Adatelemzés beadandó

Budapesti kerületek klaszterezése és a kiadó lakások árainak elemzése többváltozós lineáris regresszióval és Gradient Boosting algoritmussal

Az elemzés célja, háttere

Az elemzés fő célja, hogy segítséget nyújtson azoknak az embereknek, akik új lakásba terveznek költözni Budapesten. A bemutatott eredmények segítenek dönteni, hogy melyik kerületet válasszák lakhelyüknek. A projekt második felében pedig az albérletek árait befolyásoló néhány tényezőt vizsgáltam, illetve lehetőséget nyújt a felhasználónak, hogy predikciót, becslést készítsen az árra különböző tényezők meghatározásával.

Budapest Magyarország fővárosa, az ország kereskedelmi, politikai, kulturális és közlekedési központja. Lakosainak száma nagyjából 1,7 millió fő. Közigazgatási szempontból 23 kerületből áll, ezek közül 6 található Budán, 16 Pesten, 1 pedig a Csepel szigeten.

Elemzésem fő célja, hogy segítséget nyújtson albérletet kereső ügyfeleknek, az olyan kérdésekben, mint

- Az ár/m2 szempontjából melyik kerületben lenne érdemesebb keresgélni?
- Milyen típusú kerületben szándékozik lakást keresni az ügyfél?
- Hasonló árkategóriájú kerületek közül esetleg melyiket érdemesebb választani?

Jelenleg úgy tűnik, hogy a koronavírus járvány az albérletpiacot is érinti. Az árak csökkenésnek indultak, viszont úgy gondolom, hogy az arányok állandóak maradnak a kerületek szempontjából, így elemzésem helytálló marad a jelenlegi helyzetben is.

Adat

Adatforrások

Többféle adathalmazt használtam fel elemzésem elkészítéséhez, ezek közül vannak amelyek a lényegi információkat hordozzák, illetve vannak amelyek csak technikai szempontból voltak fontosak számomra.

1. Kaggle-ről letöltött „**flats.csv**”
    - Közel 2 éves adat, amely a Jófogás oldalon hirdetett budapesti kiadó lakások adatait tartalmazza
2. Foursquare-ről lekért budapesti helyszínek listája: API-val - bp_venues(dataframe), általam lementett csv-t használva: „**venues20200423.csv**”
    - A Foursquare egy mobilos alkalmazás, amely adott pozíció és hely alapján ajánlatot tud adni közeli helyszínekre, létesítményekre. Ennek az alkalmazásnak a számomra fontos adatait kértem le API hívásokkal.
3. „**keruletek.csv**”
    - Ezt a file-t én állítottam elő Google Maps-et használva. A budapesti kerületek neveit, illetve központjainak koordinátáit tartalmazza.

4. „**kerulet_polygon_copy.geojson**”
    - Geojson formátumú file, amely adott területek polygonjait tartalmazza, hogy azok térképen vizualizálhatók legyenek.
    - Ennek létrehozásához mellékelem a kódot („CreatingGeoJson.ipynb”), illetve az alapjául szolgáló „kerulet_polygon.txt”-t, amit github-ról töltöttem le

Adattisztítás

Adattisztítás klaszterezéshez

A klaszterezésnél az adattisztítás során az volt a fontos számomra, hogy reális értékű és méretű lakásokat hagyjak benne az adathalmazban, természetesen az extrém kiugró értékeket kezelve.

A flats.csv beolvasása után, a kiadó lakások adatait tartalmazó dataframeből a számomra jelenleg értéket hordozó változók megtartása, illetve egy új változó, a „Price/m2” képzése:

```
df = df_full[['location', 'location_str', 'price', 'size']]
df.columns = ['Coordinates', 'Location', 'Price', 'Size']
df['Price/m2'] = (df['Price']/df['Size']).round(2)
```

Ezután a „Coordinates” -ben szereplő tuple objektumokat felszabdaltam szélességi és hosszúsági fokokra, majd hozzáfűztem az adattáblához, létrehozva a „Latitude” és „Longitude” mezőket, ezután eldobtam a feleslegessé vált „Coordinates” oszlopot. Ebben a pillanatban a dataframe:

