Enhancing The Home-buying Experience with Data Science

By George Pinto

Abstract

1. In the Real Estate business we need to figure out how can use modern data science to drive sales into all types of consumer demographics while removing economic/social discrimination from our sales process and sell to people we may not have reached with a regular sales format and:

2. Use that same data science to enhance the consumer experience to the point that each home buyer feels that we were working just for them, working to find exactly the environment they were looking for at a price they could afford

3. Using the Google Geocoding API, median home price ranges by Neighborhood from Zillow's economic data for research and Sci-Kit Learn machine learning libraries, I have created a model in which I have clustered Chicago, IL neighborhoods by the things that make them fun, or interesting or dependable to people along with the price data in each location so they can find a home surrounded by the things they're looking for at a price that fits their budget.

Motivation

Psychologically, people are driven by the environment they live in, a home is a place they can eat, sleep and co-exist in while their environment is what they strive for, they want better schools, better grocery stores, better neighbors with similar interests. They want a better life. That environment they want is very precious, very personal and a much stronger impulse for buying that simply just the home itself. Yes, I want a sales solution that will increase sales, but I want a solution for them, for people, to get a lot more from the homebuying experience than what they expected -- to find everything they want in terms of the physical space surrounding their lives at a price they can afford – not just a home.

Dataset(s)

1. From https://www.zillow.com/research/data/ I obtained my custom csv file as follows: I selected the Data Type as ZHVI (Zillow Home Value Index) for the current month and the Geography data arranged by Neighborhood, then proceeded to download the csv file.

2. For the type of locations surrounding each of the Neighborhoods I used the places API from Foursquare https://developer.foursquare.com/places-api by creating an account as a developer and obtaining credentials to use their API.

3. For matching locations with latitude and longitude data I used Google's geocoding API https://developers.google.com/maps/documentation/geocoding/start

4. I also pulled data from the City of Chicago's Geographical Information systems to obtain accurate coordinates from different Chicago Neighborhoods: https://www.chicago.gov/city/en/depts/doit/provdrs/gis.html

Data Preparation and Cleaning

I started this process by using Pandas for data manipulation. When I first created the Zillow dataframe I noted the information that would be needed. First, I needed to narrow the location data to Chicago, which is our area of interest for this model:

```
Chicago_Home_Price.head()
```

Date	RegionID	RegionName	State	Metro	County	City	SizeRank	Zhvi	MoM	QoQ	YoY	5Year
2019-07-31	269592	Logan Square	IL	Chicago-Naperville-Elgin	Cook County	Chicago	50	341000	-0.001464	-0.021521	-0.003215	0.070642
2019-07-31	403169	West Rogers Park	IL	Chicago-Naperville-Elgin	Cook County	Chicago	99	210100	-0.001900	-0.001426	0.069756	0.079082
2019-07-31	269566	Albany Park	IL	Chicago-Naperville-Elgin	Cook County	Chicago	141	261900	-0.001525	-0.012443	-0.016892	0.052375
2019-07-31	269609	Uptown	IL	Chicago-Naperville-Elgin	Cook County	Chicago	145	266000	-0.001502	-0.011152	-0.006721	0.038987
2019-07-31	269589	Lake View	IL	Chicago-Naperville-Elgin	Cook County	Chicago	149	414100	-0.003130	-0.012637	-0.009330	0.027940

Data Preparation and Cleaning

Next, I found the Zillow Home Value Index data and narrowed down the dataframe to only the neighborhood locations, the state and the average home price data:

```
In [7]: Chicago_Homes = Chicago_Home_Price[['RegionName','City','State','Zhvi']]
```

Let's see what our dataset looks like now:

```
In [8]: Chicago_Homes.head()
```

Out[8]:

	RegionName	City	State	Zhvi
50	Logan Square	Chicago	IL	341000
99	West Rogers Park	Chicago	IL	210100
141	Albany Park	Chicago	IL	261900
145	Uptown	Chicago	IL	266000
149	Lake View	Chicago	IL	414100

Data Preparation and Cleaning

Next, I changed the column names to more straightforward terms that I could use later in the analysis and for easier understanding and reset the index for manipulation later. I noted I did not have longitude and latitude data, retrieving that data was next.

```
In [10]: Chicago_Homes.reset_index(drop=True)
```

Out[10]:

	Neighborhood	City	State	Average_Home_Price
0	Logan Square	Chicago	IL	341000
1	West Rogers Park	Chicago	IL	210100
2	Albany Park	Chicago	IL	261900
3	Uptown	Chicago	IL	266000
4	Lake View	Chicago	IL	414100
5	Rogers Park	Chicago	IL	179900
6	Jefferson Park	Chicago	IL	280000
7	Portage Park	Chicago	IL	276700
8	South Loop	Chicago	IL	284700
9	Humboldt Park	Chicago	IL	263400
10	Bridgeport	Chicago	IL	323100
11	Edgewater	Chicago	IL	276400

