# battle 2

August 20, 2020

# 1 Targeting Prime Location for Chinese Restaurant with K-Means Clustering

## 1.1 Introduction

This project will use various sources of data to group specific Toronto neighbourhoods into categories identifying their potential for establishing a successful chinese restaurant. The categories will be defined by aspects of the neighbourhood such as the number of chinese restaurants, household income, and amount of chinese residents.

## 1.2 Business Problem

A client wishes to open an authentic Chinese restaurant in the city of Toronto, which will evidently target a customer base majoritively made up of Chinese residents. The choice of which neighbourhood to establish the restaurant should consider competion, income, and number of Chinese residents.

### 1.2.1 Importing Libraries

[92]:
```
import pandas as pd
import numpy as np
```

### 1.2.2 Importing Social Demographic Data and Converting Into Dataframes

The social demographic of Toronto neighbourhoods is obtained from City of Toronto open data source: 'https://www.toronto.ca/city-government/data-research-maps/open-data/'

[93]:
```
# importing after-tax household income
income = pd.read_csv('toronto_income.csv')
df_income = pd.DataFrame(income)
df_income.drop(['Neighbourhood Id'], axis=1, inplace=True)

# importing the total population of Toronto neighbourhoods
pop = pd.read_csv('toronto_pop.csv')
df_pop = pd.DataFrame(pop)
df_pop.drop(['Neighbourhood Id'], axis=1, inplace=True)

# importing the population of chinese residents in Toronto neighbourhoods
```

```
chi = pd.read_csv('toronto_chi.csv')
df_chi_pop = pd.DataFrame(chi)
df_chi_pop.drop(['Neighbourhood Id'], axis=1, inplace=True)


print("{}, {}, {}".format(df_income.shape[1], df_pop.shape[1], df_chi_pop.
 ↪shape[1]))
```

```
4, 2, 2
```

### 1.2.3 Merging the Dataframes

[94]:
```
from functools import reduce
df = [df_income, df_pop, df_chi_pop]
df_merge = reduce(lambda left,right: pd.merge(left,right,on='Neighbourhood'),␣
 ↪df)
df_merge.drop(['Combined Indicators', 'Total Population_x'], axis =1, inplace␣
 ↪=True)
df_merge.rename(columns={'After-Tax Household Income': 'Household Income',␣
 ↪'Total Population_y': 'Total Pop', '   Chinese': 'Chinese Pop'},␣
 ↪inplace=True)
df_merge
```

[94]:
```
                        Neighbourhood  Household Income  Total Pop  \
0                West Humber-Clairville           59703.0    33312.0
1      Mount Olive-Silverstone-Jamestown           46986.0    32954.0
2           Thistletown-Beaumond Heights           57522.0    10360.0
3                        Rexdale-Kipling           51194.0    10529.0
4                       Elms-Old Rexdale           49425.0     9456.0
..                                   …                 …          …
135                            West Hill           46803.0    27392.0
136                               Woburn           47908.0    53485.0
137                        Eglinton East           42790.0    22776.0
138                  Scarborough Village           40181.0    16724.0
139                            Guildwood           67678.0     9917.0

     Chinese Pop
0          470.0
1          285.0
2          110.0
3          165.0
4          105.0
..           …
135        685.0
136       3715.0
137        895.0
138        390.0
```

```
139        275.0

[140 rows x 4 columns]
```

### 1.2.4 Importing Geographical Coordinates

The geographical coordinates can be imported from: https://cocl.us/Geospatial_data

[95]:
```
df_geo = pd.read_csv("https://cocl.us/Geospatial_data")
df_geo
```

[95]:
```
    Postal Code   Latitude  Longitude
0           M1B  43.806686 -79.194353
1           M1C  43.784535 -79.160497
2           M1E  43.763573 -79.188711
3           M1G  43.770992 -79.216917
4           M1H  43.773136 -79.239476
..          …        …        …
98          M9N  43.706876 -79.518188
99          M9P  43.696319 -79.532242
100         M9R  43.688905 -79.554724
101         M9V  43.739416 -79.588437
102         M9W  43.706748 -79.594054

[103 rows x 3 columns]
```

### 1.2.5 Scraping Wikipedia for Postal Codes

The list of postal codes for each neighbourhood can be scraped from wikipedia: https://en.wikipedia.org/w/index.php?title=List_of_postal_codes_of_Canada:_M&oldid=945633050

[96]:
```
df_pc = pd.read_html('https://en.wikipedia.org/w/index.php?
 ↪title=List_of_postal_codes_of_Canada:_M&oldid=945633050')[0]
df_pc = df_pc[df_pc.Borough!='Not assigned']
df_pc = pd.merge(df_pc, df_geo, left_on='Postcode', right_on='Postal Code').
 ↪drop(['Postal Code', 'Postcode', 'Borough'], axis=1)
df_pc = pd.merge(df_pc, df_merge, on='Neighbourhood', how='left').
 ↪dropna(how='any', axis=0).reset_index(drop=True)
df_pc.head()
```

