Capstone Project

The Battle of the Neighborhoods (Week 2)

This report is done within the Courser course 'Applied Data Science Capstone'

Table of Content

Introduction. The Problem Description

In this project we are going to help families with kids to plan their day-off.

In big cities, such as Moscow, there are thousands of venues that could be **interesting for families with kids**. But unfortunately, parents do not know about them. In fact, there is a very limited list of popular venues mostly placed in the downtown, that are known well to everybody. They are nice but always overcrowded and it could take more than an hour to get there through traffic jams. And parents encounter more problems when they are going to plan several activities for the same day. For example, where they could eat with kids, or what they will do if their plans, weather or mud suddenly change.

We use the data science and data visualization tools to display on the map **the clustered venues affordable for families with kids**. That clusters will include **venues for education and entertainment** and will be placed in the specified neighborhood.

Data

Taking into account the problem definition, we will have to mine two types of data:

1. We must determine how to split the Moscow area into neighborhoods with determined geo coordinates;
2. We must mine information of venues placed in some specified neighborhood.

How to Split the Moscow Area into Neighborhoods

There are several ways to determine neighborhoods in Moscow. For example, it could be done **by the municipalities**. There are 12 administrative districts in Moscow that include 146 municipalities. The list of them that includes their boundaries can be downloaded at http://gis-lab.info/qa/moscow-atd.html in ESRI Shape, GeoJSON, CSV+VRT or KML formats. The second way is **by the post offices locations**. There are 13 post regions in Moscow that include 524 post offices. The geo locations of the post offices can be downloaded at http://hubofdata.ru/dataset/ruspost-msk in JSON format.

Using the post offices locations as neighborhoods centers is easier since we can get their geo coordinates and most of the venues have postcodes (except such as parks, playgrounds etc.). For those that have not, it can be determined by the closest venues. But the size of such neighborhoods seems to be too small to get a proper point of view.

So, we will **determine the neighborhoods as the Moscow municipalities.**

To download the data for Moscow municipalities we use the link http://gis-lab.info/data/mos-adm/mo-csv.zip

Since data are stored in zip file we will need to unzip it.

Top 5 records look like:

	WKT	NAME	OKATO	OKTMO	NAME_AO	OKATO_AO	ABBREV_AO	TYPE_MO
0	MULTIPOLYGON (((36.8031012 55.4408329,36.80319...	Киевский	45298555	45945000	Троицкий	45298000	Троицкий	Поселение
1	POLYGON ((37.4276499 55.7482092,37.4284863 55....	Филёвский Парк	45268595	45328000	Западный	45268000	ЗАО	Муниципальный округ
2	POLYGON ((36.8035692 55.4516224,36.8045117 55....	Новофёдоровское	45298567	45954000	Троицкий	45298000	Троицкий	Поселение
3	POLYGON ((36.9372397 55.2413907,36.9372604 55....	Роговское	45298575	45956000	Троицкий	45298000	Троицкий	Поселение
4	POLYGON ((37.4395575 55.6273129,37.4401803 55....	"Мосрентген"	45297568	45953000	Новомосковский	45297000	Новомосковский	Поселение

Some columns have information in Cyrillic. To make data more convenient for review within this project we developed the function to **transliterate the Cyrillic symbols into English ones** with similar or alike articulation.

The final data frame looks like:

	WKT	NAME	LAT	LNG	OKATO	OKTMO	NAME_AO	OKATO_AO	ABBREV_AO	TYPE_MO
0	MULTIPOLYGON (((36.8031012 55.4408329,36.80319...	Kievskiy	0.0	0.0	45298555	45945000	Troitskiy	45298000	Troitskiy	Poselenie
1	POLYGON ((37.4276499 55.7482092,37.4284863 55....	Filyovskiy Park	0.0	0.0	45268595	45328000	Zapadnyy	45268000	ZAO	Munitsipal'nyy okrug
2	POLYGON ((36.8035692 55.4516224,36.8045117 55....	Novofyodorovskoe	0.0	0.0	45298567	45954000	Troitskiy	45298000	Troitskiy	Poselenie
3	POLYGON ((36.9372397 55.2413907,36.9372604 55....	Rogovskoe	0.0	0.0	45298575	45956000	Troitskiy	45298000	Troitskiy	Poselenie
4	POLYGON ((37.4395575 55.6273129,37.4401803 55....	"Mosrentgen"	0.0	0.0	45297568	45953000	Novomoskovskiy	45297000	Novomoskovskiy	Poselenie