Data Preparation and Cleaning

When uploading the City of Chicago's Geographical Information System's data broken down by neighborhood, I noted the location data, labeled as "geometry" below, was in Polygon form (which is a polygon shaped geometric area consisting of different points) and I needed to convert those to standard coordinates. Regular Pandas doesn't handle shapefile data manipulation for the next steps I used GeoPandas:

```
n [16]: Chicago_Locations.head()
```

ut[16]:

	PRI_NEIGH	SEC_NEIGH	SHAPE_AREA	SHAPE_LEN	geometry
0	Grand Boulevard	BRONZEVILLE	4.849250e+07	28196.837157	POLYGON ((1182322.0429 1876674.730700001, 1182...
1	Printers Row	PRINTERS ROW	2.162138e+06	6864.247156	POLYGON ((1176452.803199999 1897600.927599996,...
2	United Center	UNITED CENTER	3.252051e+07	23101.363745	POLYGON ((1165664.482500002 1902791.857299998,...
3	Sheffield & DePaul	SHEFFIELD & DEPAUL	1.048259e+07	13227.049745	POLYGON ((1167948.490499999 1914766.266999997,...
4	Humboldt Park	HUMBOLDT PARK	1.250104e+08	46126.751351	POLYGON ((1145646.625 1902270.125100002, 11456...

Data Preparation and Cleaning

The first step in retrieving the coordinates was converting the Polygons to centroids, which averages the Polygonal area data into average points for x and y or centroids, but you can see the numbers inside the POINT parentheses under geometry are still not standard coordinates:

In [18]:
```
Chicago_Locations.head()
```

Out[18]:

	PRI_NEIGH	SEC_NEIGH	SHAPE_AREA	SHAPE_LEN	geometry
0	Grand Boulevard	BRONZEVILLE	4.849250e+07	28196.837157	POINT (1179293.930098935 1875240.896042445)
1	Printers Row	PRINTERS ROW	2.162138e+06	6864.247156	POINT (1176074.080680388 1896362.291937789)
2	United Center	UNITED CENTER	3.252051e+07	23101.363745	POINT (1162397.851053501 1900257.057634569)
3	Sheffield & DePaul	SHEFFIELD & DEPAUL	1.048259e+07	13227.049745	POINT (1169201.97562045 1916790.107110357)
4	Humboldt Park	HUMBOLDT PARK	1.250104e+08	46126.751351	POINT (1152171.573477795 1907077.542505913)

Whoa! We need to convert the points to standard coordinate system:

Data Preparation and Cleaning

To convert the centroid data to standard coordinate points I used the GeoPandas to_crs method, you can see now that the numbers inside the POINT parentheses look like standard coordinates:

```
In [20]: Chicago_Locations.head()
```

Out[20]:

	PRI_NEIGH	SEC_NEIGH	SHAPE_AREA	SHAPE_LEN	geometry
0	Grand Boulevard	BRONZEVILLE	4.849250e+07	28196.837157	POINT (-87.61785967580141 41.81294893920027)
1	Printers Row	PRINTERS ROW	2.162138e+06	6864.247156	POINT (-87.62903477385433 41.87098062164045)
2	United Center	UNITED CENTER	3.252051e+07	23101.363745	POINT (-87.6791359550997 41.88196514863731)
3	Sheffield & DePaul	SHEFFIELD & DEPAUL	1.048259e+07	13227.049745	POINT (-87.653670400691 41.92718779602446)
4	Humboldt Park	HUMBOLDT PARK	1.250104e+08	46126.751351	POINT (-87.71650684946805 41.9008890219128)

Data Preparation and Cleaning

Next, I extracted the x and y coordinates from the POINT parenthesis and created columns on the right side of the dataframe for Longitude(x) and Latitude(y):

```
Chicago_Locations.head()
```

	PRI_NEIGH	SEC_NEIGH	SHAPE_AREA	SHAPE_LEN	geometry	Longitude	Latitude
0	Grand Boulevard	BRONZEVILLE	4.849250e+07	28196.837157	POINT (-87.61785967580141 41.81294893920027)	-87.617860	41.812949
1	Printers Row	PRINTERS ROW	2.162138e+06	6864.247156	POINT (-87.62903477385433 41.87098062164045)	-87.629035	41.870981
2	United Center	UNITED CENTER	3.252051e+07	23101.363745	POINT (-87.6791359550997 41.88196514863731)	-87.679136	41.881965
3	Sheffield & DePaul	SHEFFIELD & DEPAUL	1.048259e+07	13227.049745	POINT (-87.653670400691 41.92718779602446)	-87.653670	41.927188
4	Humboldt Park	HUMBOLDT PARK	1.250104e+08	46126.751351	POINT (-87.71650684946805 41.9008890219128)	-87.716507	41.900889

Data Preparation and Cleaning

Next, I narrowed the dataframe down to just the needed fields, which came down to primary neighborhood (PRI_NEIGH), Longitude and Latitude since I had retrieved the other information I needed from the Zillow dataframe:

```
Chicago_Locations.head()
```

