,
yurkokryvenchuk@gmail.com, inem.news@gmail.com.

Abstract. This article proposes an approach to analyze the behavior of groups of people and shows how to predict person location for the next month. The clustering algorithms were used in this research. Also was inspected the problem of finding associative rules. We used scalable Apriori algorithms to find the best rules. For analysis, we used the standard mlxtend library to aggregate data by cluster, user login, and time. In this article we explored apriori and *k*-means clustering algorithms to get user behavior analysis template. In the process, we looked at the problem of finding associative rules that were able to find and describe patterns in large datasets. We used scalable Apriori algorithms to find the best rules. For analysis, we used the standard mlxtend library to aggregate data by cluster, user login, and time. While working, we were faced with the problem of inaccuracy and inconsistency of data with real conditions, and were forced to reduce the minimum support for associative rules.

Keywords: Preprocessing, clustering, associative rules, Apriori algorithm.

1 Introduction

Nowadays, information technologies offer lots of opportunities for communication and social networking. Such partial isolation is a huge problem for establishing socio-communicative relationships between people. One way to partially solve this problem is to use the meet city. This application uses geolocation and artificial intelligence methods to predict and recommend meetings. For example, if you desire to talk to a similar-interest person during your lunch, this application will help [1-3].

To solve the problems described above we should use machine learning technologies. We propose to use clustering and associative rules. While working, we have formed a pattern of finding algorithms and patterns of behavior of people or groups of people united by common interests [8, 13].

2 Statement of the problem

To begin the analysis, we select the following data (Fig.1):

	A	B	C	D	E	F
1	Login	Latitude	Longitude	PlaceName	Ratting	TimeStamp
2	vetall	49.819634445545894	24.01395733984498	Driving	4	1546527439.373259
3	Sofia	49.81478797276487	24.015408279396606	Meal	3	1546532847.104377
4	vlad	54.935931447013222	83.1965436414659	Culture_places	5	1545278838.463506
5	sofia	54.93592457385262	83.19668613381981	Healthcare	4	1544973519.072424
6	vlad	54.97667698974983	83.04515953895823	Sport	5	1544842790.35071
7	sofia	54.93592457385262	83.19668613381981	Shopping	3	1544973920.063797
8	vova	54.93592457385262	83.19668613381981	Amusements	4	1544973920.063797

Fig. 1. Meetcity application input

First of all, we need to vectorize (binarize) the text data of the PlaceName field in order to be able to apply clustering using the LabelBinarizer class from the standard sklearn library [5, 6]. As a result, we obtain text format data (list) that returns the compatibility matrix of given element.

```
from sklearn.preprocessing import LabelBinarizer
encoder= LabelBinarizer()
    encoder.fit(df['PlaceName'])
    data=encoder.transform(df['PlaceName'])
    df1 = pd.DataFrame(data)
```

Latitude	Longitude	0	1	2	3	4	5	6	7	8
49,81963	24,01396	0	0	1	0	0	0	0	0	0
49,81479	24,01541	0	0	0	0	0	1	0	0	0
54,93593	83,19654	0	1	0	0	0	0	0	0	0
54,93592	83,19669	0	0	0	0	1	0	0	0	0
54,97668	83,04516	0	0	0	0	0	0	0	0	1
54,93592	83,19669	0	0	0	0	0	0	0	1	0
54,93592	83,19669	1	0	0	0	0	0	0	0	0
54,85461	83,11102	0	0	0	0	0	0	1	0	0
54,85982	83,09263	0	0	1	0	0	0	0	0	0
54,85466	83,11106	0	0	0	1	0	0	0	0	0
54,85484	83,11109	0	0	1	0	0	0	0	0	0
54,86953	83,09487	0	0	0	0	0	1	0	0	0
54,85984	83,09783	0	1	0	0	0	0	0	0	0
54,93565	83,19666	0	0	0	0	1	0	0	0	0
54,9943	82,87084	0	0	0	0	0	0	0	0	1
54,93565	83,19666	0	0	0	0	0	0	0	1	0
54,93592	83,19645	1	0	0	0	0	0	0	0	0
54,86259	83,09817	0	0	0	0	0	0	1	0	0
54,93599	83,19643	0	0	1	0	0	0	0	0	0
54,94531	83,18996	1	0	0	0	0	0	0	0	0
54,85447	83,11037	0	0	0	0	0	0	1	0	0
54,85961	83,10668	0	0	1	0	0	0	0	0	0
54,93589	83,1965	0	0	0	1	0	0	0	0	0
54,85448	83,11044	0	0	1	0	0	0	0	0	0
54,85544	83,11205	0	1	0	0	0	0	0	0	0

Fig. 2. Results of binarization of meetcity application data

In Fig.2 columns from 0 to 8 are vectors in multidimensional space. A value of 1 in these columns belongs to a particular value of the previous Placename field for a concrete type.