So, we have managed to:

- get the list of the Moscow municipalities as our neighborhoods;
- get the data about the neighborhoods boundaries as polygons descriptions;

  A polygon description is a list of points that are connected one by one with direct lines. In our case each polygon consists of about 500 and more point. Each point is represented by a couple that is latitude and longitude coordinates separated by space symbol. Since we downloaded .csv file polygon description is in string format;

- replace the Cyrillic symbols in the neighborhoods data with English ones similar or alike in articulation.

How to Mine Information of Venues Placed in Some Specified Neighborhood

We can do it with the Foursquare API **and the method called** 'explore'**.**
With that method we can download the data of venues we are interested in placed no farther that some specified distance form some specified geo coordinates.

To get venue data we will form the url request for the Foursquare API that looks like:

'https://api.foursquare.com/v2/venues/explore?&client_id=VC5VAI2CNSBEQ1BIWEREK0RDJX2VK4WVRYEXQTSLB4Q4XXFB&client_secret=HRGWOCZXY555UGMEUD4APX5LYXYTREVQNOLEQZJT1YFRW3U3&v=20180605&ll=55.7504461,37.6174943&radius=1000&limit=20'

The data venue looks like:

{'reasons': {'count': 0, 'items': [{'summary': 'This spot is popular', 'type': 'general', 'reasonName': 'globalInteractionReason'}]}, 'venue': {'id': '4da9654f43a1128196dbea8b', 'name': 'Cathedral Square (Соборная площадь)', 'location': {'address': 'Соборная пл.', 'lat': 55.75067703638466, 'lng': 37.61744217329331, 'labeledLatLngs': [{'label': 'display', 'lat': 55.75067703638466, 'lng': 37.61744217329331}], 'distance': 25, 'postalCode': '101000', 'cc': 'RU', 'neighborhood': 'Krasnaya Ploshchad', Moscow', 'city': 'Москва', 'state': 'Москва', 'country': 'Россия', 'formattedAddress': ['Соборная пл.', '101000, Москва', 'Россия']}, 'categories': [{'id': '4bf58dd8d48988d164941735', 'name': 'Plaza', 'pluralName': 'Plazas', 'shortName': 'Plaza', 'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/parks_outdoors/plaza_', 'suffix': '.png'}, 'primary': True}], 'photos': {'count': 0, 'groups': []}}, 'referralId': 'e-0-4da9654f43a1128196dbea8b-1'},

We can see that our request brings significant amount of data including for those venues we are not interested in, such as boutiques, for example. Of course, we can use this raw data for the following sorting and picking up the venues affordable for families with kids, but in this case we will have to deal with huge data massive and encounter significant computational resources needs. Besides, 'explore' method deals with a point coordinates as a center of investigating area, whereas our neighborhoods are determined by polygon boundaries.

In fact, that means that we have to solve two problems:

1. How to find the center for each of our neighborhoods if we have just their boundaries as polygons?
2. How to download from the Foursquare API the data just for venues we are interested in?