	Location	Price	Size	Price/m2	Latitude	Longitude
0	Budapest, XIV. kerület, Istvánmező, Thököly...	148000.0	60.0	2466.67	47.5101	19.0937
1	Budapest, XV. kerület, Rákospalota, Dessewf...	170000.0	100.0	1700.00	47.5499	19.1263
2	Budapest, VII. kerület, VII. Kerület	190000.0	106.0	1792.45	47.5012	19.0712
3	Budapest, VII. kerület, VII. Kerület	190000.0	106.0	1792.45	47.5012	19.0712
4	Budapest, V. kerület	372132.0	106.0	3510.68	47.4951	19.054

Az egyik legfontosabb lépés a későbbiekben kulcsmezőként szolgáló „District” mező bevezetése volt. Ezt úgy állítottam elő, hogy egy függvény segítségével, ha a „Location” változóban, amely utcaneveket is tartalmazhatott a kerületek neve mellett, szerepelt például a „Budapest, I. kerület” string, akkor az általam képzett „District” változó értéke az adott kerület lett. Fontos volt, hogy mivel a Foursquare API a „Budapest XX. kerulete” formátumú stringeket várja el a lekérdezéshez, már itt is azokkal dolgozzak, hogy később ezek megfeleltethetőek legyenek egymással.

Az alábbi dataframe állt elő:

	Price	Size	Price/m2	Latitude	Longitude	District
0	148000.0	60.0	2466.67	47.5101	19.0937	Budapest XIV. kerulete
1	170000.0	100.0	1700.00	47.5499	19.1263	Budapest XV. kerulete
2	190000.0	106.0	1792.45	47.5012	19.0712	Budapest VII. kerulete
3	190000.0	106.0	1792.45	47.5012	19.0712	Budapest VII. kerulete
4	372132.0	106.0	3510.68	47.4951	19.054	Budapest V. kerulete

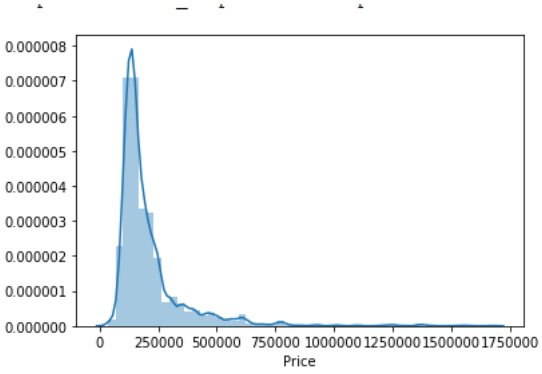
Következett a hiányzó adat, illetve a kiugró értékek kezelése. A hiányzó adatokat tartalmazó sorokat eldobtam, mivel kis arányban voltak jelen az adatban.

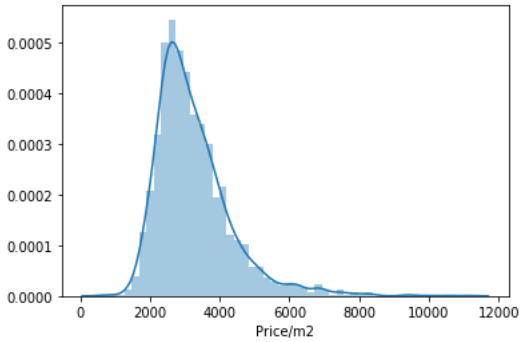
A kiugró értékek kezelésénél elsőnek a „Price/m2” mező értékeit kezdtem el elemezni, mivel ez jól reprezentálja a „Price” és a „Size” változót is. Kiírattam a 20 legnagyobb és 20 legkisebb értéket, majd a felső és alsó 0,5% határát. Ezek alapján úgy látszott, hogy nagyjából a 10 000 Ft/m2-nél nagyobb, illetve az 1000 Ft/m2-nél kisebb értéket tartalmazó sorokat kell eldobnom. Fontos szempont volt, hogy mielőtt ezt megteszem, ránézzek konkrétan azokra a sorokra, amiket ki akarok vonni a további elemzésből. Itt az látszott, hogy melyek a lakás mérete, a kerület és az ár alapján a valóban irreális sorok, amelyeket el fogok dobni.

Ezek alapján döntöttem úgy, hogy a határ felülről 12 000 Ft/m2, alulról pedig 400 Ft/m2 lesz.