	PRI_NEIGH	Longitude	Latitude
0	Grand Boulevard	-87.617860	41.812949
1	Printers Row	-87.629035	41.870981
2	United Center	-87.679136	41.881965
3	Sheffield & DePaul	-87.653670	41.927188
4	Humboldt Park	-87.716507	41.900889

Data Preparation and Cleaning

I proceeded to change the name of the column (PRI_NEIGH) to Neighborhood which would be very important to the next step in the process:

```
Chicago_Locations.head()
```

	Neighborhood	Longitude	Latitude
0	Grand Boulevard	-87.617860	41.812949
1	Printers Row	-87.629035	41.870981
2	United Center	-87.679136	41.881965
3	Sheffield & DePaul	-87.653670	41.927188
4	Humboldt Park	-87.716507	41.900889

Data Preparation and Cleaning

The Neighborhood fields allowed me to merge the dataframe created with the Zillow data with the dataframe created from the City of Chicago data:

```
In [30]: Chicago_Home_Locations = pd.merge(Chicago_Locations, Chicago_Homes, on='Neighborhood')

In [31]: Chicago_Home_Locations
```

Out[31]:

	Neighborhood	Longitude	Latitude	City	State	Average_Home_Price
0	Printers Row	-87.629035	41.870981	Chicago	IL	257600
1	Humboldt Park	-87.716507	41.900889	Chicago	IL	263400
2	Burnside	-87.596476	41.728182	Chicago	IL	94400
3	Hegewisch	-87.546578	41.660534	Chicago	IL	127900
4	Oakland	-87.603216	41.823750	Chicago	IL	285400
5	Woodlawn	-87.601686	41.778787	Chicago	IL	152800
6	Portage Park	-87.763399	41.954028	Chicago	IL	276700
7	Hermosa	-87.734740	41.924347	Chicago	IL	234100

Research

My intention is to create a data science model to incorporate what people want surrounding their home in terms of their daily activities and interests (the lifestyle choices they want) but also take into consideration their budget. The intention is to create choices, the same type of quality choices for neighborhoods but at different average home price ranges, so that the entire process becomes home-buyer oriented. The home-buyer should be able to do everything they want to do in their daily lives, and the real estate agency should be able to find those choices, find a neighborhood with the lifestyle features the home-buyer wants, but at a price that fits their budget.

Methods

1. Pandas library was used for data manipulation
2. GeoPandas library was used to extract Geospatial data
3. Folium library was used to visually show the location data on maps in an interactive way
4. The Foursquare API was used to extract venue/location information surrounding each neighborhood and the location data for said location/venues
5. Sklearn Machine Learning library was used to import the K-Means algorithm which was used to create clusters around the types of venues that surround each neighborhood
6. Matplotlib library was used to extract some additional meaningful data by using visualizations after creating the K-Means model

Methods

The Algorithm

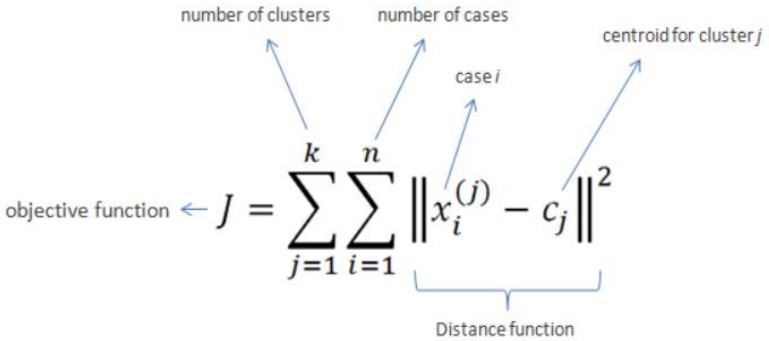
K-Means starts by randomly defining *k* centroids. From there, it works in iterative (repetitive) steps to perform two tasks:

Assign each data point to the closest corresponding centroid, using the standard Euclidean distance. In layman's terms: the straight-line distance between the data point and the centroid.
For each centroid, calculate the mean of the values of all the points belonging to it. The mean value becomes the new value of the centroid.

Once step 2 is complete, all the centroids have new values that correspond to the means of all of their corresponding points. These new points are put through steps one and two producing yet another set of centroid values. This process is repeated over and over until there is no change in the centroid values, meaning that they have been accurately grouped. Or, the process can be stopped when a previously determined maximum number of steps has been met (References, 5).

In our case, the data points are the combinations of frequencies of categories of venues within each neighborhood. You can think of the combinations as groupings where we consider each frequency or likelihood of each category of venue from occurring in that neighborhood. The centroid or center of each cluster will be the average of the most likely combinations of categories for the neighborhoods in that cluster (as above the mean of the values of all points belonging to it).

