William Schneider

# Recommending the ideal neighborhood using dual-supervision machine learning techniques

## Business Problem

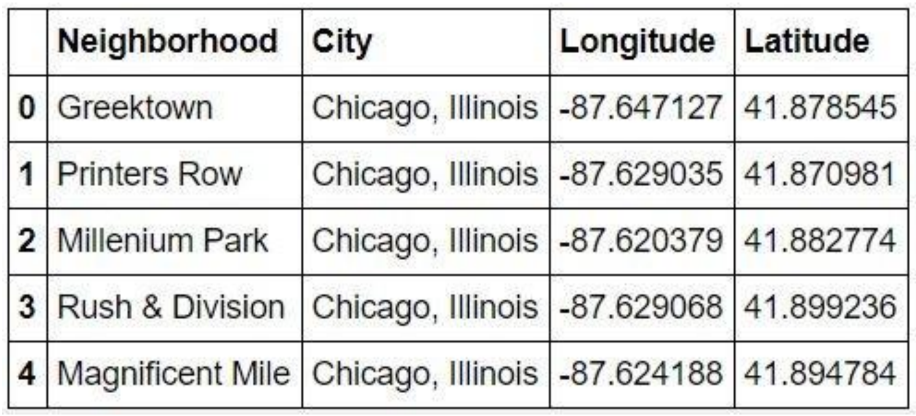
Moving to a new city can be an exciting adventure but finding a place to live can be more difficult than one might think. So often are neighborhoods advertised and characterized by single or marginal features, other attractive aspects of the area might be overshadowed, for better or for worse. What if there was a way to increase home buyer input to find the ideal neighborhood?

The project described in this writing aims to solve this issue by utilizing recommendation engine. The objective of this engine is recommended ideal neighborhoods for users based on Kmeans Clustering. The engine is further influenced by traditional content-based recommendation methods. The purpose of the engine is to combine user inputs for both neighborhood and venue preferences (i.e. Shops, parks, etc.) to predict the ideal living area within a set of North American city data.

## Data

The data used in this project consists of various neighborhood data in the select North American cities. The model also uses business and venue data in the surrounding area of the neighborhoods.

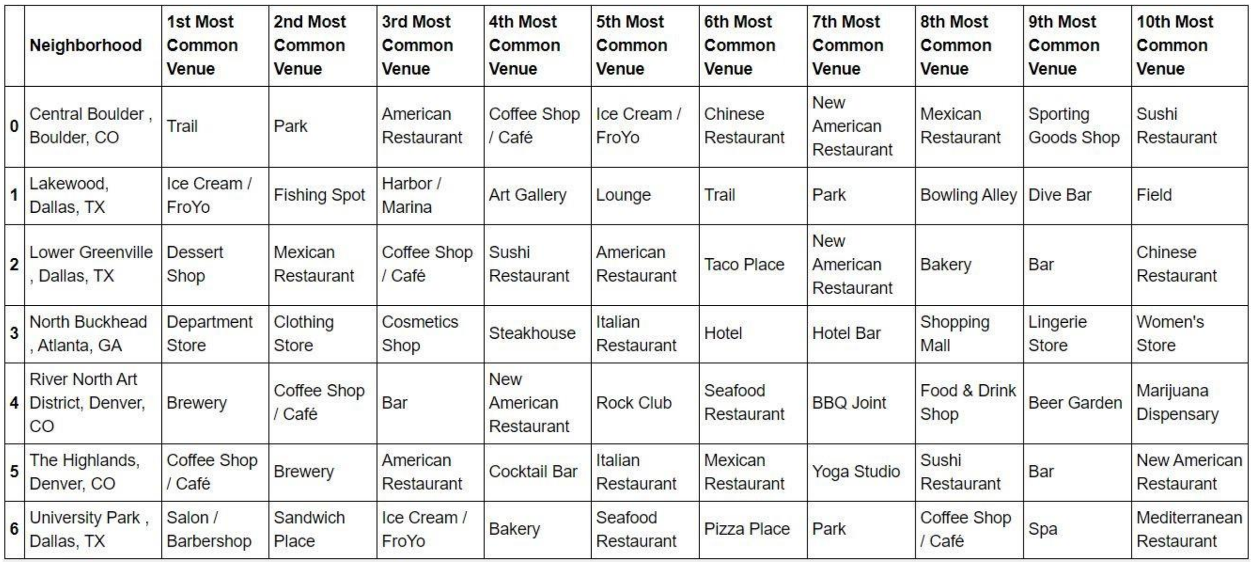
### Clustering Data "Test Data" – Destination City Neighborhood



|   | Neighborhood | City | Longitude | Latitude |
|---|---|---|---|---|
| 0 | Greektown | Chicago, Illinois | -87.647127 | 41.878545 |
| 1 | Printers Row | Chicago, Illinois | -87.629035 | 41.870981 |
| 2 | Millenium Park | Chicago, Illinois | -87.620379 | 41.882774 |
| 3 | Rush & Division | Chicago, Illinois | -87.629068 | 41.899236 |
| 4 | Magnificent Mile | Chicago, Illinois | -87.624188 | 41.894784 |

The dataset consists of cities that the user is proposing to move to. It is the dataset that the clusters are conditioned to. This data was obtained from various city open-data websites/repositories. Spatial coordinate in its raw form was in .shp format.(More details on this data formatting process is detailed under **Methodology**.