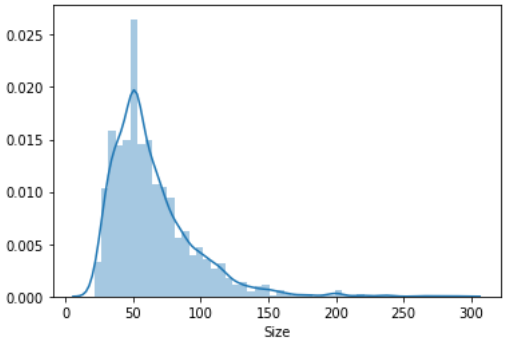
Ezután megvizsgáltam a „Price” változót, azaz a lakások árainak alakulását, hogy esetleg kell-e még ez alapján tovább tisztítanom az adatot. Itt nem volt szükség további adatok eldobására.

Az Ár, illetve az Ár/m2 a következőképpen alakult:





A lakások alapterületének szempontjából, az előforduló legnagyobb és legkisebb értékek figyelembevételével kivontam a további elemzésből az olyan alapterülettel rendelkező lakásokat, amelyek vagy irreálisan kicsi, vagy túl nagy alapterülettel rendelkeztek. Így a „Size ” változónál a 20 m2 és 300 m2 közötti értékkel rendelkező lakásokat hagytam meg.



	Price	Size	Price/m2
count	2.769000e+03	2769.000000	2769.000000
mean	2.086312e+05	63.946912	3238.120813
std	1.515488e+05	30.937864	1074.178976
min	3.500000e+04	21.000000	647.060000
25%	1.300000e+05	44.000000	2500.000000
50%	1.600000e+05	56.000000	3000.000000
75%	2.300000e+05	76.000000	3714.290000
max	1.570000e+06	291.000000	11111.110000

Az így előállt „df” nevű adattáblát csoportosítottam groupby függvény-nyel a „District” változóra, ez alapján képezve a kerületenkénti átlagokat.

Ezzel megkaptuk a „mean_per_district” táblát, amely a számunkra fontos, kerületenkénti „Price/m2” mezőt tartalmazza.

A mean_per_district tábla:

	District	Price	Size	Price/m2
0	Budapest I. kerulete	245088.227642	72.170732	3395.196098
1	Budapest II. kerulete	278012.063415	80.243902	3400.325073
2	Budapest III. kerulete	213905.120000	64.266667	3094.314667
3	Budapest IV. kerulete	126759.259259	52.462963	2469.308704
4	Budapest IX. kerulete	189058.045455	63.187500	3097.740455
5	Budapest V. kerulete	310687.336391	76.376147	4062.948563
6	Budapest VI. kerulete	250112.108000	68.960000	3625.957320
7	Budapest VII. kerulete	186481.327660	61.489362	3201.294000

Klaszterezés alapjául használt Foursquare-ről lekért adatok megtisztítása:

A bp_venues alaptáblából kiindulva, amely a kerületekben található létesítményeket tartalmazza, elkészítettem a klaszterezéshez használt táblát. Ennek lépései a következők voltak:

A különböző létesítmény kategóriákra dummy változókat képeztem, azaz 1-es ha található az adott kategóriából létesítmény a kerületben, és 0 az értéke, ha nem. Ehhez a dataframehez még hozzáadtam a Kerületek neveit 'District' mezőként, amelyet egyből a tábla elejére mozgattam.

Ezután ezt a táblát csoportosítottam a kerületekre, ezzel megkapva az egyes kategóriák gyakoriság értékeit. Létrejött a „bp_grouped" tábla. Mivel így itt 0-1 közötti értékek jöttek létre, további normalizálásra nem volt szükség. A modell illesztése előtt ebből ebből a dataframeből hozom létre a „District" mező elhagyásával a „bp_group_clustering"-et.

A klaszterezéshez használt, bp_group_clustering tábla egy részlete:

Airport Terminal	American Restaurant	Aquarium	Arcade	Art Gallery	Art Museum	Arts & Crafts Store	Arts & Entertainment	Asian Restaurant	Athletics & Sports	I
0.000000	0.000000	0.000000	0.00	0.000000	0.01	0.000000	0.00	0.000000	0.000000	
0.000000	0.010000	0.000000	0.00	0.000000	0.01	0.000000	0.00	0.000000	0.010000	
0.000000	0.010000	0.000000	0.00	0.020000	0.00	0.010000	0.01	0.010000	0.000000	
0.000000	0.000000	0.000000	0.00	0.010000	0.00	0.000000	0.00	0.000000	0.010000	
0.000000	0.000000	0.000000	0.00	0.000000	0.01	0.000000	0.00	0.010000	0.000000	
0.000000	0.000000	0.000000	0.01	0.000000	0.01	0.000000	0.00	0.000000	0.000000	
0.000000	0.000000	0.000000	0.01	0.000000	0.00	0.000000	0.00	0.000000	0.000000	

A következő tábla, ami később fontos lesz, a „district_venues_sorted", ez tartalmazza kerületenként a top10 létesítmény kategóriát. Majd a vizualizációnál, klaszterek vizsgálatánál lesz szerepe. Ezt a gyakoriság értékek kerületenkénti csökkenő rendezésével állítottam elő.

