

Final Capstone Project

IBM Data Science Professional Certificate

Matthew G. Sullivan

Section 1 - About this Report

Introduction

This notebook is the final assignment for the Capstone course for the IBM Data Science Professional Certificate. The IBM Data Science Professional Certificate Course immerses students in the topic of data science, focusing on technologies utilized by Watson Studio on IBM Cloud. The course teaches how to use Python, Jupyter Notebooks and related technologies to retrieve, format and analyze data using numerous statistical and machine learning models, and report the data such that it can be utilized by stakeholders for particular purposes.

In this assignment, we are to use the Places API from FourSquare to analyze location data for particular neighborhoods and apply that to assist solving a problem identified by the student. This report contains my final assignment for this course and program

Business Problem

The Parkrose neighborhood in Portland, OR has a plethora of available business spaces. The Parkrose Neighborhood Association has been coordinating with the Parkrose Business Council to attract new businesses, but that has had mixed results.

- Some of the new businesses that have opened have languished or closed, resulting in less increases in business density than targeted
- Existing businesses are not seeing benefit in increased sales from any changes in business density
- Parking has become more scarce
- Neighborhood residents complain that they have not seen a noticeable improvement to quality of life from the presence of these businesses

The two organizations have decided to halt the current initiatives and engage in a more data driven approach and a mission with increased clarity.

The objectives of this approach will be:

- To improve the quality of life residents of the neighborhood
- To attract businesses to Parkrose which better serve the needs of the residents
- To improve the economy by increasing the amount of income generated in Parkrose which stays in the neighborhood
- To grow the economy sustainably, in a way which does not impact shared resources that are depended upon, such as on-street parking
- To create a feedback cycle which attracts residents who will spend locally and attract premium employers

Considering these goals, the organizations have decided to narrow in on metrics scoring the pedestrian, bike and public transit services in the area. The overall hypotheses is that improving these qualities will result in attracting more upwardly mobile residents, less contention for decreasing availability of parking, less overall driving, improved health and overall improved ratings on quality of life from members of the community.

This report will help provide the necessary data for these plans. The questions this report will answer is:

1. Which Portland neighborhoods are classified as similar to Parkrose based demographic clustering
2. How does Parkrose compare in walkability to the other neighborhoods in that cluster
3. What are the most common non-residential location categories in neighborhoods of the same cluster, and how does Parkrose compare
4. What are the most common non-residential location categories in neighborhoods with higher walk \ bike \ transit scores

Data

The demographic data used in this report will include

- % of total residents and gender split in each age range
- % of 1 person households and the gender split of each
- % of family households with married couples
- % of non-family households
- % Households owned with a mortgage or a loan
- % Households owned free and clear
- % Households renter occupied

The demographic data will be a CSV file compiled from data available at this address:

<https://www.portlandoregon.gov/civic/article/376383> (<https://www.portlandoregon.gov/civic/article/376383>)

The data source for walk, bike and transit scores will be scraped from this web site: <https://www.pdxmonthly.com/articles/2018/3/27/portland-neighborhoods-by-the-numbers-2018-the-city> (<https://www.pdxmonthly.com/articles/2018/3/27/portland-neighborhoods-by-the-numbers-2018-the-city>)

The data source for the location information in the report will be the Places API from FourSquare. The data from that data source which will be used in this analysis includes:

- The name of the place
- The geographic coordinates of the place
- The category of the place

Caveats, Assumptions and Disclaimers

- The veracity of this data, particularly the demographic data, has not been verified.
- Some demographic data is intentionally left out. The hypotheses is that racial data is not as important as age data and household ownership status when determining the types of businesses that encourage car-free transportation. It may be worth conducting future analysis to test this hypotheses.
- Additional demographic data may be available to make a more robust study on this subject. It would be worthwhile to re these results with those from studies using additional or different data points.
- The original data for the demographic CSV was in an Excel. The data has been pre-formatted to remove columns and categorical headers. While this could be done in Pandas, it was quicker and easier to do so in Excel and made it possible to do more within a limited timeframe.
- Walk scores, bikeability scores and transit scores are already calculated using undisclosed algorithms, and accepted at face value

Section 2 - Methodology

Methodology

The primary approach to this analysis and report is clustering and comparison. There will be two cluster determinations using the K-Means algorithm, and based on these clusters there will be a comparison using basic means.