- **User Input Data** – Ideal City preferences



| | Neighborhood | 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 | Central Boulder , Boulder, CO | Trail | Park | American Restaurant | Coffee Shop / Café | Ice Cream / FroYo | Chinese Restaurant | New American Restaurant | Mexican Restaurant | Sporting Goods Shop | Sushi Restaurant |
| 1 | Lakewood, Dallas, TX | Ice Cream / FroYo | Fishing Spot | Harbor / Marina | Art Gallery | Lounge | Trail | Park | Bowling Alley | Dive Bar | Field |
| 2 | Lower Greenville , Dallas, TX | Dessert Shop | Mexican Restaurant | Coffee Shop / Café | Sushi Restaurant | American Restaurant | Taco Place | New American Restaurant | Bakery | Bar | Chinese Restaurant |
| 3 | North Buckhead , Atlanta, GA | Department Store | Clothing Store | Cosmetics Shop | Steakhouse | Italian Restaurant | Hotel | Hotel Bar | Shopping Mall | Lingerie Store | Women's Store |
| 4 | River North Art District, Denver, CO | Brewery | Coffee Shop / Café | Bar | New American Restaurant | Rock Club | Seafood Restaurant | BBQ Joint | Food & Drink Shop | Beer Garden | Marijuana Dispensary |
| 5 | The Highlands, Denver, CO | Coffee Shop / Café | Brewery | American Restaurant | Cocktail Bar | Italian Restaurant | Mexican Restaurant | Yoga Studio | Sushi Restaurant | Bar | New American Restaurant |
| 6 | University Park , Dallas, TX | Salon / Barbershop | Sandwich Place | Ice Cream / FroYo | Bakery | Seafood Restaurant | Pizza Place | Park | Coffee Shop / Café | Spa | Mediterranean Restaurant |

User input neighborhoods and zip codes. Lat/Long values were obtained from Google Mapping service for sake of time, but can be achieved with ArcGIS, especially with increased neighborhood/zip code inputs

- **Geospatial Data Manipulations** – Applied to both Clustering Data and User Input Data

Manipulations to the location coordinates of the data to format data into uniform features, as well as obtain zipcode information for each neighborhood. The spatial coordinate manipulation of select neighborhoods was accomplished via ArcGIS Post API requests and the service's find_centroids() function. The ArcGIS package was utilized to publish spatial data frames to package online mapping service. from there, the centroid function was called to the posted data, and returned in a CSV format.

**find_centroid()**: call function allowed easy manipulation of spatial data frame into a latitude-longitude coordinate system

William Schneider

```
#### Query Feature Layer from ArcGis Map repository

#### Call query results to a feature item
feature_service_item = search_results[0]

#### Call feature item as feature layer
feature_layer = feature_service_item.layers[0]
feature_layer


#### Create centroids from Neighborhood polygons
sf_centroids = analysis.find_centroids(feature_layer,
                                       point_location=True,
                                       output_name='San Fran Centroids')
sf_centroids
```