Adattisztítás regresszióhoz

Első körben a teljes flats.csv fileból indultam el, majd bevontam a számomra érdekesebb változókat:

```
# Az érdekesebb változók bevonása
dfreg = df_full[['condition', 'elevator', 'garden', 'price', 'rooms', 'size', 'type']]
```

Hiányzó adatok vizsgálata során az látszott, hogy a garden változónak körülbelül fele hiányzó adat, így azt kivettem az elemzésből.

Mivel a regressziós modellben nem használható nominális változó, így kódolnom kellett ezeket, erre a következő kódot használtam:

```
# szótárak a nominális változók kódolásához
conditions = {'felújítandó': 1,'jó állapotú': 2, 'felújított' : 3, 'újszerű' : 4, 'új építésű' : 5}
elevators = {'nincs' : 1, 'van' : 2}
types = {'panel' : 1, 'tégla' : 2,'egyéb' : 3}

# alkalmazzuk a dataframe-re
dfreg.condition = [conditions[item] for item in dfreg.condition]
dfreg.elevator = [elevators[item] for item in dfreg.elevator]
dfreg.type = [types[item] for item in dfreg.type]

dfreg.head()
```

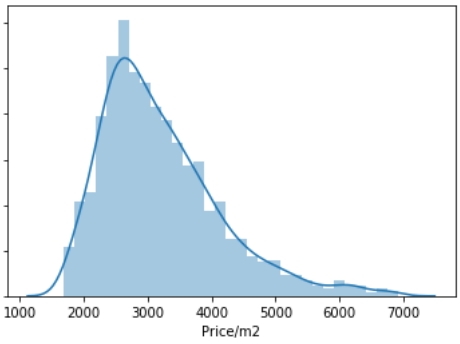
A dfreg tábla:

	condition	elevator	price	rooms	size	type
0	5	2	148000.0	2.0	60.0	2
1	3	1	170000.0	2.0	100.0	2
2	2	2	190000.0	3.0	106.0	2
3	2	2	190000.0	3.0	106.0	2
4	2	2	372132.0	3.0	106.0	2

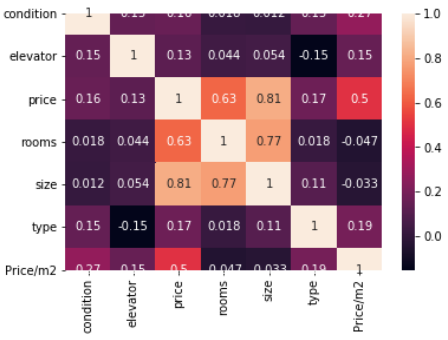
Az így létrejött integer típusú változókat konvertáltam float-ra. Ezután következett a kiugró értékek kezelése, melynek során azokat a lakásokat, ahol az ár a felső, illetve az alsó 1%-ba esik, kidobtam. Később előállítottam a Price/m2 változót itt is, és ugyanígy jártam el vele is. A „size” változó elemzésénél egy nagyon magas értéket találtam, amit szintén eltávolítottam az elemzésből.

A kapott tábla statisztikái, illetve az Ár/m2 eloszlása:

	condition	elevator	price	rooms	size	type	Price/m2
count	2612.000000	2612.000000	2612.000000	2612.000000	2612.000000	2612.000000	2612.000000
mean	2.981623	1.564319	200897.739663	2.049005	62.985835	1.922665	3204.163446
std	0.870628	0.495941	123145.185923	0.919537	30.076823	0.353555	933.324381
min	1.000000	1.000000	75000.000000	1.000000	14.000000	1.000000	1686.750000
25%	2.000000	1.000000	130000.000000	1.000000	43.000000	2.000000	2539.680000
50%	3.000000	2.000000	150000.000000	2.000000	55.000000	2.000000	3000.000000
75%	3.000000	2.000000	229250.000000	3.000000	75.000000	2.000000	3703.700000
max	5.000000	2.000000	930000.000000	5.000000	389.000000	3.000000	6923.080000



Boxplot diagrammon ábrázoltam az árat a különböző változók függvényében, illetve az összes változóra páronként létrehoztam a regressziós pontdiagramot, hogy lássam, hogyan alakulnak.



