1. On the first page of your submission, create one plot that communicates your recommendation and your reasoning for it.

- Based on clustering analysis, the **Upper East Side-Carnegie Hill (i.e., Carnegie Hill)** area was identified as a prime location (Green) for a new Trader Joe's.
    - The closest Trader Joe's (Blue) is at 405 E 59th St, New York, NY 10022. According to Google Maps, it would take 13 minutes by driving and 26 minutes on public transport for the customers to get there.
- **Customer Potential**:
    - These neighbourhoods exhibit a high "People per Shop" ratio (291.5), suggesting a strong potential to attract customers from existing competitors.
    - 25.4% of the population falls within the target age range of 25-44 (Reuter, 2021), with a slightly higher median age of 47.9.
- **Demographic Factors**: A median household income of $302,939 indicates a high purchasing power among potential customers, exceeding the target income level of $80,000 (Reuter, 2021).
- **Competitive Landscape**: The area has a high concentration of Stoop Line Stands of 51 (Red), Trader Joe's can differentiate itself with its unique product offerings and pricing strategy.
- **Drawback**: The new proposed store is not located within the FRESH Food Stores Zoning Boundaries, which means that potential development, expansion, or renovation of a new Trader Joe's may not be eligible for tax incentives.

Other indicators

- The Upper East Side-Carnegie Hill neighbourhood has a predominantly non-Hispanic/Latino population (92.9%), with 44.1% male and 55.9% female, indicating a slight female skew.
- The neighbourhood has a relatively high employment rate of 61.2%, but only 10% of employed residents earn within the $75,000-$99,000 income bracket. The majority (97.9%) of employed individuals have health insurance coverage.
- 51.4% of the population lives in families, suggesting a strong family-oriented community.
- The median household income is significantly higher than the national average, standing at $302,939. The median family income is also substantial at $200,000, indicating a strong economic foundation in the area.
- The neighbourhood has a high median house price ($2,000,000) and rent ($2,251), with limited parking availability of 37.2 people per parking while 35.3% owned a car. Occupancy rates are high (77.2%), indicating strong demand for housing in the area.

2. Show us how you came to your recommendation. Include why you used certain data, visualizations, data cleaning, etc. Discuss any assumptions made and clearly cite data and other public information sources used.

The source code is available in Appendix B.

The primary goal of this analysis is to identify the optimal location at the NTA level (Neighborhood Tabulation Area) within New York City's five boroughs for establishing a new Trader Joe's supermarket. To achieve this objective, two primary datasets were used:

1. **The list of legally operating businesses in New York (DCWP, 2024)**:
    a. The dataset was filtered to include only businesses operating in New York City with a Neighborhood Tabulation Area (NTA) value, resulting in 137,476 companies.
    b. The dataset does not contain the list of Trade Joe's stores.
2. **The demographic profiles of five years estimated between 2012 and 2016 for Neighborhood Tabulation Areas (NTAs) in New York City from 2012 to 2016 (DCP, 2020)**:
    a. This includes data on demographics, economics, housing, and sociology for each NTA.
    b. Please note that these estimates may not reflect the most recent trends, and utilising more up-to-date data is highly recommended to capture changes that have occurred since 2020.

The demographic profiles contain 4,831 features, including estimates, margins of error, coefficients of variation, percentages, and percentage margins of error. To streamline the analysis and identify key opportunities, I filtered the data and focused on the most relevant variables.

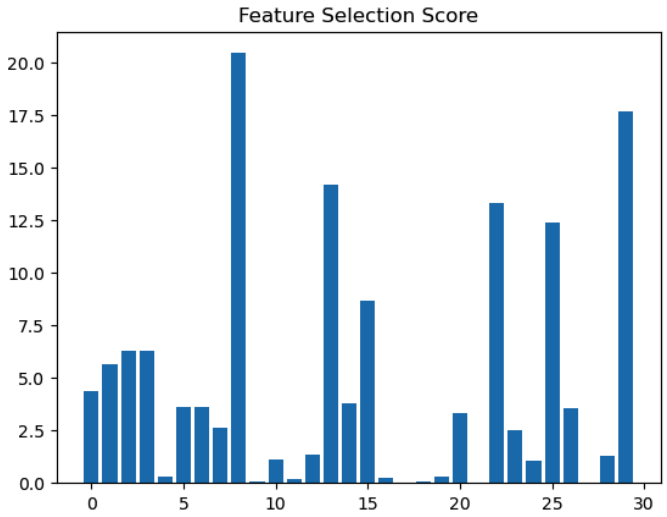
First, I determined the opportunity indicator for this scenario based on two primary factors:

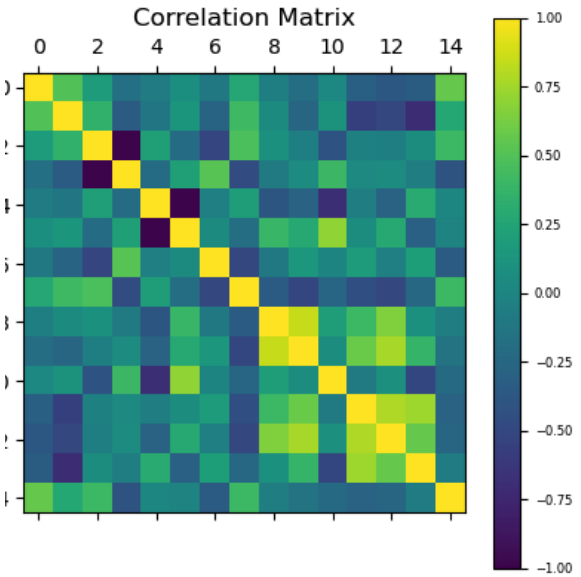
1. **Demographic Alignment**: We focused on neighbourhoods with a high concentration of Trader Joe's target demographic, specifically individuals aged 25-44 (Reuter, 2021).
2. **Competitive Landscape**: To assess competition, I calculated the total number of stores within the "Stoop Line Stand" and "General Vendor" industries per NTA. A lower number of competing stores would indicate a less competitive market.

To quantify the potential customer base in each neighbourhood, I created a "**People_Per_Shop**" variable. This metric was calculated by dividing the population of residents aged 25-44 by the number of competitive stores. A higher People_Per_Shop value suggests a denser potential customer base and a greater opportunity for a new Trader Joe's location.

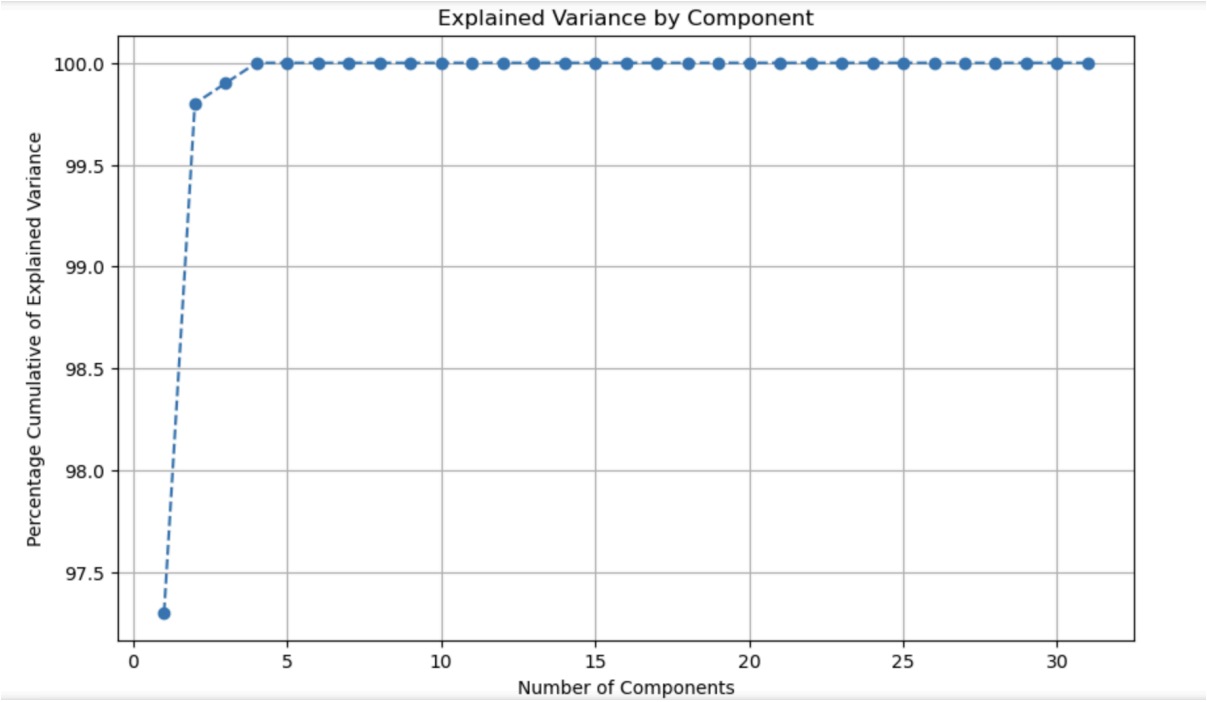
During the feature engineering process, I analysed the demographic data and selected relevant features as demographic and economic indicators as outlined in Appendix A. To optimise the feature selection, I employed the **SelectKBest** technique from the scikit-learn

library. This method ranks input variables based on their correlation with the target variable (People_Per_Shop) and selects the top 15 most informative features.





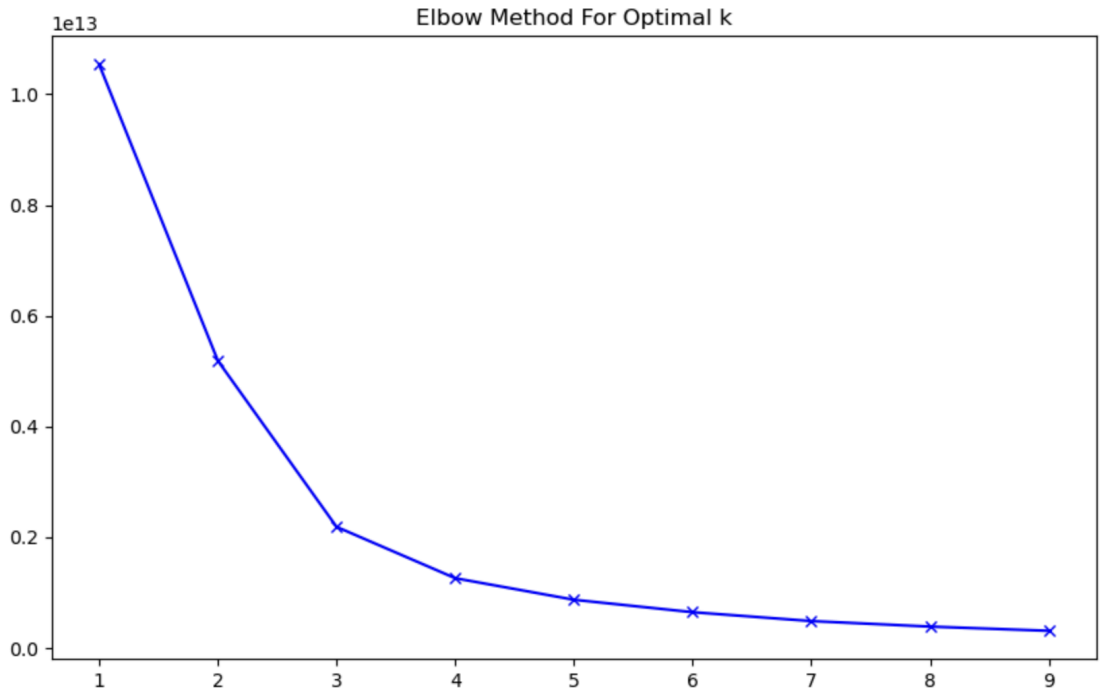
While the SelectKBest method initially selected 15 features, I found a high correlation among these variables. Hence, the idea of using SelectKBest was discarded. To address this issue, I implemented **Principal Component Analysis (PCA)** to reduce dimensionality. PCA transforms the original features into a new set of uncorrelated components.