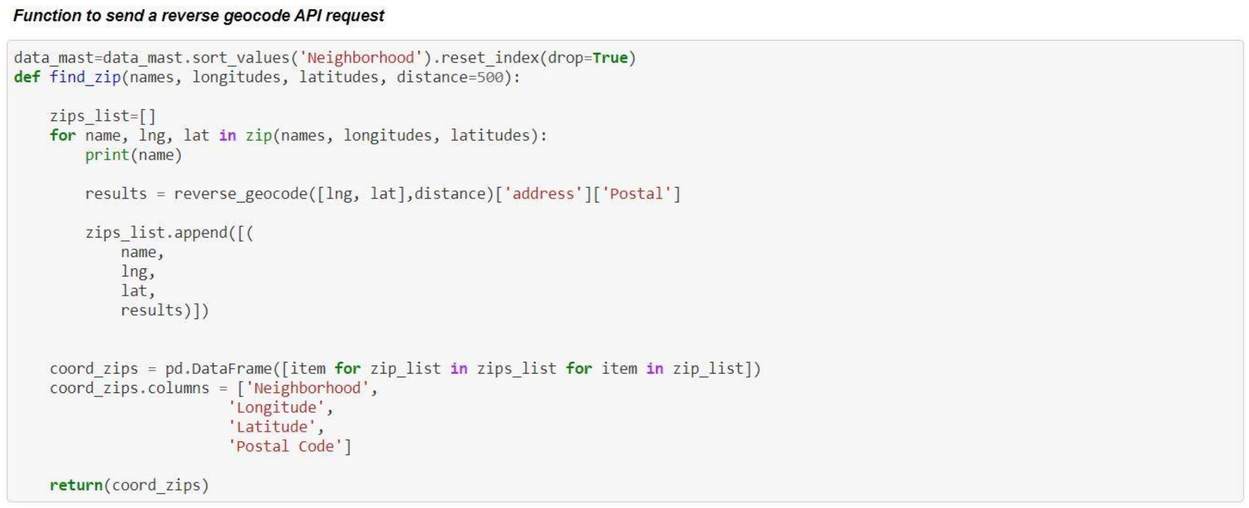
**San Fran Centroids**

Feature Layer Collection by WWSscript
Last Modified: August 22, 2020

| | OBJECTID | nbrhood | nid | sfar_distr | Shape__Area | Shape__Length | ORIG_FID | SHAPE |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Merced Manor | 3f | District 3 - Southwest | 390698.472656 | 2537.859137 | 45 | {"x": -122.47848975999995, "y": 37.73289811600... |
| **1** | 2 | Monterey Heights | 4m | District 4 - Twin Peaks West | 368775.332031 | 2554.749361 | 38 | {"x": -122.46120047599999, "y": 37.73268986700... |
| **2** | 3 | Balboa Terrace | 4a | District 4 - Twin Peaks West | 349939.445312 | 2597.270308 | 3 | {"x": -122.46845959699999, "y": 37.73139107400... |
| **3** | 4 | Anza Vista | 6a | District 6 - Central North | 463146.500000 | 2829.807921 | 2 | {"x": -122.44335277599998, "y": 37.78100556800... |
| **4** | 5 | Alamo Square | 6e | District 6 - Central North | 520422.613281 | 3166.780152 | 1 | {"x": -122.43446523499995, "y": 37.77656580100... |

(See data_format.ipynb in Git gist under "Content/Cities" for detailed code)

- **Reverse Geocoding**



***Function to send a reverse geocode API request***

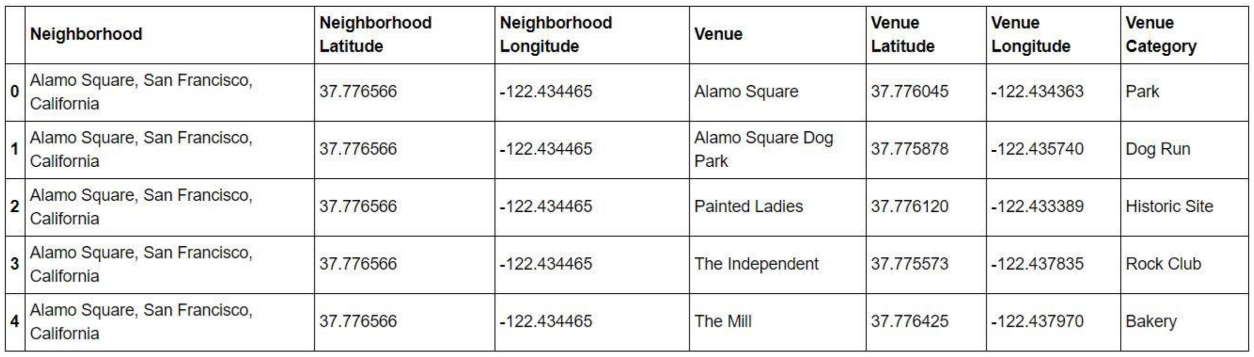
```
data_mast=data_mast.sort_values('Neighborhood').reset_index(drop=True)
def find_zip(names, longitudes, latitudes, distance=500):

    zips_list=[]
    for name, lng, lat in zip(names, longitudes, latitudes):
        print(name)

        results = reverse_geocode([lng, lat],distance)['address']['Postal']

        zips_list.append([(
            name,
            lng,
            lat,
            results)])


    coord_zips = pd.DataFrame([item for zip_list in zips_list for item in zip_list])
    coord_zips.columns = ['Neighborhood',
                          'Longitude',
                          'Latitude',
                          'Postal Code']

    return(coord_zips)
```

Reverse Geocoding was used to obtain the zip codes for given latitude-longitude coordinates. Zip codes were not used in this analysis, but have been included for ease of access in future updates to the model. The **Discussion** session below proposes even more considerations of zip code utilization.