Methodology

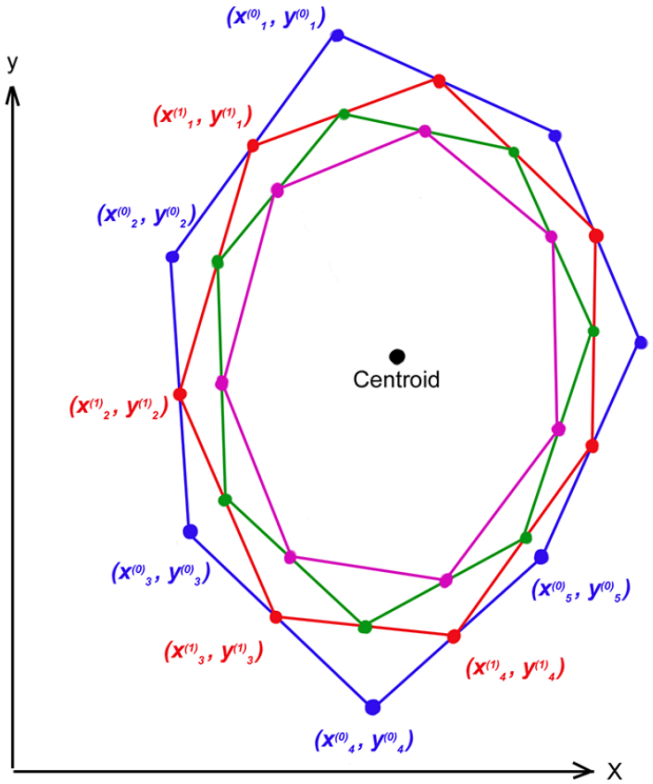
How To Find The Center For Each Of Our Neighborhoods.

To do so:

- We will transform polygon that is a string into a list of float pairs that represent longitude and latitude.

- The idea of centroid finding is the following:

  Polygon is a set of lines connecting points one by one. We will find the middle of each such line and redraw the polygon by connecting them. That's an iteration procedure. At every iteration polygon will be convoluting around some point that is centroid we are looking for. The indicator of the process called 'gradient' is an averages of longitude and latitude for all points. If after some iteration gradient stayed the same, we can tell that centroid was found. The Picture 1 below illustrates the algorithm.



Iteration	Polygon
0	$[(x_1,y_1), (x_2, y_2), \dots , (x_m,y_m)]$
1	$[(x_1,y_1), (x_2, y_2), \dots , (x_m,y_m)]$
2	$[(x_1,y_1), (x_2, y_2), \dots , (x_m,y_m)]$
3	$[(x_1,y_1), (x_2, y_2), \dots , (x_m,y_m)]$
…	…
n	$[(x^{(n)}_1,y^{(n)}_1), (x^{(n)}_2, y^{(n)}_2), \dots , (x^{(n)}_m,y^{(n)}_m)]$

$$(x^{(j)}_i, y^{(j)}_i) = \frac{(x^{(j-1)}_i, y^{(j-1)}_i) + (x^{(j-1)}_{i+1}, y^{(j-1)}_{i+1})}{2}, \; i \neq m$$

$$(x^{(j)}_m, y^{(j)}_m) = \frac{(x^{(j-1)}_m, y^{(j-1)}_m) + (x^{(j-1)}_1, y^{(j-1)}_1)}{2}$$

$$\text{gradient}^{(j)} = \frac{1}{m}\left(\sum_{i=1}^{m} x^{(j)}_i \, , \sum_{i=1}^{m} y^{(j)}_i\right)$$

$$\triangle_{grad}{}^{(j)} = \text{gradient}^{(j)} - \text{gradient}^{(j-1)}$$

$$(x_{centroid}, y_{centroid}) = \text{gradient}^{(j)}_{\triangle_{grad}{}^{(j)} = 0}$$

Picture 1

Note: This algorithm is rather resource intensive concerning that we have to find centroids for 146 neighborhoods with 500+ points polygon each.

Now we have a list of the neighborhoods with determined coordinates of center for each of them in the columns 'LAT' and 'LNG'. The first of the two problems is solved and the data looks like:

WKT	NAME	LAT	LNG	OKATO	OKTMO	NAME_AO	OKATO_AO	ABBREV_AO	TYPE_MO
MULTIPOLYGON (((36.8031012 55.4408329,36.80319...	Kievskiy	55.383952	36.909133	45298555	45945000	Troitskiy	45298000	Troitskiy	Poselenie
POLYGON ((37.4276499 55.7482092,37.4284863 55....	Filyovskiy Park	55.748470	37.476145	45268595	45328000	Zapadnyy	45268000	ZAO	Munitsipal'nyy okrug
POLYGON ((36.8035692 55.4516224,36.8045117 55....	Novofyodorovskoe	55.420282	36.974195	45298567	45954000	Troitskiy	45298000	Troitskiy	Poselenie
POLYGON ((36.9372397 55.2413907,36.9372604 55....	Rogovskoe	55.228378	37.037273	45298575	45956000	Troitskiy	45298000	Troitskiy	Poselenie
POLYGON ((37.4395575 55.6273129,37.4401803 55....	"Mosrentgen"	55.621927	37.465978	45297568	45953000	Novomoskovskiy	45297000	Novomoskovskiy	Poselenie

How to download from the Foursquare API the data just for venues we are interested in?