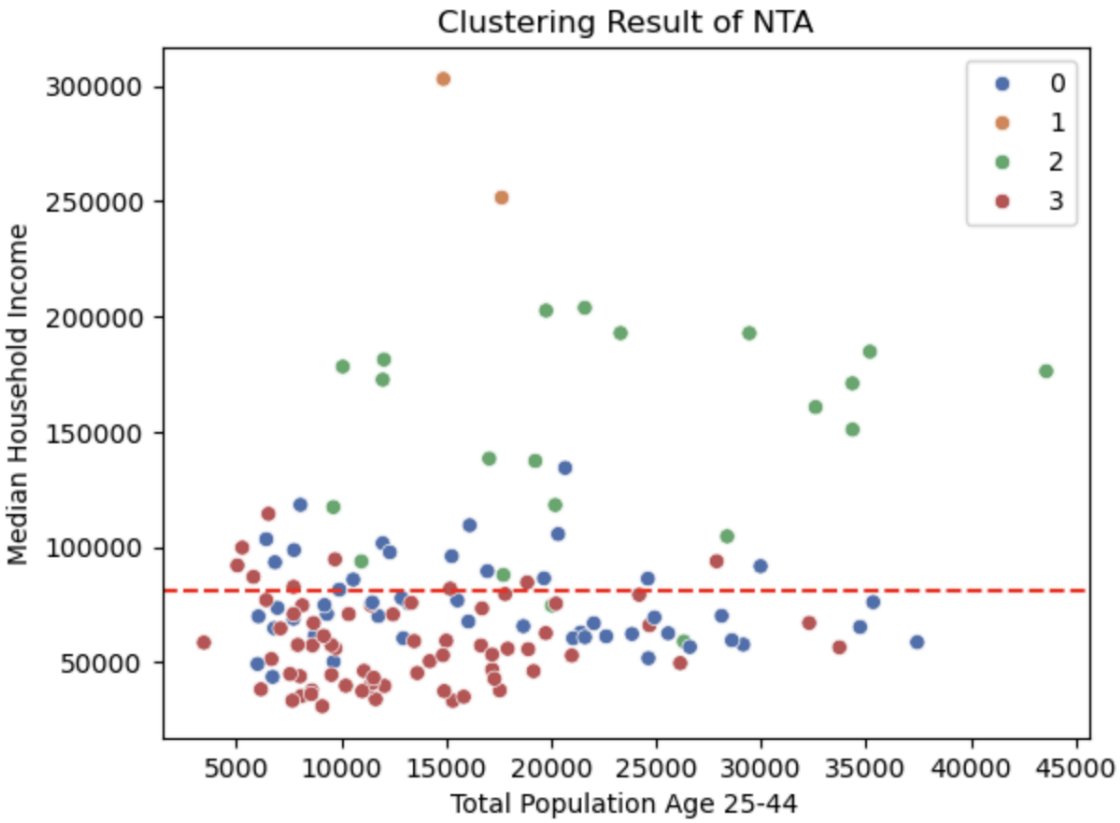
Visualising the explained variance ratio in the graph shows that retaining only 4 out of the 31 original features was sufficient to capture approximately 100% of the variation in the data. Therefore, I applied PCA to reduce the dimensions of features to 4 features while while preserving information.

After completing the feature engineering process, the next step is to explore the relationship among the selected features. Given the recommendation-based nature of the problem and the absence of a clear target variable, I decided to employ an **unsupervised learning** approach. By clustering similar neighbourhoods together, I could extract insights into the underlying relationships between different features.

There are various clustering algorithms. To simplify the analysis, I initially employed the **K-means** clustering algorithm, a common partition-based method. The algorithm requires us to determine the optimal number of clusters. Using elbow visualization and silhouette scores, we were able to identify the ideal number of clusters (k). This analysis helped us understand the relationship between the number of clusters and the model's performance.



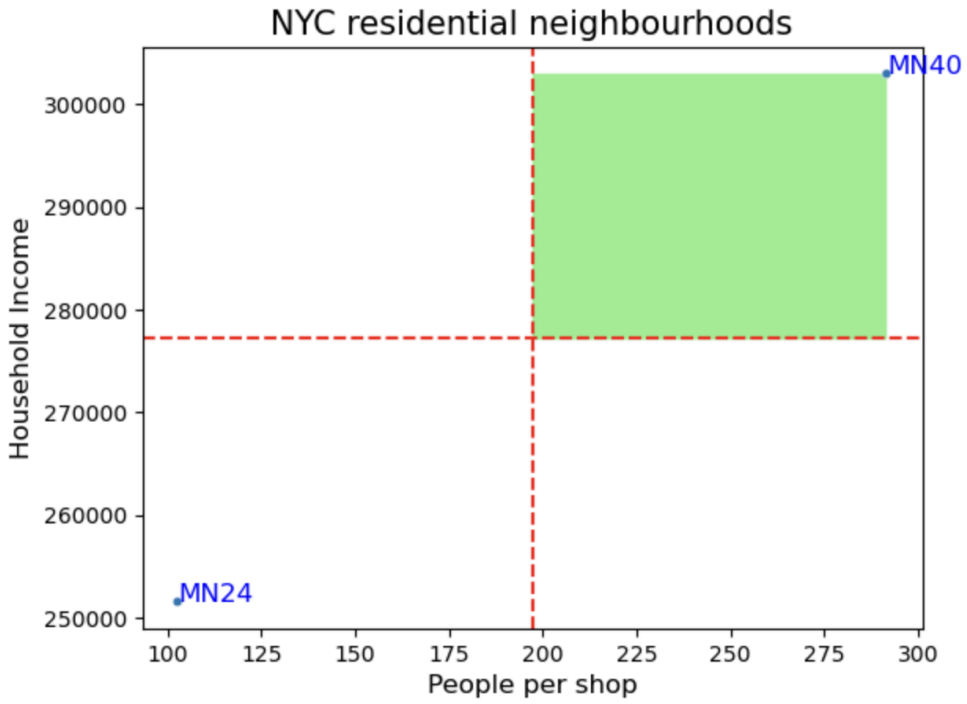
While the elbow method and silhouette score suggested an optimal number of clusters as 3, I want a clustering solution with 4 clusters (the second best) to gain additional insights into how the 195 NTAs could be grouped.



The visualisation shows how the NTA was clustered together based on total population and median household income. Generally, the domain expertise would guide the cluster selection, given the extensive number of metrics (31), I employed a simplified approach

```
Variable: Num_NTA Cluster: 3
Variable: Avg_Pop_Total_25_44 Cluster: 2
Variable: Avg_Pop_Pct_25_44 Cluster: 2
Variable: Avg_Pop_Male Cluster: 0
Variable: Avg_Pop_Female Cluster: 1
Variable: Avg_Median_Age Cluster: 1
Variable: Avg_Pop_Pct_Hisp_Latin Cluster: 3
Variable: Avg_Pop_Pct_Non_Hisp_Latin Cluster: 1
Variable: Avg_Pop_Pct_White Cluster: 1
Variable: Avg_Pop_Pct_Black Cluster: 3
Variable: Avg_Pop_Pct_American Cluster: 3
Variable: Avg_Pop_Pct_Asia Cluster: 0
Variable: Avg_Pop_Pct_Native Cluster: 3
Variable: Avg_Pop_Pct_Employ Cluster: 2
Variable: Avg_Pop_Pct_Employ_Salary Cluster: 1
Variable: Avg_Pop_Pct_Household_Earn_75_99 Cluster: 0
Variable: Avg_Pop_Pct_Family_Earn_75_99 Cluster: 0
Variable: Avg_Median_Household_Income Cluster: 1
Variable: Avg_Median_Family_Income Cluster: 1
Variable: Avg_Median_Earning Cluster: 1
Variable: Avg_People_Per_Parking Cluster: 0
Variable: Avg_Pop_Pct_Healthy Cluster: 1
Variable: Avg_Pct_Occupied_House Cluster: 3
Variable: Avg_Median_Room Cluster: 0
Variable: Avg_Median_House Cluster: 1
Variable: Avg_Median_Rent Cluster: 1
Variable: Avg_Pop_Pct_With_Car Cluster: 0
Variable: Avg_Pop_Pct_with_Family Cluster: 3
Variable: Avg_Pop_Pct_with_Couple Cluster: 1
Variable: Avg_Pop_Pct_With_Degree Cluster: 1
Variable: Avg_Shop_Count Cluster: 1
Variable: Avg_People_Per_Shop Cluster: 3
```

By counting the number of times each cluster was assigned the highest value for various variables (excluding the total number of competitors), I found that Cluster 1 received the most votes, despite only two NTAs being assigned to this cluster.



	GeoID	GeogName	Borough	Median_Household_Income	People_Per_Shop
1	MN40	Upper East Side-Carnegie Hill	Manhattan	302939.0	291.588235
0	MN24	SoHo-TriBeCa-Civic Center-Little Italy	Manhattan	251546.0	102.610465

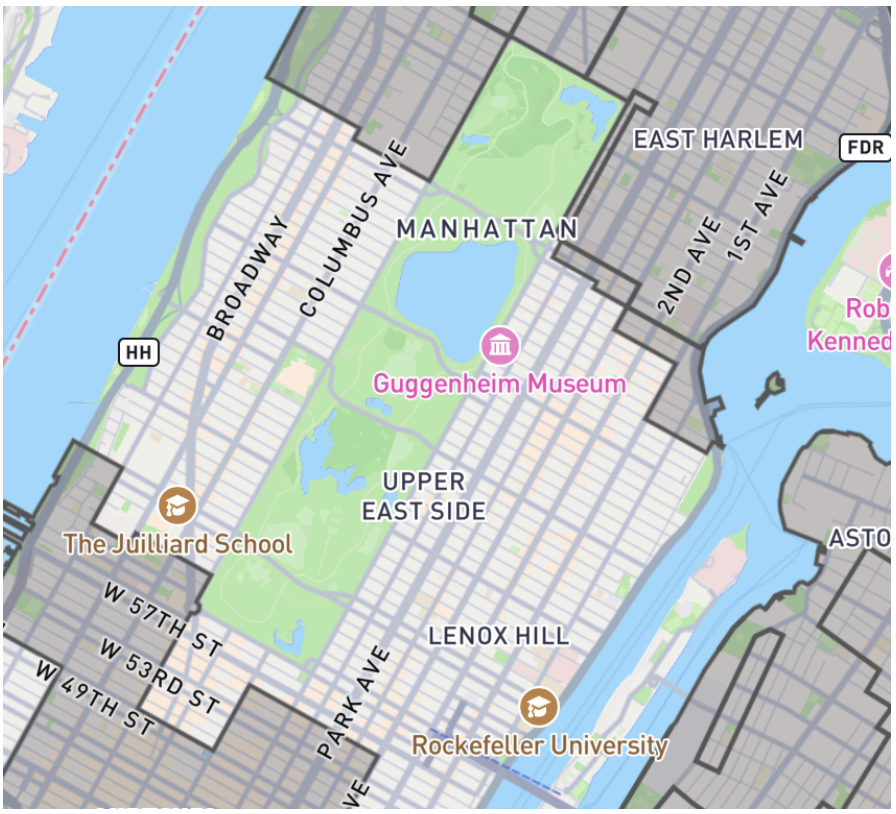
To select the optimal location within Cluster 1, I focused on the Upper East Side-Carnegie Hill (MN40) and SoHo-TriBeCa-Civic Center-Little Italy (MN24) neighbourhoods. Considering median household income and People per Shop, the analysis revealed that the Upper East **Side-Carnegie Hill neighbourhood (MN40)** offers the most promising potential. With a **People_per_Shop value of 291.5** and an **average household income of $302,939**, it outperforms SoHo-TriBeCa-Civic Center-Little Italy (MN24), which has a lower average household income of $251,546 and a higher level of competition with a People per Shop value of 102.6.

In addition, I utilised the **Folium** package to visualise the location of the Upper East Side-Carnegie Hill neighbourhood (green) on a map. The visualisation shows that the proposed location is located at a significant distance from the nearest Trader Joe's store (blue). However, there are several Stoop Line Stands (red) as competitors located along the route.

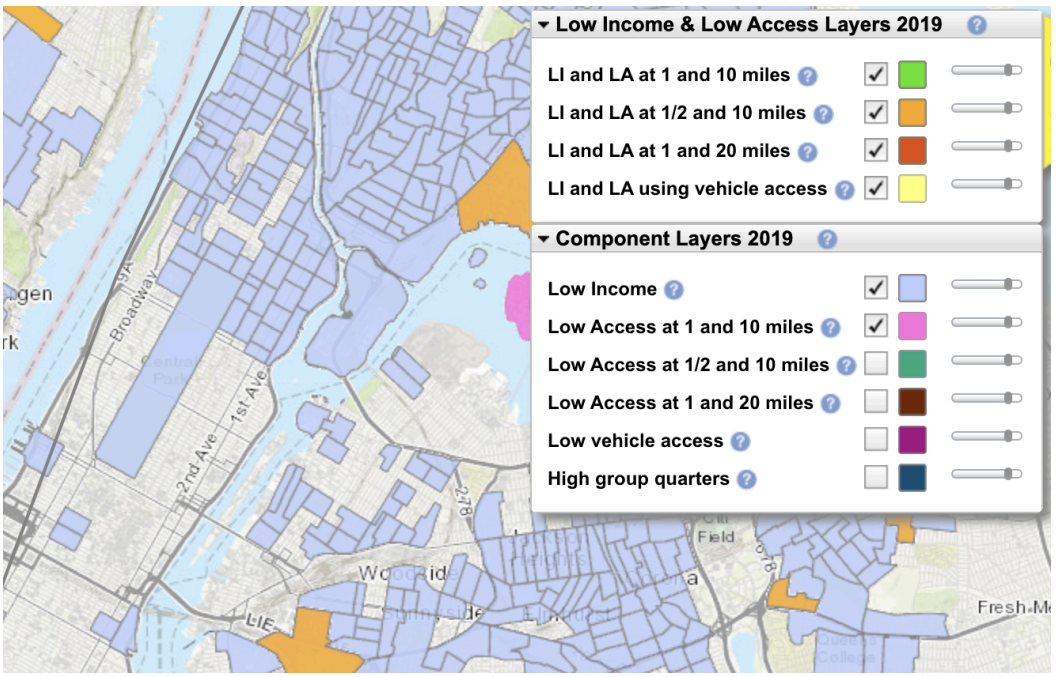
There are other clustering algorithms that can be explored. I also implemented **agglomerative clustering**, a hierarchical clustering method. Interestingly, agglomerative clustering yielded a slightly higher Silhouette Coefficient of 0.543 compared to K-means (0.527). Additionally, the Adjusted Rand Score of 0.824 indicated strong agreement between the cluster labels generated by the two algorithms.