- **ZipWho**

A geographical demographics API service. Gets data based on zip code input. Unable to use data in current versions of the model due to inability to call API with non-USA zip codes (Canada)

- **Foursquare Data** – Venue-Neighborhood Information



| | Neighborhood | Neighborhood Latitude | Neighborhood Longitude | Venue | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|---|---|---|---|
| 0 | Alamo Square, San Francisco, California | 37.776566 | -122.434465 | Alamo Square | 37.776045 | -122.434363 | Park |
| 1 | Alamo Square, San Francisco, California | 37.776566 | -122.434465 | Alamo Square Dog Park | 37.775878 | -122.435740 | Dog Run |
| 2 | Alamo Square, San Francisco, California | 37.776566 | -122.434465 | Painted Ladies | 37.776120 | -122.433389 | Historic Site |
| 3 | Alamo Square, San Francisco, California | 37.776566 | -122.434465 | The Independent | 37.775573 | -122.437835 | Rock Club |
| 4 | Alamo Square, San Francisco, California | 37.776566 | -122.434465 | The Mill | 37.776425 | -122.437970 | Bakery |

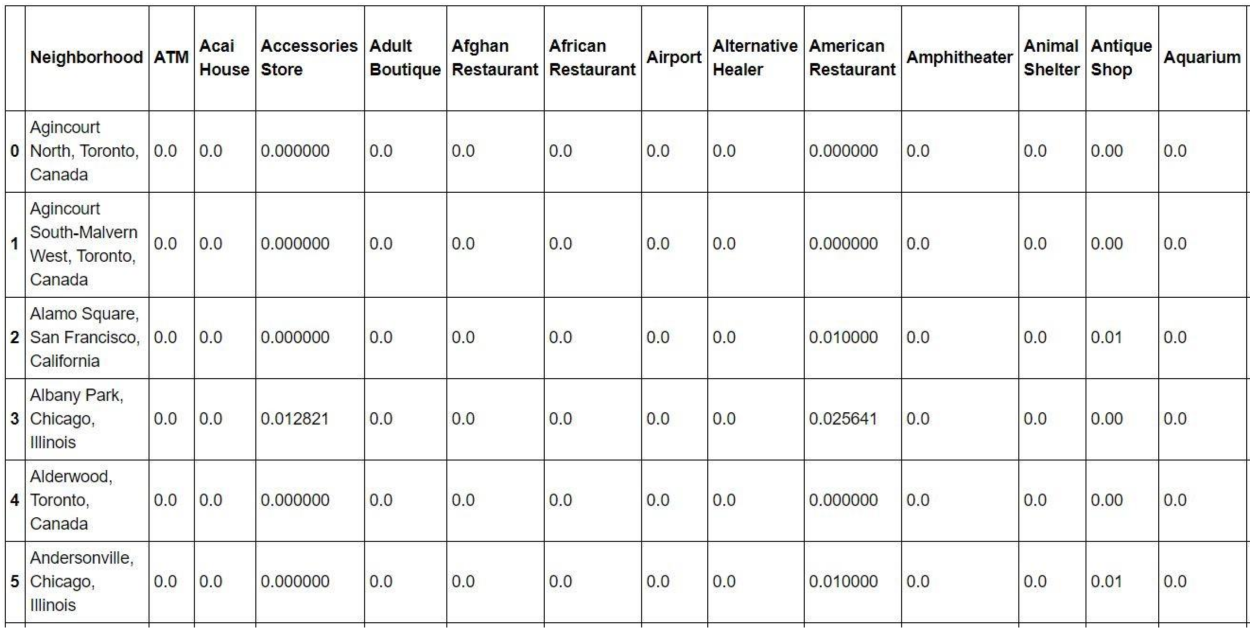
Foursquare is a location data and technology platform. The key sections of the platform used in this model revolve around venue data and information for areas surrounding given location coordinates.

**Data Acquisition**: In order to acquire the Foursquare data for this uses in the recommendation engine, REST API calls were used with the given conditions: Latitude, Longitude, radius, and limit features. (For both User Input and Test datasets) "Limit" is the maximum number of venues to be return for a given set of input coordinates. "Radius" is the maximum distance surrounding the input coordinates that venue data will be returned.

## Methodology

### Exploratory Data Analysis & Pre-Processing

After the data has been pulled, the data underwent a varying stage of data pre-processing and exploratory analysis, throughout the whole process of the model. One-hot encoding is a data pre-processing technique that was instrumental to transforming a list of returned venues and venue types, to a categorical feature occurrence for each venue category occurrence. The mean (“frequency”) of each venue category was then found with respect to each neighborhood. This measure is also referred to as the Mean Boolean Value (MBV). The MBV is normalized by nature.