Now let us clarify what venue categories we are interested in. To do so we can look through the content by the link https://developer.foursquare.com/docs/resources/categories that includes categories id.

After looking it through we can determine that we are looking for venues from the categories that could be affordable for families with kids:

Venue type	Catergory Id
Parks	4bf58dd8d48988d163941735
Entertaiment centers	4bf58dd8d48988d1e1931735
Amusement parks	4bf58dd8d48988d182941735
Playgrounds	4bf58dd8d48988d1e7941735
Museums	4bf58dd8d48988d181941735
Cinema	4bf58dd8d48988d17f941735
Kids cafe	4bf58dd8d48988d1d0941735

Further we'll pick up venues we are interested in for a neighborhood called 'Akademichesky' as an example. In fact, it can be any other municipality or current user's location.

To do it we have to add to the request url additional parameter describing id of the categories we are looking for and shown in the table above.

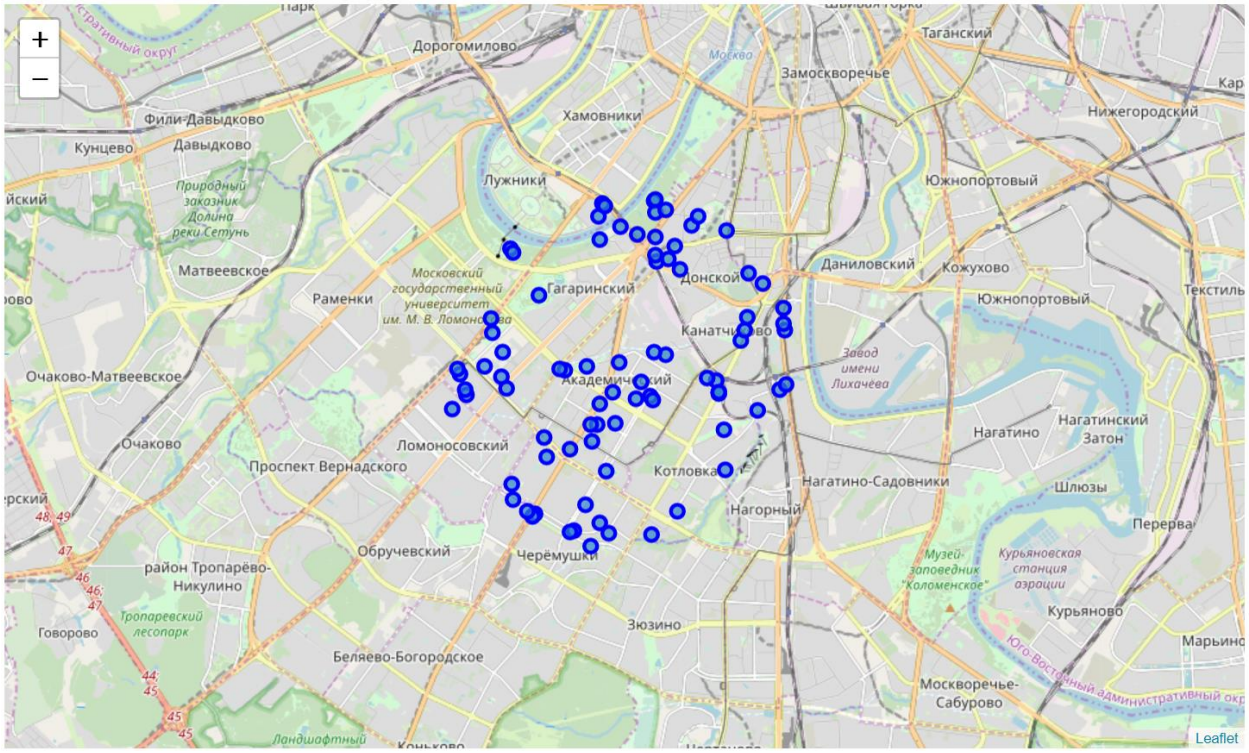
Let’s look at the first 10 venues found for the neighborhood.