	GeoID	GeogName	Borough	Median_Household_Income	People_Per_Shop
1	MN40	Upper East Side-Carnegie Hill	Manhattan	302939.0	291.588235
0	MN24	SoHo-TriBeCa-Civic Center-Little Italy	Manhattan	251546.0	102.610465

Despite the change in the clustering algorithm, the final recommendation remained consistent, suggesting that both East Side-Carnegie Hill (MN40) and SoHo-TriBeCa-Civic Center-Little Italy (MN24) are potential locations for a new Trader Joe's.



Other indicators such as FRESH Food Stores Zoning Boundaries (DCP, 2013) offer potential tax incentives for grocery store development, expansion, or renovation. I used **shapely** analysed GeoJson data to confirm if the proposed location will lie in these boundaries. Unfortunately, the Upper East Side-Carnegie Hill neighbourhood falls outside of these boundaries. As a result, a new Trader Joe's in this area may not be eligible for such incentives. Again, this indicator was created in 2013 so there is a possibility that the boundaries are changing.



Another relevant indicator is the food access index for low-income and other census tracts, as measured by the USDA (2021). I did a quick examination of the interactive map and revealed that the proposed area is not located within the designated food desert boundaries. This suggests that the low-income population in the area has reasonable access to supermarkets and grocery stores for purchasing essential goods.

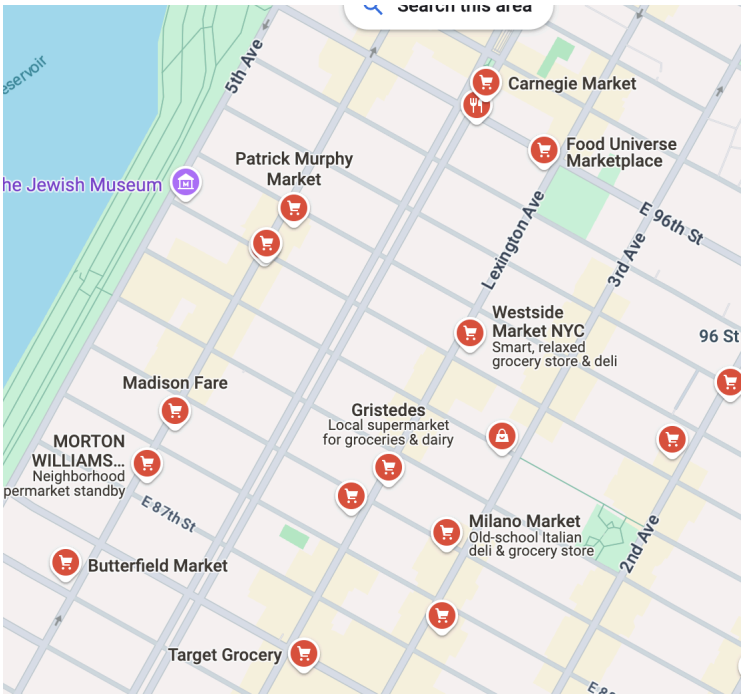
3. If you were given more time and permitted access to proprietary and/or third-party data sources obtained for a fee, discuss what else might you investigate in order to answer this question.

While the datasets provided valuable insights, several limitations were identified during the analysis. These include the potential for missing or incomplete data, as well as the lack of up-to-date information regarding economic, demographic, and social indicators at the NTA level.

I would recommend utilising third-party sources, such as the **Google Maps API**, to obtain up-to-date information on surrounding supermarkets, grocery stores, and Stoop Line Stands in each neighbourhood. We can potentially gain valuable insights, including:

- **List of competitors**: We can obtain a more comprehensive list of competitors, allowing us to better assess the competitive landscape and identify potential opportunities for Trader Joe's to attract customers.
    - Determine the distance between proposed locations and existing grocery stores.
    - Identify the density and types of competitors in each neighbourhood. Allow Trader Joe's to understand their market position with different criteria e.g., price vs brand.

- Identify nearby amenities that may attract customers, such as parks, schools, or entertainment venues.



:

- **Google Review:** Analysing customer reviews can provide valuable insights of the strengths and pain points of existing stores in the area. By understanding customer perceptions of factors such as pricing, product diversity, and customer service, Trader Joe's can develop strategies to differentiate itself and address unmet market needs.

In addition to identifying the potential opportunities in the area, we should also consider the costs and barriers associated with establishing a new store:

- **Financial Feasibility:** Understanding Trader Joe's current financial position, including income statement and cash flow, to determine the viability of opening a new store.
- **Set-Up Costs:** Estimating the costs associated with acquiring land or renting a space, obtaining necessary licenses, and hiring staff.
- **Infrastructure:** The accessibility of public transportation, road network connectivity, and transportation costs and efficiency for transporting goods from warehouses.
- **Crime Rate:** Considering the potential impact of crime, particularly shoplifting, on store profitability.

Additionally, conducting user behaviour research could provide valuable insights into customer behaviour, particularly spending decisions and interactions with the store. This research could explore factors such as transportation choices (walking or driving), average shopping expenditures, and the influence of economic conditions on the shopping experience.

References

Brownlee, J. (2020). *How to Perform Feature Selection for Regression Data*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/feature-selection-for-regression-data/.

Department of Agriculture (USDA) (2021). Food Access Research Atlas - Data.gov. [online] Data.gov. Available at: https://catalog.data.gov/dataset/food-access-research-atlas.

Department of City Planning (DCP) (2020). *Demographic Profiles of ACS 5 Year Estimates at the Neighborhood Tabulation Area (NTA) level*. [online] Cityofnewyork.us. Available at: https://data.cityofnewyork.us/City-Government/Demographic-Profiles-of-ACS-5-Year-Estimates-at-th/8cwr-7pqn/about_data [Accessed 5 Sep. 2024].

Department of City Planning (DCP) (2013). *FRESH Food Stores Zoning Boundaries*. Online] Cityofnewyork.us. Available at: https://data.cityofnewyork.us/Business/FRESH-Food-Stores-Zoning-Boundaries/w9uz-8epq [Accessed 6 Sep. 2024].

Department of Consumer and Worker Protection (DCWP) (2024). *Legally Operating Businesses | NYC Open Data*. [online] data.cityofnewyork.us. Available at: https://data.cityofnewyork.us/Business/Legally-Operating-Businesses/w7w3-xahh/about_data.

Macias, E. (2021). *Store locations*. [online] Medium. Available at: https://towardsdatascience.com/store-locations-d1025df22865.

Reuter, D. (2021). *Meet the typical Trader Joe's shopper: a younger, married, college-educated person earning over $80,000*. [online] Business Insider. Available at: https://www.businessinsider.com/typical-trader-joes-shopper-demographic-younger-married-earning-80k-income-2021-9.

Appendix

A. A list of variables used in this recommendation study.

Feature Name	Description	Selected by SelectKBest
Pop_Total_25_44	Total population of people aged between 25-44	Y
Pop_Pct_25_44	% population of people aged between 25-44	Y
Pop_Male	% male population	Y
Pop_Female	% female population	Y
Median_Age	Median age of the total population	
Pop_Pct_Hisp_Latin	% Hispanic/Latino (of any race) population	Y
Pop_Pct_Non_Hisp_Latin	% non-Hispanic/Latino population	
Pop_Pct_White	% White population	
Pop_Pct_Black	% Black or African American population	Y
Pop_Pct_American	% American Indian and Alaska Native population	
Pop_Pct_Asia	% Asia population	
Pop_Pct_Native	% native population	
Pop_Pct_Employ	% Employment	
Pop_Pct_Employ_Salary	% employment and earn a salary	Y

Pop_Pct_Household_Earn_75_99	% population with households earning between 75,000 and 99,000	Y
Pop_Pct_Family_Earn_75_99	% population with families earning between 75,000 and 99,000	Y
Median_Household_Income	Median household income	
Median_Family_Income	Median family income	
Median_Earning	Median earning ($)	
Pop_Pct_Healthy	% employed population with health insurance coverage	Y
Pct_Occupied_House	% occupied house	
Median_Room	Median number of rooms	Y
Median_House	Median house price	
Median_Rent	Median rent	
Pop_Pct_With_Car	% population own the car	Y
People_Per_Parking	Number of people (age 25-44) per parking lot or garage	
Pop_Pct_with_Family	% Household population with Family	Y
Pop_Pct_with_Couple	% Household population as married-couple	
Pop_Pct_With_Degree	% Household population hold Bachelor's degree or higher	

Shop_Count	Number of competitors	Y
People_Per_Shop	Number of people (age 25-44) per competitor shop	

B. The source code on Jupyter notebook

recommend_location

September 6, 2024

[41]:
```
import pandas as pd
import folium
import ast
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import json

from shapely.geometry import shape, GeometryCollection, Point
from collections import Counter

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

1 Load the data

[2]:
```
# load the demographic profiles dataset from
# https://data.cityofnewyork.us/City-Government/
  ↪Demographic-Profiles-of-ACS-5-Year-Estimates-at-th/8cwr-7pqn/about_data
df_pop = pd.read_excel('./data/ACS5yrNTA/demo_2016acs5yr_nta.xlsx')
print(df_pop.shape)

df_econ = pd.read_excel('./data/ACS5yrNTA/econ_2016acs5yr_nta.xlsx')
print(df_econ.shape)

df_house = pd.read_excel('./data/ACS5yrNTA/hous_2016acs5yr_nta.xlsx')
print(df_house.shape)

df_soc = pd.read_excel('./data/ACS5yrNTA/soc_2016acs5yr_nta.xlsx')
print(df_soc.shape)
```

```
(195, 484)
(195, 659)
```

```
(195, 524)
```

```
/Users/pornpanitrasivisith/opt/anaconda3/envs/pantalone/lib/python3.10/site-
packages/openpyxl/reader/workbook.py:118: UserWarning: Print area cannot be set
to Defined name: #N/A.
  warn(f"Print area cannot be set to Defined name: {defn.value}.")
```

```
(195, 2714)
```

[3]:
```
# load the demographic profiles dataset from
# https://data.cityofnewyork.us/Business/Legally-Operating-Businesses/w7w3-xahh/
  ↪about_data
df_bus = pd.read_csv('./data/Legally_Operating_Businesses_20240905.csv')
print(df_bus.shape)
```

```
(281413, 27)
```

```
/var/folders/0f/0432hb3j0_97yqpm39c2hwlw0000gn/T/ipykernel_11469/940827208.py:3:
DtypeWarning: Columns (19,20) have mixed types. Specify dtype option on import
or set low_memory=False.
  df_bus = pd.read_csv('./data/Legally_Operating_Businesses_20240905.csv')
```

[4]:
```
# check if there is any trader joe
df_bus[df_bus['Business Name'].str.lower().str.contains('trader') &␣
  ↪df_bus['Business Name'].str.lower().str.contains('joe')]
```

[4]:
```
Empty DataFrame
Columns: [DCA License Number, License Type, License Expiration Date, License
Status, License Creation Date, Industry, Business Name, Business Name 2, Address
Building, Address Street Name, Secondary Address Street Name, Address City,
Address State, Address ZIP, Contact Phone Number, Address Borough, Borough Code,
Community Board, Council District, BIN, BBL, NTA, Census Tract, Detail,
Longitude, Latitude, Location]
Index: []

[0 rows x 27 columns]
```

[5]:
```
# filter business operating in NYC only
print(df_bus['Address Borough'].value_counts())

# filter the dataset to only keep 5 boroughs, remove rows with misspelling
NYC_BOROUGHS = ['Queens', 'Brooklyn', 'Manhattan', 'Bronx', 'Staten Island']
df_bus = df_bus[df_bus['Address Borough'].isin(NYC_BOROUGHS)]
print(df_bus.shape)
```

```
Address Borough
Brooklyn         50346
Queens           45800
Manhattan        40435
```