Methods



Algorithm

1. Clusters the data into k groups where k is predefined.
2. Select k points at random as cluster centers.
3. Assign objects to their closest cluster center according to the *Euclidean distance* function.
4. Calculate the centroid or mean of all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

(References, 6)

Methods

Choosing *k*

The algorithm explained above finds clusters for the number *k* that we chose. So, how do we decide on that number?

To find the best *k* we need to measure the quality of the clusters. The most traditional and straightforward method is to start with a random *k*, create centroids, and run the algorithm as we explained above. A sum is given based on the distances between each point and its closest centroid. As an increase in clusters correlates with smaller groupings and distances, this sum will always decrease when *k* increases; as an extreme example, if we choose a *k* value that is equal to the number of data points that we have, the sum will be zero.

The goal with this process is to find the point at which increasing *k* will cause a very small decrease in the error sum, while decreasing *k* will sharply increase the error sum. This sweet spot is called the “elbow point.” (References, 5)

Methods

For plotting *k* on the horizontal axis better results may be obtained by using either the sum of square distances or the sum of squared errors on the y axis, either method might offer a cleaner break at the elbow and an easier way to identify the best *k* :

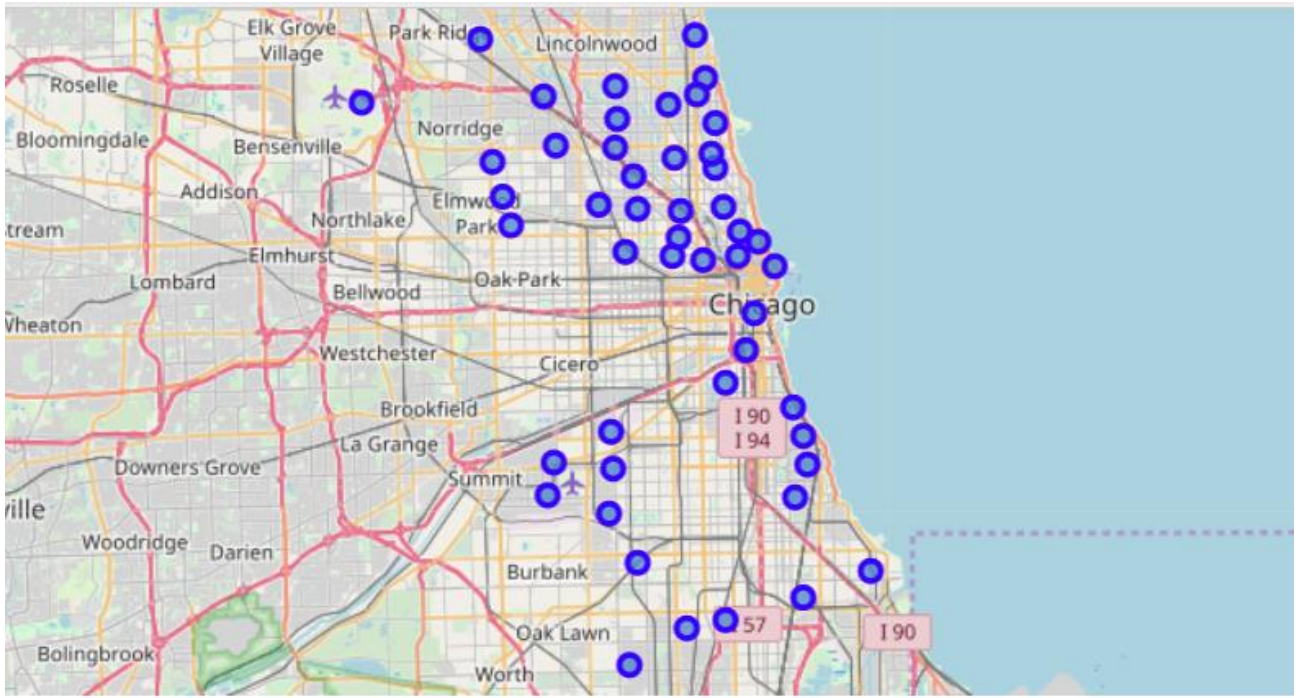
$$SSE = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

$$SST = \sum_{i=1}^{m} (y_i - \bar{y})^2$$

If a clear *k* is still difficult to discern by the elbow method, the silhouette method can be used or you can use it as well, and the code is available at https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html. Generally, you want the *k* with the highest coefficient as it represents correct classification of the clusters.