	id	name	categories	lat	lng
0	59a2a8c3c0cacb5d0ff796eb	Академический парк	Park	55.691777	37.568886
1	4e7b0341b0fbf3d6be9917d3	Парк «Новые Черёмушки»	Park	55.693547	37.589786
2	4fb90be5e4b0c86152260256	Двор с фонтаном	Playground	55.692286	37.577203
3	4fbbaa56e4b0c852de6b3a36	Сквер «200 лет А. С. Пушкину»	Park	55.687814	37.575538
4	54ae6d16498e18b8bcce5838	Эндорфин квест	Arcade	55.680578	37.570033
5	4f8c484ee4b0e67b87bb83e1	Аллея	Park	55.678210	37.558306
6	4e527a86d4c075ade75b3f06	Парк Дворца Пионеров	Park	55.702327	37.556238
7	4ed851fcbe7be28335359dac	Молодёжная улица	Park	55.693817	37.546515
8	56d097b7cd10ac3eb8a5d793	Батутный парк «Небо»	Athletics & Sports	55.687681	37.603730
9	57c85b83498eec7de643b658	Kuzina	Dessert Shop	55.688509	37.547529

The total number of the venues found is 84 that is rather big. So, it will be more convenient to analyze them after procedure of clustering.

Clustering and Map Compilation Section¶

Let's look how the found venues of the neighborhood are distributed on the map.



We see that our venues are distributed into five clusters. To split the venues into clusters we'll use K-Means method to group together the venue situated close to each other.
This approach will help customer to pick up the most appropriate cluster concerning distance and a set of the venues in it.

The array below is the result of splitting 84 venues into 5 clusters and indicates the cluster labels for each of the venue.

```
array([3, 3, 3, 3, 1, 4, 4, 0, 4, 3, 0, 3, 2, 3, 3, 3, 3, 2, 3, 3, 4,
       3, 4, 2, 1, 4, 2, 3, 4, 3, 2, 2, 1, 2, 2, 3, 1, 1, 2, 0, 4, 0, 0,
       0, 2, 0, 4, 0, 1, 0, 0, 0, 2, 2, 1, 1, 1, 3, 4, 0, 0, 0, 1, 0, 0,
       4, 2, 2, 0, 4, 1, 2, 0, 3, 4, 2, 2, 1, 1, 0, 1], dtype=int32)
```

Results

The first ten rows of the dataset is looking now so:

	Cluster Labels	id	name	categories	lat	lng
0	3	59a2a8c3c0cacb5d0ff796eb	Академический парк	Park	55.691777	37.568886
1	3	4e7b0341b0fbf3d6be9917d3	Парк «Новые Черёмушки»	Park	55.693547	37.589786
2	3	4fb90be5e4b0c86152260256	Двор с фонтаном	Playground	55.692286	37.577203
3	3	4fbbaa56e4b0c852de6b3a36	Сквер «200 лет А. С. Пушкину»	Park	55.687814	37.575538
4	3	54ae6d16498e18b8bcce5838	Эндорфин квест	Arcade	55.680578	37.570033
5	1	4f8c484ee4b0e67b87bb83e1	Аллея	Park	55.678210	37.558306
6	4	4e527a86d4c075ade75b3f06	Парк Дворца Пионеров	Park	55.702327	37.556238
7	4	4ed851fcbe7be28335359dac	Молодёжная улица	Park	55.693817	37.546515
8	0	56d097b7cd10ac3eb8a5d793	Батутный парк «Небо»	Athletics & Sports	55.687681	37.603730
9	4	57c85b83498eec7de643b658	Kuzina	Dessert Shop	55.688509	37.547529

We can see in the list the parks, a trampoline venue, a playground, a quest room and a dessert shop.

