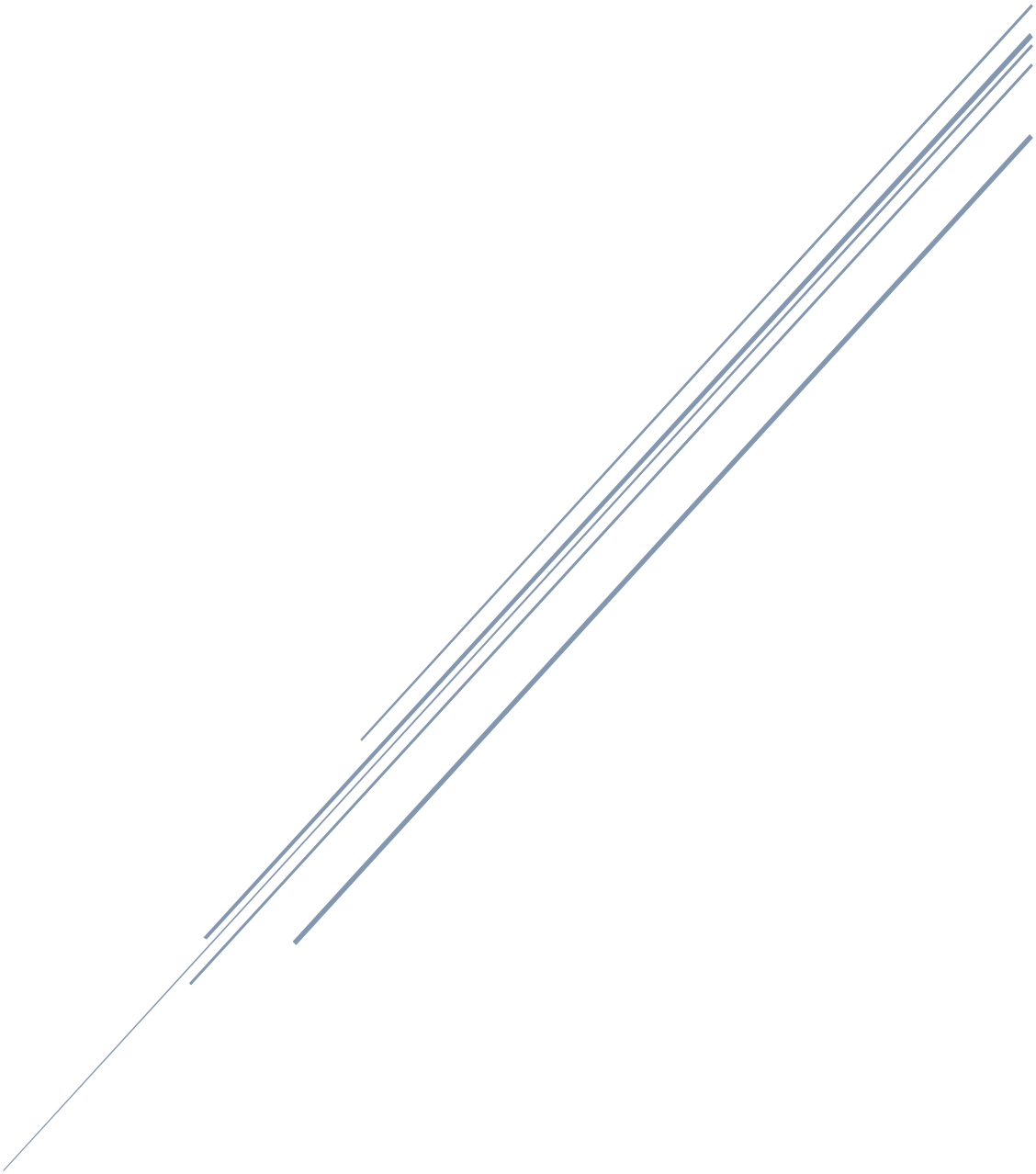
BATTLE OF THE NEIGHBORHOODS

A check-in-based recommender system



by Gergely Karácsonyi

Capstone of IBM Data Science Professional course

Table of Contents

Introduction....2

Background....2

Data Description....2

Municipal area data (districts and neighborhoods)....2

Nominatim....2

Foursquare venue data....3

Personal check-in data....3

Real estate prices....3

Data usage....4

Methodology....4

Results....8

Discussion....8

Conclusion....8

References used....9

Introduction

Background

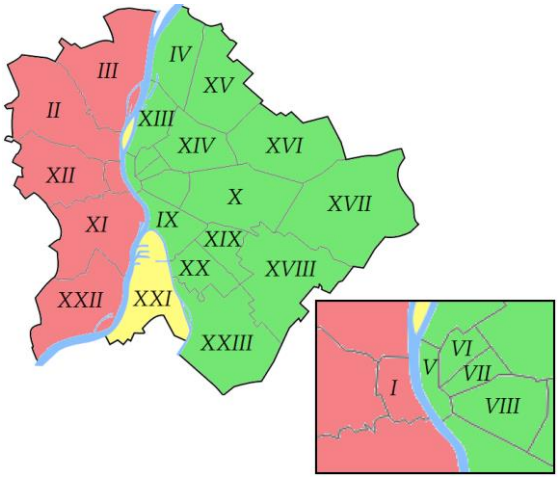
Moving to a big city is not simple. Without any local knowledge, it can be very challenging to pick the right area to buy or rent our new home. In this study I will set up a recommender system which uses Foursquare API and various location based data as input to help us select a proper place to live.

Data Description

For the study I selected Budapest, the capital of Hungary. It's the most populous city in the country with a population of approx. 1.7M. I chose Budapest because it's a popular place to live, with a thousands of venues and sights. I utilized the following data for the study:

Municipal area data (districts and neighborhoods)

Budapest has 23 districts, pictured on the right. Each district can be associated with one or more neighbourhoods so we will dig into neighborhood level later if possible, because district-level resolution is quite coarse. I've extracted the data (both the district and the neighborhood names) from Wikipedia.[1] and after that I've done a little cleanup and left only the names in the dataframe.



	District
0	I.
1	II.
2	III.
3	IV.
4	V.

Nominatim

I used Openstreetmap's Nominatim engine [2] to assign geocoordinates to each district and neighborhood.

	District	Latitude	Longitude
0	I.	47.499163	19.035143
1	II.	47.538887	18.982636
2	III.	47.568691	19.027668
3	IV.	47.577779	19.093164
4	V.	47.499945	19.050549

Foursquare venue data

I used data from Foursquare API to get the most common venue categories for each neighborhood. [3]

	District	District Latitude	District Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	I.	47.499163	19.035143	Dísz tér	47.499100	19.036163	Plaza
1	I.	47.499163	19.035143	Halászbástya | Fisherman's Bastion (Halászbástya)	47.502029	19.035058	Historic Site