```
Bronx            24695
Outside NYC      14775
Staten Island     9265
MANHATTAN           46
QUEENS              17
BROOKLYN            15
BRONX                9
Name: count, dtype: int64
(170541, 27)
```

[6]:
```
# We want to focus on recommending the specific NTA (Neighborhood Tabulation
 ↪Area) where the business is located.
# remove row with missing NTA
df_bus = df_bus.dropna(subset=['NTA'])
print(df_bus.shape)

df_bus[['Address Borough', 'NTA']].value_counts()
```

```
(137476, 27)
```

[6]:
```
Address Borough  NTA
Manhattan        MN17    4222
                 MN13    2288
                 MN24    1936
Queens           QN70    1905
                 QN61    1904
                         …
Brooklyn         MN24       1
Manhattan        BK78       1
                 BK91       1
Brooklyn         MN19       1
Queens           MN36       1
Name: count, Length: 209, dtype: int64
```

[7]:
```
# check how many industries
df_bus['Industry'].value_counts()
```

[7]:
```
Industry
Tobacco Retail Dealer                37491
Home Improvement Contractor          21261
Secondhand Dealer - General          12621
Electronics Store                    11933
Stoop Line Stand                      7376
Laundry                               6832
Electronic & Appliance Service        5325
Laundries                             4353
Laundry Jobber                        4339
```

```
Sidewalk Cafe                    3728
Garage                           3106
Electronic Cigarette Dealer      2923
Secondhand Dealer - Auto         2322
Parking Lot                      1720
Dealer In Products               1600
Tow Truck Company                1232
Employment Agency                1102
Special Sale                     1005
Amusement Device Temporary        979
Amusement Device Portable         942
Pawnbroker                        766
Cabaret                           602
Debt Collection Agency            496
Pedicab Business                  422
Amusement Device Permanent        414
Bingo Game Operator               294
Games of Chance                   294
Garage and Parking Lot            257
Process Serving Agency            227
Pool or Billiard Room             210
Catering Establishment            205
Car Wash                          164
Tow Truck Exemption               147
Gaming Cafe                       130
Storage Warehouse                 118
Scrap Metal Processor             111
Amusement Arcade                   92
Horse Drawn Cab Owner              90
Auction House Premises             69
Sightseeing Bus                    50
Scale Dealer Repairer              45
General Vendor Distributor         22
Booting Company                    15
Commercial Lessor                  15
Newsstand                          14
Secondhand Dealer - Firearms       11
Ticket Seller Business              6
Name: count, dtype: int64
```

2 Feature Engineering

[8]:
```
# Assume that the trader joe's lies in between  Stoop Line Stand  and General␣
 ↪Vendor industries
# [Note that general vendor was filtered as they are outside the NYC]
# let's use the sum of businesses of two industries as an indicator of␣
 ↪supermarket success

df_sup_success = df_bus[df_bus['Industry'].isin(['Stoop Line Stand', 'General␣
 ↪Vendor'])]
df_sup_success = df_sup_success[['Address Borough', 'NTA', 'Business Name',␣
 ↪'Industry']]
df_sup_success = df_sup_success.groupby(['NTA'])['Business Name'].count().
 ↪reset_index()
df_sup_success = df_sup_success.rename(columns={'Business Name':'Shop_Count',␣
 ↪'NTA':'GeoID'})
df_sup_success.head(5)
```

[8]:

	GeoID	Shop_Count
0	BK09	12
1	BK17	58
2	BK19	173
3	BK21	24
4	BK23	5

[9]:
```
# get the parking lot count per NTA
df_parking = df_bus[df_bus['Industry'].isin(['Garage', 'Parking Lot', 'Garage␣
 ↪and Parking Lot'])]
df_parking = df_parking[['Address Borough', 'NTA', 'Business Name', 'Industry']]
df_parking = df_parking.groupby(['NTA'])['Business Name'].count().reset_index()
df_parking = df_parking.rename(columns={'Business Name':'Parking_Count', 'NTA':
 ↪'GeoID'})
df_parking.head(5)
```

[9]:

	GeoID	Parking_Count
0	BK09	38
1	BK17	9
2	BK19	4
3	BK21	12
4	BK23	10

[10]:
```
# create NTA-level data
df_nta = pd.merge(df_pop, df_sup_success, on=['GeoID'])
df_nta = pd.merge(df_nta, df_parking, on=['GeoID'])
df_nta = pd.merge(df_nta, df_econ, on='GeoID')
```

```
df_nta = df_nta.drop(columns=['GeoType_x', 'GeoType_y', 'GeogName_x',␣
 ↪'GeogName_y', 'Borough_x', 'Borough_y'])
df_nta = pd.merge(df_nta, df_house, on='GeoID')
df_nta = pd.merge(df_nta, df_soc, on='GeoID')
df_nta = df_nta.drop(columns=['GeoType_x', 'GeoType_y', 'GeogName_x',␣
 ↪'GeogName_y', 'Borough_x', 'Borough_y'])
df_nta.shape
```

[10]: (141, 4368)

[11]:
```
df_nta.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141 entries, 0 to 140
Columns: 4368 entries, GeoID to UnclsNRZ
dtypes: float64(3511), int64(856), object(1)
memory usage: 4.7+ MB
```

[12]:
```
# create new features
df_nta['Pop_Total_25_44'] = df_nta.apply(lambda x: x['Pop25t29E'] +␣
 ↪x['Pop30t34E'] + x['Pop35t39E'] + x['Pop40t44E'], axis=1)
df_nta['Pop_Pct_25_44'] = df_nta.apply(lambda x: x['Pop_Total_25_44'] /␣
 ↪x['Pop_1E'] , axis=1)
df_nta['Pop_Male'] = df_nta.apply(lambda x: x['MaleE'] / x['Pop_1E'] , axis=1)
df_nta['Pop_Female'] = df_nta.apply(lambda x: x['FemE'] / x['Pop_1E'] , axis=1)
df_nta['Median_Age'] = df_nta['MdAgeE']
df_nta['Pop_Pct_Hisp_Latin'] = df_nta.apply(lambda x: x['Hsp1E'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_Non_Hisp_Latin'] = df_nta.apply(lambda x: x['NHspE'] /␣
 ↪x['Pop_2E'], axis=1)
df_nta['Pop_Pct_White'] = df_nta.apply(lambda x: x['WtNHE'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_Black'] = df_nta.apply(lambda x: x['BlNHE'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_American'] = df_nta.apply(lambda x: x['AIANNHE'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_Asia'] = df_nta.apply(lambda x: x['AsnNHE'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_Native'] = df_nta.apply(lambda x: x['NHPINHE'] / x['Pop_2E'],␣
 ↪axis=1)
df_nta['Pop_Pct_Employ'] = df_nta.apply(lambda x: x['CvEm16pl1E'] /␣
 ↪x['Pop16plE'], axis=1)
df_nta['Pop_Pct_Employ_Salary'] = df_nta.apply(lambda x: x['PrvWSWrkrE'] /␣
 ↪x['CvEm16pl4E'], axis=1)
df_nta['Pop_Pct_Household_Earn_75_99'] = df_nta.apply(lambda x: x['HHI75t99E'] /
 ↪ x['HH2E'], axis=1) #earning $75,000 to $99,999
```

```
df_nta['Pop_Pct_Family_Earn_75_99'] = df_nta.apply(lambda x: x['FamI75t99E'] /␣
 ↪x['Fam2E'], axis=1) #earning $75,000 to $99,999
df_nta['Median_Household_Income'] = df_nta['MnHHIncE']
df_nta['Median_Family_Income'] = df_nta['MdFamIncE']
df_nta['Median_Earning'] = df_nta['MdEWrkE']
df_nta['Pop_Pct_Healthy'] = df_nta.apply(lambda x: x['EmHInsE'] / x['CvLFEmE'],␣
 ↪axis=1) #healthy with insurance coverage of overall employment
df_nta['Pop_Pct_With_Car'] = df_nta.apply(lambda x: (x['Vhcl1AvE'] +␣
 ↪x['Vhcl2AvE'] + x['Vhcl3plAvE']) / x['OcHU4E'], axis=1)
df_nta['Pct_Occupied_House'] = df_nta.apply(lambda x: x['OcHU1E'] / x['HU1E'],␣
 ↪axis=1)
df_nta['Median_Room'] = df_nta['MdRmsE']
df_nta['Median_House'] = df_nta['MdVlE'] #house price
df_nta['Median_Rent'] = df_nta['MdGRE']
df_nta['Pop_Pct_with_Family'] = df_nta.apply(lambda x: x['Fam1E'] / x['HH1E'],␣
 ↪axis=1)
df_nta['Pop_Pct_with_Couple'] = df_nta.apply(lambda x: x['MrdFamE'] /␣
 ↪x['HH1E'], axis=1)
df_nta['Pop_Pct_With_Degree'] = df_nta.apply(lambda x: x['EA_BchDHE'] /␣
 ↪x['EA_P25plE'], axis=1)
df_nta['People_Per_Shop'] = df_nta.apply(lambda x: (x['Pop_Total_25_44'] /␣
 ↪x['Shop_Count']) if x['Shop_Count'] > 0 else x['Pop_Total_25_44'], axis=1)
df_nta['People_Per_Parking'] = df_nta.apply(lambda x: ((x['Pop_Total_25_44'] *␣
 ↪x['Pop_Pct_With_Car']) / x['Parking_Count']) if x['Parking_Count'] > 0 else␣
 ↪0, axis=1)

filtered_columns = ['Pop_Total_25_44', 'Pop_Pct_25_44', 'Pop_Male',␣
 ↪'Pop_Female', 'Median_Age',
                        'Pop_Pct_Hisp_Latin', 'Pop_Pct_Non_Hisp_Latin',␣
 ↪'Pop_Pct_White', 'Pop_Pct_Black',
                     'Pop_Pct_American', 'Pop_Pct_Asia', 'Pop_Pct_Native',
                     'Pop_Pct_Employ', 'Pop_Pct_Employ_Salary',␣
 ↪'Pop_Pct_Household_Earn_75_99', 'Pop_Pct_Family_Earn_75_99',
                     'Median_Household_Income', 'Median_Family_Income',␣
 ↪'Median_Earning', 'People_Per_Parking',
                     'Pop_Pct_Healthy', 'Pct_Occupied_House', 'Median_Room',␣
 ↪'Median_House', 'Median_Rent', 'Pop_Pct_With_Car',
                    'Pop_Pct_with_Family', 'Pop_Pct_with_Couple',␣
 ↪'Pop_Pct_With_Degree', 'Shop_Count', 'People_Per_Shop', 'GeoID']

df_nta = df_nta[filtered_columns]
```

[13]:
```
df_nta.describe()
```

[13]:
```
       Pop_Total_25_44  Pop_Pct_25_44    Pop_Male  Pop_Female  Median_Age  \
count       141.000000     141.000000  141.000000  141.000000  141.000000
```

mean	16084.865248	0.322150	0.476259	0.523741	35.804255
std	8383.992699	0.067413	0.024014	0.024014	5.023756
min	3460.000000	0.199396	0.405374	0.447971	19.300000
25%	9326.000000	0.278074	0.461103	0.508270	32.800000
50%	14848.000000	0.306920	0.476797	0.523203	35.000000
75%	20674.000000	0.353908	0.491730	0.538897	38.300000
max	43597.000000	0.522566	0.552029	0.594626	57.800000

	Pop_Pct_Hisp_Latin	Pop_Pct_Non_Hisp_Latin	Pop_Pct_White	\
count	141.000000	141.000000	141.000000	
mean	0.318375	0.681625	0.309179	
std	0.225222	0.225222	0.267171	
min	0.027231	0.121075	0.009473	
25%	0.130770	0.460005	0.059919	
50%	0.220358	0.779642	0.219758	
75%	0.539995	0.869230	0.533173	
max	0.878925	0.972769	0.947759	

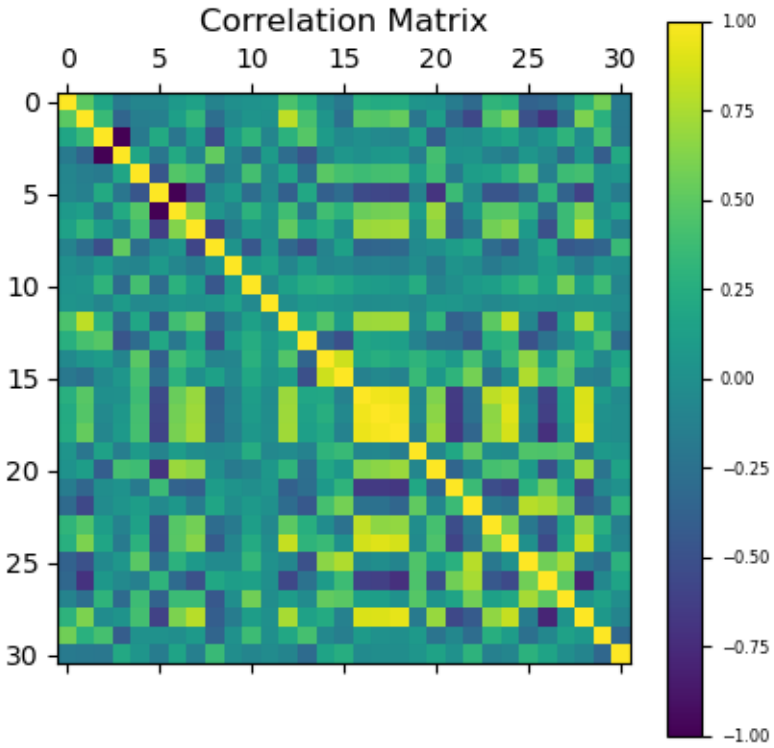
	Pop_Pct_Black	Pop_Pct_American	…	Pct_Occupied_House	Median_Room	\
count	141.000000	141.000000	…	141.000000	141.000000	
mean	0.210242	0.001670	…	0.914621	3.976596	
std	0.230328	0.001797	…	0.040714	0.591202	
min	0.005468	0.000000	…	0.691320	2.800000	
25%	0.034394	0.000487	…	0.899330	3.600000	
50%	0.111066	0.001076	…	0.925796	3.900000	
75%	0.295650	0.002248	…	0.939145	4.200000	
max	0.889189	0.008993	…	0.965416	5.700000	

	Median_House	Median_Rent	Pop_Pct_With_Car	Pop_Pct_with_Family	\
count	1.410000e+02	141.000000	141.000000	141.000000	
mean	5.579921e+05	1364.390071	0.434875	0.603866	
std	2.661271e+05	408.922273	0.197576	0.133373	
min	1.096480e+05	620.000000	0.126626	0.216916	
25%	3.876940e+05	1108.000000	0.270372	0.537256	
50%	5.148060e+05	1302.000000	0.391197	0.640015	
75%	6.843430e+05	1523.000000	0.559543	0.683944	
max	2.000000e+06	2931.000000	0.896510	0.849989	

	Pop_Pct_with_Couple	Pop_Pct_With_Degree	Shop_Count	People_Per_Shop
count	141.000000	141.000000	141.000000	141.000000
mean	0.348798	0.351970	47.921986	787.563607
std	0.124267	0.209393	48.602626	1336.120611
min	0.146943	0.073298	1.000000	50.508671
25%	0.250179	0.201790	17.000000	286.468750
50%	0.319097	0.287973	31.000000	441.181818
75%	0.429271	0.464265	59.000000	706.424242
max	0.708358	0.846186	251.000000	11416.000000