[96]:
```
      Neighbourhood   Latitude  Longitude  Household Income  Total Pop  \
0  Victoria Village  43.725882 -79.315572           43743.0    17510.0
1             Rouge  43.806686 -79.194353           72784.0    46496.0
2           Malvern  43.806686 -79.194353           53425.0    43794.0
3    Highland Creek  43.784535 -79.160497           87321.0    12494.0
4   Flemingdon Park  43.725900 -79.340923           43511.0    21933.0
```

```
   Chinese Pop
0        730.0
1       2100.0
2       3275.0
3        955.0
4       1015.0
```

### 1.2.6 Calculate Percentage of Chinese Residents for Each Neighbourhood

[97]:
```
df_pc['Pop Percent of Chinese'] = df_pc['Chinese Pop'] / df_pc['Total Pop'] *␣
 ↪100
df_pc.drop(['Total Pop', 'Chinese Pop'], axis=1, inplace=True)
df_pc.head()
```

[97]:
```
      Neighbourhood   Latitude  Longitude  Household Income  \
0  Victoria Village  43.725882 -79.315572           43743.0
1             Rouge  43.806686 -79.194353           72784.0
2           Malvern  43.806686 -79.194353           53425.0
3    Highland Creek  43.784535 -79.160497           87321.0
4    Flemingdon Park  43.725900 -79.340923           43511.0

   Pop Percent of Chinese
0                4.169046
1                4.516518
2                7.478193
3                7.643669
4                4.627730
```

### 1.2.7 Folium Map

[99]:
```
# get location of Toronto using geopy
!pip install geopy
from geopy.geocoders import Nominatim
address = 'Toronto'
geolocator = Nominatim(user_agent='to_explorer')
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

print("The geographical coordinates of Toronto are {}, {}".format(latitude,␣
 ↪longitude))
```

```
Requirement already satisfied: geopy in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (2.0.0)
Requirement already satisfied: geographiclib<2,>=1.49 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from geopy) (1.50)
The geographical coordinates of Toronto are 43.6534817, -79.3839347
```

[100]:

```python
!python -m pip install folium
import folium
map_Toronto = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, neighbourhood in zip(df_pc['Latitude'], df_pc['Longitude'],␣
 ↪df_pc['Neighbourhood']):
  label = '{}'.format(neighbourhood)
  label = folium.Popup(label)
  folium.CircleMarker(
      [lat,lng],
      radius=8,
      color='blue',
      popup=label,
      fill_color='#3186cc',
      fill_opacity=0.7,
      fill=True

  ).add_to(map_Toronto)

map_Toronto
```

```
Requirement already satisfied: folium in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (0.11.0)
Requirement already satisfied: requests in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from folium) (2.24.0)
Requirement already satisfied: jinja2>=2.9 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from folium) (2.11.2)
Requirement already satisfied: numpy in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from folium) (1.18.5)
Requirement already satisfied: branca>=0.3.0 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from folium) (0.4.1)
Requirement already satisfied: chardet<4,>=3.0.2 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from requests->folium)
(3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from requests->folium)
(2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from requests->folium)
(1.25.9)
Requirement already satisfied: idna<3,>=2.5 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from requests->folium)
(2.10)
Requirement already satisfied: MarkupSafe>=0.23 in
./anaconda2/envs/p36workshop/lib/python3.6/site-packages (from
jinja2>=2.9->folium) (1.1.1)
```

[100]: <folium.folium.Map at 0x7f3e19d96630>

### 1.2.8 Obtain Venue Categories with FourSquare

Foursquare API is used to obtain venue categories for each neighbourhood.

[101]:
```
# define Foursquare API credentials and version
CLIENT_ID = 'ET3KHGBHTXDBZVWUM3UT5UXO5SGGIUW4LP2GADYNGF3D1L4W' # your␣
↪Foursquare ID
CLIENT_SECRET = 'CLY1WZIGAIWLN42TZ1ZDFZYCJMC0GDV1JJNZCTOTMSYEXBNH' # your␣
↪Foursquare Secret
VERSION = '20200801' # Foursquare API version
```

### 1.2.9 Obtain Top 100 Venues within 1500 Metre Radius of Toronto

[102]:
```
import requests
LIMIT = 100
radius = 1500
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
    )

def getNearbyVenues(names, latitudes, longitudes, radius=1500):
  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
```

```
    results = 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 results
        ])

  nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in␣
↪venue_list])
  nearby_venues.columns = ['Neighbourhood',
                           'Neighbourhood Latitude',
                           'Neighbourhood Longitude',
                           'Venue',
                           'Venue Latitude',
                           'Venue Longitude',
                           'Venue Category']

  return(nearby_venues)
```