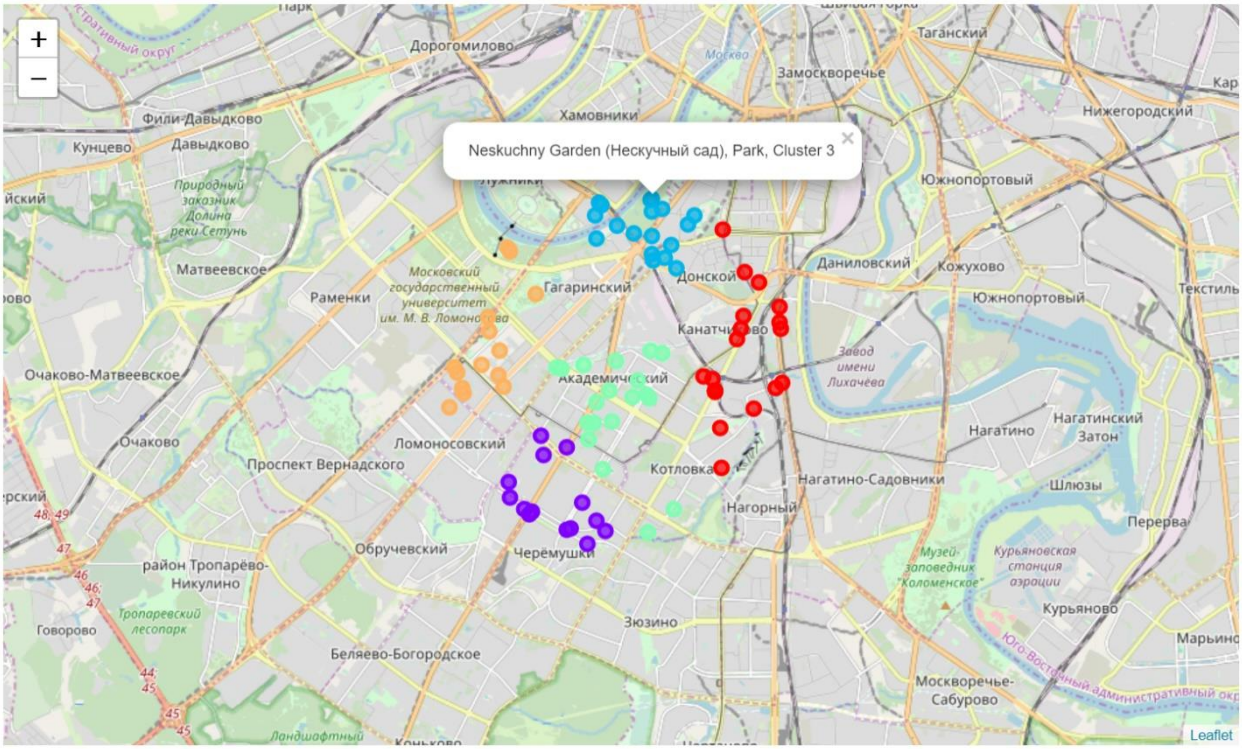
> *There is no sense to transliterate the venue names since they remained sensless for English speaking person. For example, 'Батутный парк "Небо" is the trampline park called 'Sky'. Its transliteration will be 'Batutny park "Nebo" and is useless. Some venues in Foursquare (mostly placed in the center of Moscow) has both Russian and English names slashed.*

Let's look at the list of the venue categories to ensure that all the venues are suitable for families with kids.

```
Index(['Arcade', 'Art Museum', 'Athletics & Sports', 'Cupcake Shop',
       'Dessert Shop', 'Frozen Yogurt Shop', 'Ice Cream Shop', 'Movie Theater',
       'Multiplex', 'Museum', 'Park', 'Pie Shop', 'Playground',
       'Science Museum', 'Shopping Mall', 'Theme Park Ride / Attraction'],
      dtype='object', name='categories')
```

It looks like all the venues are affordable for families with kids. The 'Shopping Mall' got in the list since there are cinema multiplexes there together with kids' cafe and some attractions.

Let's visualize the results of clustering on the map. We'll also add labels for the venue marks to make the map more convenient.



Discussion

Looking at the map we can see that the most interesting are the clusters 2 and 3 since they are compact. That means that it will be a little more convenient to move from one venue to other by walk.
Let's make a list of the venues for the clusters 2 and 3 separately.