2	I.	47.499163	19.035143	Stand25 Bisztró	47.497673	19.032679	Bistro
3	I.	47.499163	19.035143	Várnegyed	47.501195	19.032261	Scenic Lookout
4	I.	47.499163	19.035143	Honda Dream	47.498561	19.031825	Motorcycle Shop

Personal check-in data

First I wanted to use some custom rating data, but Foursquare API provides the check-ins of a given user – unfortunately, at this very moment only self check-ins are available. This dataframe looks like this:

[39]:

	Checkins
0	Café
1	Supermarket
2	Café
3	Café
4	Diner

These are venue category names which we will use for matching.

Real estate prices

I've found house-pricing data for each district on a hungarian website [4] but I've had to merge them into one dataframe with the districts. The numbers mean average HUF (hungarian forints) / square meters (at this very moment 100000 HUF = 328.28 USD = 290.73 EUR approx.)

Here is the dataframe head:

[40]:

	District	AvgPrice
0	I.	1023321
1	II.	892150
2	III.	754493
3	IV.	591152
4	V.	1222010

Data usage

We'll create clusters of the districts based on their most common venue categories. The user can get information about these clusters and by using the recommender system she can get recommendations matching her persona. This persona will be created by classification based on her venue check-ins. After she got a district recommendation, she can dig deeper if possible, to neighborhood granularity (It's not always possible, given there are districts which have only one neighborhood - practically the whole district is the neighborhood itself). Then we could repeat the process to get an even more specific recommendation. In the end, we'll visualize the results on a map, along with the real estate prices, just to make the choice more comfortable.

Methodology

After scraping the data some cleaning is necessary to get rid of unwanted columns such as district population, venue data noise, etc. We're recommended to keep the following features:

- district numbers
- real estate prices
- venue categories (by district)
- geographical coordinates (both districts and venues)
- check-in categories

After cleanup we can start to process the venue data. We have to group the data by venue category to get the exact number of each venue per category.