Findings

This is the map created with Folium showing the locations (Neighborhoods) from the merged dataframe we saw on the last data preparation slide:



Findings

I used the Foursquare API to find a limit of 100 venues/locations along a 500-meter radius (1km diameter) from the centroid (average latitude and longitude) of each neighborhood – as you can see, we can even use the API to extract individual venue/location information:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Printers Row	41.870981	-87.629035	Jazz Showcase	41.871650	-87.628707	Jazz Club
1	Printers Row	41.870981	-87.629035	Lou Malnati's Pizzeria	41.871588	-87.627425	Pizza Place
2	Printers Row	41.870981	-87.629035	UMAI Japanese Kitchen & Sushi	41.872213	-87.630724	Japanese Restaurant
3	Printers Row	41.870981	-87.629035	Bikram Yoga Chicago	41.871956	-87.629126	Yoga Studio
4	Printers Row	41.870981	-87.629035	Kriser's Natural Pet	41.869137	-87.627229	Pet Service
5	Printers Row	41.870981	-87.629035	Gordo's Homemade Ice Cream Bars	41.872627	-87.629281	Ice Cream Shop
6	Printers Row	41.870981	-87.629035	Kasey's Tavern	41.873317	-87.629284	Pub
7	Printers Row	41.870981	-87.629035	The Foundry - Printers Row CrossFit	41.872990	-87.630571	Gym / Fitness Center
8	Printers Row	41.870981	-87.629035	Sandmeyer's Bookstore	41.872996	-87.629287	Bookstore
9	Printers Row	41.870981	-87.629035	First Draft	41.873259	-87.630563	Sports Bar
10	Printers Row	41.870981	-87.629035	Artist & Craftsman Supply	41.871088	-87.625827	Arts & Crafts

Findings

I then proceeded to one hot encode the data frame for each category and then take the average frequency of occurrence of each category for each neighborhood.

[52]:
```
Chicago_grouped = Chicago_onehot.groupby('Neighborhood').mean().reset_index()
Chicago_grouped.head()
```

t[52]:

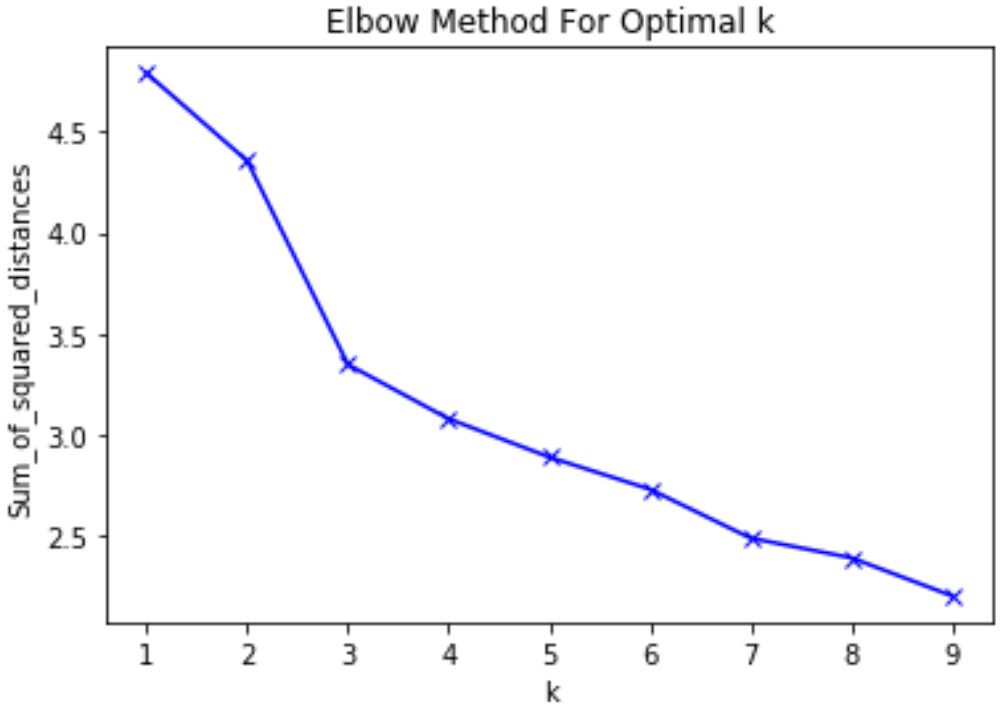
	Neighborhood	Yoga Studio	Accessories Store	African Restaurant	Airport Lounge	Airport Terminal	American Restaurant	Antique Shop	Arcade	Argentinian Restaurant	...	Vegetari / Vega Restaura
0	Albany Park	0.0	0.153846	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0
1	Andersonville	0.0	0.000000	0.0	0.0	0.0	0.0	0.016667	0.0	0.0	...	0
2	Archer Heights	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0
3	Ashburn	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0
4	Avondale	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0

Findings

I used this dataframe (previous slide) to train the K-Means model; I used a loop to run K-Means at multiple ranges of *k* (number of clusters) plotted against the sum of the within clusters squared distance (the distance from each centroid which is now composed of the clusters of most frequently occurring venues and locations in the neighborhoods), we want to minimize *k* (minimize within clusters square distance and maximize between clusters square distance so that the centroids represent true averages of different groups). I plotted the data and chose the *k* at which the curve breaks at the 'elbow' – see explanation from slide # 18 if you want more detail on this (you can see the curve changes after this point).

