Predicting which country to choose for residence

Introduction

Every year, there are many people happen to change their residency around the world due to Client's work location. One such family of Indian origin got an offer to choose either Chicago, USA or Toronto, CA as their new work location. Both Chicago and Toronto has accountable number of Indian population. A city with more Indian stuffs will be preferred over the other. Therefore, it is important to predict accurately whether Chicago or Toronto has more Indian culture around.

Machine learning allows for the creation of computational models capable of identifying patterns in multi-dimensional datasets. This project aims to leverage venue data from Foursquare 'Places API' and a machine learning algorithm called 'k-means clustering' to identify 'Chicago City, USA' and 'Toronto city, CA' neighborhoods of 'Indian Restaurant'.

Problem

Data that might contribute to determine which location to choose include indian groceries, food, temple, school, commute, population, home prices, crime rate, etc. Now we are going to predict which location is better for an Indian family to live, considering the availability of Indian Food.

Interest

Obviously, any Indian who travels to these two cities would also be interested in knowing where their cuisine is located. Others who are interested might be those who wish to explore new cuisine food.

Data

Data Sources

Chicago,USA data contains Chicago city's Borough, Neighbourhood, Latitude and Longitude. Chicago data can be found here. Toronto,CA data contains Toronto city's Postcode, Borough, Neighbourhood, Latitude and Longitude. Toronto data can be found here and here.

Data Cleaning

For chicago data, scrapped table containing Neighbourhood and Community area using pandas read_html() method. Renamed column names 'Community area' as 'Borough'. It contained only Borough and Neighbourhood names. Grouped 'Neighbourhood' values on unique 'Borough' values and listed them as one value separated by commas. To get venue details, we first need latitude and longitude coordinates of each borough. I used google search method to obtain gps coordinates of all boroughs and saved them as a .csv file. Extracted it as a new dataframe.

Both dataframe have Borough column as common, so merged both dataframe on Borough values to obtain gps coordinates. The final dataframe *chicago_data* looks like this.

	Borough	Neighbourhood	Latitude	Longitude
0	ALBANY PARK	Albany Park,Mayfair,North Mayfair,Ravenswood M...	41.9683	-87.7280
1	ARCHER HEIGHTS	Archer Heights	41.8079	-87.7236
2	ARMOUR SQUARE	Armour Square,Chinatown,Wentworth Gardens	41.8408	-87.6340
3	ASHBURN	Ashburn,Ashburn Estates,Beverly View,Crestline...	41.7479	-87.7072
4	AUBURN GRESHAM	Auburn Gresham,Gresham	41.7434	-87.6562

Chicago city data

After cleaning there were 81 records and 4 features(Borough, Neighbourhood, Latitude, Longitude).

For Canada data, scrapped the wikipedia page with postcode, borough and neighbourhood column values using pandas read_html() method. There are few rows that has “Not assigned” value in Borough column. Delete those rows using drop() function. One entry has “Not assigned” value in Neighbourhood column. Replace its value with the value in its Borough column. Apply groupby() method, to group neighbourhoods with same postcode as one row.

Create a new dataframe to read .csv file that contains Toronto city’s postcode and gps coordinates. Merging both dataframe on Postcode, to obtain gps coordinates of each postcode. The final dataframe *canada_data* looks like this.

	Postcode	Borough	Neighbourhood	Latitude	Longitude
0	M1B	Scarborough	Rouge,Malvern	43.806686	-79.194353
1	M1C	Scarborough	Highland Creek,Rouge Hill,Port Union	43.784535	-79.160497
2	M1E	Scarborough	Guildwood,Morningside,West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

Toronto city data

After cleaning there were 99 records and 5 features(Postcode, Borough, Neighbourhood, Latitude, Longitude).

Foursquare API

Foursquare Places API is used to obtain data related to ‘venues’ in any given location using its gps coordinates and a specific url created for API call. It is important to note that Foursquare defines a ‘venue’ as a place that one can go to, or check-in to, and that a ‘venue’ is not necessarily a specific venue but can be any establishment such as a restaurant or type of retail shop, etc. Each Foursquare ‘venue’ is assigned a ‘category’ and each ‘category’ is associated

with a particular ‘categoryID’. After every API call, use *requests library get()* function to return a json file with all venue information for entire location specified in the url.