[47]:

Venue Category	District	District Latitude	District Longitude	Venue	Venue Latitude	Venue Longitude
Afghan Restaurant	1	1	1	1	1	1
Airport	2	2	2	2	2	2
Airport Food Court	1	1	1	1	1	1
Airport Service	1	1	1	1	1	1
Airport Terminal	1	1	1	1	1	1
...	...	...	...	...	...	...
Wine Bar	12	12	12	12	12	12
Wine Shop	7	7	7	7	7	7
Yoga Studio	6	6	6	6	6	6
Zoo	1	1	1	1	1	1
Zoo Exhibit	2	2	2	2	2	2

With this dataframe we are able to do One Hot Encoding, converting the individual numbers into boolean values per category.

[48]:

	District	Afghan Restaurant	Airport	Airport Food Court	Airport Service	Airport Terminal	American Restaurant	Amphitheater	Aquarium	Arcade	...
0	I.	0	0	0	0	0	0	0	0	0	...
1	I.	0	0	0	0	0	0	0	0	0	...
2	I.	0	0	0	0	0	0	0	0	0	...
3	I.	0	0	0	0	0	0	0	0	0	...
4	I.	0	0	0	0	0	0	0	0	0	...

5 rows × 262 columns

After that, the next step is to calculate the mean of the venues by district.

[49]:

	District	Afghan Restaurant	Airport	Airport Food Court	Airport Service	Airport Terminal	American Restaurant	Amphitheater	Aquarium	Arcade	...
0	I.	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.00	...
1	II.	0.0	0.027778	0.0	0.0	0.0	0.027778	0.0	0.0	0.00	...
2	III.	0.0	0.000000	0.0	0.0	0.0	0.016667	0.0	0.0	0.00	...
3	IV.	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.00	...
4	IX.	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.01	...

5 rows × 262 columns

In order to create proper clusterization, we have to create a dataframe of the most popular venues of the districts:

[50]:

	District	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
0	I.	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout
1	II.	Scenic Lookout	Grocery Store	Park	Pizza Place	Tram Station
2	III.	Bus Stop	Supermarket	Electronics Store	Grocery Store	Flower Shop
3	IV.	Bus Stop	Dessert Shop	Athletics & Sports	Grocery Store	Park
4	IX.	Park	Music Venue	Bakery	Fast Food Restaurant	Coffee Shop

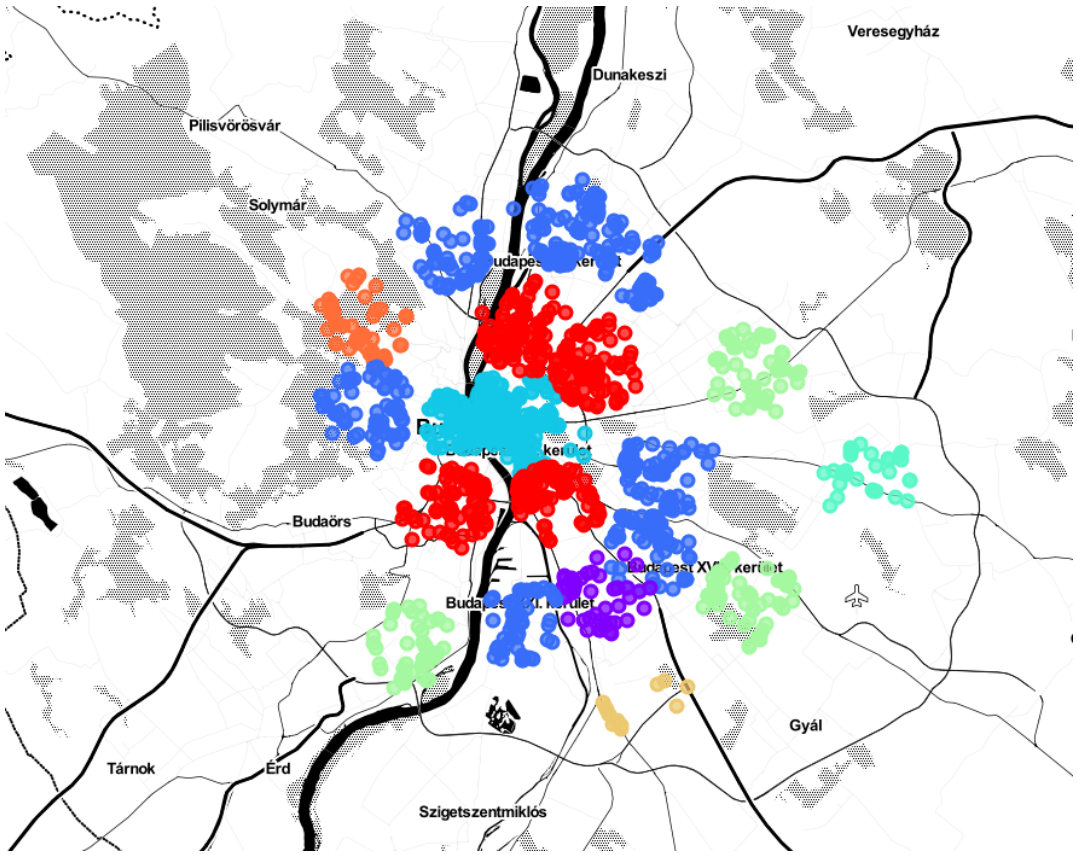
For the better result we have to standardize our data a little bit with StandardScaler.