```
[8 rows x 31 columns]
```

[14]:
```
plt.matshow(df_nta.drop(columns=['GeoID']).corr())
cb = plt.colorbar()
cb.ax.tick_params(labelsize=6)
plt.title('Correlation Matrix');
```



[15]:
```
# https://machinelearningmastery.com/feature-selection-for-regression-data/
# Feature selection instead, takes the target into consideration.
# It will rank your input variables in terms of how useful they are to predict␣
 ↪the target value.
# it is different from PCA that try to reduce dimensionality by exploring how␣
 ↪one feature of the data is expressed
# in terms of the other features(linear dependency)

def select_features(X_train, y_train, k):
 fs = SelectKBest(score_func=f_regression, k=k)

 # learn relationships from training data
 fs.fit(X_train, y_train)
```

```
 # transform train input data
 X_train_fs = fs.transform(X_train)

 return X_train_fs, fs
```

[16]:
```
# remove columns that contain all missing rows
df_nta = df_nta.drop(columns=df_nta.columns[df_nta.isnull().all()].to_list())

# replace missing value with mean
for col in df_nta.columns:
    if col not in ['GeoID', 'Address Borough', 'Industry']:
        col_mean = df_nta[col].mean()
        df_nta[col].fillna(col_mean, inplace=True)

print(df_nta.columns[df_nta.isnull().any()])
```

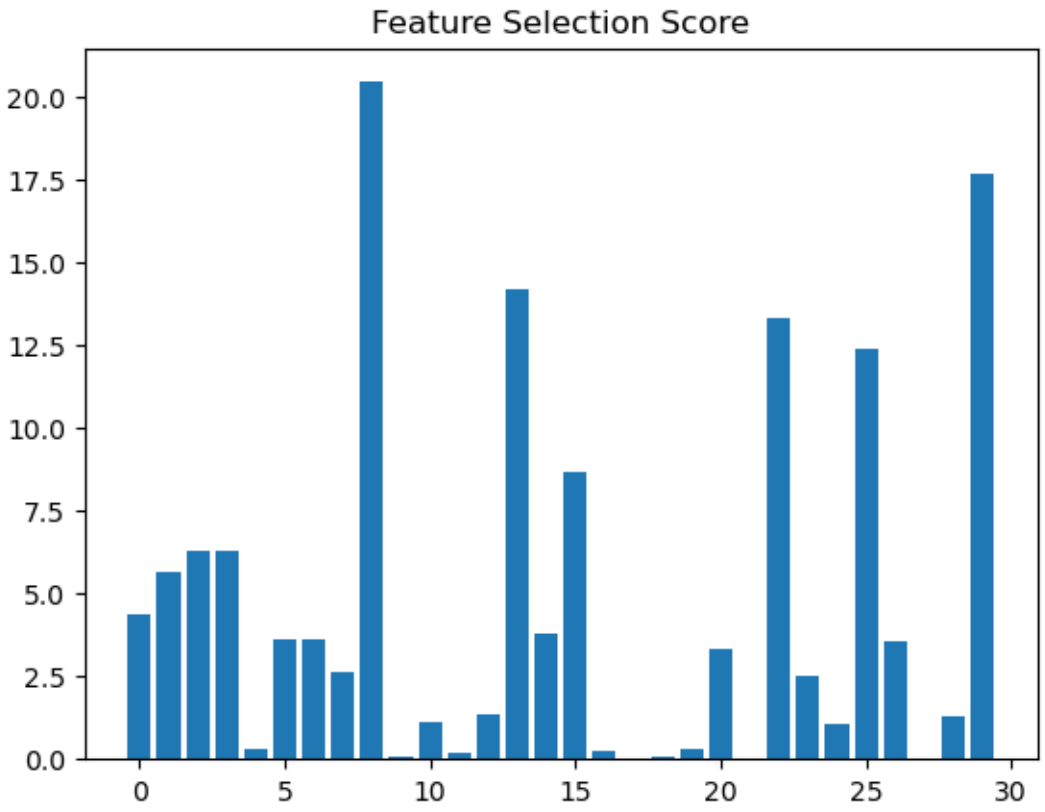
```
Index([], dtype='object')
```

```
/var/folders/0f/0432hb3j0_97yqpm39c2hwlw0000gn/T/ipykernel_11469/687371754.py:8:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df_nta[col].fillna(col_mean, inplace=True)
```

[17]:
```
# feature selection
X_train_fs, fs = select_features(df_nta.drop(columns=['People_Per_Shop',␣
 ↪'GeoID']), df_nta['People_Per_Shop'], 15)
cols = df_nta.drop(columns=['People_Per_Shop', 'GeoID']).columns
# # what are scores for the features
for i in range(len(fs.scores_)):
 print('Feature %s: %f' % (cols[i], fs.scores_[i]))
# plot the scores
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.title('Feature Selection Score')
plt.show()
```

```
Feature Pop_Total_25_44: 4.361144
Feature Pop_Pct_25_44: 5.655758
Feature Pop_Male: 6.313239
Feature Pop_Female: 6.313239
```

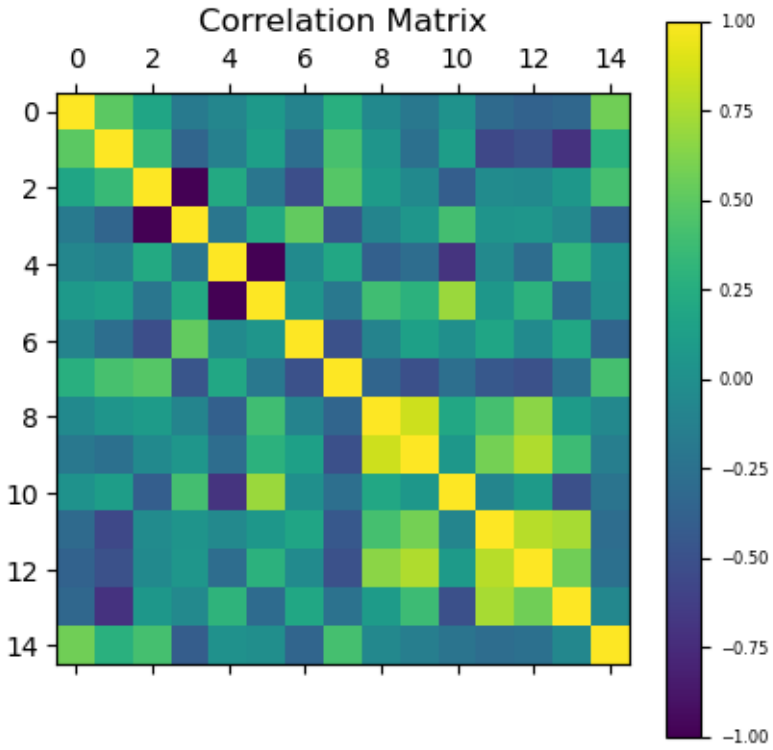
```
Feature Median_Age: 0.286427
Feature Pop_Pct_Hisp_Latin: 3.613611
Feature Pop_Pct_Non_Hisp_Latin: 3.613611
Feature Pop_Pct_White: 2.643762
Feature Pop_Pct_Black: 20.460530
Feature Pop_Pct_American: 0.070569
Feature Pop_Pct_Asia: 1.145907
Feature Pop_Pct_Native: 0.212032
Feature Pop_Pct_Employ: 1.377569
Feature Pop_Pct_Employ_Salary: 14.183344
Feature Pop_Pct_Household_Earn_75_99: 3.799188
Feature Pop_Pct_Family_Earn_75_99: 8.678893
Feature Median_Household_Income: 0.246639
Feature Median_Family_Income: 0.040072
Feature Median_Earning: 0.073440
Feature People_Per_Parking: 0.301660
Feature Pop_Pct_Healthy: 3.312111
Feature Pct_Occupied_House: 0.025695
Feature Median_Room: 13.313625
Feature Median_House: 2.489113
Feature Median_Rent: 1.064887
Feature Pop_Pct_With_Car: 12.372560
Feature Pop_Pct_with_Family: 3.565232
Feature Pop_Pct_with_Couple: 0.015439
Feature Pop_Pct_With_Degree: 1.298991
Feature Shop_Count: 17.674911
```



[18]:
```
selected_features = []
for idx, col in enumerate(cols):
    if fs.get_support()[idx]:
        selected_features.append(col)
selected_features
```

[18]:
```
['Pop_Total_25_44',
 'Pop_Pct_25_44',
 'Pop_Male',
 'Pop_Female',
 'Pop_Pct_Hisp_Latin',
 'Pop_Pct_Non_Hisp_Latin',
 'Pop_Pct_Black',
 'Pop_Pct_Employ_Salary',
 'Pop_Pct_Household_Earn_75_99',
 'Pop_Pct_Family_Earn_75_99',
 'Pop_Pct_Healthy',
 'Median_Room',
 'Pop_Pct_With_Car',
 'Pop_Pct_with_Family',
 'Shop_Count']
```

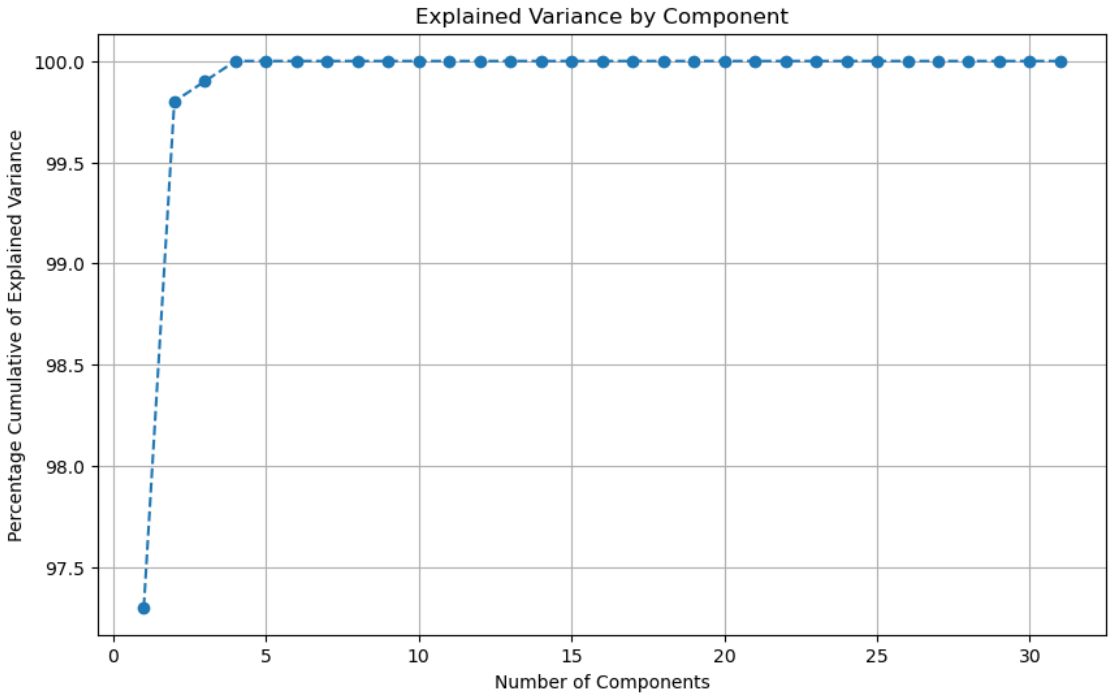
[19]:
```
# still indicates highly correlated result
plt.matshow(df_nta[selected_features].corr())
cb = plt.colorbar()
cb.ax.tick_params(labelsize=6)
plt.title('Correlation Matrix');
```



3 PCA

[20]:
```
# reducing the dimensionality of data, decorrelate these variables