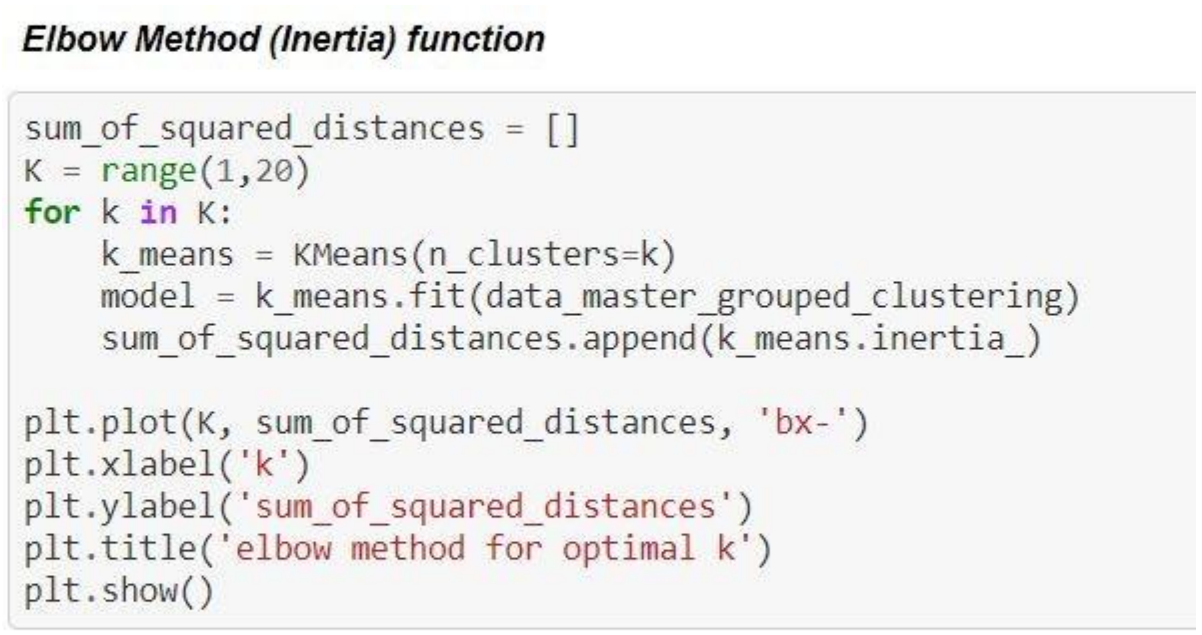


| | Neighborhood | ATM | Acai House | Accessories Store | Adult Boutique | Afghan Restaurant | African Restaurant | Airport | Alternative Healer | American Restaurant | Amphitheater | Animal Shelter | Antique Shop | Aquarium |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Agincourt North, Toronto, Canada | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.00 | 0.0 |
| 1 | Agincourt South-Malvern West, Toronto, Canada | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.00 | 0.0 |
| 2 | Alamo Square, San Francisco, California | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.010000 | 0.0 | 0.0 | 0.01 | 0.0 |
| 3 | Albany Park, Chicago, Illinois | 0.0 | 0.0 | 0.012821 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.025641 | 0.0 | 0.0 | 0.00 | 0.0 |
| 4 | Alderwood, Toronto, Canada | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.00 | 0.0 |
| 5 | Andersonville, Chicago, Illinois | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.010000 | 0.0 | 0.0 | 0.01 | 0.0 |

### K-Means Clustering

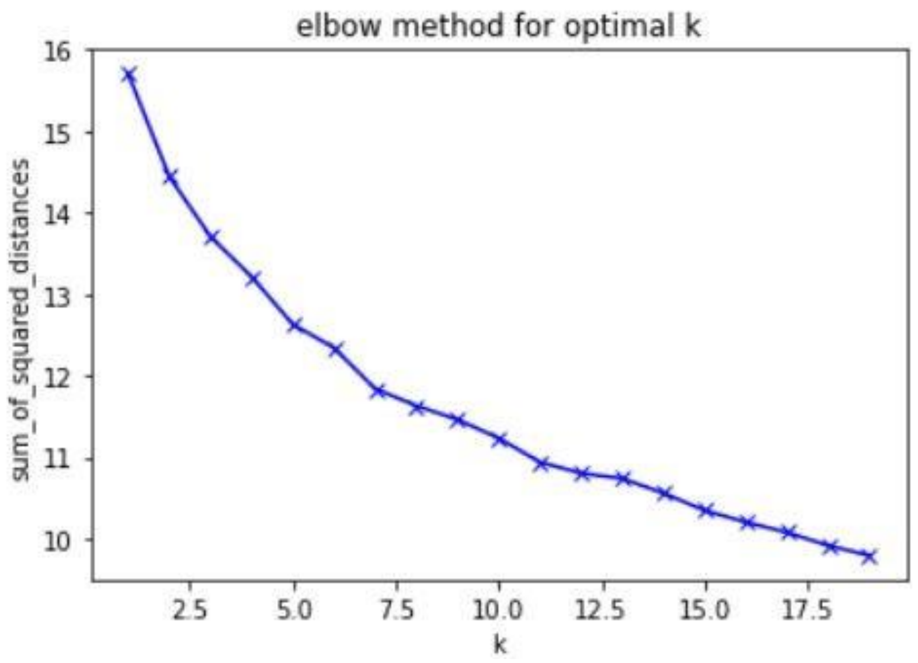
Kmeans clustering techniques were used during the first stage of the recommendation engine. Kmeans was utilized on the data via the scikit.learn package for Python. The Kmeans model clusters data points based on the distance of the samples from each other. The end goal of a Kmeans is to minimize intra-cluster distances and maximize inter-cluster distances.

Since the Kmeans algorithm run hundreds of times by default, one can utilize the Elbow Method to determine the correct number of clusters. The elbow method is essentially observing for k-clusters, where sum of squared distances (SSD) is decreasing in a linear fashion. The elbow methods used in this analysis were Inertia and Euclidean Distance (ED), but the latter acted more as a confirmation tool. Inertia is the SSD of samples to their closest cluster center.