Utolsó ellenőrzésként korrelációs mátrixot is bekértem. Feltártam általa az esetleges multikollinearitást. Itt látszott, hogy a szobák száma és az alapterület szorosan függenek egymástól, ezért csak a „size” változót hagytam bent később a modell tanításánál.

Használt módszerek

Klaszterezés: egy dimenziócsökkentő eljárás, amelynek célja, hogy adatpontokat homogén csoportokba soroljunk. Az egyes klasztereken belüli adatok valamilyen dimenzió szerint hasonlítanak egymáshoz, ugyanakkor e dimenzió mentén különböznek a többi klaszter elemeitől. A különböző klaszterező módszerek alapja az, hogy az elemek különbözőségének megállapítására egy függvényt, egy távolság metrikát definiál, ezzel értelmezhetővé válik az elemek „távolsága”.

Én a K-közép (k-means) klaszterelemzést használtam, amely a nem hierarchikus módszerek közé tartozik. A lényege ennek a módszernek, hogy ha adott n különböző megfigyelésünk, ahol minden megfigyelés egy d dimenziós valós vektor, akkor ezeket hogyan rakhatjuk k db csoportba úgy, hogy az elemeknek a megfelelő csoportközéptől (centroid) vett négyzetes hibaösszegei minimálisak legyenek. Csak vektortérben megadott elemek csoportosítására használható, nominális skálán mért változóknál ez az algoritmus nem alkalmazható. Az algoritmus lépései a következőek (MacQueen, 1967):

•Kiválasztja a klaszterek számát (k).

•Véletlenszerűen létrehoz k számú klasztert, és meghatározza minden klaszter közepét, vagy azonnal létrehoz k véletlenszerű klaszter középpontot.

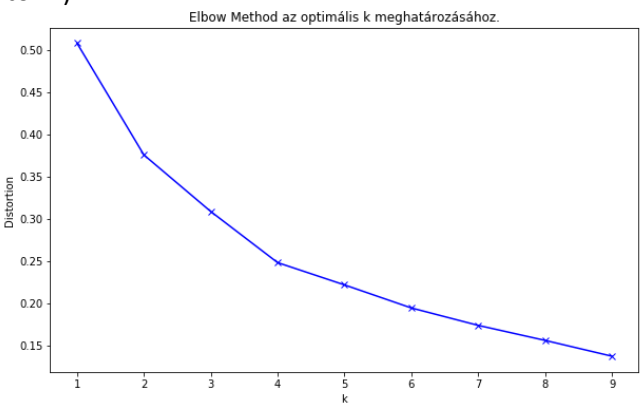
•Minden egyes pontot abba a klaszterbe sorol, amelynek középpontjához a legközelebb helyezkedik el.

•Kiszámolja az új klaszter középpontokat.

•Addig ismétli az előző két lépést (iterál), amíg valamilyen konvergencia kritérium nem teljesül (általában az, hogy a besorolás nem változik).

A klaszterek számának megválasztásához használtam az Elbow-methodot, amely egy grafikont jelent, amin megjelenítjük a magyarázott varianciát a klaszterek számának függvényében, és ahol kirakzolódik a könyék alak, azt választjuk.

Jelen esetben 4-nek választottam a klaszterek számát.



Megvalósítás:

k = 4-gyel a következő kód illeszti a modellt a bp_grouped_clustering adathalmazra, amely alapján a budapesti kerületek 4 klaszterbe fogja besorolni. Ezután csak összefűztem a kívánt adatokat, amelyeket meg akarok jeleníteni a klaszterek mellett, majd előállt az eredménytábla.

```
kclusters = 4

kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(bp_grouped_clustering)

kmeans.labels_[0:23]
```

```
array([0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 1,
       2])
```

Többváltozós lineáris regresszió:

Lineáris regresszióelemzésnek nevezik azt a statisztikai eljárást, amellyel megtalálhatjuk a pontdiagram pontjaira bizonyos értelemben legjobban illeszkedő egyenest.