1. Clustering will be performed to determine undefined demographic classification for Portland neighborhoods. The intent is to leverage multi-dimensional comparison to discover patterns of similar neighborhoods. This determination will allow us to see which neighborhoods are demographically similar based on the data collected.
2. Clustering will be used to determine which neighborhoods are similar in terms the top categories of businesses located in the neighborhood
3. The walk, bike and transit scores for each neighborhood will be compared to determine how Parkrose compares in terms of car-free transportation options
4. Clustering will be used to determine which neighborhoods are similar in terms of most common place categories
5. The walk, bike and transit scores for each neighborhood will be compared to determine if there is a correlation with place categories and walkability
6. The most common place categories for neighborhoods of the same demographic clusters with higher walk/bike/transit scores will be compared to the most common place categories for Parkrose
7. All of this will be reviewed for apparent meaning

Caveats, Assumptions and Disclaimers

- It is assumed that having the right businesses nearby results in less transportation by car. There is not likely to be a causative relationship there. Even if a strong correlation is found, additional research and analysis is necessary there
- There are other aspects to place categories that could be considered. The most obvious is business density. The most common category metric was chosen because there are more options to change that by attracting particular categories of businesses / places. Obviously, any chamber of commerce would want to increase business density as much as possible, but it is more easily said than done
- It is assumed that decreasing transportation by car would increase the quality of life. That is highly debatable, and may not be easy to empirically prove or disprove.
- There are other clustering and classification algorithms that could be used. K-Means was selected because much of the work needed to employ this algorithm was completed in prior work. A more robust analysis would cross-compared analyses using alternative algorithms and approaches
- Some amount of subjective reasoning would be used to determine the meaning of the data discovered through this analysis. This introduces risks from bias and unfounded conclusions

Part 1 - Data Import and Wrangling

Demographic Data Import

First we bring in the data necessary. A CSV containing demographic data has already been uploaded to this project. Here is the code for importing the CSV as a Pandas dataframe.

```

In [17]: import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_3feabad67fd94abf8d6d1eda273012e2 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='7n0p6mmrfI9ixzLsn00BMT01tmAdku03WNuyJUbCtZmW',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_3feabad67fd94abf8d6d1eda273012e2.get_object(Bucket='ibmdatasciencecapstonproject-donotdelete-pr-jxbz6zjdb6uts0',Key='portland_neighborhoods_census_data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_demographics = pd.read_csv(body)
df_demographics.head()

```

Out[17]:

	Unnamed: 0	population_total	population_male_under_5	population_female_Under_5	population_male_5_to_9	population_female_
0	NEIGHBORHOOD ASSOCIATION	P0010001	P0120003	P0120027	P0120004	P0120004
1	ALAMEDA	5214	217	179	214	
2	ARBOR LODGE	6153	173	180	166	
3	ARDENWALD-JOHNSON CREEK	4748	141	149	143	
4	ARGAY	6006	205	179	201	

5 rows × 52 columns



Demographic Data Cleanup

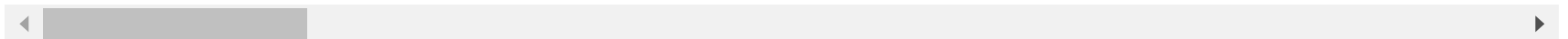
Next, lets rename column 0, set it as the index, drop the first row, and drop NaN rows

```
In [18]: df_demographics.rename(columns={'Unnamed: 0': 'neighborhood'}, inplace=True)
df_demographics.set_index('neighborhood', inplace=True)
df_demographics=df_demographics.drop(df_demographics.index[0])
df_demographics=df_demographics.dropna()
df_demographics.head()
```

Out[18]:

	population_total	population_male_under_5	population_female_Under_5	population_male_5_to_9	population_female_5_to_9
neighborhood					
ALAMEDA	5214	217	179	214	183
ARBOR LODGE	6153	173	180	166	140
ARDENWALD-JOHNSON CREEK	4748	141	149	143	121
ARGAY	6006	205	179	201	175
ARLINGTON HEIGHTS	718	33	21	26	27