[103]:
```
# apply the function to create a new dataframe for each neighbourhood
toronto_venues = getNearbyVenues(names=df_pc['Neighbourhood'],
                                 latitudes=df_pc['Latitude'],
                                 longitudes=df_pc['Longitude'])

print(toronto_venues.shape)
```

```
Victoria Village
Rouge
Malvern
Highland Creek
Flemingdon Park
Humewood-Cedarvale
Markland Wood
Guildwood
Morningside
West Hill
The Beaches
Woburn
Hillcrest Village
Bathurst Manor
Thorncliffe Park
```

```
Scarborough Village
Henry Farm
Little Portugal
Ionview
Kennedy Park
Bayview Village
Oakridge
Humber Summit
Cliffcrest
Mount Dennis
Weston
Dorset Park
Forest Hill North
Willowdale West
Roncesvalles
Agincourt North
Milliken
New Toronto
Alderwood
Long Branch
(1818, 7)
```

[104]:
```
toronto_venues.groupby('Neighbourhood').count()
```

[104]:
```
                     Neighbourhood Latitude  Neighbourhood Longitude  Venue  \
Neighbourhood
Agincourt North                          77                       77     77
Alderwood                                45                       45     45
Bathurst Manor                           40                       40     40
Bayview Village                          15                       15     15
Cliffcrest                               40                       40     40
Dorset Park                              56                       56     56
Flemingdon Park                          85                       85     85
Forest Hill North                       100                      100    100
Guildwood                                32                       32     32
Henry Farm                               63                       63     63
Highland Creek                           10                       10     10
Hillcrest Village                        53                       53     53
Humber Summit                            18                       18     18
Humewood-Cedarvale                       91                       91     91
Ionview                                  38                       38     38
Kennedy Park                             38                       38     38
Little Portugal                         100                      100    100
Long Branch                              45                       45     45
Malvern                                  35                       35     35
Markland Wood                            39                       39     39
Milliken                                 77                       77     77
```

| | | | |
|---|---|---|---|
| Morningside | 32 | 32 | 32 |
| Mount Dennis | 39 | 39 | 39 |
| New Toronto | 39 | 39 | 39 |
| Oakridge | 36 | 36 | 36 |
| Roncesvalles | 100 | 100 | 100 |
| Rouge | 35 | 35 | 35 |
| Scarborough Village | 33 | 33 | 33 |
| The Beaches | 100 | 100 | 100 |
| Thorncliffe Park | 93 | 93 | 93 |
| Victoria Village | 54 | 54 | 54 |
| West Hill | 32 | 32 | 32 |
| Weston | 52 | 52 | 52 |
| Willowdale West | 40 | 40 | 40 |
| Woburn | 36 | 36 | 36 |

| | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|
| Neighbourhood | | | |
| Agincourt North | 77 | 77 | 77 |
| Alderwood | 45 | 45 | 45 |
| Bathurst Manor | 40 | 40 | 40 |
| Bayview Village | 15 | 15 | 15 |
| Cliffcrest | 40 | 40 | 40 |
| Dorset Park | 56 | 56 | 56 |
| Flemingdon Park | 85 | 85 | 85 |
| Forest Hill North | 100 | 100 | 100 |
| Guildwood | 32 | 32 | 32 |
| Henry Farm | 63 | 63 | 63 |
| Highland Creek | 10 | 10 | 10 |
| Hillcrest Village | 53 | 53 | 53 |
| Humber Summit | 18 | 18 | 18 |
| Humewood-Cedarvale | 91 | 91 | 91 |
| Ionview | 38 | 38 | 38 |
| Kennedy Park | 38 | 38 | 38 |
| Little Portugal | 100 | 100 | 100 |
| Long Branch | 45 | 45 | 45 |
| Malvern | 35 | 35 | 35 |
| Markland Wood | 39 | 39 | 39 |
| Milliken | 77 | 77 | 77 |
| Morningside | 32 | 32 | 32 |
| Mount Dennis | 39 | 39 | 39 |
| New Toronto | 39 | 39 | 39 |
| Oakridge | 36 | 36 | 36 |
| Roncesvalles | 100 | 100 | 100 |
| Rouge | 35 | 35 | 35 |
| Scarborough Village | 33 | 33 | 33 |
| The Beaches | 100 | 100 | 100 |
| Thorncliffe Park | 93 | 93 | 93 |

```
Victoria Village                   54               54               54
West Hill                          32               32               32
Weston                             52               52               52
Willowdale West                    40               40               40
Woburn                             36               36               36
```

#### 1.2.10 Apply One Hot Encoding for the Analysis of Venue Categories for Each Neighbourhood

[105]:
```python
# one hot encoding
toronto_onehot = pd.get_dummies(toronto_venues[['Venue Category']], prefix='',␣
 ↪prefix_sep='')

# add neighbourhood column back to data frame
toronto_onehot['Neighbourhood'] = toronto_venues['Neighbourhood']

# move neighbourhood column to the first column
cols = list(toronto_onehot)
cols.insert(0, cols.pop(cols.index('Neighbourhood')))
toronto_onehot = toronto_onehot.loc[:,cols]
toronto_onehot
```