pca = PCA()
pca.fit(df_nta.drop(columns=['GeoID']))
per_var = np.round(pca.explained_variance_ratio_*100, decimals = 1)

plt.figure(figsize = (10,6))
plt.plot(range(1, len(per_var)+1), per_var.cumsum(), marker = "o", linestyle =␣
↪"--")
plt.grid()
plt.ylabel("Percentage Cumulative of Explained Variance")
plt.xlabel("Number of Components")
plt.title("Explained Variance by Component")
```

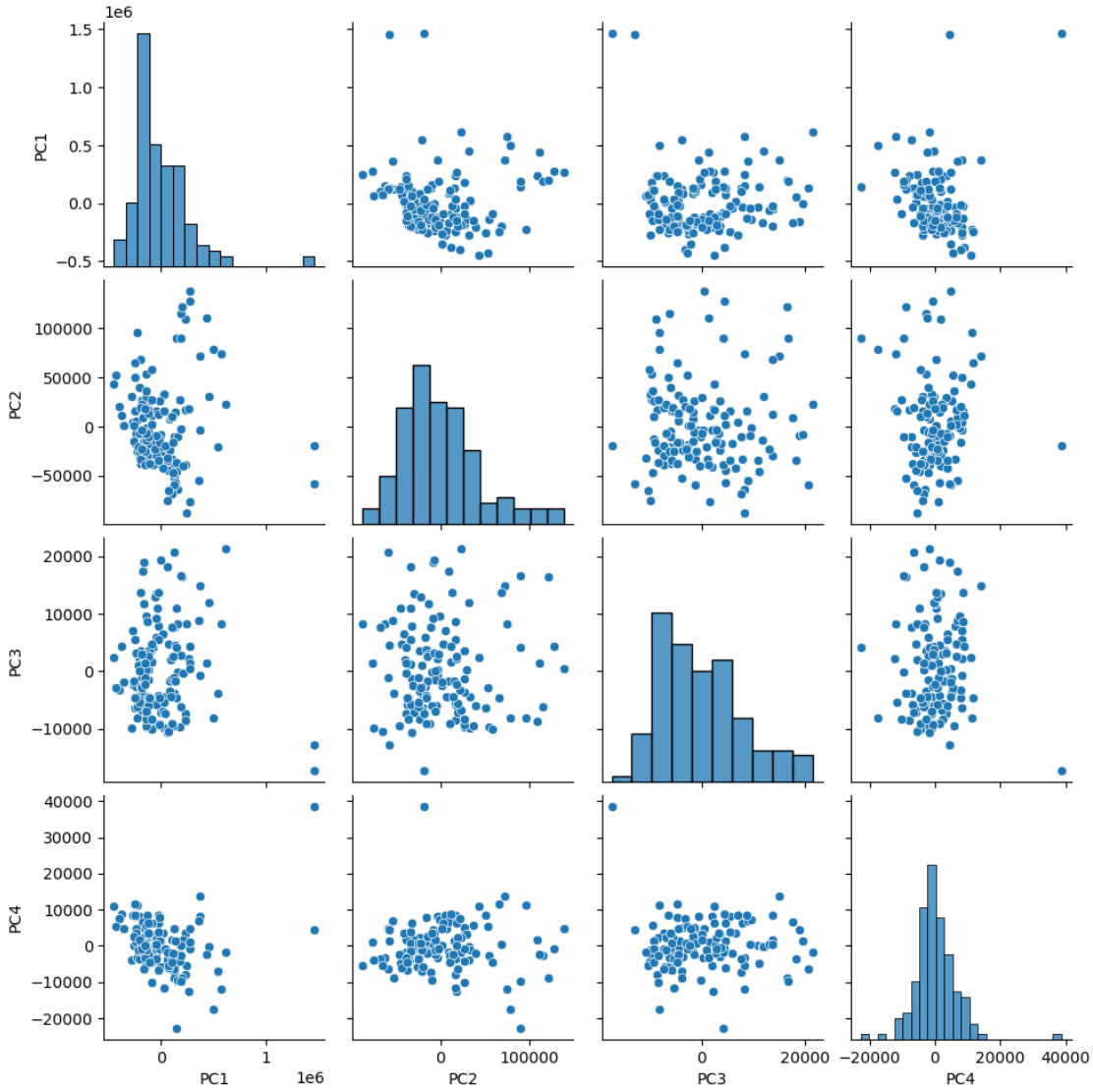
```
plt.show()
```



[21]:
```
# it's possible to retain almost 100% of the variation by using only 4 out of␣
↪the 31 features
pca = PCA(n_components = 4)
pca.fit(df_nta.drop(columns=['GeoID']))
pca_features = pca.transform(df_nta.drop(columns=['GeoID']))
pca_features.shape
```

[21]: (141, 4)

[44]:
```
sns.pairplot(pd.DataFrame(pca_features, columns=['PC1', 'PC2', 'PC3', 'PC4']))
plt.show()
```



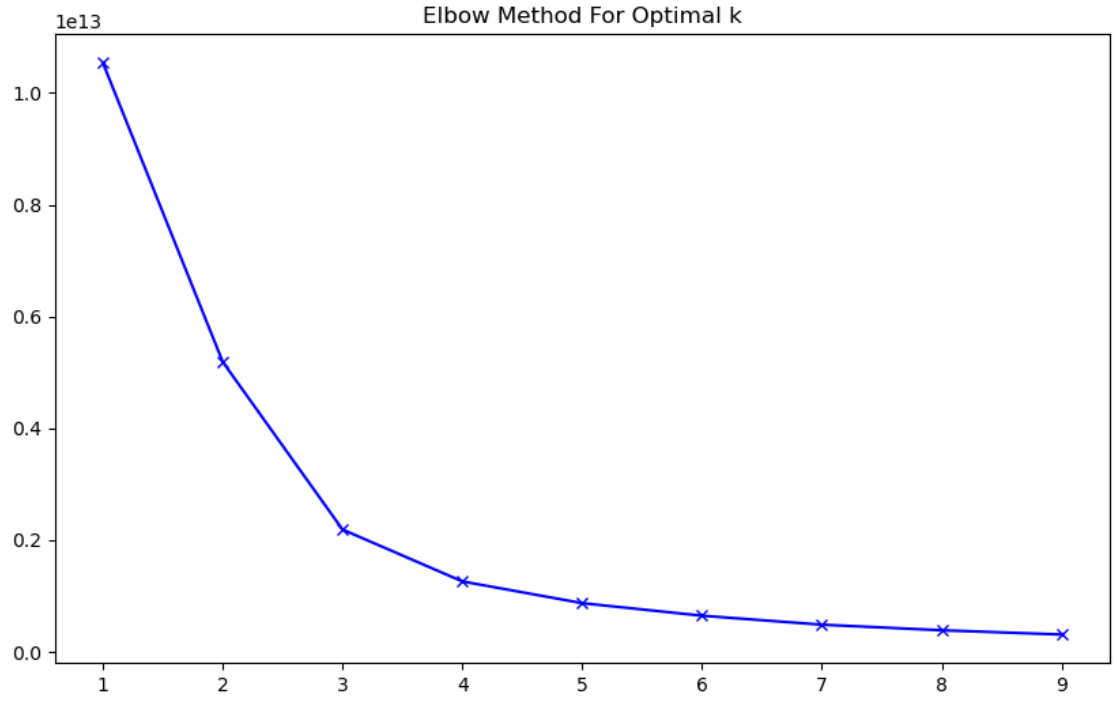
4 Clustering

[22]:
```
# Apply clustering on feature selection
sum_of_squared_distances = []

for k in range(1,10): # maximum of 10 clusters
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(pca_features)
    sum_of_squared_distances.append(kmeans.inertia_)
    if k > 1:
        print(f"K: {k} Score: {silhouette_score(pca_features, kmeans.
↪fit_predict(pca_features))}")
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(1,10), sum_of_squared_distances, 'bx-')
plt.title('Elbow Method For Optimal k')
```

```
K: 2 Score: 0.5545612178520679
K: 3 Score: 0.581433678237405
K: 4 Score: 0.5300248551230026
K: 5 Score: 0.44373916962487037
K: 6 Score: 0.4386562492295527
K: 7 Score: 0.49168855900972935
K: 8 Score: 0.44884829385421543
K: 9 Score: 0.46280595459102264
```

[22]: `Text(0.5, 1.0, 'Elbow Method For Optimal k')`



[23]:
```
k_clusters = 4

kmeans = KMeans(n_clusters=k_clusters, random_state=100).fit(pca_features)
```

```
/Users/pornpanitrasivisith/opt/anaconda3/envs/pantalone/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[24]:
```
nyc_cluster = df_nta.copy()
nyc_cluster['cluster'] = kmeans.labels_
nyc_cluster
```

[24]:

	Pop_Total_25_44	Pop_Pct_25_44	Pop_Male	Pop_Female	Median_Age	\
0	24615	0.311055	0.478037	0.521963	39.5	
1	25584	0.339680	0.484492	0.515508	29.1	
2	18681	0.290678	0.486766	0.513234	39.6	
3	28128	0.306920	0.487212	0.512788	37.8	
4	26331	0.248610	0.510155	0.489845	25.6	
..	…	…	…	…	…	
136	11003	0.294237	0.480439	0.519561	31.4	
137	10220	0.285858	0.478435	0.521565	33.0	
138	7933	0.278968	0.467068	0.532932	34.0	
139	17193	0.264333	0.461541	0.538459	34.7	
140	12467	0.282929	0.465278	0.534722	38.9	

	Pop_Pct_Hisp_Latin	Pop_Pct_Non_Hisp_Latin	Pop_Pct_White	Pop_Pct_Black	\
0	0.178608	0.821392	0.634329	0.019094	
1	0.194933	0.805067	0.350607	0.405826	
2	0.123283	0.876717	0.465153	0.008262	
3	0.145975	0.854025	0.426773	0.008740	
4	0.096721	0.903279	0.757131	0.008932	
..	…	…	…	…	
136	0.707046	0.292954	0.018986	0.231154	
137	0.646090	0.353910	0.033061	0.251091	
138	0.636249	0.363751	0.039420	0.151352	
139	0.252156	0.747844	0.023461	0.692157	
140	0.174451	0.825549	0.162922	0.593795	

	Pop_Pct_American	…	Median_House	Median_Rent	Pop_Pct_With_Car	\
0	0.002287	…	687001.0	1385.0	0.548691	
1	0.000876	…	722811.0	1165.0	0.312752	
2	0.000000	…	657026.0	1209.0	0.540046	
3	0.001211	…	708946.0	1259.0	0.526765	
4	0.001322	…	815579.0	1341.0	0.483225	
..	…	…	…	…	…	
136	0.000882	…	213234.0	1098.0	0.228866	
137	0.000587	…	453009.0	1108.0	0.394016	
138	0.000000	…	425647.0	1308.0	0.522319	
139	0.003413	…	378872.0	1204.0	0.512452	
140	0.000272	…	407578.0	1296.0	0.667160	

	Pop_Pct_with_Family	Pop_Pct_with_Couple	Pop_Pct_With_Degree	\
0	0.565443	0.429271	0.464265	
1	0.557965	0.273034	0.321807	
2	0.733583	0.531257	0.301847	

```
3               0.729948              0.537698              0.262382
4               0.800452              0.680694              0.188132
..                   …                     …                     …
136             0.650325              0.232837              0.139636
137             0.683944              0.280236              0.092649
138             0.696242              0.339579              0.185358
139             0.680864              0.278872              0.187137
140             0.697100              0.350198              0.274353

     Shop_Count  People_Per_Shop  GeoID  cluster
0           152       161.940789   BK31        0
1            25      1023.360000   BK75        0
2           105       177.914286   BK29        0
3           173       162.589595   BK28        0
4            99       265.969697   BK88        2
..          …              …    …      …
136          32       343.843750   BX63        3
137          40       255.500000   BX08        3
138          30       264.433333   BX59        3
139          42       409.357143   BX44        3
140          10      1246.700000   BX62        3

[141 rows x 33 columns]
```

5 Analyse clustering result

[25]:
```
# merge the NTA metadata
nyc_cluster = pd.merge(nyc_cluster, df_pop[['GeogName', 'GeoID', 'Borough']],␣
 ↪on = 'GeoID')
nyc_cluster.head(5)
```