First we create a url with our client id, client secret, version for using Foursquare API. Extract venues list within specified radius for first neighbourhood of chicago dataframe.

```
LIMIT = 100
radius = 750
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
      CLIENT_ID, CLIENT_SECRET, VERSION, n_latitude, n_longitude, radius, LIMIT)
url
```

```
'https://api.foursquare.com/v2/venues/explore?&client_id=3UWNFX0SRVQJMVVCJ3B2C5KOWK5FB1ETLMHFII8N0BJX0PN5&client_secret=QW2CZDT4IFFE1ZZBMM1LC
VDSUV40IOIDYBG3PUVHQU3HSY2H&v=20180605&ll=41.9683,-87.728&radius=750&limit=100'
```

Example Foursquare API url

We define a function and extract venue details for all Neighbourhoods in Chicago dataframe and Canada dataframe. A dataset of all venues associated with each Chicago city neighbourhood and Toronto city neighbourhood was created by recursively sending *get requests* to the Foursquare API.

The function getNearbyVenues() recursively sends a *get* requests to Foursquare for each neighbourhood that requests all nearby venues. While looping through each neighborhood from the chicago and canada dataset, the function appends each venue entry to a list and, after looping through each neighborhood, creates a DataFrame of all of the results. Included for each entry in the dataset are neighborhood name and gps coordinates, and venue name, gps coordinates and category.

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        result = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in result])
```

Extracting venue list for each neighbourhood

The resulting data frame for chicago neighbourhoods looks like:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Albany Park,Mayfair,North Mayfair,Ravenswood M...	41.9683	-87.728	Starbucks	41.968911	-87.728817	Coffee Shop
1	Albany Park,Mayfair,North Mayfair,Ravenswood M...	41.9683	-87.728	Lawrence Fish Market	41.968280	-87.726250	Seafood Restaurant
2	Albany Park,Mayfair,North Mayfair,Ravenswood M...	41.9683	-87.728	Marie's Pizza & Liquors	41.968132	-87.731533	Pizza Place
3	Albany Park,Mayfair,North Mayfair,Ravenswood M...	41.9683	-87.728	Ssyal Korean Restaurant and Ginseng House	41.968172	-87.733207	Korean Restaurant

The canada neighbourhoods data frame looks like:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Rouge,Malvern	43.806686	-79.194353	Wendy's	43.807448	-79.199056	Fast Food Restaurant
1	Highland Creek,Rouge Hill,Port Union	43.784535	-79.160497	Royal Canadian Legion	43.782533	-79.163085	Bar
2	Highland Creek,Rouge Hill,Port Union	43.784535	-79.160497	Scarborough Historical Society	43.788755	-79.162438	History Museum
3	Guildwood,Morningside,West Hill	43.763573	-79.188711	Swiss Chalet Rotisserie & Grill	43.767697	-79.189914	Pizza Place

Methodology

In this project we will direct our efforts on predicting which city would be better for an *Indian* origin family to reside. We will limit our analysis to availability of *Indian Restaurant* in each neighbourhood of *Chicago* and *Toronto* city.

As first step we have collected the required data: location and category details of every venue located within 750m in each neighbourhood.

Second step in our analysis will be creating *cluster groups* of locations using ***K-means*** algorithm. We will present a map, to visualize all cluster groups in different color on Chicago and Toronto city map respectively. We use ***Folium*** library to generate maps.

Third step in our analysis will be exploring *Indian Restaurants* from the resulting data frame with all venue locations. We filter only rows that have venue category value as "*Indian Restaurant"* and make them a new data frame. We also visualize the neighbourhood with *Indian Restaurant* on map. Then calculate the count of *Indian Restaurant* and group them by neighbourhood. A ***bar chart*** created with Neighbourhood and Count along x-axis and y-axis respectively.

Exploratory Data Analysis

To predict a better place, we here consider "Indian Restaurant" factor alone. So from the resulting data frame we use entries with venue category values as "Indian Restaurant". Once we create a new dataframe with venue details using **Foursquare API**, we find the count of venues listed for each neighbourhood.

Checking how many venues have been returned for each neighbourhood

```
chicago_venues.groupby('Neighbourhood').count()
```

Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Albany Park,Mayfair,North Mayfair,Ravenswood Manor	35	35	35	35	35	35
Altgeld Gardens,Eden Green,Golden Gate,Riverdale	1	1	1	1	1	1
Andersonville,Edgewater,Edgewater Beach,Edgewater Glen,Lakewood / Balmoral	45	45	45	45	45	45
Archer Heights	26	26	26	26	26	26

Number of venues extracted for each neighbourhood

One-Hot-Encoding Venue Categories

A one-hot-encoding representation of each entry was created using *Pandas "get_dummies()"* function. The result was a dataframe of venues where each venue category is represented by value '1' in the column of matching venue category.