[105]:
```
          Neighbourhood  Afghan Restaurant  American Restaurant  Amphitheater  \
0      Victoria Village                  0                    0             0
1      Victoria Village                  0                    0             0
2      Victoria Village                  0                    0             0
3      Victoria Village                  0                    0             0
4      Victoria Village                  0                    0             0
…                   …                  …                    …           …
1813        Long Branch                  0                    0             0
1814        Long Branch                  0                    0             0
1815        Long Branch                  0                    0             0
1816        Long Branch                  0                    0             0
1817        Long Branch                  0                    0             0

      Antique Shop  Art Gallery  Art Museum  Arts & Crafts Store  \
0                0            0           0                    0
1                0            0           0                    0
2                0            0           0                    0
3                0            0           0                    0
4                0            0           0                    0
…             …          …          …                    …
1813             0            0           0                    0
1814             0            0           0                    0
1815             0            0           0                    0
1816             0            0           0                    0
1817             0            0           0                    0
```

```
      Asian Restaurant  Athletics & Sports  …  Vietnamese Restaurant  \
0                    0                   0  …                      0
1                    0                   0  …                      0
2                    0                   0  …                      0
3                    0                   0  …                      0
4                    0                   0  …                      0
…                  …                 …   …                    …
1813                 0                   0  …                      0
1814                 0                   0  …                      0
1815                 0                   0  …                      0
1816                 0                   0  …                      0
1817                 0                   0  …                      0

      Volleyball Court  Warehouse Store  Whisky Bar  Wings Joint  \
0                    0                0           0            0
1                    0                0           0            0
2                    0                0           0            0
3                    0                0           0            0
4                    0                0           0            0
…                  …              …         …          …
1813                 0                0           0            0
1814                 0                0           0            0
1815                 0                0           0            0
1816                 0                0           0            0
1817                 0                0           0            0

      Women's Store  Xinjiang Restaurant  Yoga Studio  Zoo  Zoo Exhibit
0                 0                    0            0    0            0
1                 0                    0            0    0            0
2                 0                    0            0    0            0
3                 0                    0            0    0            0
4                 0                    0            0    0            0
…               …                  …          …  …          …
1813              0                    0            0    0            0
1814              0                    0            0    0            0
1815              0                    0            0    0            0
1816              0                    0            0    0            0
1817              0                    0            0    0            0

[1818 rows x 216 columns]
```

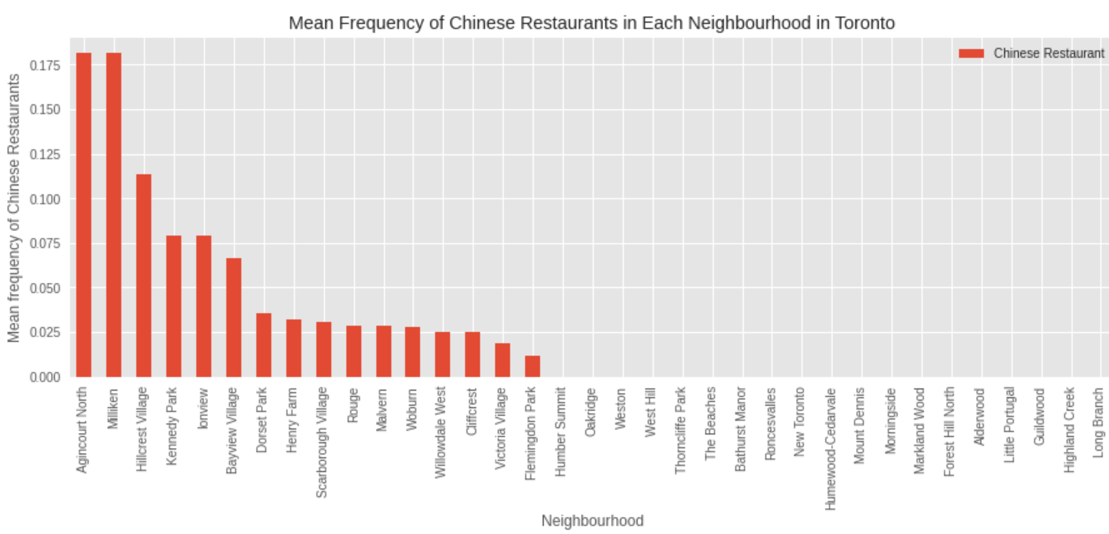
### 1.2.11 Calculate the Mean Frequency of Chinese Restaurants for Each Neighbourhood and Produce Bar Graph