Since the k-means algorithm is unstable to anomalies, the data must first be normalized. To do this, we should apply the MinMax Scaling method which moves all points of space to the limit [0,1].

$$X_{norm} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

To do this, we will use the Normalizer class from the sklearn library from sklearn.preprocessing import Normalizer

```
scaler= Normalizer()
scaler.fit(df1)
scaled_df=scaler.transform(df1)
df1= pd.DataFrame(scaled_df)
```

As a result of using of the Normalizer class, we obtain the output data of the preprocessing function (Fig.3)

	A	B	C	D	E	F	G	H	I	J	K	L
1	Login	0	1	2	3	4	5	6	7	8	9	10
2	Afanasy	1,978736	3,083825	0,006985	0,004463	0,003343	0,002229	0	0	0	0	0,020079
3	Anastasia	1,967644	2,980301	0	0,002866	0,02007	0	0	0	0	0,007164	0,005738
4	Anatoly	1,73155	2,622649	0,004302	0	0,001435	0,011471	0,005738	0	0	0,005738	0,002866
5	Andriy	2,430502	3,522298	0,005731	0	0,018649	0	0	0,005738	0,01621	0	0
6	Anton	2,343627	3,229045	0	0	0,014166	0,00678	0,020067	0	0,005731	0	0
7	Antonina	2,42582	3,149958	0,009241	0,020076	0,010547	0,006687	0	0	0	0	0
8	Boris	2,40508	3,325081	0	0	0,001255	0,012542	0,008666	0	0,006276	0,015056	0
9	Daryna	2,203472	3,338128	0	0	0,025104	0	0	0	0,015063	0	0
10	Denys	2,024071	3,057154	0	0,016716	0,013424	0,006687	0	0	0	0	0
11	Dmytry	2,203886	3,337855	0	0	0,01003	0,01339	0	0,016718	0	0	0
12	Evgen	1,836428	2,78164	0	0	0	0,013389	0,01003	0,01003	0	0	0
13	Ganna	2,13485	3,233655	0	0	0,012538	0	0	0,015063	0,00251	0,003764	0,005017
14	Gennady	1,652475	2,503681	0	0,030124	0	0	0	0	0	0	0
15	Georgy	2,203296	3,338244	0	0	0	0	0,040166	0	0	0	0
16	Glib	2,754128	4,1728	0	0	0	0	0	0	0	0	0,050208
17	Oleg	2,754546	4,172524	0,025075	0	0	0	0	0	0,025104	0	0
18	Olga	1,984568	3,00336	0,008042	0	0,020077	0	0	0	0	0,008026	0
19	Roman	1,947138	2,779741	0,008024	0	0	0,014052	0	0	0,006024	0,009492	0
20	Sofia	2,26375	2,225007	0,007522	0	0,007522	0	0	0,01356	0	0	0,013867
21	Valentin	2,276551	3,281191	0,008024	0	0,010034	0,009492	0	0	0,016067	0	0
22	Valery	1,744073	2,681388	0,01003	0	0,01003	0	0,002141	0	0	0,010043	0
23	Varvara	2,093683	3,170968	0	0	0,008031	0	0	0	0,01003	0	0,02006
24	Victor	2,619361	3,962214	0,012578	0	0	0,010032	0	0	0,012552	0,012552	0
25	Vitaliy	1,762993	2,67036	0,014051	0	0,012043	0	0	0	0,006021	0	0
26	Volodymir	2,313453	3,505162	0	0,010041	0,01003	0,004012	0	0	0,018068	0	0
27	max	2,13208	3,379753	0	0	0	0,008025	0,012537	0	0,020071	0	0

Fig. 3. Output data of the preprocessing function

In Fig. 3. we received data as a result of normalization and grouping by the value of login (Fig. 2). Columns 0-10 are measurements of multidimensional space, and

data is a reflection of vectors in them. All data should be normalized using the MinMax Scaling method [12-15].