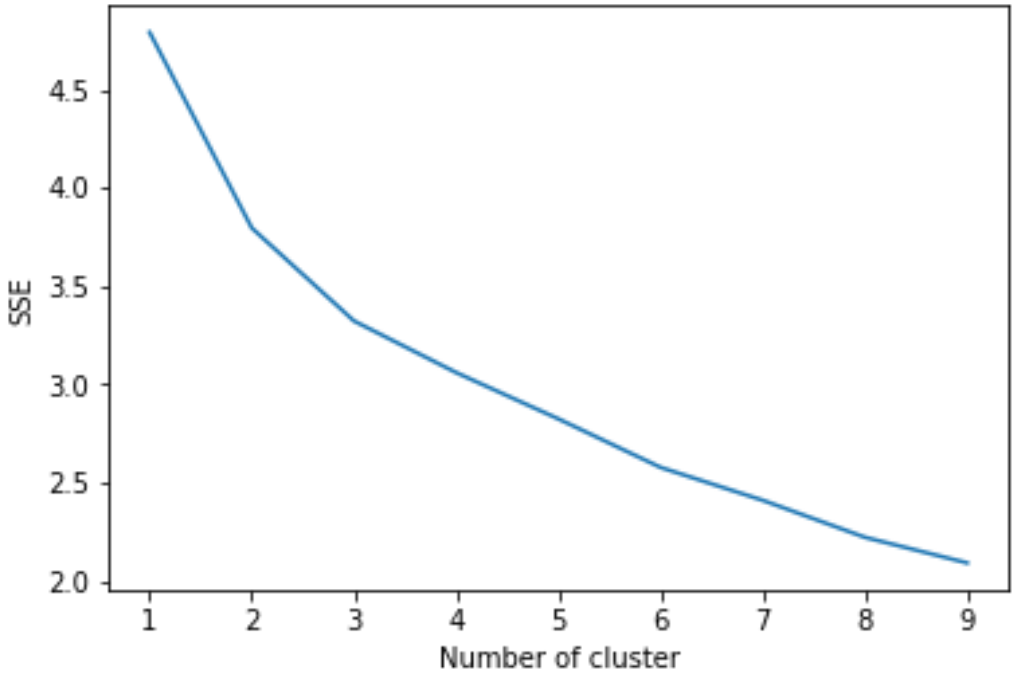
Findings

The value I chose was 3. You can see the curve changes right at this point (breaks at elbow):



Findings

To confirm my findings I also ran *k* against SSE (sum of squared errors) which sometimes gives a smoother curve, 3 looks about right:



Findings

One last test for confirmation -- I ran silhouette scores for each *k*, you choose the highest value, the test had the following results also pointing at *k* = 3 as the right value:

```
For n_clusters=2, The Silhouette Coefficient is 0.7907509958882437
For n_clusters=3, The Silhouette Coefficient is 0.8091254703246521
For n_clusters=4, The Silhouette Coefficient is 0.7816514235003741
For n_clusters=5, The Silhouette Coefficient is 0.7387882400303358
For n_clusters=6, The Silhouette Coefficient is 0.7468002094876061
For n_clusters=7, The Silhouette Coefficient is 0.7275258137231541
For n_clusters=8, The Silhouette Coefficient is 0.683992846115916
For n_clusters=9, The Silhouette Coefficient is 0.6507627446806667
For n_clusters=10, The Silhouette Coefficient is 0.6269933396915032
```

Findings

After creating the K-Means model I was able to use the K-Means cluster labels and add them to the original data frame to match the Neighborhoods to their clusters:

	Neighborhood	Longitude	Latitude	City	State	Average_Home_Price	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4 C
0	Printers Row	-87.629035	41.870981	Chicago	IL	257600	0	Gym / Fitness Center	Pizza Place	Clothing Store	Tac
1	Humboldt Park	-87.716507	41.900889	Chicago	IL	263400	0	Music Venue	Video Store	Gym	
2	Burnside	-87.596476	41.728182	Chicago	IL	94400	3	Convenience Store	Home Service	Motel	Inte
3	Hegewisch	-87.546578	41.660534	Chicago	IL	127900	1	Baseball Field	Women's Store	Electronics Store	Foc
4	Oakland	-87.603216	41.823750	Chicago	IL	285400	3	Beach	Park	Public Art	