[106]:
```
toronto_grouped = toronto_onehot.groupby('Neighbourhood').mean().reset_index()
toronto_grouped = toronto_grouped[['Neighbourhood', 'Chinese Restaurant']]
toronto_grouped.set_index('Neighbourhood', inplace=True)
toronto_grouped
```

[106]:

| | Chinese Restaurant |
|---|---|
| Neighbourhood | |
| Agincourt North | 0.181818 |
| Alderwood | 0.000000 |
| Bathurst Manor | 0.000000 |
| Bayview Village | 0.066667 |
| Cliffcrest | 0.025000 |
| Dorset Park | 0.035714 |
| Flemingdon Park | 0.011765 |
| Forest Hill North | 0.000000 |
| Guildwood | 0.000000 |
| Henry Farm | 0.031746 |
| Highland Creek | 0.000000 |
| Hillcrest Village | 0.113208 |
| Humber Summit | 0.000000 |
| Humewood-Cedarvale | 0.000000 |
| Ionview | 0.078947 |
| Kennedy Park | 0.078947 |
| Little Portugal | 0.000000 |
| Long Branch | 0.000000 |
| Malvern | 0.028571 |
| Markland Wood | 0.000000 |
| Milliken | 0.181818 |
| Morningside | 0.000000 |
| Mount Dennis | 0.000000 |
| New Toronto | 0.000000 |
| Oakridge | 0.000000 |
| Roncesvalles | 0.000000 |
| Rouge | 0.028571 |
| Scarborough Village | 0.030303 |
| The Beaches | 0.000000 |
| Thorncliffe Park | 0.000000 |
| Victoria Village | 0.018519 |
| West Hill | 0.000000 |
| Weston | 0.000000 |
| Willowdale West | 0.025000 |
| Woburn | 0.027778 |

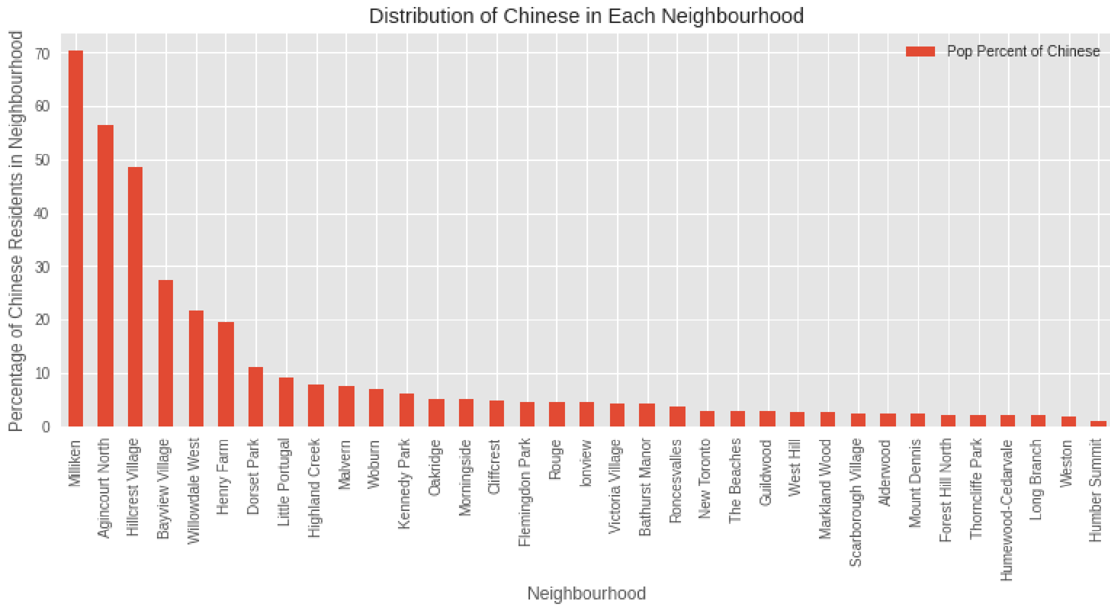
[107]:
```python
# sort by descending order
toronto_grouped.sort_values(by='Chinese Restaurant', ascending=False, inplace=True)

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot')
toronto_grouped.plot(kind='bar', figsize=(15,5))

plt.title('Mean Frequency of Chinese Restaurants in Each Neighbourhood in Toronto')
plt.xlabel('Neighbourhood')
plt.ylabel('Mean frequency of Chinese Restaurants')

plt.show()
```