In order for the system to be dynamic and allow users to change their behavior and, as a consequence, to move to other clusters, their number should be determined automatically. For this purpose it is necessary to cluster on n different quantities of clusters, and then to find the point of inflection of the function of dispersion, which is "elbow". This number will be the optimal number of clusters at the moment [4, 7].

An alternative to the reproduction of the “elbow” dispersion function is the use of gap statistics [6-8], which are generated on the basis of resampling and Monte Carlo simulation procedures.

Let $E*n\{log(W*k)\}$ mean an estimate of the average dispersion $W*k$ obtained by the bootstrap method when k clusters are generated by random sets of objects from an input sample of size n. Then the statistic $Gapn(k)=E\{n\{log(W*k)\}-log(Wk)$ determines the deviation of the observed dispersion Wk from its expected value, provided that the null hypothesis holds that the input data form only one cluster.

To do this, we cluster the data into different number of clusters, find the dispersion and find the point of inflection by the “elbow” method.

```
def clasternumber(df1):
# find number of clusters
   cluster_range = range(1, 10)
   cluster_errors = []

   for num_clusters in cluster_range:
      clusters = KMeans(num_clusters)
      clusters.fit(df1)
      cluster_errors.append(clusters.inertia_)
```

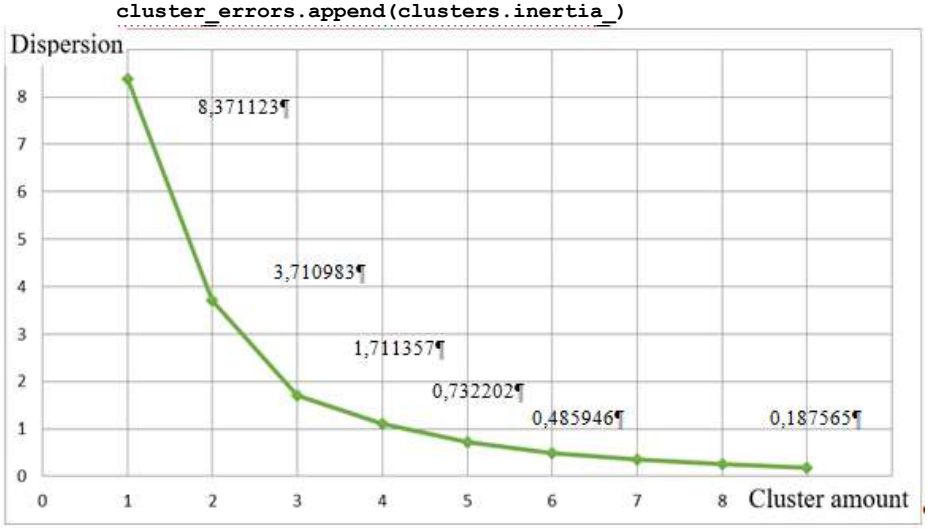

Fig. 4. Dispersion for n clusters

Find the inflection point of the function. It corresponds to the number of clusters 4. Therefore, the given dataset must be divided into 4 clusters for best results.

Clustering - grouping objects based on the similarity of their properties, so that each cluster consists of similar objects, and the objects of different clusters differ significantly [12-18]. Clustering helps us understand natural grouping or data structure. The purpose of clustering is to determine the internal grouping of multiple unlabeled data segments. With the help of clustering we can solve the following problems of data analysis: search and discovery of knowledge, grouping and recognition of objects: search for representatives of homogeneous groups (reducing the dimensionality of data), search for natural clusters and description of their unknown properties, search for useful and appropriate grouping, search for unusual objects. (detection of emissions) [2, 19].

Unlike hierarchical methods, when clusters are built up step by step, split clustering methods examine all segments at once. In doing so, they either attempt to identify clusters by iteratively moving points between subsets, or to identify clusters as areas that are densely filled with objects. The first kind of algorithms belong to partitioning methods with movement (partitioning relocation clustering). They, in turn, are divided into probabilistic, k-means and k-medoids methods, and concentrate on adjusting points to corresponding clusters, with a tendency to construct spherical segments. The second type separation algorithms belong to the group of density-based partitioning methods. They try to identify dense cohesive data components that are flexible in terms of their shape. This group is less sensitive to emissions and may find irregular clusters [6-10].

