# Home Selection in Chicago, IL

## Introduction

People look at a variety of factors when deciding to rent/buy a home in a new town. Location is a major driver of this decision

A lot of manual effort goes into this process along with realtor fees. For e.g. people may want to rent in a neighborhood that has coffee shops and grocery stores and a train station

In this problem, I will be analyzing Chicago neighborhoods to provide recommendations on neighborhoods to rent/buy homes based availability of combination of certain venues per the user's preference

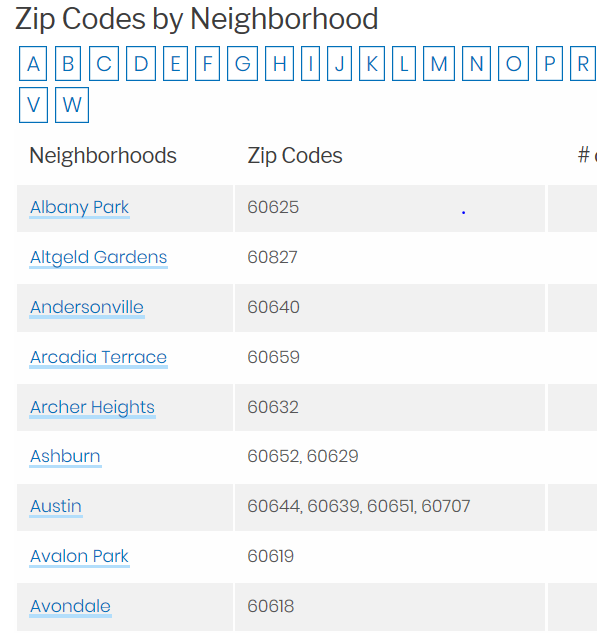
This analysis would be of interest to anyone who is looking to move into the city and rent/buy a home

## Data

Chicago has been selected to be the city whose neighborhoods will be analyzed

Three sets of data will be required for this problem:

* Chicago neighborhoods and associated zip codes: will be scraped from <https://www.dreamtown.com/maps/chicago-zipcode-map>
* The data will be converted to a dataframe and a longitude/latitude will be appended to the table using Python package: uszipcodes
* For each neighborhood, the nearby venues and associated latitudes/longitudes and categories are obtained using the FOURSQUARE API



## Methodology

### Data Cleansing & Wrangling

The table is cleaned by combining the neighborhoods and zip codes into one table. The # of listings are removed. Using the “USZIPCODE” / SearchEngine package/module, we get the latitude and longitude of each of the zip codes. Neighborhoods that have more than one zip code are excluded from the analysis



### Clustering of Similar Neighborhoods

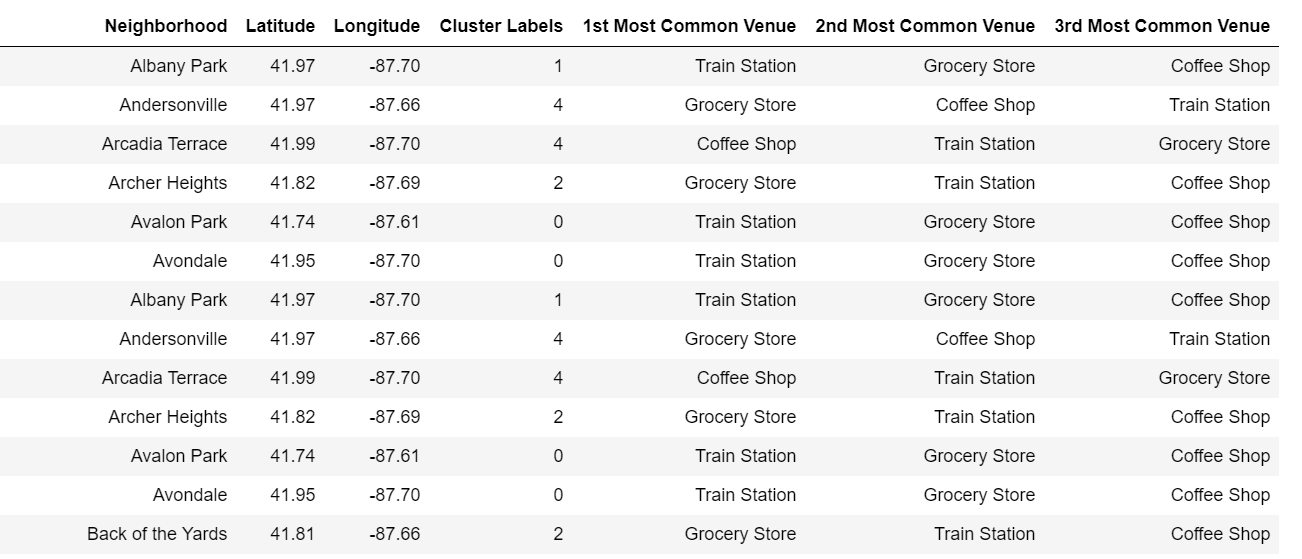
Need to cluster together similar neighborhoods – similarity is determined based on the number of a type of venue (e.g. grocery store, coffee shop). Foursquare API provides the category/ number of venues in a neighborhood along with latitude and longitude coordinates. We then estimate the total number of venue categories in each neighborhood. We also follow the one-hot encoding process to create a table enumerating the category of the venues in each neighborhood. We estimate a normalized frequency of occurrence of a category within a neighborhood and create a new dataframe.