```
# one hot encoding
chicago_onehot = pd.get_dummies(chicago_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
chicago_onehot['Neighbourhood'] = chicago_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [chicago_onehot.columns[-1]] + list(chicago_onehot.columns[:-1])
chicago_onehot = chicago_onehot[fixed_columns]

chicago_onehot.head()
```

	Neighbourhood	ATM	Adult Boutique	African Restaurant	American Restaurant	Antique Shop	Arcade	Arepa Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports	Auditorium	Auto Garage	Aut
0	Albany Park,Mayfair,North Mayfair,Ravenswood M...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Albany Park,Mayfair,North Mayfair,Ravenswood M...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	Albany Park,Mayfair,North Mayfair,Ravenswood M...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Chicago neighborhoods' one hot encoding

A data frame showing the top ten venue categories for each neighbourhood was created.

Lets create a new dataframe with top 10 venues in each neighbourhood

```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighbourhoods_venues_sorted = pd.DataFrame(columns=columns)
neighbourhoods_venues_sorted['Neighbourhood'] = chicago_grouped['Neighbourhood']

for ind in np.arange(chicago_grouped.shape[0]):
    neighbourhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(chicago_grouped.iloc[ind, :], num_top_venues)

neighbourhoods_venues_sorted.head()
```

	Neighbourhood	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	Albany Park,Mayfair,North Mayfair,Ravenswood M...	Mexican Restaurant	Pizza Place	Hookah Bar	Korean Restaurant	Dive Bar	Discount Store	Chinese Restaurant	Donut Shop	Taco Place	Fried Chicken Joint
1	Altgeld Gardens,Eden Green,Golden Gate,Riverdale	Grocery Store	Yoga Studio	Electronics Store	Food Truck	Food Court	Food & Drink Shop	Food	Flower Shop	Fish & Chips Shop	Financial or Legal Service
2	Andersonville,Edgewater,Edgewater Beach,Edgewa...	Gym	Sushi Restaurant	Wine Bar	Optical Shop	Mexican Restaurant	Bank	Indian Restaurant	Gym / Fitness Center	Diner	Hot Dog Joint

Chicago neighborhoods' top 10 venue categories

Predictive Modeling

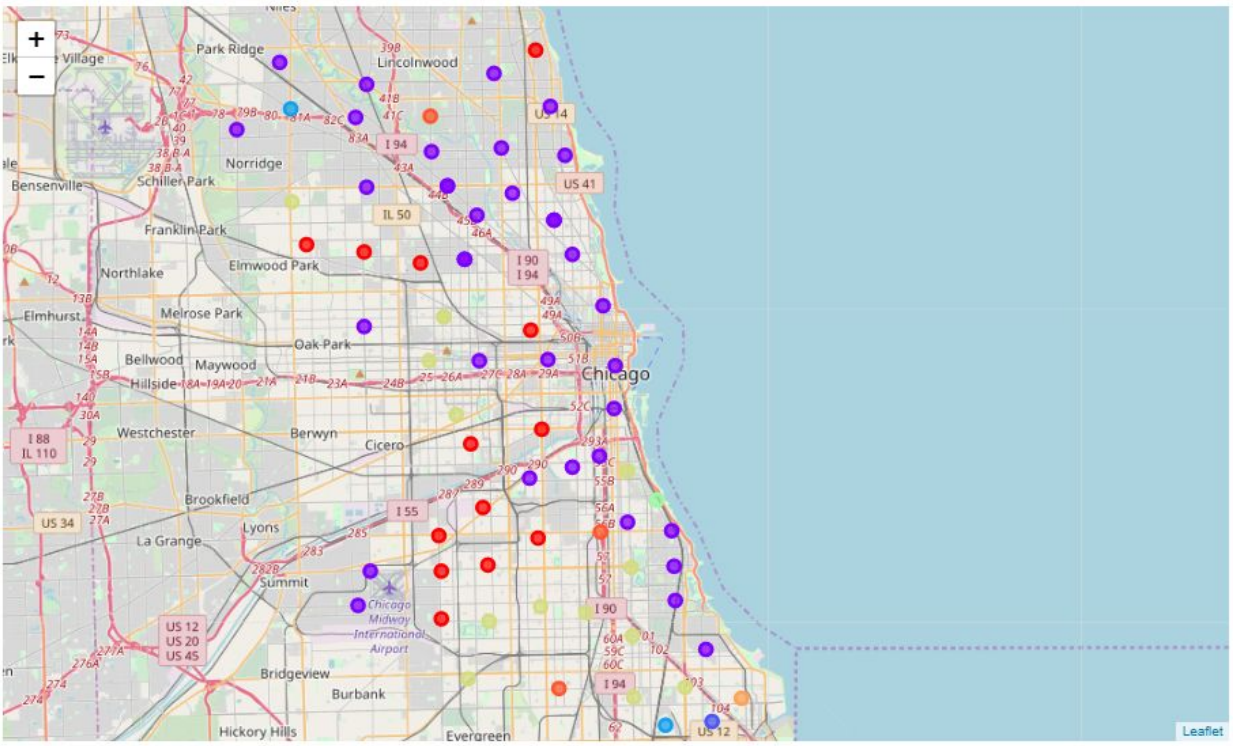
Scikit-learn K-Means clustering algorithm is used to determine similar neighborhoods based on mean value of venue categories. The image below shows the data being scaled and the K-Means model being created:

```
# set number of clusters
kclusters = 10

chicago_grouped_clustering = chicago_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(chicago_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

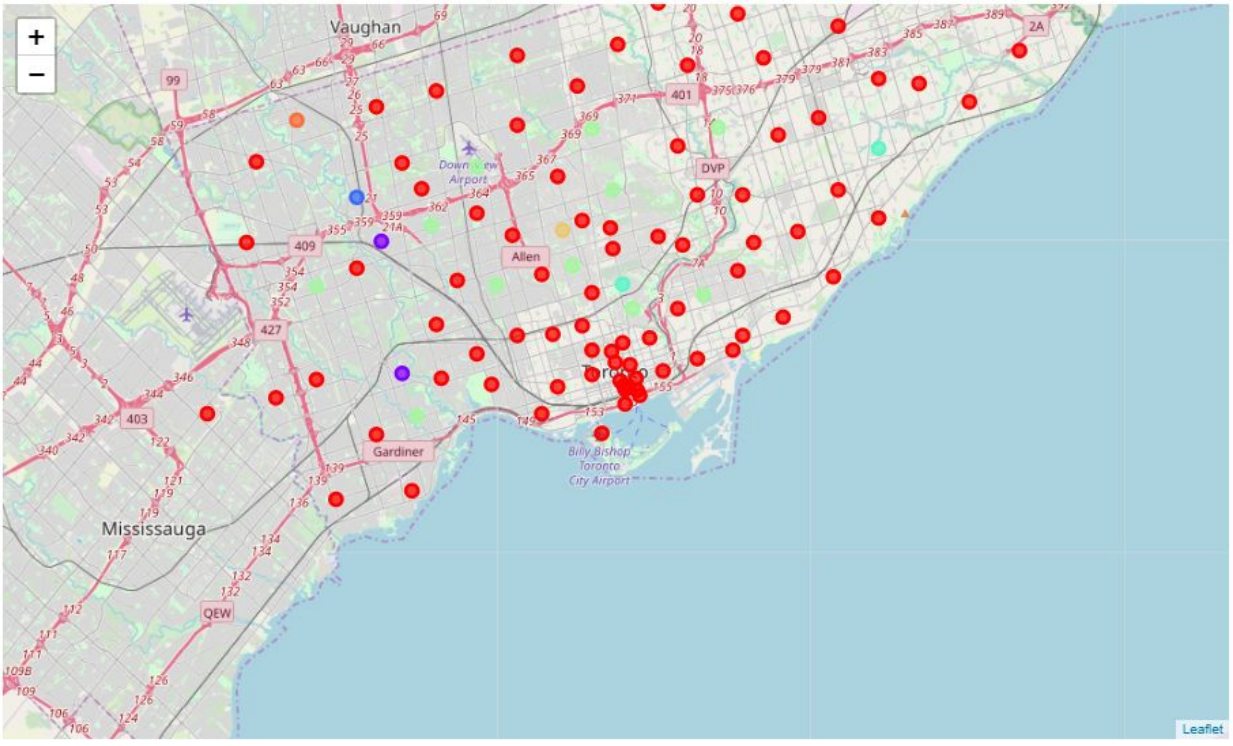
```
array([5, 1, 5, 7, 5, 5, 3, 5, 5, 5], dtype=int32)
```

Clustering neighbourhood data

A data frame is created by merging neighbourhood location data with cluster labels and top 10 venue categories. We then visualize the resulting cluster groups using different color for each cluster on map using Folium package.