[25]:
```
   Pop_Total_25_44  Pop_Pct_25_44  Pop_Male  Pop_Female  Median_Age  \
0            24615       0.311055  0.478037    0.521963        39.5
1            25584       0.339680  0.484492    0.515508        29.1
2            18681       0.290678  0.486766    0.513234        39.6
3            28128       0.306920  0.487212    0.512788        37.8
4            26331       0.248610  0.510155    0.489845        25.6

   Pop_Pct_Hisp_Latin  Pop_Pct_Non_Hisp_Latin  Pop_Pct_White  Pop_Pct_Black  \
0            0.178608                0.821392       0.634329       0.019094
1            0.194933                0.805067       0.350607       0.405826
2            0.123283                0.876717       0.465153       0.008262
3            0.145975                0.854025       0.426773       0.008740
4            0.096721                0.903279       0.757131       0.008932
```

```
   Pop_Pct_American  …  Pop_Pct_With_Car  Pop_Pct_with_Family  \
0          0.002287  …          0.548691             0.565443
1          0.000876  …          0.312752             0.557965
2          0.000000  …          0.540046             0.733583
3          0.001211  …          0.526765             0.729948
4          0.001322  …          0.483225             0.800452

   Pop_Pct_with_Couple  Pop_Pct_With_Degree  Shop_Count  People_Per_Shop  \
0             0.429271             0.464265         152       161.940789
1             0.273034             0.321807          25      1023.360000
2             0.531257             0.301847         105       177.914286
3             0.537698             0.262382         173       162.589595
4             0.680694             0.188132          99       265.969697

   GeoID  cluster          GeogName   Borough
0   BK31        0         Bay Ridge  Brooklyn
1   BK75        0           Bedford  Brooklyn
2   BK29        0  Bensonhurst East  Brooklyn
3   BK28        0  Bensonhurst West  Brooklyn
4   BK88        2      Borough Park  Brooklyn

[5 rows x 35 columns]
```

[26]:
```python
# create new dataframe to summarise each label
filtered_columns.pop()

results = []
for cluster in sorted(nyc_cluster.cluster.unique()):
    df = nyc_cluster[nyc_cluster.cluster == cluster]
    nta_names = df.GeogName.unique()
    nta_ids = df.GeoID.unique()
    cluster_dict = {
        'cluster': cluster,
        'NTA_Name': df.GeogName.unique(),
        'NTA_ID': df.GeoID.unique(),
        'Num_NTA': df.GeoID.nunique()
    }
    for col in filtered_columns:
        cluster_dict[f'Avg_{col}'] = df[col].mean()
    results.append(cluster_dict)
    print(f"Group {cluster}")
```

```
Group 0
Group 1
Group 2
Group 3
```

[27]:
```
df_results = pd.DataFrame(results)
df_results.head()
```

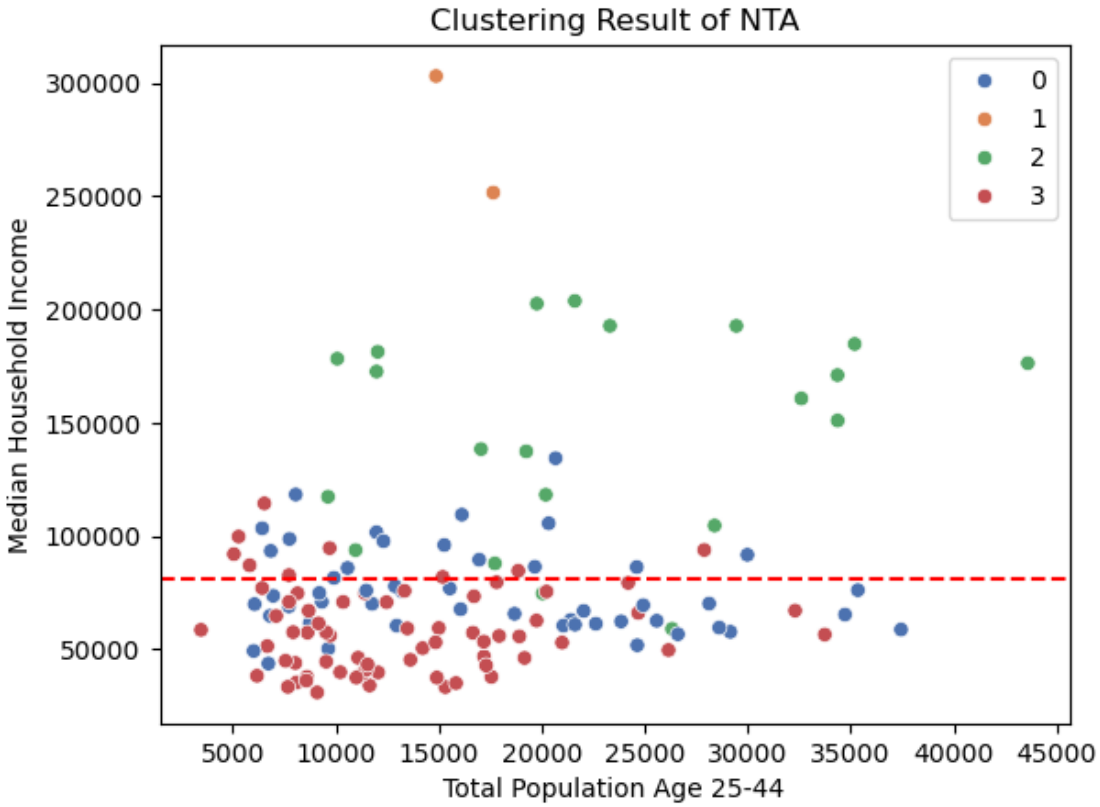
[27]:

	cluster	NTA_Name	NTA_ID	Num_NTA	Avg_Pop_Total_25_44	Avg_Pop_Pct_25_44	Avg_Pop_Male	Avg_Pop_Female	Avg_Median_Age	Avg_Pop_Pct_Hisp_Latin	…	Avg_Pct_Occupied_House	Avg_Median_Room	Avg_Median_House	Avg_Median_Rent	Avg_Pop_Pct_With_Car	Avg_Pop_Pct_with_Family	Avg_Pop_Pct_with_Couple	Avg_Pop_Pct_With_Degree	Avg_Shop_Count	Avg_People_Per_Shop
0	0	[Bay Ridge, Bedford, Bensonhurst East, Bensonh…	[BK31, BK75, BK29, BK28, BK19, BK77, BK78, BK6…	50	17102.640000	0.331125	0.484837	0.515163	35.846000	0.249277	…	0.918420	4.102000	6.168269e+05	1364.860000	0.471497	0.613026	0.401405	0.352989	60.460000	621.422852
1	1	[SoHo-TriBeCa-Civic Center-Little Italy, Upper…	[MN24, MN40]	2	16260.000000	0.332222	0.468898	0.531102	42.000000	0.068271	…	0.809973	3.800000	2.000000e+06	2197.500000	0.304679	0.471211	0.404373	0.795576	111.500000	197.099350
2	2	[Borough Park, Brooklyn Heights-Cobble Hill, C…	[BK88, BK09, BK33, BK38, BK68, BK73, BK37, BK6…	21	22759.095238	0.412403	0.478205	0.521795	36.128571	0.152747	…	0.870738	3.404762	8.822295e+05	1925.380952	0.273841	0.425934	0.316080	0.674728	50.000000	586.792662
3	3	[Brownsville, East New York, East New York (Pe…	[BK81, BK82, BK85, BK95, BK58, BK96, BK21, BK1…	68	13270.191176	0.287383	0.469567	0.530433	35.491176	0.427687	…	0.928458	4.066176	3.721872e+05	1166.294118	0.461507	0.655981	0.318585	0.238498	36.191176	989.095314

[4 rows x 35 columns]

[28]:
```
# visualise the clsutering result
fig, ax = plt.subplots()
sns.scatterplot(data=nyc_cluster, x='Pop_Total_25_44',␣
 ↪y='Median_Household_Income', hue='cluster', palette='deep')

avg_income = nyc_cluster['Median_Household_Income'].mean()
plt.axhline(y=avg_income, color='r', linestyle='--')

plt.legend()
plt.title('Clustering Result of NTA')
plt.xlabel('Total Population Age 25-44')
plt.ylabel('Median Household Income')
```

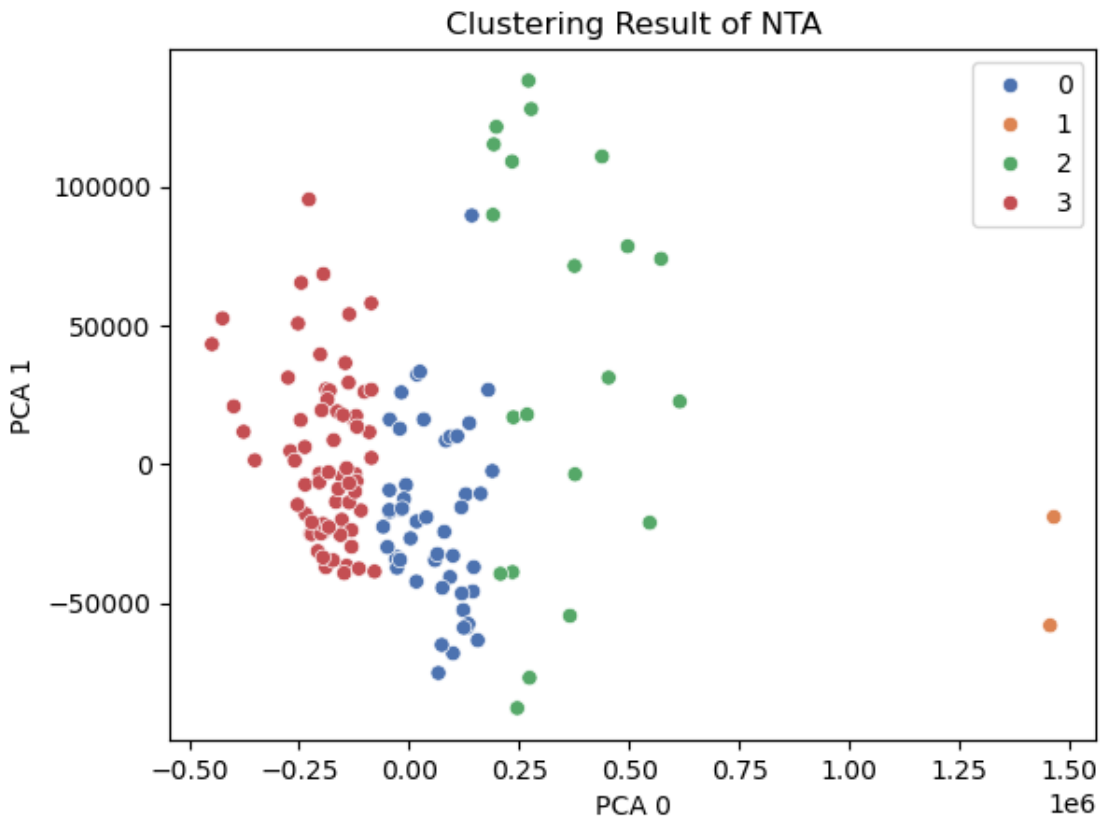
[28]:
```
Text(0, 0.5, 'Median Household Income')
```



[51]:
```
# visualise the clsutering result using PCA

df_pca = pd.DataFrame(pca_features)
df_pca['cluster'] = kmeans.labels_

fig, ax = plt.subplots()
```

```
sns.scatterplot(data=df_pca, x=0, y=1, hue='cluster', palette='deep')

plt.legend()
plt.title('Clustering Result of NTA')
plt.xlabel('PCA 0')
plt.ylabel('PCA 1')
```

[51]: Text(0, 0.5, 'PCA 1')