Separating moving clustering methods have several advantages: linear complexity of the algorithm; relative scalability and simplicity; good fit to data with compact, well-separated spherical clusters. The disadvantages include: a significant decrease in performance on high-dimensional data; the need to indicate the number of clusters in advance; sensitivity to initialization, noise and emissions; possibility to get to local optima's; inability to cope with clusters of different shapes and densities [11, 14, 16].

To divide users into clusters, you need to use the k-means algorithm, and then apply the resulting data to the Apriori algorithm to obtain user behavior patterns [1, 15].

The principle of the k-means algorithm is to find the following cluster centers and sets of elements of each cluster in the presence of some function *F(°)*, which expresses the quality of the current division of the set into k clusters, when the total quadratic deviation of the cluster elements from the centers of these clusters is smallest:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \qquad (2)$$

Where k – cluster amount, S_i – obtained clusters, $i = 1, 2, \dots, k$, μ - vectors` center of mass, $x_i \in S_i$.

```
{\displaystyle V=\sum _{i=1}^{k}\sum _{x_{j}\in S_{i}}(x_{j}-\mu _{i})^{2}}
```

At first step of the k-means algorithm, we select the cluster centers arbitrarily. Then, for each element of the set, we calculate the distance from the centers and attach each element to the cluster. For each of the obtained cluster, we calculate new

value of the center, trying to minimize the function *F (°)*. After that the procedure of redistributing the elements between the clusters is repeated.

Algorithm of Clustering using the k-means scheme:

- select k information points as cluster centers until the process of changing cluster centers is completed;
- compose each information point with a cluster whose distance to the center is minimal;
- ensure that each cluster contains at least one point. To do this, each empty cluster must be supplemented by an arbitrary point located "far" from the center of the cluster;
- replace the center of each cluster with the mean value of the cluster elements;

The main advantages of the k-means method are its simplicity and speed of execution. The k-means method is more convenient for clustering large numbers of observations than the method of hierarchical cluster analysis (in which the dendrograms become overloaded and lose clarity).

One of the disadvantages of the simple method is the violation of the connectivity of elements of one cluster, so different modifications of the method, as well as its fuzzy *k*-means methods, are developed, in which, at the first stage of the algorithm, the membership of one element of a set to several clusters is allowed (with varying degrees of affiliation).

Despite the obvious advantages of the method, it also has significant disadvantages:

- The result of the classification strongly depends on the random starting positions of the cluster centers
- The algorithm is sensitive to emissions, which can distort the mean value
- The researcher should determine the number of clusters in advance.

For clustering, we pass to the function numerical normalized data, which are the coordinates of user vectors in multidimensional space based on their activity:

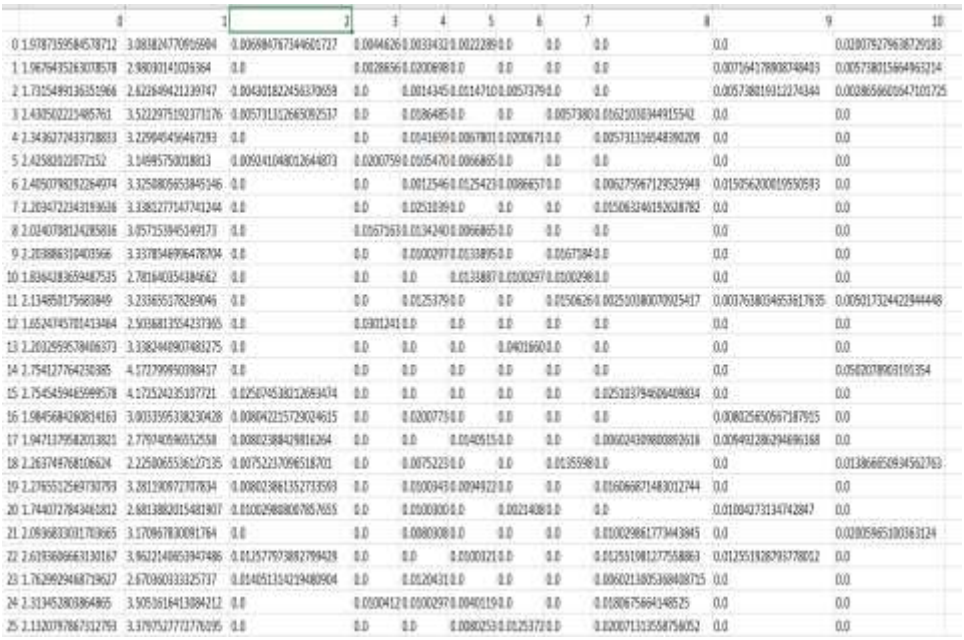

Fig. 5. Input data for clustering

1. K-means algorithm using python and sklearn library def kmeans(n,df1):

```
df1.index = np.arange(len(df1))
# clustering
n_clusters = 4
km = KMeans(n_clusters=n_clusters)
data= KMeans.fit_predict(km,df1)
data=pd.DataFrame(data)
centroids= km.cluster_centers_
centroids=pd.DataFrame(centroids)
```

2. At the output, each element is assigned to a specific cluster:

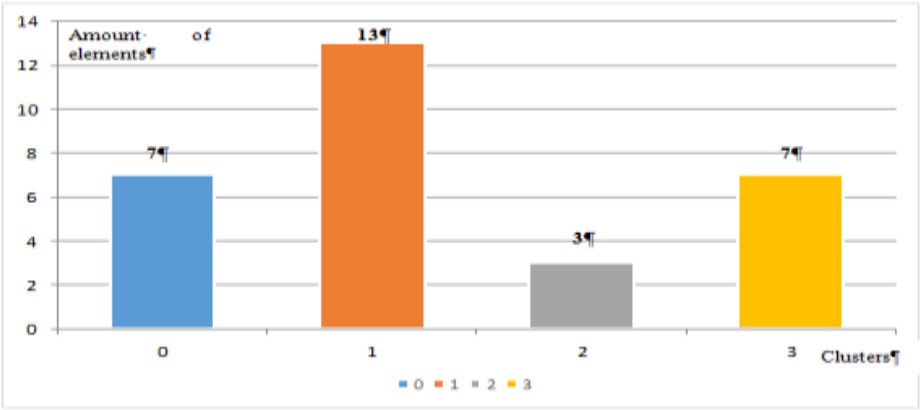

Fig. 6. K-means algorithm result displayed in diagram

As a result, we obtained data that is divided into 4 clusters. The size of each section corresponds to the number of values in the cluster. Next, we need to use this information to find the rules of conduct. Therefore, after clustering, we should start looking for associative rules.

Affinity analysis is one of the common methods of Data Mining. The purpose of this method is to investigate the relationship between events that occur together. Its purpose is to identify associations between different events, that means to find rules for quantifying the relationship between two or more events. These rules are called association rules.

The basic concepts in associative rule theory are subject set and transaction. A subject set is some non-empty set of elements that transactions can include:

$$I=\{i_1,i_2,\ldots,i_k,\ldots,i_n\}, \tag{3}$$

where i_k - elements included in the subject sets, k=1..n, n is the number of elements of the set I.

Transaction is a set that has some elements of set I that occur together. The transaction also has a unique TID (Transaction ID).

There is a certain set of transactions in the database:

$$T=\{t1,\ t2,\ldots,\ ti,\ldots,\ tm\}, \tag{4}$$

where t_i – some transaction, m – amount of transactions.

Between transaction elements we can set some regularities in the form of associative rules: $X=\left\{i_k \middle| i_k \in I\right\}$, called a condition and $Y=\left\{j_k \middle| j_k \in I\right\}$, called a consequence. We should also say that the same set should not be included in the antecedent and consequent at the same time: $i_k \neq j_k$.

The associative rule describes the relationship between sets of subjects which respond to consequence rule and are written down as $X \rightarrow Y$. The sets X and Y must not intersect: $X \cap Y=\varnothing$. The main indicators of the importance of an associative rule are support and confidence.

We should distinguish between support for recruitment and support for associative rule. In two cases, it is defined as the ratio of the number of transactions having the specified amount of items to the total number of transactions. The only difference is that the number of transactions that have the corresponding set is taken to calculate support for the set, and the number of transactions that have both a condition and a consequence at the same time to calculate the support of the associative rule.

Lets assume that we have some set X and associative rule $X \rightarrow Y$. Then support of set X is:

$$Supp(X)=\frac{\left|X(t)\right|}{\left|T\right|}, \tag{5}$$

Where $X(t)=\left\{t \in T \middle| X \in t\right\}$.