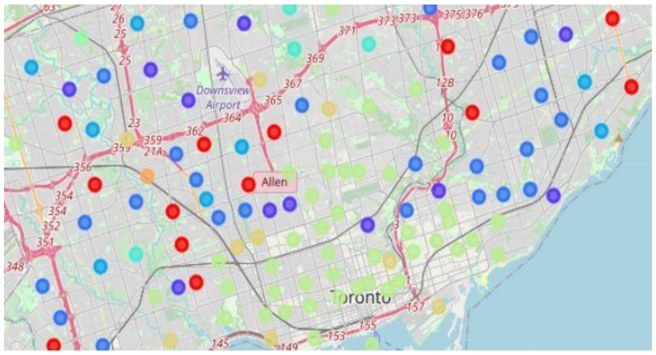


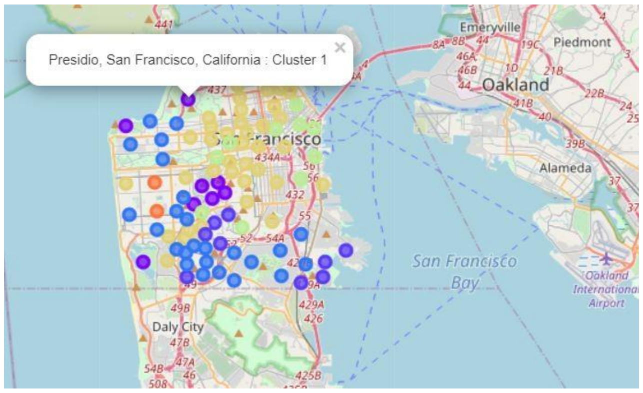
***Elbow Method (Inertia) function***

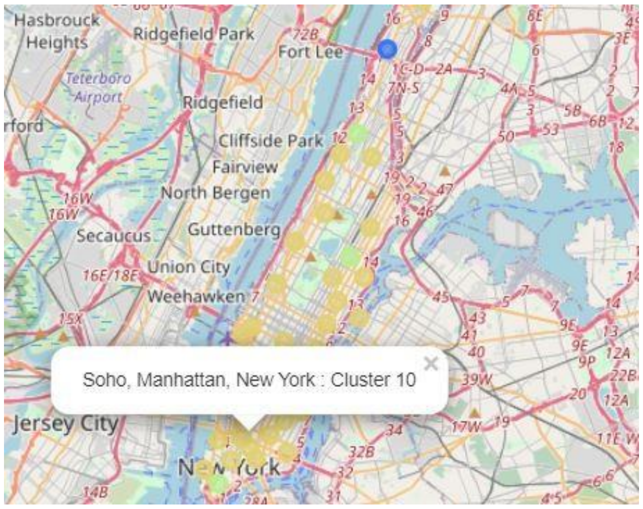
```
sum_of_squared_distances = []
K = range(1,20)
for k in K:
    k_means = KMeans(n_clusters=k)
    model = k_means.fit(data_master_grouped_clustering)
    sum_of_squared_distances.append(k_means.inertia_)

plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('sum_of_squared_distances')
plt.title('elbow method for optimal k')
plt.show()
```

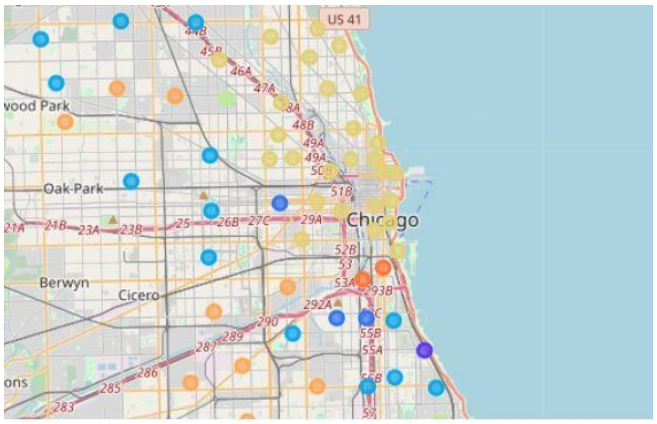


## Initial Cluster Visualizations (Map)




Presidio, San Francisco, California : Cluster 1
San Francisco
Oakland
Emeryville
Piedmont
Alameda
Daly City
San Francisco Bay
Oakland International Airport