### 1.2.12 Produce Bar Graph for the Percentage of Chinese Residents In Total Population for Each Neighbourhood

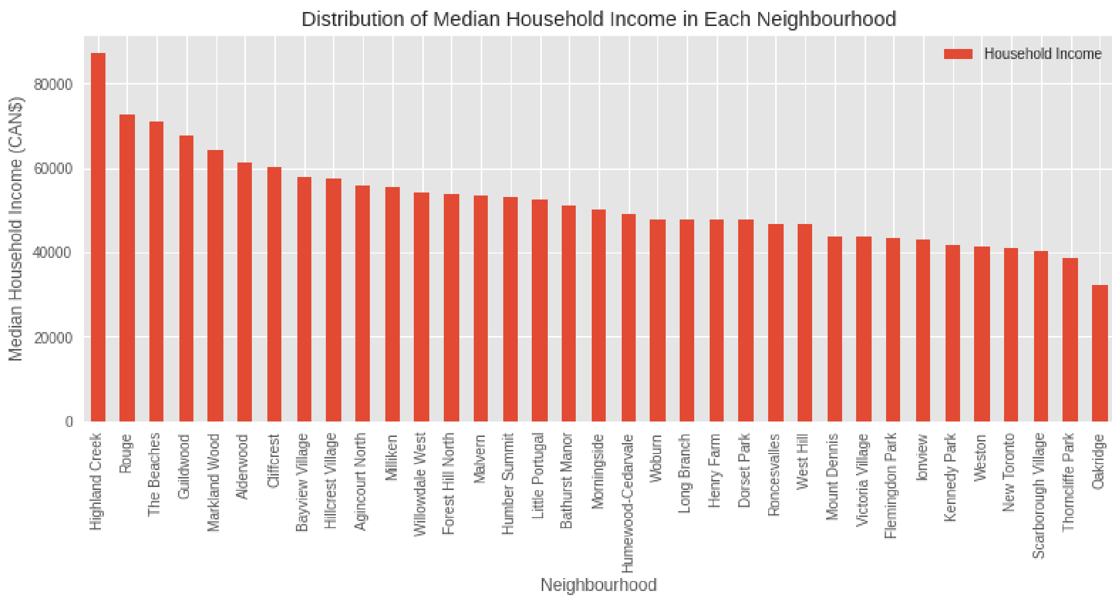
[108]:
```python
df_chi_pop = df_pc[['Neighbourhood', 'Pop Percent of Chinese']].set_index('Neighbourhood')

# sort by descending order
df_chi_pop.sort_values(by='Pop Percent of Chinese', ascending=False, inplace=True)

#plot bar graph
```

```
df_chi_pop.plot(kind='bar', figsize=(13,5))
plt.title('Distribution of Chinese in Each Neighbourhood')
plt.xlabel('Neighbourhood')
plt.ylabel('Percentage of Chinese Residents in Neighbourhood')

plt.show()
```




### 1.2.13 Produce Bar Graph for Median Income per Household for Each Neighbourhood

[109]:
```
df_income = df_pc[['Neighbourhood', 'Household Income']].
 ↪set_index('Neighbourhood')

# sort by descending order
df_income.sort_values(by='Household Income', ascending=False, inplace=True)

#plot bar graph
df_income.plot(kind='bar', figsize=(13,5))
plt.title('Distribution of Median Household Income in Each Neighbourhood')
plt.xlabel('Neighbourhood')
plt.ylabel('Median Household Income (CAN$)')

plt.show()
```




### 1.2.14 Merge Into a Final Database

[110]:
```
# merge df with toronto_grouped
df_final = pd.merge(df_pc, toronto_grouped, on='Neighbourhood')
df_final.head()
```

[110]:
```
        Neighbourhood   Latitude  Longitude  Household Income  \
0    Victoria Village  43.725882 -79.315572           43743.0
1               Rouge  43.806686 -79.194353           72784.0
2             Malvern  43.806686 -79.194353           53425.0
3      Highland Creek  43.784535 -79.160497           87321.0
4     Flemingdon Park  43.725900 -79.340923           43511.0

   Pop Percent of Chinese  Chinese Restaurant
0                4.169046            0.018519
1                4.516518            0.028571
2                7.478193            0.028571
3                7.643669            0.000000
4                4.627730            0.011765
```

### 1.2.15 Clustering of Neighbourhoods

**Normalise Final Dataset**

[111]:
```
from sklearn.preprocessing import StandardScaler
X = df_final.values[:,3:]
X = np.nan_to_num(X)
Cluster = StandardScaler().fit_transform(X)
```

```
Cluster
```