Support for the associative rule will be equal to:

$$Supp(X \to Y) = \frac{\left|X(t) \cap Y(t)\right|}{\left|T\right|}, \tag{6}$$

where $X(t) = \left\{t \in T \middle| X \in t\right\}$, $Y(t) = \left\{t \in T \middle| Y \in t\right\}$.

Because modern database sizes can reach large enough volumes (up to gigabytes and terabytes), finding associstive rules requires efficient algorithms that are scalable and allow you to find a solution to a given problem in an acceptable time.

Apriori algorithm was designed for relational databases and allows you to generate frequent datasets from transaction tables.

Apriori algorithm uses iterative approach. At first step it finds single-element frequent datasets denoted by the set *L1*. At the next step dataset *L1* is used to find frequent sets that has two elements, from which we form dataset *L2* which is used to find dataset *L3* and so on. To increase the productivity of frequent datasets generation the anti-monotony property is used. This is based on the following observation: if some dataset is not frequent: $\sup\{I\} < \min \quad \sup$, when if we add some specific object *i* we will get new dataset which also would not be frequent:

$$\sup\{I \cup \{i\}\} < \min \quad \sup. \tag{7}$$

Using the specified property, frequent *k*-element sets of *Lk* data can be obtained by combining frequent *(k-1)* element sets. Moreover, in order for some *k*-element set *Lk* to be included in frequent sets *Lk*, all of its *(k -1)* element subsets must also be frequent. If at least one of them is not a frequent set, *Lk* must be excluded from the set of frequent subject sets.

This observation contributes to the creation of a plurality of candidates of *k*-elemental *Ck*, sets, which will be a subset of *Lk*. This subset is obtained by removing from the *Ck* infrequent datasets, which is the result of checking the support values of each of the candidates *ck, (ck* $\in$ *Ck,)*. Based on the antimonotonic property, the set *Ck* is generated in two steps. In the first step, the candidate is generated by joining the members of the set of frequent sets C_{k-1}, where two members can be joined if they have *k-2* common elements, ie:

$$L_{k-1} \cup L_{k-1} = \left\{A \cup B \middle| A, B \subset L_{k-1} \middle| A \wedge B \middle| = k-2\right\} \tag{7}$$

The next step is to remove from the set of C_k members that include (*k*-1)- elemental data sets that are not frequent.

3 Numerical experiments

We used the standard mlxtend library to aggregate data by cluster, user login, and time.

	Class	Login	PlaceName	TimeStamp
2	0	Afanasy	Sport	2018-12-15 02:31:28
3		Afanasy	Culture_pl	2018-12-19 10:10:04
4		Afanasy	Sport	2018-12-15 00:19:41
5		Afanasy	Sport	2018-12-19 08:41:07
6		Afanasy	Sport	2018-12-17 11:51:46
7		Afanasy	Amusemer	2018-12-16 14:35:52
8		Afanasy	Driving	2018-12-14 12:37:15
9		Afanasy	Finance	2018-12-17 11:51:46
10		Afanasy	Amusemer	2018-12-16 14:55:18
11		Afanasy	Sport	2018-12-15 02:31:28
12		Afanasy	Culture_pl	2018-12-19 10:10:04
13		Afanasy	Sport	2018-12-15 00:19:41
14		Afanasy	Sport	2018-12-19 08:41:07
15		Afanasy	Sport	2018-12-17 11:51:46
16		Afanasy	Amusemer	2018-12-16 14:35:52
17		Afanasy	Driving	2018-12-14 12:37:15
18		Afanasy	Finance	2018-12-17 11:51:46
19		Afanasy	Amusemer	2018-12-16 14:55:18
20		Boris	Shopping	2018-12-16 16:15:23
21		Boris	Shopping	2018-12-14 13:39:52
22		Boris	Finance	2018-12-15 03:13:56
23		Boris	Shopping	2018-12-18 02:32:27
24		Boris	Recreatior	2018-12-18 02:52:28
25		Boris	Finance	2018-12-17 15:21:13
26		Boris	Driving	2018-12-18 02:32:27
27		Boris	Healthcare	2018-12-19 11:08:13