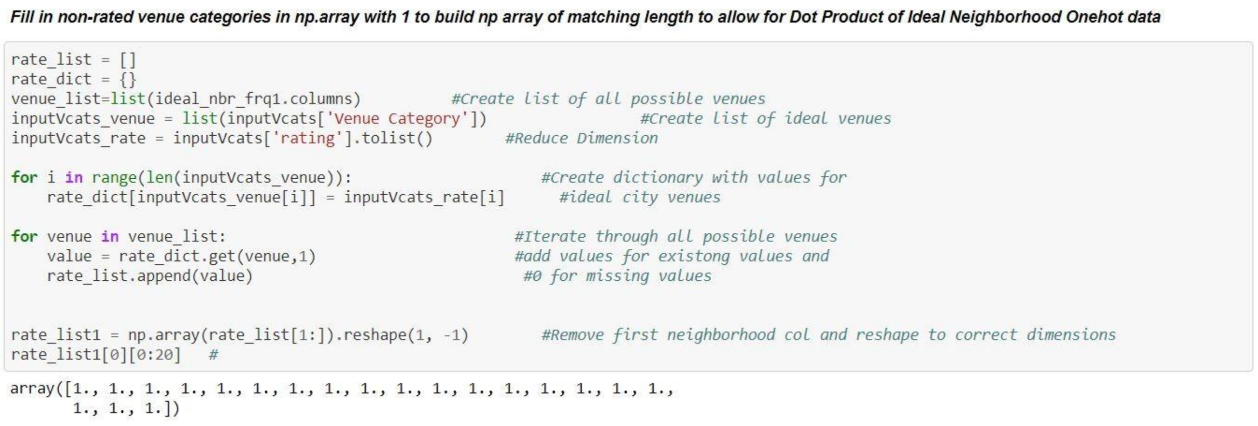


North American cities in the clustering dataset were clustered under one set, rather than by each city. This was done to ensure the cluster are homogeneous with respect to each city.

Recommended clusters were determined using the Kmeans.predict() function on user's average neighborhood preferences. (See the referenced Jupyter Notebook for a more detailed look at the Python code)

## Traditional Recommendation Method (Content-Item Based Recommending System)

Once the neighborhood cluster is predicted by the user's ideal neighborhood preferences, venue categories are ranked via the user input and weighted against the predicted cluster's frequency. The objective of this weighing is to highlight a user's preference of a venue category. The true frequency of the category is not changed. The engine was designed so that the user need not input a rank for a venue category that is indifferent to their preferences. Categories that are not ranked are automatically given a neutral value of 1 so that they are normalized relative to the rest of the data.



***Fill in non-rated venue categories in np.array with 1 to build np array of matching length to allow for Dot Product of Ideal Neighborhood Onehot data***

```
rate_list = []
rate_dict = {}
venue_list=list(ideal_nbr_frq1.columns)          #Create list of all possible venues
inputVcats_venue = list(inputVcats['Venue Category'])                #Create list of ideal venues
inputVcats_rate = inputVcats['rating'].tolist()        #Reduce Dimension

for i in range(len(inputVcats_venue)):                      #Create dictionary with values for
    rate_dict[inputVcats_venue[i]] = inputVcats_rate[i]      #ideal city venues

for venue in venue_list:                                 #Iterate through all possible venues
    value = rate_dict.get(venue,1)                       #add values for existong values and
    rate_list.append(value)                               #0 for missing values


rate_list1 = np.array(rate_list[1:]).reshape(1, -1)        #Remove first neighborhood col and reshape to correct dimensions
rate_list1[0][0:20]   #
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1.])
```

After obtaining a ranking list, the User Profile is be generated. The User Profile is the Dot product of the input rankings and the predicted cluster venue frequency data. The profile was then multiplied element-wise on the frequency data, producing Weighted Venues Matrix (WVM). The sum of the matrix rows gives the Weighted Average for each Neighborhood. This average was then merged into the recommended cluster data frame, and sorted descending to observe the most preferred clustered neighborhood based on venue preference. Table size and values were tested manually to ensure accuracy when merging rankings into predicted cluster data.