In this analysis, we want to cluster neighborhoods based on the number of coffee shops, grocery stores and train stations they encompass. The clustering is based on k-means clustering algorithm – we start with 5 clusters. The idea is that neighborhoods that have most of the three venue categories (coffee shops, grocery stores, train stations) will be clustered together as opposed to neighborhoods that don’t have any of the three.

## Results

The main results are as follows:

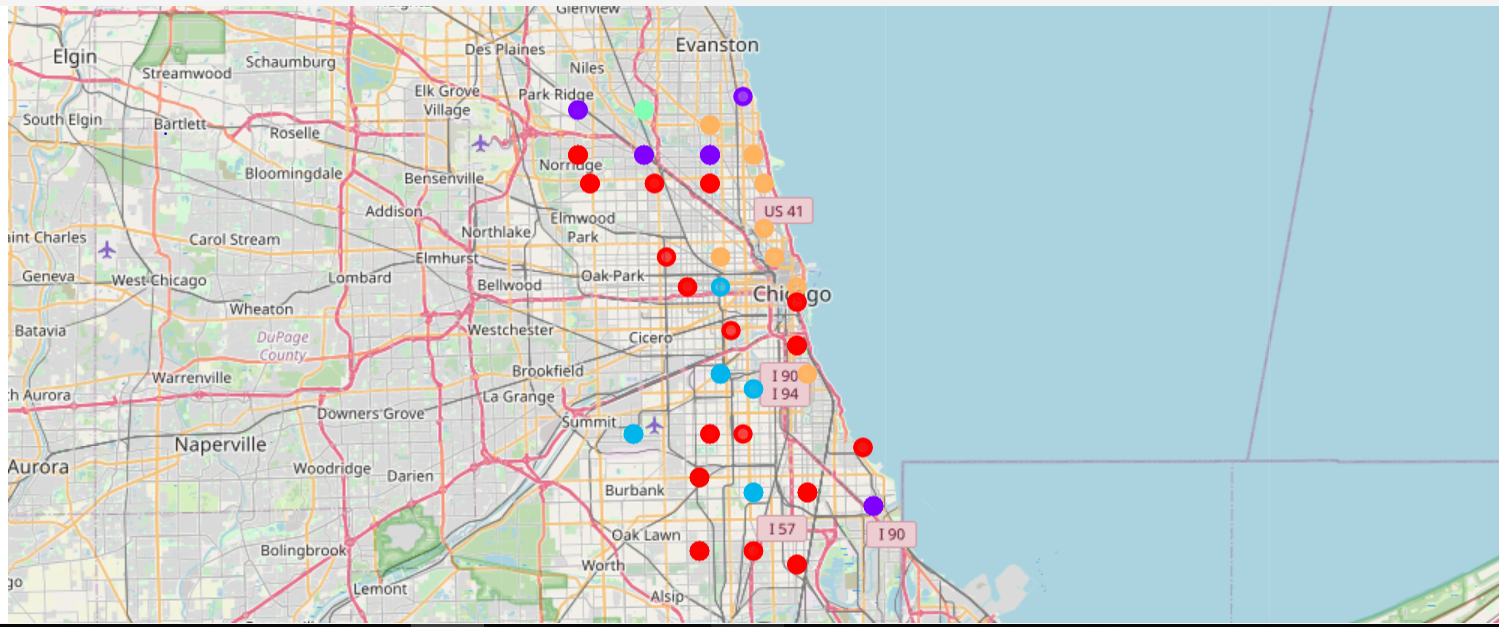
We tabulate neighborhoods and the clusters they belong to and the most common venues in them