Fig. 7. Data preparation for the accurate prediction

After that we need to use apriori class for finding associative dependencies
def rules(data,df2):

```
df2['TimeStamp'] = pd.to_datetime(df2['TimeStamp'], unit='s')
df2=pd.merge(df2, data,on='Login')
df2.set_index(['Class','Login'], inplace=True)
df2.sort_index(inplace=True)
del df2['Longitude']
del df2['Latitude']
del df2['Ratting']
df2=df2.groupby('TimeStamp')['PlaceName'].apply(list)
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ary = te.fit(df2).transform(df2)
df = pd.DataFrame(te_ary, columns=te.columns_)

from mlxtend.frequent_patterns import apriori
rules=apriori(df, min_support=0.01, use_colnames=True)
```

4 Analysis of the results

Algorithm result is shown below:

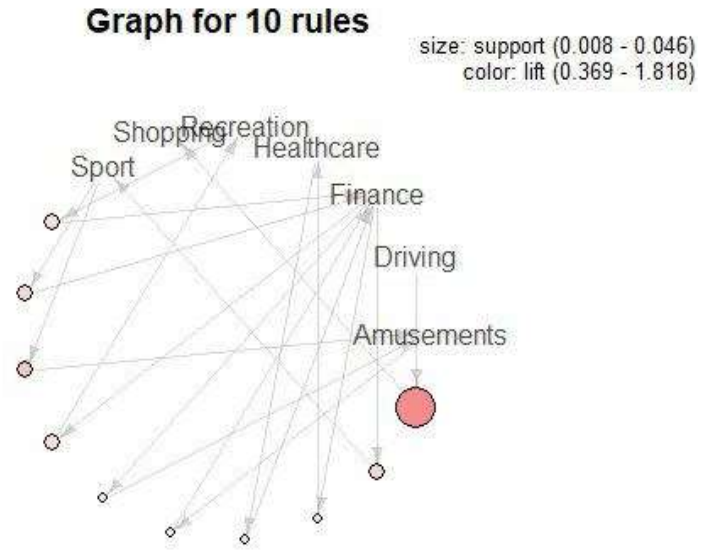

Fig. 8. Graph representation.

In Fig. 8 we can see the work of the associative rule algorithm, where the diameter of a circle means the support (frequency) of a certain rule, and the arrows to and from the circle, respectively, indicate the sequence of elements in the rule. As we can see, the Driving-Shopping rule (with the largest and brightest circle) is most likely. It has 0.046 support and 1.818 color saturation. the least likely is the Amusements-Finance rule with 0.008 support and a circle saturation of 0.369.

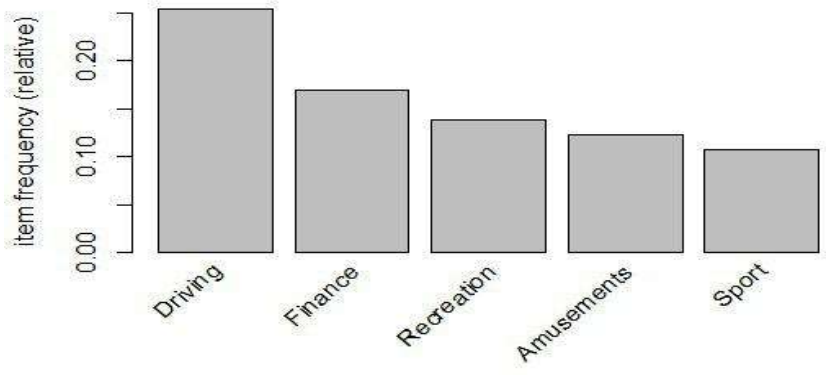

Fig. 9. Most popular places due to apriori algorithm

In Fig. 9 is a diagram showing the 5 most popular places. They are in descending order. Therefore, it can be assumed that, due to the use of meetcity and its geolocation capabilities to predict and recommend meetings, Driving is considered the most visited place.

5 Conclusion

In this article we explored apriori and *k*-means clustering algorithms to get user behavior analysis template. In the process, we looked at the problem of finding associative rules that were able to find and describe patterns in large datasets. We used scalable Apriori algorithms to find the best rules. For analysis, we used the standard mlxtend library to aggregate data by cluster, user login, and time.

While working, we were faced with the problem of inaccuracy and inconsistency of data with real conditions, and were forced to reduce the minimum support for associative rules.