[29]:
```
# for each attribute give me the label that has the highest average
best_cluster_per_feature = {}
for col in df_results.columns[3:]:
    if col != 'Avg_shop_count':
        max_index = df_results[col].idxmax()
        cluster = df_results.loc[max_index]['cluster']
    else:
        min_index = df_results[col].idxmin()
        cluster = df_results.loc[min_index]['cluster']
    print(f"Variable: {col} Cluster: {cluster}")
    best_cluster_per_feature[col] = cluster
```

```
Variable: Num_NTA Cluster: 3
Variable: Avg_Pop_Total_25_44 Cluster: 2
Variable: Avg_Pop_Pct_25_44 Cluster: 2
Variable: Avg_Pop_Male Cluster: 0
Variable: Avg_Pop_Female Cluster: 1
Variable: Avg_Median_Age Cluster: 1
Variable: Avg_Pop_Pct_Hisp_Latin Cluster: 3
Variable: Avg_Pop_Pct_Non_Hisp_Latin Cluster: 1
Variable: Avg_Pop_Pct_White Cluster: 1
Variable: Avg_Pop_Pct_Black Cluster: 3
Variable: Avg_Pop_Pct_American Cluster: 3
Variable: Avg_Pop_Pct_Asia Cluster: 0
Variable: Avg_Pop_Pct_Native Cluster: 3
Variable: Avg_Pop_Pct_Employ Cluster: 2
Variable: Avg_Pop_Pct_Employ_Salary Cluster: 1
Variable: Avg_Pop_Pct_Household_Earn_75_99 Cluster: 0
Variable: Avg_Pop_Pct_Family_Earn_75_99 Cluster: 0
Variable: Avg_Median_Household_Income Cluster: 1
Variable: Avg_Median_Family_Income Cluster: 1
Variable: Avg_Median_Earning Cluster: 1
Variable: Avg_People_Per_Parking Cluster: 0
Variable: Avg_Pop_Pct_Healthy Cluster: 1
Variable: Avg_Pct_Occupied_House Cluster: 3
Variable: Avg_Median_Room Cluster: 0
Variable: Avg_Median_House Cluster: 1
Variable: Avg_Median_Rent Cluster: 1
Variable: Avg_Pop_Pct_With_Car Cluster: 0
Variable: Avg_Pop_Pct_with_Family Cluster: 3
Variable: Avg_Pop_Pct_with_Couple Cluster: 1
Variable: Avg_Pop_Pct_With_Degree Cluster: 1
Variable: Avg_Shop_Count Cluster: 1
Variable: Avg_People_Per_Shop Cluster: 3
```

[30]:
```
best_cluster = Counter(best_cluster_per_feature.values())
best_cluster = max(best_cluster, key=best_cluster.get)
best_cluster
```

[30]:
```
1
```

[31]:
```
nyc_cluster_result = nyc_cluster[nyc_cluster.cluster == best_cluster]
nyc_cluster_result = nyc_cluster_result.reset_index()
nyc_cluster_result.head(5)
```

[31]:

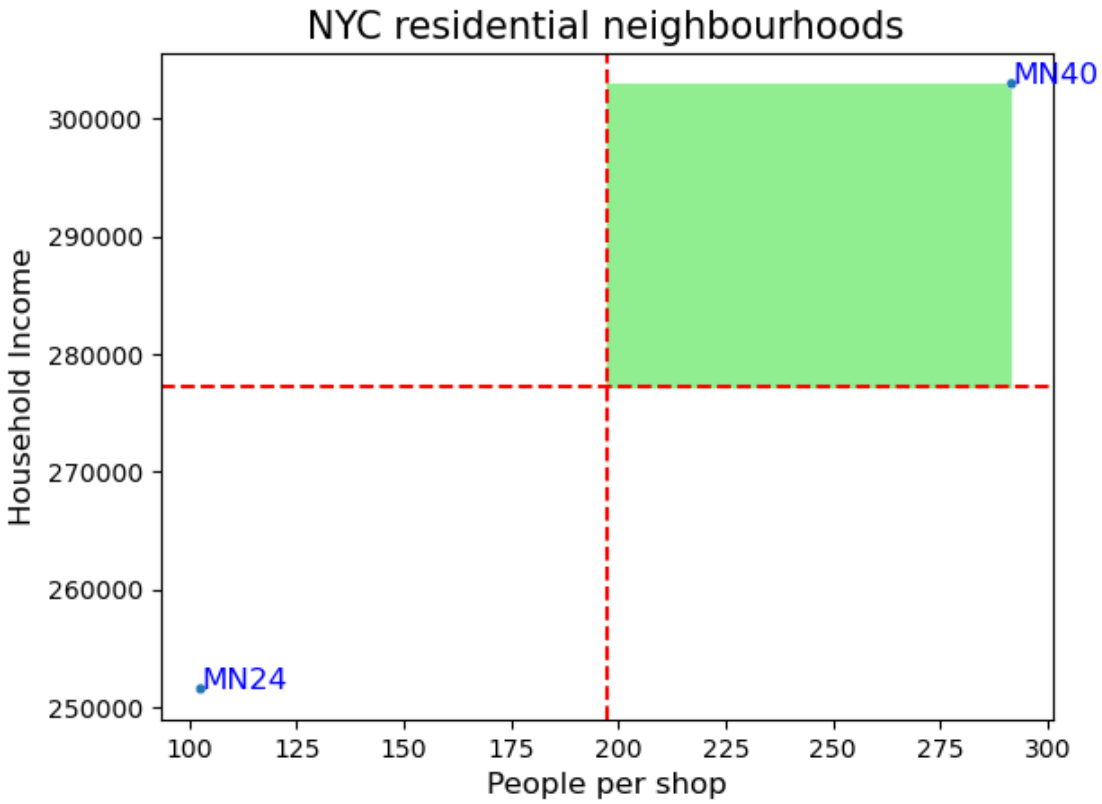
	index	Pop_Total_25_44	Pop_Pct_25_44	Pop_Male	Pop_Female	Median_Age	\
0	60	17649	0.410642	0.497196	0.502804	36.1	
1	62	14871	0.253802	0.440599	0.559401	47.9	

```
   Pop_Pct_Hisp_Latin  Pop_Pct_Non_Hisp_Latin  Pop_Pct_White  Pop_Pct_Black  \
0            0.066056                0.933944       0.652644       0.023802
1            0.070486                0.929514       0.836363       0.010274

   …  Pop_Pct_With_Car  Pop_Pct_with_Family  Pop_Pct_with_Couple  \
0  …           0.256548             0.427970             0.355533
1  …           0.352811             0.514452             0.453213

   Pop_Pct_With_Degree  Shop_Count  People_Per_Shop  GeoID  cluster  \
0             0.746168         172       102.610465   MN24        1
1             0.844983          51       291.588235   MN40        1

                                GeogName    Borough
0  SoHo-TriBeCa-Civic Center-Little Italy  Manhattan
1           Upper East Side-Carnegie Hill  Manhattan

[2 rows x 36 columns]
```

[40]:
```
# https://towardsdatascience.com/store-locations-d1025df22865
def scatter_text(x, y, text_column, data, title, xlabel, ylabel):
    p1 = sns.scatterplot(x=x, y=y, data=data, size=8, legend=False)

    texts = [p1.text(data[x][line], data[y][line],
             data[text_column][line], horizontalalignment='left',
             size='large', color='blue') for line in range(0,data.shape[0])]

    plt.title(title, size=15)
    plt.xlabel(xlabel, size=12)
    plt.ylabel(ylabel, size=12)

    return p1

# Using the average people per shop and average income as the threshold
avg_people_per_shop = nyc_cluster_result['People_Per_Shop'].mean()
max_people_per_shop = nyc_cluster_result['People_Per_Shop'].max()
avg_income = nyc_cluster_result['Median_Household_Income'].mean()
max_income = nyc_cluster_result['Median_Household_Income'].max()

plt.axhline(y=avg_income, color='r', linestyle='--')
plt.axvline(x=avg_people_per_shop, color='r', linestyle='--')

# Highlight the optimal quadrant
plt.fill_between(x=np.arange(avg_people_per_shop, max_people_per_shop),␣
 ↪y1=avg_income, y2=max_income, color='lightgreen')

# Draw and label the neighbourhood points
scatter_text('People_Per_Shop', 'Median_Household_Income', 'GeoID',
```

```
            data = nyc_cluster_result,
            title = 'NYC residential neighbourhoods',
            xlabel = 'People per shop',
            ylabel = 'Household Income')
```

[40]:
```
<Axes: title={'center': 'NYC residential neighbourhoods'}, xlabel='People per
shop', ylabel='Household Income'>
```



[33]:
```
top_result = nyc_cluster_result.loc[nyc_cluster_result.People_Per_Shop.idxmax()]
top_result
```

[33]:

index	62
Pop_Total_25_44	14871
Pop_Pct_25_44	0.253802
Pop_Male	0.440599
Pop_Female	0.559401
Median_Age	47.9
Pop_Pct_Hisp_Latin	0.070486
Pop_Pct_Non_Hisp_Latin	0.929514
Pop_Pct_White	0.836363
Pop_Pct_Black	0.010274

```
Pop_Pct_American                                    0.000222
Pop_Pct_Asia                                        0.061407
Pop_Pct_Native                                           0.0
Pop_Pct_Employ                                      0.612262
Pop_Pct_Employ_Salary                               0.830222
Pop_Pct_Household_Earn_75_99                        0.099536
Pop_Pct_Family_Earn_75_99                           0.064134
Median_Household_Income                            302939.0
Median_Family_Income                               200000.0
Median_Earning                                      93707.0
People_Per_Parking                                 37.210269
Pop_Pct_Healthy                                     0.979035
Pct_Occupied_House                                  0.772489
Median_Room                                              4.3
Median_House                                     2000000.0
Median_Rent                                         2251.0
Pop_Pct_With_Car                                    0.352811
Pop_Pct_with_Family                                 0.514452
Pop_Pct_with_Couple                                 0.453213
Pop_Pct_With_Degree                                 0.844983
Shop_Count                                                51
People_Per_Shop                                   291.588235
GeoID                                                   MN40
cluster                                                    1
GeogName                          Upper East Side-Carnegie Hill
Borough                                             Manhattan
Name: 1, dtype: object
```

[34]:
```
# Top 10 recommendations sorted by opportunity value
nyc_cluster_result.sort_values('People_Per_Shop', ascending=False)[['GeoID',␣
 ↪'GeogName', 'Borough', 'Median_Household_Income', 'People_Per_Shop']].head()
```

[34]:
```
  GeoID                                GeogName    Borough  \
1  MN40          Upper East Side-Carnegie Hill  Manhattan
0  MN24  SoHo-TriBeCa-Civic Center-Little Italy  Manhattan

   Median_Household_Income  People_Per_Shop
1                 302939.0       291.588235
0                 251546.0       102.610465
```

6 Plot the Recommendation on Map

[35]:
```
# filter the list of shops in the recommended area
df_bus_shop = df_bus[df_bus['Industry'].isin(['Stoop Line Stand', 'General␣
 ↪Vendor'])]
df_bus_shop = df_bus_shop[['Address Borough', 'NTA', 'Business Name',␣
 ↪'Industry', 'Location']]
df_bus_shop = df_bus_shop[df_bus_shop['NTA'] == top_result.GeoID]

print("Numer of potential competitors: ", df_bus_shop.shape[0])
df_bus_shop.head(5)
```

```
Numer of potential competitors:  51
```

[35]:

	Address Borough	NTA	Business Name	Industry	Location
27186	Manhattan	MN40	Dean & DeLuca	Stoop Line Stand	(40.78129989454938, -73.95990144939537)
27521	Manhattan	MN40	89TH ST. MARKET INC.	Stoop Line Stand	(40.78060823292779, -73.95269467250661)
59148	Manhattan	MN40	LEE, HYUNKYU	Stoop Line Stand	(40.77437746126572, -73.9593098884484)
59452	Manhattan	MN40	HMS FOOD CORP.	Stoop Line Stand	(40.763724869216475, -73.96710656691472)
62356	Manhattan	MN40	1200 THIRD AVE. CORP.	Stoop Line Stand	(40.76818616406079, -73.9617469087358)

[42]:
```
# Upper East Side-Carnegie Hill
top_location = (40.784441381713364, -73.95504886126758)

# https://locations.traderjoes.com/ny/new-york/
trade_joes_locs = [
    (40.77868573574113, -73.98165974388836),
    (40.7422363888565, -73.9938099666359),
    (40.74220387389157, -73.99384215314335),
    (40.71662851329304, -73.98622463224902),
    (40.808942800620464, -73.9447919734825),
    (40.74404302923888, -73.97841446875535),
    (40.72588669406502, -74.00444429177101),
    (40.73346852919018, -73.9875103647856),
    (40.759841323514586, -73.9612664819784),
    (40.79093941325663, -73.96869388621614)
]

mapit = folium.Map(top_location, zoom_start=12) #initial location
folium.Marker(location=top_location,
```

```
                tooltip="New Trader Joe's",
                icon=folium.Icon(color='green')).add_to(mapit)
for coord in df_bus_shop['Location']:
    coord = ast.literal_eval(coord)
    folium.Marker(location=[coord[0], coord[1]],
                  tooltip='Competitor',
                  icon=folium.Icon(color='red')).add_to(mapit)
for coord in trade_joes_locs:
    folium.Marker(location=[coord[0], coord[1]],
                  tooltip="Trader Joe's",
                  icon=folium.Icon(color='blue')).add_to(mapit)
mapit.save("recommended_location.html")
```

[43]:
```
# check if the location is in the FRESH Food Stores Zoning Boundaries
# https://data.cityofnewyork.us/Business/FRESH-Food-Stores-Zoning-Boundaries/
 ↪w9uz-8epq

with open('./data/FRESH Food Stores Zoning Boundaries.geojson', 'r') as f:
    js = json.load(f)

point = Point(40.784441381713364, -73.95504886126758)

for feature in js['features']:

    polygon = shape(feature['geometry'])

    if polygon.contains(point):
        print ('Found containing polygon:', feature)

# not return any output means that the Upper East Side-Carnegie Hill is not in␣
 ↪FRESH Food Area
```

[ ]:
```

```