Alapegyenlete: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_i X_i + \varepsilon$

Y a függő változó, Xi a független változók, β_0 a metszet, β_i a regressziós együtthatók ε a hibatag.

Választ próbál adni arra, hogy a független változók egységnyi változása, a függő változó milyen mérvű megváltozását vonhatja maga után. A változók mögött meghúzódó rejtett tendenciák feltárása révén magyarázó modell/ek kialakítását teszi lehetővé.

A modell illesztése a legkisebb átlagos négyzetes távolság kiszámításán alapul. A legkisebb négyzetek módszerének lényege, hogy minimalizálja a tényleges és a becsült paraméterrel illesztett modellek négyzetes eltérését, azaz az eltérések négyzetösszegét a lehető legkisebbre redukálja, ezzel minimalizálja a becslésből eredő torzításokat.

Mean squared error (átlagos négyzetes eltérés) képlete:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Megvalósítás: Definiáltam a független változók mátrixát, illetve a célváltozót.

```
X = dfreg[['condition', 'elevator',
                'size', 'type']]
y = dfreg['price']
```

Ezután felszabdaltam az állományokat training, illetve test adatok halmazára, beállítottam, hogy az adatok mekkora részét használja a train-eléshez.

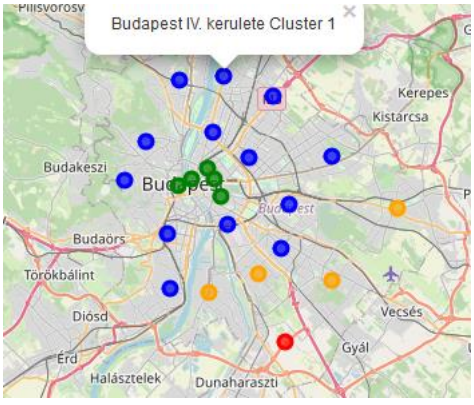
```
# dataset felszabdalása
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

A modell definiálása, normalizálással, majd illesztjük a training adatokra.

```
lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=True)
```

Eredmények, következtetések:



Klaszterezés: A klaszterezés eredményeit térképen vizualizáltam, amelyen jól látszik, hogy melyik kerület, melyik klaszterbe tartozik, melyik kerületekkel hasonló, illetve különböző a bemeneti adatok alapján.

Első körben az látszik ebből, hogy az adott kerületben található létesítmények alapján, azt mondhatjuk, hogy egy klaszterbe kerültek a belvárosi kerületek, egy másik klaszterbe a kicsit távolabb lévő, de mégis forgalmas kerületek, a maradék kettő klaszterben pedig a külvárosi kerületek találhatóak, még inkább elszeparálva, egymagában az utolsó klaszterben a XXIII. kerület. Megnéztem az egyes klaszterekben a leggyakoribb létesítményeket azért, hogy jobban megértsem a szétválogatás logikáját.

1. Klaszter