References

1. Estivill-Castro, V., Lee, I.: Amoeba: Hierarchical clustering based on spatial proximity using Delaunay diagram. In: 9th Intern. Symp. on spatial data handling, pp. 26–41, Beijing, China (2000)
2. Kang, H.-Y., Lim, B.-J., Li K.-J.: P2P Spatial query processing by Delaunay triangulation , Lecture notes in computer science, vol. 3428, pp. 136–150, Springer, Heidelberg (2005)
3. Boehm, C., Kailing, K., Kriegel, H., Kroeger P.: Density connected clus-tering with local subspace preferences. In: Proc. of the 4th IEEE Intern. conf. on data mining, pp. 27–34, Los Alamitos: IEEE Computer Society (2004)
4. Wang, Y., Wu, X.: Heterogeneous spatial data mining based on grid, Lecture notes in computer science, vol. 4683, pp. 503–510,Springer/Heidelberg (2007)
5. Harel, D., Koren, Y.: Clustering spatial data using random walks. In: Proc. of the 7th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 281–286, San Francisco, California (2001)
6. Turton, I., Openshaw, S., Brunsdon, C. et al.: Testing spacetime and more complex hyperspace geographical analysis tools. In: Innovations in GIS 7, pp. 87–100, L.: Taylor & Francis (2000)
7. Tung, A.K. H., Hou, J., Han, J.: Spatial clustering in the presence of obstacles. In: The 17th Intern. conf. on data engineering (ICDE'01), pp. 359–367, Heidelberg (2001)
8. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic sub-space clustering of high dimensional data. In: Data mining knowledge discovery, vol. 11(1), pp. 5–33 (2005)
9. Guimei, L., Jinyan, L., Sim, K., Limsoon, W.: Distance based subspace clustering with flexible dimension partitioning. In: Proc. of the IEEE 23rd Intern. conf. on digital object identifier, vol. 15, pp. 1250-1254 (2007)
10. Aggarwal, C., Yu, P.: Finding generalized projected clusters in high dimensional spaces. In: ACM SIGMOD Intern. conf. on management of data, pp. 70-81 (2000)
11. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M. : A Monte Carlo algorithm for fast projective clustering. In: ACM SIGMOD Intern. conf. on management of data, pp. 418–427, Madison, Wisconsin, USA (2002)

12. Ankerst, M., Ester, M., Kriegel, H.-P.: Towards an effective cooperation of the user and the computer for classification. In: Proc. of the 6th ACM SIGKDD Intern. conf. on knowledge discovery and data mining, pp. 179-188, Boston, Massachusetts, USA (2000)
13. Peuquet, D.J.: Representations of space and time, N. Y., Guilford Press (2002)
14. Guo, D., Peuquet, D.J., Gahegan, M.: ICEAGE: Interactive clustering and exploration of large and high-dimensional geodata. Geoinfor-matica, vol. 3, N. 7, pp. 229-253 (2003)
15. Boyko, N.: A look trough methods of intellectual data analysis and their applying in informational systems. In: Scientific and Technical Conference “Computer Sciences and Information Technologies (CSIT), 2016 XIth International, pp. 183-185, IEEE (2016).
16. Shakhovska, N., Boyko, N., Zasoba, Y., Benova, E.: Big data processing technologies in distributed information systems. Procedia Computer Science, 10th International conference on emerging ubiquitous systems and pervasive networks (EUSPN-2019), 9th International conference on current and future trends of information and communication technologies in healthcare (ICTH-2019), Vol. 160, 2019, pp. 561–566, Lviv, Ukraine (2019)
17. Boyko, N., Shakhovska, Kh., Mochurad, L., Campos, J.: Information System of Catering Selection by Using Clustering Analysis, Proceedings of the 1st International Workshop on Digital Content & Smart Multimedia (DCSMart 2019), pp. 94-106, Lviv, Ukraine (2019)
18. Kunanets, N., Vasiuta, O., Boiko, N.: Advanced Technologies of Big Data Research in Distributed Information Systems, Proceedings of the 14th International conference "Computer sciences and Information technologies" (CSIT 2019), pp. 71-76, Lviv, Ukraine, (2019)
19. Boyko, N., Basystiuk, O.: Comparison Of Machine Learning Libraries Performance Used For Machine Translation Based On Recurrent Neural Networks, 2018 IEEE Ukraine Student, Young Professional and Women in Engineering Congress (UKRSYW), pp.78-82, Kyiv, Ukraine (2018)