**Neighborhoods with cluster labels (0 to 4)**

The clusters can also be depicted on a map (each neighborhood is a color coded dot) where the neighborhoods with same color coding form a cluster

**Map showing neighborhood clusters (each neighborhoods is a color coded dot)**

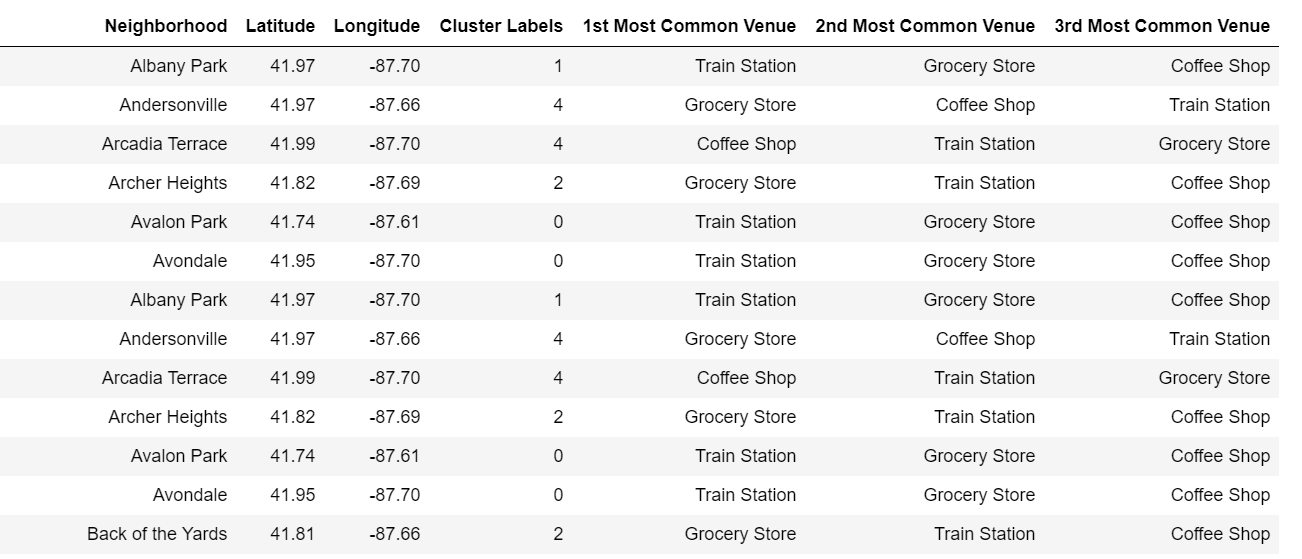


## Discussion

We create a table showing the normalized frequency of the venue categories in each cluster:



We can see from the table above that the total normalized frequency of the venue categories in each cluster – cluster 0 does not have a train station nor coffee shop, so we can probably skip all neighborhoods in that cluster. Cluster 1 seems to be a blend with reasonable scores for each venue category – so that can be explored further.



For example from the above table, we see Albany Park may be a candidate for a neighborhood to rent/buy a home in.

## Conclusion

We can use k-means clustering technique to cluster neighborhoods in a city that include a certain combination of venue categories important to home renter/buyer. This saves manual effort and resources of looking into each and every neighborhood in a city. In this analysis we used Train station, coffee shop and grocery store as the 3 venue categories.

In future analysis, we can increase or decrease the number of clusters to get a better sense of the neighborhood clustering. Furthermore, we can extend the analysis to provide an html interface where the user can select the type of venues they are interested in and the back-end Python ‘engine’ code can be run on the fly to identify neighborhoods they should be looking at first