```
cluster1 = bp_merged.loc[bp_merged['Cluster Labels'] == 0, bp_merged.columns[[0] + list(range(5, bp_merged.shape[1]))]]
cluster1
```

	District	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
0	Budapest I. kerulete	Bakery	Hotel	Ice Cream Shop	Coffee Shop	Hungarian Restaurant	Café	Bistro	Gym / Fitness Center	Italian Restaurant	Playground
4	Budapest V. kerulete	Coffee Shop	Bakery	Hotel	Park	Ice Cream Shop	Pizza Place	Theater	Historic Site	Dessert Shop	Breakfast Spot
5	Budapest VI. kerulete	Coffee Shop	Hotel	Ice Cream Shop	Bar	Pizza Place	Hungarian Restaurant	Dessert Shop	Beer Bar	Restaurant	Plaza
6	Budapest VII. kerulete	Coffee Shop	Hotel	Ice Cream Shop	Restaurant	Bar	Bakery	Pizza Place	Theater	Burger Joint	Plaza
7	Budapest VIII. kerulete	Coffee Shop	Beer Bar	Park	Theater	Burger Joint	Bakery	Hotel	Bar	Soup Place	Café

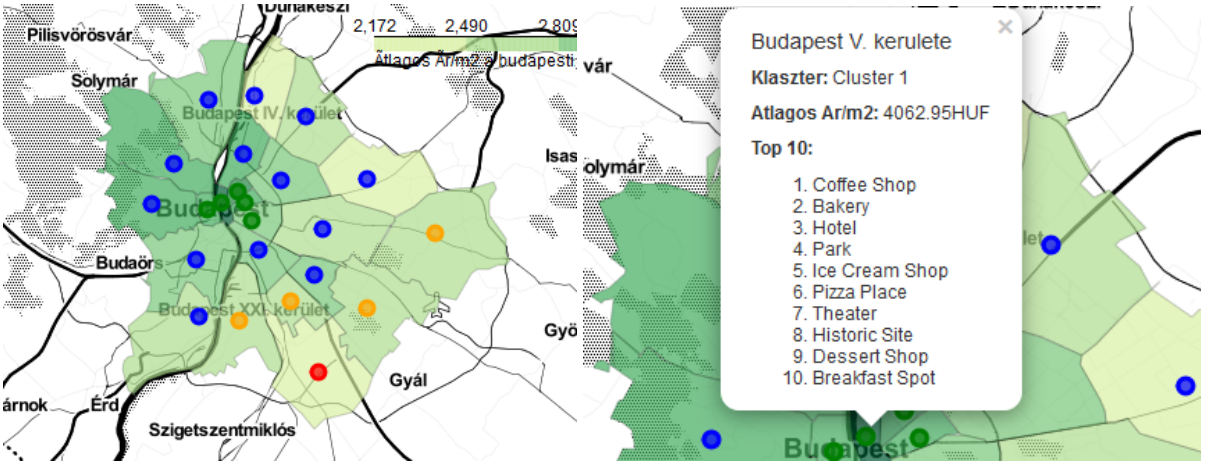
Például az első klaszterbe láthatóan a belvárosi, nagyon forgalmas, főleg turisták által kedvel és látogatott kerületek vannak. Az itt található létesítmények általában hotelek, kávézók, szórakozóhelyek, fagyizók.

A második klaszterbe tartozó 13 kerület hasonlósága, hogy egy zónával távolabb helyezkednek el a centrumtól. Itt már több park, bevásárlóközpont, konditerem, gyorsétterem található, de még ugyancsak jelen vannak nagy számban a kávézók, szórakozóhelyek.

A harmadik klaszterbe tartozó külvárosi kerületek hasonlósága, hogy a rengeteg buszmegállón kívül, általában boltokat, zöldségeseket, pékségeket találunk itt, illetve parkokat is nagy számban.

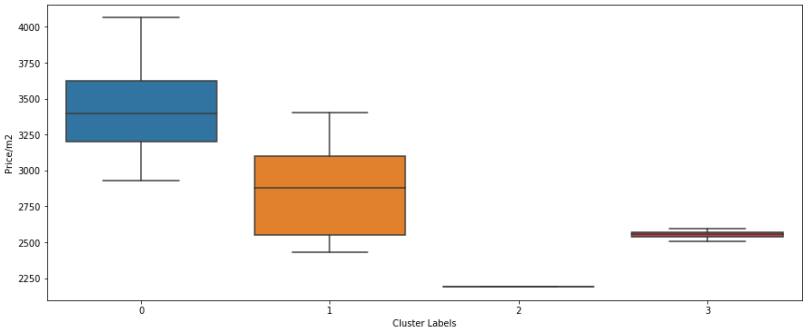
Az utolsó klaszterbe egyetlen kerület, a XXIII. kerület található. Mivel jelentősen különbözik a többi kerülettől és viszonylag messze található Budapest központjától, sok a buszmegálló, a vonatállomás, illetve a 4. leggyakoribb létesítmény a benzinkút. Ezen kívül éttermek, büfék itt is megtalálhatóak.

Miután megismertem a klaszterek tulajdonságait, megjelenítettem egy térképen a lakások Ár/m2 mutatóit, illetve a klasztereket úgy, hogy kiírja a kerület leggyakoribb létesítményeit, illetve a pontos Ár/m2-t is, ha az adott kerületre kattintunk.



Elemzésemmel arra szerettem volna rávilágítani, hogy ha már albérletet keresünk, és tudjuk azt is, hogy milyen típusú kerületben szeretnénk lakni, akkor jobb rálátást kapjunk az adott stílusú kerületek áraira.

Ha megnézzük a mellékelt boxplot diagramot, akkor láthatjuk, hogy melyik klaszterben mennyi az Ár/m2 alakulása. Így például ha az ügyfél a belvárosban szeretne lakni, akkor mondhatjuk azt a térkép alapján, hogy ne az 5. kerületben keressen, amelyben 4000 Ft/m2 fölött van az általagos ár, hanem próbáljon keresgélni esetleg a VII. kerületben, ahol már 3200 Ft/m2-ért találhat kiadó lakást magának és mégis hasonló a két kerület.



Nyilván még rengeteg szempontot figyelembe kell venni egy albérlet keresésénél, de kiindulási alapnak megfelelő ez a modell.

Többváltozós lineáris regresszió:

A regressziós modellel vizsgálni akartam, hogy az adott adat alapján mely tényezők alakítják az árakat és milyen hatásuk van rá.

A kapott eredmények alapján felírható az egyenes egyenlete:

***Y* = -136362,68 + 16783,21**condition* + 19320,33**elevator* + 3275,29**size*+ 26506,15**type* + ε**