Chicago cluster group map



Toronto cluster group map

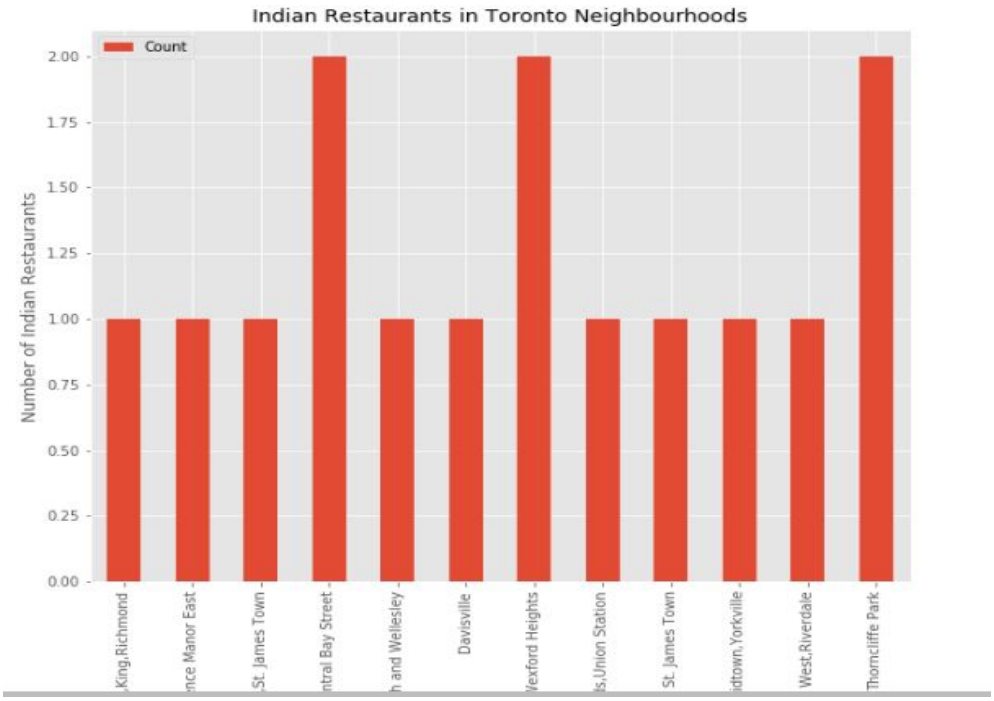
We now query venues dataframe resulted from FourSquare API Call, to extract only places that have category value as "*Indian Restaurant*". Then count the number of restaurants in each neighbourhood by applying count method.

```
canada_graph = canada_indian.groupby(['Neighbourhood'], axis = 0).count()
canada_graph.drop(['Neighbourhood Latitude','Neighbourhood Longitude','Venue Latitude','Venue Longitude','Venue Category'], axis =1, i
canada_graph.rename(columns={'Venue':'Count'}, inplace =True)
canada_graph
```

Neighbourhood	Count
Adelaide,King,Richmond	1
Bedford Park,Lawrence Manor East	1
Cabbagetown,St. James Town	1
Central Bay Street	2
Church and Wellesley	1
Davisville	1
Dorset Park,Scarborough Town Centre,Wexford Heights	2
Harbourfront East,Toronto Islands,Union Station	1
St. James Town	1
The Annex,North Midtown,Yorkville	1
The Danforth West,Riverdale	1
Thorncliffe Park	2

Toronto Indian restaurant count in each neighbourhood

Create a bar graph representing Neighbourhood names on x-axis and Number of Indian Restaurant on y-axis.



Toronto Indian Restaurant Count

Results and Discussion

In this study, we analyzed the total number of *Indian Restaurants* in *Chicago* and *Toronto* city using *Foursquare API* with its gps coordinates. I identified that 5 boroughs in Chicago city have *Indian Restaurants* where as in Toronto city, 6 boroughs have *Indian Restaurants* scattered around. I feel Toronto city has more boroughs that has *Indian Restaurant*. But still Chicago city has more number of *Indian Restaurant*. I built k-means clustering to cluster venues around each city. This model can be very helpful in helping an Indian origin family to choose a new location for residence. For example, it could help identify any specific venue category, plan for tourists travelling to these places, stakeholders who wish to open an Indian cuisine restaurant, etc.

Conclusion

Purpose of this project was to identify Chicago and Toronto areas that has more *Indian Restaurants* in order to aid stakeholders in narrowing down the search for optimal location to reside. By calculating the total number of *Indian Restaurants* from Foursquare data we have identified all neighbourhoods that has *Indian Restaurants.*Final decision on optimal residence location will be made by stakeholders on specific characteristics of neighbourhoods, taking into consideration additional factors like social and economic dynamics, population, housing price, crime data, commute, etc. These interactions data are obviously more difficult to extract and quantify, but if optimized, could bring significant improvements to the model.