[111]:
```
array([[-0.76318401, -0.38981414, -0.19312015],
       [ 1.89181483, -0.36812483,  0.02175792],
       [ 0.12196797, -0.18325575,  0.02175792],
       [ 3.22082273, -0.1729267 , -0.58894816],
       [-0.78439402, -0.36118291, -0.33748095],
       [-0.25953784, -0.52186104, -0.58894816],
       [ 1.11591265, -0.49627376, -0.58894816],
       [ 1.4250119 , -0.47695494, -0.58894816],
       [-0.18484571, -0.33535299, -0.58894816],
       [-0.48343137, -0.49395114, -0.58894816],
       [ 1.72478605, -0.47205102, -0.58894816],
       [-0.38240958, -0.21648344,  0.00479386],
       [ 0.51115327,  2.38177002,  1.83083063],
       [-0.09278332, -0.39246963, -0.58894816],
       [-1.22925557, -0.5214099 , -0.58894816],
       [-1.08883072, -0.50448473,  0.05877041],
       [-0.40517377,  0.57271454,  0.08961415],
       [ 0.03913925, -0.07434681, -0.58894816],
       [-0.83376213, -0.37549125,  1.09852916],
       [-0.94301194, -0.27274741,  1.09852916],
       [ 0.54278543,  1.05954145,  0.83603269],
       [-1.82953526, -0.33219729, -0.58894816],
       [ 0.10798034, -0.58217764, -0.58894816],
       [ 0.75817668, -0.35038273, -0.05458034],
       [-0.75888716, -0.50769266, -0.58894816],
       [-0.98140936, -0.53382478, -0.58894816],
       [-0.40782502,  0.03524628,  0.17443444],
       [ 0.17252458, -0.52087855, -0.58894816],
       [ 0.19519734,  0.69706291, -0.05458034],
       [-0.47611758, -0.41660689, -0.58894816],
       [ 0.34759853,  2.87909816,  3.29736323],
       [ 0.30837831,  3.74630299,  3.29736323],
       [-1.02684631, -0.46490476, -0.58894816],
       [ 0.85124471, -0.5076418 , -0.58894816],
       [-0.4032539 , -0.52624686, -0.58894816]])
```

[112]:
```
df_normalised= pd.DataFrame(Cluster)
df_normalised.rename(columns={0:'Household Income', 1:'% Chinese', 2:'Num of␣
 ↪Chinese Restaurants'}, inplace=True)
df_normalised.head()
```

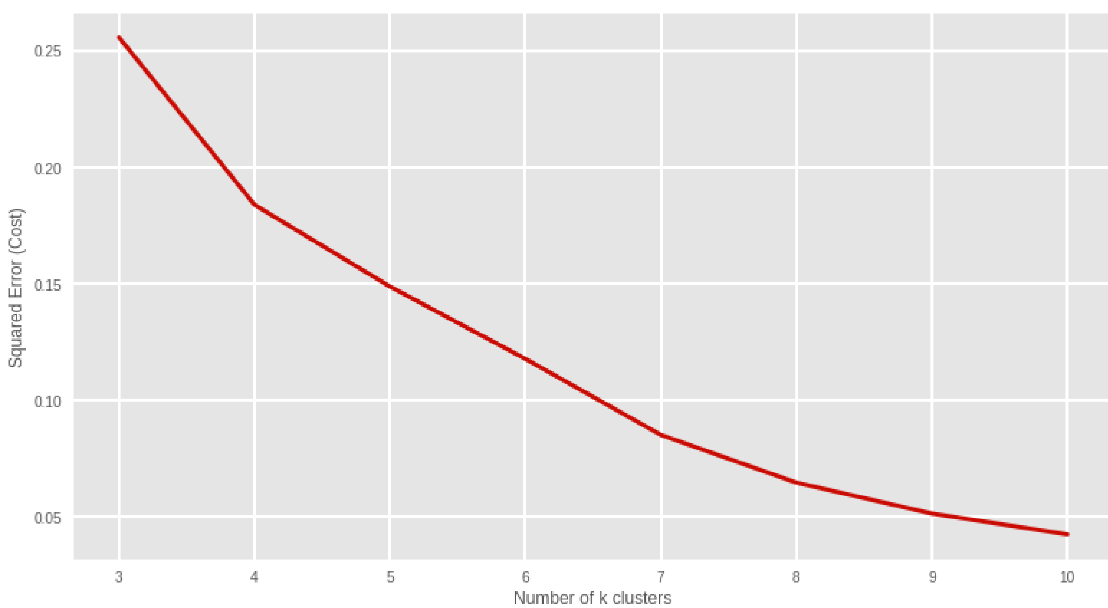
[112]:

| | Household Income | % Chinese | Num of Chinese Restaurants |
|---|---|---|---|
| 0 | -0.763184 | -0.389814 | -0.193120 |
| 1 | 1.891815 | -0.368125 | 0.021758 |
| 2 | 0.121968 | -0.183256 | 0.021758 |

| | | | |
|---|---|---|---|
| 3 | 3.220823 | -0.172927 | -0.588948 |
| 4 | -0.784394 | -0.361183 | -0.337481 |