We will use K-Means clustering to determine the venue clusters. But first it's recommended to select a proper K value to work with. How can we do that? A lot of ways, e.g. the Elbow-method or Silhouette-score. For this study I've chosen *Gap Statistic* method [5]. It returns us the optimal K value. With this K we are able to get this dataframe of venues with clusters assigned:

[56]:

	District	District Latitude	District Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue	Cluster Labels
0	I.	47.499163	19.035143	Dísz tér	47.499100	19.036163	Plaza	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout	Bistro	Historic Site	Hungarian Restaurant	Italian Restaurant	Dessert Shop	3
1	I.	47.499163	19.035143	Halászbástya | Fisherman's Bastion (Halászbástya)	47.502029	19.035058	Historic Site	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout	Bistro	Historic Site	Hungarian Restaurant	Italian Restaurant	Dessert Shop	3
2	I.	47.499163	19.035143	Stand25 Bisztró	47.497673	19.032679	Bistro	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout	Bistro	Historic Site	Hungarian Restaurant	Italian Restaurant	Dessert Shop	3
3	I.	47.499163	19.035143	Várnegyed	47.501195	19.032261	Scenic Lookout	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout	Bistro	Historic Site	Hungarian Restaurant	Italian Restaurant	Dessert Shop	3
4	I.	47.499163	19.035143	Honda Dream	47.498561	19.031825	Motorcycle Shop	Bakery	Coffee Shop	Hotel	Gym / Fitness Center	Scenic Lookout	Bistro	Historic Site	Hungarian Restaurant	Italian Restaurant	Dessert Shop	3

This looks quite busy for now but with Folium visualization we can check the clusters on a map. (Monochrome palette is for the sake of visibility).



So, what can we do with these clusters? Well, simply cross-check them with our existing venue check-in dataframe! For the sake of simpicity, we're checking only the top 3 categories for now.

[61]:

	District	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	Cluster Labels
1199	XIV.	Bakery	Dessert Shop	Gym / Fitness Center	0
1134	XIII.	Coffee Shop	Gym / Fitness Center	Park	0
1133	XIII.	Coffee Shop	Gym / Fitness Center	Park	0
1132	XIII.	Coffee Shop	Gym / Fitness Center	Park	0
1131	XIII.	Coffee Shop	Gym / Fitness Center	Park	0

After the cross-check we get the cluster number which contains the most venues matching to our preferences. With this cluster number we can select the proper districts to offer:

[66]:

	District	Latitude	Longitude
2	III.	47.568691	19.027668
3	IV.	47.577779	19.093164
9	X.	47.482235	19.156494
11	XII.	47.504800	18.982815
14	XV.	47.562714	19.140218
18	XIX.	47.449333	19.144120
20	XXI.	47.424317	19.069214

But we don't stop here, no. Let's combine this dataframe with the real estate prices for a better visualization.