```
# tengelymetszet (b0)
print(lin_reg.intercept_)
```

```
-136362.68370721652
```

```
# Az együtthatók kiíratása
coeff_df = pd.DataFrame(lin_r
coeff_df
```

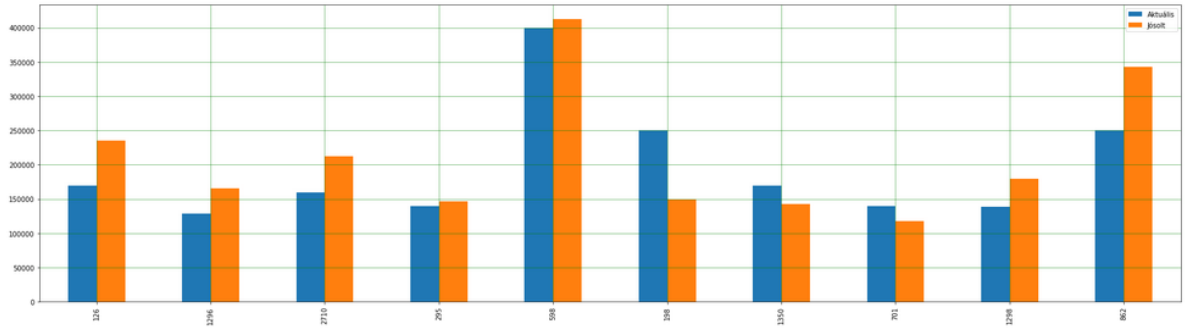
	Coefficient
condition	16783.211010
elevator	19320.329505
size	3275.293578
type	26506.148770

Ezek alapján azt mondhatjuk, hogy a kapott mintából kiindulva a regressziós modellt figyelembe véve, ha például minden tényező változatlan és a lakás alapterülete 1 m2-rel nő, akkor a lakás ára körülbelül 3275 Forinttal lesz több.

Ha a condition változót vesszük figyelembe, akkor azt mondhatom, hogy ha a lakás állapota egyel jobb, például ha nem felújított, hanem újszerű, akkor a lakás 16783 Forinttal kerül többe átlagosan.

```
MAE: 44544.96969474073
MSE: 3961208467.4710355
RMSE: 62938.1320621373
R2 Square 0.7260728470750181
```

A modell kiértékelésénél megállapíthatjuk, hogy elég nagy hibával dolgozik a modell, ennek oka többek között, hogy sok dummy változót vettünk be a modell építésébe, amelyek nem a legszerencsésebbek regressziós elemzésnél. Az R2 viszonylag magas, a regressziós egyenlet a függő változó varianciájának 73%-át tudja megmagyarázni. Viszont az RMSE értéke magas, nagy hibával dolgozik a modell, egy részlet az aktuális és előrejelzett értékekből (kékkel az aktuális, naranccsal a jósolt érték):



Összegzés:

A budapesti albérletpiac elemzése alapján azt gondolom, hogy nagyon jó alternatívák vannak adott kerületekre, illetve a nagy árkülönbségek miatt érdemes megnézni, hogy melyik is pontosan az a kerület, ahol az ügyfélnek valóban érdemes nézelődnie, hogy ár-érték arányban megfeleljen a lakás az elvárásainak. Ha megvan a választott kerület, akkor a regressziós modell alapján egyfajta becslést lehet adni az ügyfél igényei alapján a majdani lakás árára, amely akár módosíthatja a döntést.

Lakatos Áron, QZJ9VP