#### 1.2.16 Calculate Optimal Number of Clusters for K-Means Clustering Process Via Elbow Method

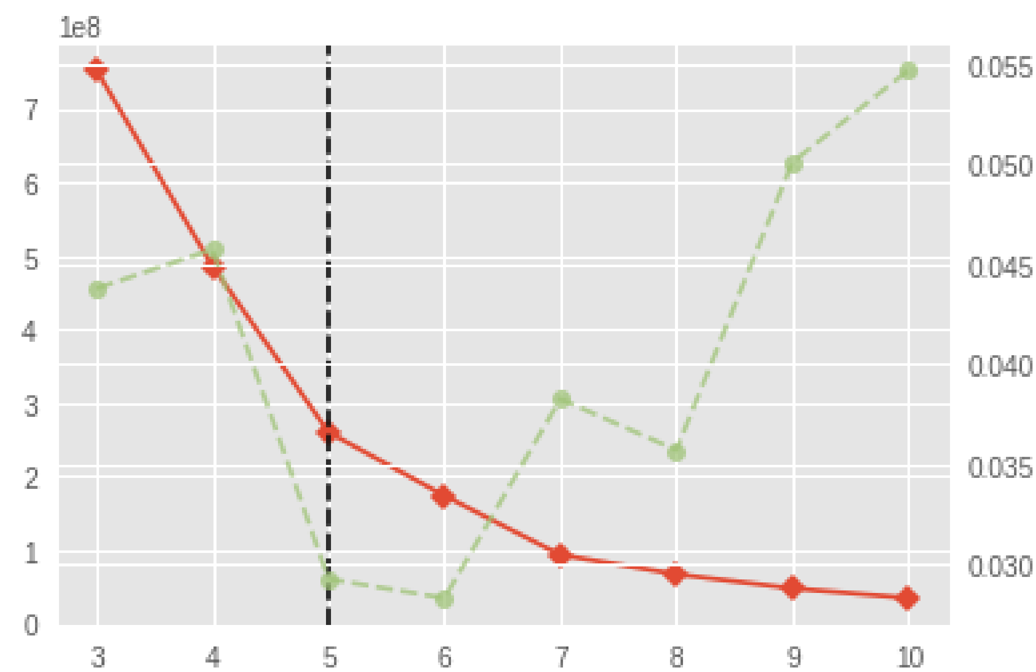
[114]:
```
from sklearn.cluster import KMeans

error_cost=[]

for i in range(3, 11):
  KM = KMeans(n_clusters=i, max_iter=100)
  try:
    KM.fit(df_normalised)

  except ValueError:
    print('Error on line', i)

  # calculate squared error for the clustered points
  error_cost.append(KM.inertia_ / 100)

# plot the K values against the squared error cost
plt.figure(figsize=(13,7))
plt.plot(range(3,11), error_cost, color='r', linewidth=3)
plt.xlabel('Number of k clusters')
plt.ylabel('Squared Error (Cost)')
plt.grid(color='white', linestyle='-', linewidth=2)

plt.show()
```



[115]:
```python
from yellowbrick.cluster import KElbowVisualizer

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(3,11))

visualizer.fit(X)
visualizer
```

```
/home/green/anaconda2/envs/p36workshop/lib/python3.6/site-
packages/sklearn/base.py:213: FutureWarning: From version 0.24, get_params will
raise an AttributeError if a parameter cannot be retrieved as an instance
attribute. Previously it would return None.
  FutureWarning)
```

[115]:
```
KElbowVisualizer(ax=<matplotlib.axes._subplots.AxesSubplot object at
0x7f3e199b7160>,
                 k=None, model=None)
```




[116]:
```python
# set number of clusters
kclusters = 5

# run k-means clustering
```

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

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

[116]: array([1, 0, 3, 0, 1, 3, 0, 0, 3, 3], dtype=int32)

### 1.2.17 Application of K-Means Clustering of Neighbourhoods

[117]:
```
df_normalised.drop(['Household Income', '% Chinese'], axis=1, inplace=True)
df_clustered = pd.merge(df_pc, df_normalised, left_index=True, right_index=True)
df_clustered.insert(0, 'Cluster Label', kmeans.labels_)
```

### 1.2.18 Create Map of Clusters

[118]:
```
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# create map
map_clusters = folium.Map(location=[latitude,longitude], zoom_start=11)

# set color schemes for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df_clustered['Latitude'],␣
↪df_clustered['Longitude'], df_clustered['Neighbourhood'],␣
↪df_clustered['Cluster Label']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

[118]: <folium.folium.Map at 0x7f3e19834940>

### 1.2.19 Database of Each Cluster

Each cluster can be summarised by taking the mean of each feature

[119]:
```
df_clustered.groupby('Cluster Label').mean()
```

[119]:
```
                Latitude  Longitude  Household Income  Pop Percent of Chinese  \
Cluster Label
0              43.712847 -79.326213      70739.833333                3.754950
1              43.702079 -79.378180      40520.500000                3.173783
2              43.811422 -79.310869      56346.333333               58.513507
3              43.718483 -79.357954      51104.083333                4.254163
4              43.760245 -79.328692      48715.000000               14.996728

               Num of Chinese Restaurants
Cluster Label
0                               -0.487164
1                               -0.427071
2                                2.808519
3                               -0.444047
4                                0.540427
```