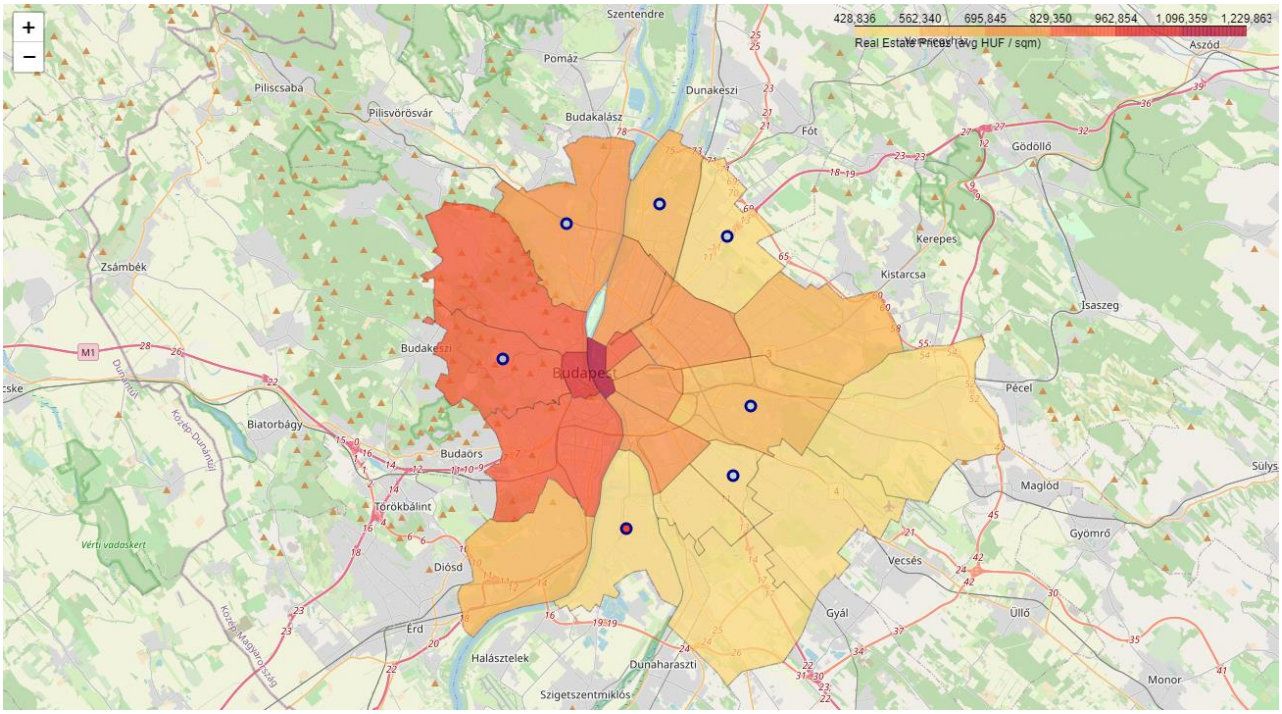
[69]:

	District	Latitude	Longitude	AvgPrice
6	XXI.	47.424317	19.069214	493899
5	XIX.	47.449333	19.144120	518294
4	XV.	47.562714	19.140218	534810
2	X.	47.482235	19.156494	577233
1	IV.	47.577779	19.093164	591152
0	III.	47.568691	19.027668	754493
3	XII.	47.504800	18.982815	937865

And this leads us to…

Results

So here we are at the doorstep of our final conclusion. All we have to do is to put everything on a map. Let's see:



What can we see here? The system recommended a few districts to live in. These are marked with the little circles on the map. The recommendation is based on our Foursquare check-ins: we collected the categories of the venues we've checked in before and crosschecked it against the venue clusters which were created by clusterization of the most common venue categories per district. So the dots mean districts having most venues matching our previous check-ins. The area colorization shows the prices according to the scale in the top right. The darker a color is, the more expensive the district is. So we highlighted the marker of the cheapest district with red to give a little bit more visual aid.

Discussion

This system has two drawbacks, a minor and a major one. The minor is, that it's tailored specifically to Budapest – but with some modifications, it can be used with any major city as well.

The major: it only works if the user has some Foursquare track record from before. If he's not, the system can be modified, but the results will be less personalized.

There is a lot of room for other improvements: we can consider more factors when selecting a district: crime data, average income, etc. And also, right now all check-ins are treated equally – perhaps it's a good idea to weigh them.

Conclusion

Although it's quite a basic tool which responds to a niche demand, it can be useful for Foursquare users planning to move to a different city.

References used

[1] https://en.wikipedia.org/wiki/List_of_districts_in_Budapest

[2] https://nominatim.openstreetmap.org/

[3] https://developer.foursquare.com/docs/places-api/

[4] https://www.ingatlannet.hu/statisztika/Budapest

[5] https://anaconda.org/milesgranger/gap-statistic/notebook