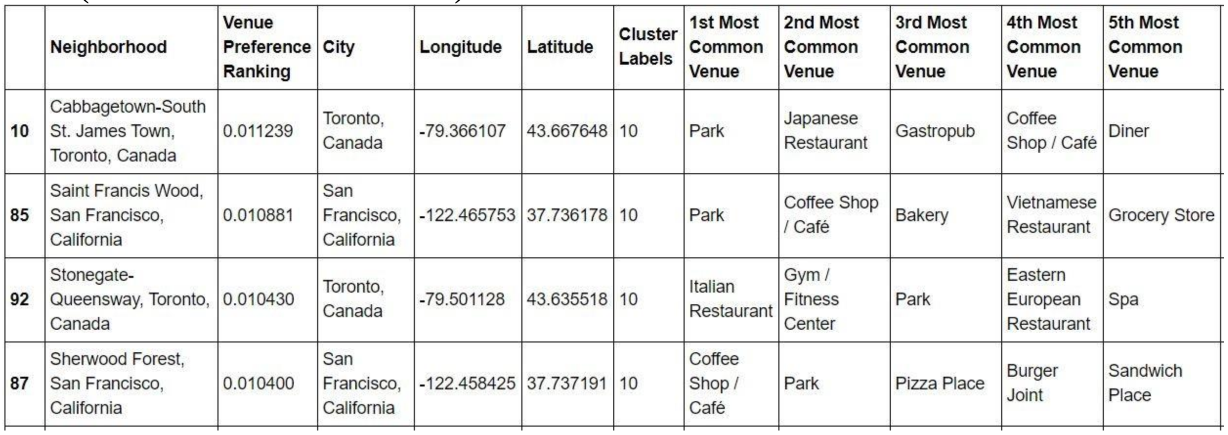
## Results

Early versions of the model generated neighborhood recommendations solely based on Kmeans clustering techniques. Repeated tests of the recommendation engine, proved to generally recommend cities based on accurate neighborhood clusters, but was too high level to decipher any meaning. The Kmeans Clustering algorithm help group city neighborhoods by venue preference. Including additional item-based recommendation via venue category frequencies, provided an extra layer of user preference and increased engine accuracy. The current engine can find the most ideal/preferred neighborhoods within a cluster via venue category rankings.

### Recommended Neighborhoods

The clustering and content-based recommendation were also run at opposite stages of the model process. That is, the Weighted Venues Matrix was generated and ranked on test data before clustering. This method has the potential benefit of influencing clusters before a prediction is made based on the user’s ideal preferences.

Clusters did appear to change, and influence was not factored within predicted-cluster neighborhood recommendation. At this stage, running the recommendation stages in the reversed order (Content-Based then Kmeans) does not benefit the model.



| | Neighborhood | Venue Preference Ranking | City | Longitude | Latitude | Cluster Labels | 1st Most Common Venue | 2nd Most Common Venue | 3rd Most Common Venue | 4th Most Common Venue | 5th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | Cabbagetown-South St. James Town, Toronto, Canada | 0.011239 | Toronto, Canada | -79.366107 | 43.667648 | 10 | Park | Japanese Restaurant | Gastropub | Coffee Shop / Café | Diner |
| **85** | Saint Francis Wood, San Francisco, California | 0.010881 | San Francisco, California | -122.465753 | 37.736178 | 10 | Park | Coffee Shop / Café | Bakery | Vietnamese Restaurant | Grocery Store |
| **92** | Stonegate-Queensway, Toronto, Canada | 0.010430 | Toronto, Canada | -79.501128 | 43.635518 | 10 | Italian Restaurant | Gym / Fitness Center | Park | Eastern European Restaurant | Spa |
| **87** | Sherwood Forest, San Francisco, California | 0.010400 | San Francisco, California | -122.458425 | 37.737191 | 10 | Coffee Shop / Café | Park | Pizza Place | Burger Joint | Sandwich Place |

## Discussion

Currently, the model only recommends on venue categories using Foursquare data. Using other geo-information APIs, such as Google or ArcGIS, may provide more accurate venue data for each neighborhood. Expenses for more API requests would need to be evaluated since these services carry a premium call rate. Upon further review of the initial clusters, denser packed areas tend to cluster together, even though the clustering is not factoring location coordinates. Reduction of the radius variable in the Foursquare API call could help to remedy this issue; by reducing the radius in which venues are returned for a given coordinate input, venue-neighborhood overlapping would be limited. however this would affect isolate sprawling or large neighborhoods with respect to number of initial centroid clusters.

Demographic data would positively influence the model's recommendations. Using demographic data, we can use other preferred metrics, such as the median income as well as the cost of living, to influence the model. Ranking/filtering the recommendation results on such metrics can provide value to the user via preference results feasibility, as well as highlight area of greater buyer/renter value.

In further updates, the Kmeans engine could also be influenced by demographic data such as income, cost of living, public service budgets, etc.

## Conclusion

This engine aims to recommend ideal neighborhoods based on a user's list of ideal neighborhood inputs, as well as rankings of nearby venues in the recommended neighborhoods. Initial results were at high-level value, and did not provide further recommendation, leaving the user with a frequently lengthy list of neighborhoods in the recommended cluster. Content-Based Filtering, a traditional recommendation system technique, was used further to provide more specific results, accomplished via item-item based recommendation. The engine used is Kmeans, so it is unsupervised. The engine does have room for refinement, notably it could benefit from demographic preferences for the user, in order to meet their monetary expectations of living.

William Schneider

# Source and References

## Source Notebook and Project Gist:

- Project Jupyter Notebook (Master) -- https://github.com/schwill2018/AppliedDSCap/blob/master/Master_final.ipynb
- Project/Model Gist -- https://github.com/schwill2018/AppliedDSCap

## References:

- Neighbourhood Profiles -- https://open.toronto.ca/dataset/neighbourhood-profiles/
- Analysis Neighborhoods -- https://data.sfgov.org/Geographic-Locations-and-Boundaries/Analysis-Neighborhoods/p5b7-5n3h
- Boundaries - Neighborhoods - https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Neighborhoods/bbvz-uum9
- 2014 New York City Neighborhood Names -- https://geo.nyu.edu/catalog/nyu_2451_34572

## Packages:

- Pandas, Numpy, Sci-kit learn, Matplotlib, ArcGIS, Foursquare