Findings

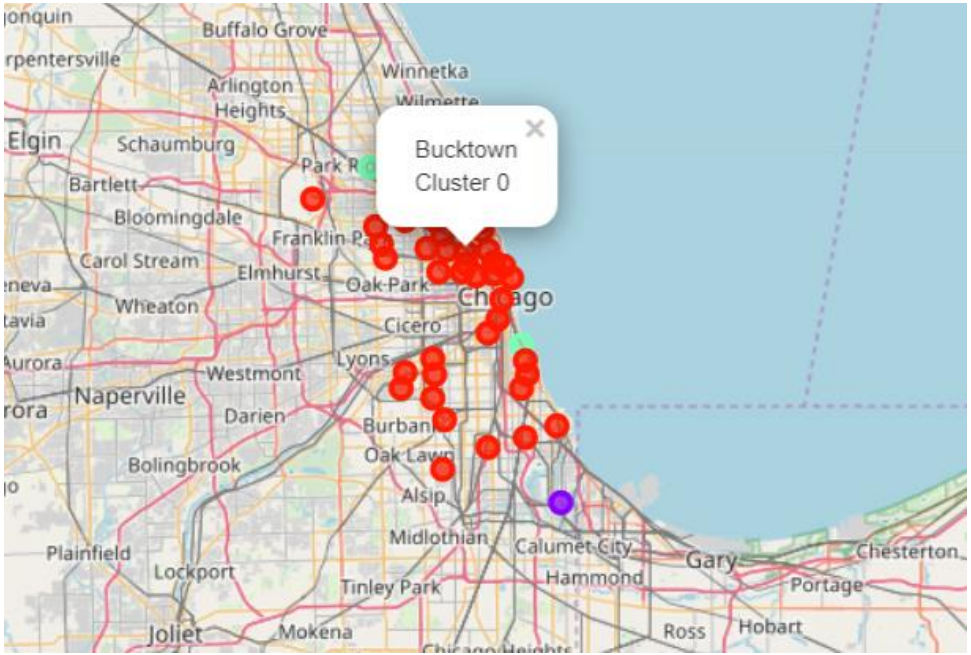
Here is the Folium map of the clusters so you can see how they are distributed:

Findings

This is a lot of data that can be manipulated a lot of different ways. We want to see if we did the research we set out to do. So let’s pretend we are a customer and we like Bucktown but cannot afford a home for ~500,000 dollars, we want to see what homes exist within the same cluster of venues/locations and we want choices that fit our budget. Now that we have our model, we can extract the information the customer would be interested in. **Let’s carefully examine the data in the next three slides.**

Findings

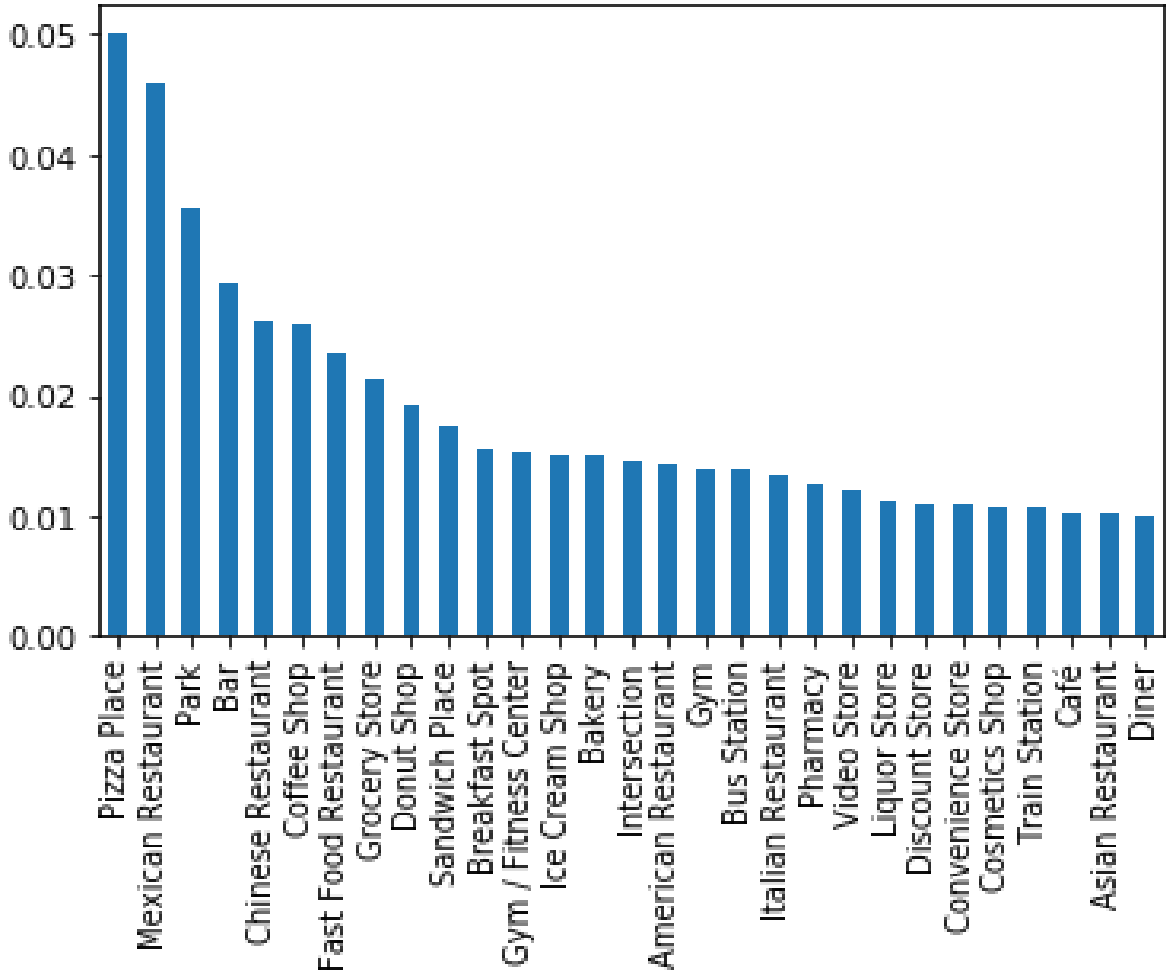
Let’s check in what cluster K-Means placed Bucktown, we can pull the information and inform the customer that the same color clusters (Red) are similar neighborhoods:



Findings

CLUSTER 0

Venues/Locations by frequency



Findings

CLUSTER 0 Average Home Prices by Neighborhood

	Neighborhood	Average_Home_Price	Cluster Labels
0	Mount Greenwood	232500	0
1	O'Hare	147700	0
2	Edgewater	276400	0
3	Lake View	414100	0
4	Lincoln Park	453100	0
5	Lincoln Square	331200	0
6	Washington Heights	103300	0
7	Hyde Park	235400	0
8	Bucktown	457400	0
9	Wrigleyville	438900	0
10	Andersonville	320700	0
11	Archer Heights	173100	0
12	Bridgeport	323100	0
13	West Elsdon	183000	0

Conclusions

Here is the scenario based on our data, we were looking to offer an alternative to Bucktown:

In our Venues/Locations by Frequency Bar Graph (slide # 31), we found lifestyle choices of plenty of bars, coffee shops and a plethora of different international restaurant choices, along with gyms, fitness centers and access to public transportation all offered around homes located in cluster 0. We mentioned that the average cost of a home in Bucktown was perhaps outside of the proposed customer budget, so we can recommend for example, to explore homes around O'Hare airport (see Average Home Prices Table for cluster 0 – slide # 32); the average home price in Bucktown is $457400 (index 8) , vs O'hare at $147700 (index 1). O'Hare's average home price is about a third of the price of Bucktown with the same type of lifestyle choices/environment in the surrounding area.

The Research was successful in showing how it could be used to center the sales process around lifestyle choices of homebuyers and their budget.

Limitations

- The first limitation I can mention is the source data itself. In developing an app or an effective model to scale we would need to make sure the data is complete. Recall in the data cleaning section when we narrowed down the Zillow Data to the final dataframe for that first piece of data, that dataframe was 129 rows (or 129 observations). Our final dataframe obtained from the City of Chicago's Geographical Information Systems to match latitudes and longitudes was 98 rows. These unequal sizes were true before we attempted to do anything that would change the row size (meaning the neighborhood names between the two data sources were not pulled from a standard source and/or they didn't have a unique identifier – i.e. the region ID for the Zillow data was not in the City of Chicago Data for example). After merging the two dataframes, we ended up with 50 cities (meaning the names of the neighborhoods also were not all equal). These issues were not significant for a sample but in order to show that our model would work, building the model to scale for commercial use would benefit from standardization of the neighborhood identifiers. I also noted that you can find services on the internet that will give you complete geographical information for a charge, but I did not verify this.

Limitations

- Secondly, it is unlikely that homebuyers would be interested only in lifestyle choices such as the ones shown by this model (although I do think that the power of an API such as Foursquare's in creating a model is clearly shown), real estate customers have a lot of questions that if answered could really elevate the entire process to a customer centric approach, here are some examples of the detail that could be added: 1)people want to know that their neighborhoods are safe including crime data and details like sex offender information would elevate the process, 2)People also want to know that their area has excellent schools for their children to attend. 3)Excellent healthcare options for anything that might come up including hospitals and access to medicine. 4) Quality of utilities in the neighborhood, gas, water, light, etc. There may be many other variables that are important to people that may be factored into the model. I just wanted to state how comprehensive the process could be for it to be compelling to even the most discerning customers.

Acknowledgements

I want to thank Alex Aklson and Polong Lin for making this class available for those of us interested in data science. This is particularly helpful for a career changer such as myself that needs to develop the experience to work in the field. I'm very grateful for the availability of this certification to demonstrate skills without yet having the work experience. I want to thank Coursera's awesome staff. I want to thank my wife for looking at and criticizing all my projects and for being incredibly supportive. Lastly, I want to encourage all data science students to keep moving forward, the field can be very challenging, but every small success is very rewarding in its own right, thank you all for reviewing my work and for your helpful comments.

References

1. https://www.zillow.com/research/data/
2. https://developer.foursquare.com/places-api
3. https://developers.google.com/maps/documentation/geocoding/start
4. https://www.chicago.gov/city/en/depts/doit/provdrs/gis.html
5. https://blog.easysol.net/machine-learning-algorithms-3/
6. https://www.saedsayad.com/clustering_kmeans.htm