Cluster #2

	name	categories	lat	lng
13	Детская площадка	Playground	55.711351	37.582281
18	Клаустрофобия	Arcade	55.712843	37.596536
24	Prostokvest	Arcade	55.714083	37.571709
27	Детская Площадка	Playground	55.716544	37.586733
31	Донской сквер	Park	55.714075	37.598430
32	CityQuest	Arcade	55.716126	37.572923
34	Neskuchny Garden (Нескучный сад)	Park	55.716678	37.586922
35	Сквер у метро Ленинский проспект	Park	55.707312	37.587318
39	Шоколадница	Dessert Shop	55.707768	37.590500
45	Итальянская кондитерская	Dessert Shop	55.711032	37.586948
53	Детская площадка у Андреевских прудов	Playground	55.710598	37.572182
54	Народный парк "Бульвар Архитекторов"	Park	55.709798	37.591990
67	Детская Площадка 2	Playground	55.714793	37.586821
68	Детская площадка "Стройка"	Playground	55.715172	37.589886
72	Baskin Robbins (Баскин Роббинс)	Ice Cream Shop	55.706239	37.593475
76	Музей Ар Деко	Art Museum	55.715618	37.573741
77	Экоцентр «Воробьёвы горы»	Science Museum	55.712644	37.577606
82	Линдфорс	Pie Shop	55.708414	37.587119

The list of the venue categories in Cluster #2 is:

```
Index(['Arcade', 'Art Museum', 'Dessert Shop', 'Ice Cream Shop', 'Park',
       'Pie Shop', 'Playground', 'Science Museum'],
      dtype='object', name='categories')
```

Quantity of venues categories in Cluster #2 is 8.

Cluster #3

	name	categories	lat	lng
0	Академический парк	Park	55.691777	37.568886
1	Парк «Новые Черёмушки»	Park	55.693547	37.589786
2	Двор с фонтаном	Playground	55.692286	37.577203
3	Сквер «200 лет А. С. Пушкину»	Park	55.687814	37.575538
4	Эндорфин квест	Arcade	55.680578	37.570033
10	Сквер на винокурова	Park	55.689456	37.583141
12	детская площадка	Playground	55.686949	37.581865
14	Салют	Movie Theater	55.682933	37.571550
15	Квест Клуб	Arcade	55.687250	37.585670
16	Квест Белый Лебедь	Arcade	55.686636	37.586239
17	Тютчевский сквер	Park	55.686196	37.572171
19	Детская Площадка Ул. Шверника	Playground	55.693883	37.586755
20	Парк «Сосенки»	Park	55.670098	37.592706
22	Детская площадка	Playground	55.683160	37.576410
28	Лермонтовский сквер	Park	55.683000	37.569900
30	Палеопарк	Park	55.691128	37.562875
36	Новочеремушкинская Аллея	Park	55.676064	37.573920
58	Сквер	Park	55.666480	37.585908
74	Государственный Дарвиновский музей / State Dar...	Science Museum	55.691300	37.561420

The list of the venue categories in Cluster #2 is:

```
Index(['Arcade', 'Movie Theater', 'Park', 'Playground', 'Science Museum'], dtype='obje
ct', name='categories')
```

Quantity of venues categories in Cluster #3 is 5.

So, Cluster #2 is more affordable than Cluster #3 since it consists of the venues from more categories (8 vs 5) what means more flexibility and choice for families.

Conclusion

We have managed within the project to solve the following problems:

1. We found the way to determine neighborhoods for Moscow area and download corresponding geo data;
2. We transliterated the names of municipalities by replacing Cyrillic symbols with English ones;
3. We developed the algorithm and code for finding geo coordinates of polygon centroids on the base of boundaries polygon by iteration procedure of polygon convolution;
4. We determined the target categories of venues that are affordable for families with kids;
5. We got data of the venues we are interested in through Foursquare API;
6. We clustered the venues according their locations;
7. We displayed the map with marks indicating venues and clusters with pop-up labels;
8. We analyze the results and found the most promising and convenient cluster of venues for families with kids.