5 rows × 51 columns



```
In [112]: df_demographics.columns
```

```
Out[112]: Index(['population_total', 'population_male_under_5',
                'population_female_Under_5', 'population_male_5_to_9',
                'population_female_5_to_9', 'population_male_10_to_14',
                'population_female_10_to_14', 'population_male_15_to_17',
                'population_female_15_to_17', 'population_male_18_to_19',
                'population_female_18_and_19', 'population_male_20_to_24',
                'population_female_20_to_24', 'population_male_25_to_29',
                'population_female_25_to_29', 'population_male_30_to_34',
                'population_female_30_to_34', 'population_male_35_to_39',
                'population_female_35_to_39', 'population_male_40_to_44',
                'population_female_40_to_44', 'population_male_45_to_49',
                'population_female_45_to_49', 'population_male_50_to_54',
                'population_female_50_to_54', 'population_male_55_to_59',
                'population_female_55_to_59', 'population_male_60_to_64',
                'population_female_60_to_64', 'population_male_65_to_69',
                'population_female_65_to_69', 'population_male_70_to_74',
                'population_female_70_to_74', 'population_male_75_to_79',
                'population_female_75_to_79', 'population_male_80_to_84',
                'population_female_80_to_84', 'population_male_85_and_over',
                'population_female_85_and_over', 'multiperson_nonfamily_households',
                'home_owned_with_mortgage_or_loan', 'home_owned_free_and_clear',
                'home_rented', 'all_one_person_household', 'all_family_household',
                'total_households', 'all_non_family_household', 'WALK_SCORE'],
                dtype='object')
```

NOTE: These column names are very long, and it could be worthwhile to make them short for better display in tabular form.

The columns need to be a numeric type.

TIP This may seem like a LOT of code. Excel is your friend here. You can output the column names using `df.columns`, and then copy that output into excel so that you can use string concatenation to build the code you want.


```

In [19]: df_demographics['population_total'] =pd.to_numeric(df_demographics['population_total'])
df_demographics['population_male_under_5'] =pd.to_numeric(df_demographics['population_male_under_5'])
df_demographics['population_female_Under_5'] =pd.to_numeric(df_demographics['population_female_Under_5'])
df_demographics['population_male_5_to_9'] =pd.to_numeric(df_demographics['population_male_5_to_9'])
df_demographics['population_female_5_to_9'] =pd.to_numeric(df_demographics['population_female_5_to_9'])
df_demographics['population_male_10_to_14'] =pd.to_numeric(df_demographics['population_male_10_to_14'])
df_demographics['population_female_10_to_14'] =pd.to_numeric(df_demographics['population_female_10_to_14'])
df_demographics['population_male_15_to_17'] =pd.to_numeric(df_demographics['population_male_15_to_17'])
df_demographics['population_female_15_to_17'] =pd.to_numeric(df_demographics['population_female_15_to_17'])
df_demographics['population_male_18_to_19'] =pd.to_numeric(df_demographics['population_male_18_to_19'])
df_demographics['population_female_18_and_19'] =pd.to_numeric(df_demographics['population_female_18_and_19'])
df_demographics['population_male_20_to_24'] =pd.to_numeric(df_demographics['population_male_20_to_24'])
df_demographics['population_female_20_to_24'] =pd.to_numeric(df_demographics['population_female_20_to_24'])
df_demographics['population_male_25_to_29'] =pd.to_numeric(df_demographics['population_male_25_to_29'])
df_demographics['population_female_25_to_29'] =pd.to_numeric(df_demographics['population_female_25_to_29'])
df_demographics['population_male_30_to_34'] =pd.to_numeric(df_demographics['population_male_30_to_34'])
df_demographics['population_female_30_to_34'] =pd.to_numeric(df_demographics['population_female_30_to_34'])
df_demographics['population_male_35_to_39'] =pd.to_numeric(df_demographics['population_male_35_to_39'])
df_demographics['population_female_35_to_39'] =pd.to_numeric(df_demographics['population_female_35_to_39'])
df_demographics['population_male_40_to_44'] =pd.to_numeric(df_demographics['population_male_40_to_44'])
df_demographics['population_female_40_to_44'] =pd.to_numeric(df_demographics['population_female_40_to_44'])
df_demographics['population_male_45_to_49'] =pd.to_numeric(df_demographics['population_male_45_to_49'])
df_demographics['population_female_45_to_49'] =pd.to_numeric(df_demographics['population_female_45_to_49'])
df_demographics['population_male_50_to_54'] =pd.to_numeric(df_demographics['population_male_50_to_54'])
df_demographics['population_female_50_to_54'] =pd.to_numeric(df_demographics['population_female_50_to_54'])
df_demographics['population_male_55_to_59'] =pd.to_numeric(df_demographics['population_male_55_to_59'])
df_demographics['population_female_55_to_59'] =pd.to_numeric(df_demographics['population_female_55_to_59'])
df_demographics['population_male_60_to_64'] =pd.to_numeric(df_demographics['population_male_60_to_64'])
df_demographics['population_female_60_to_64'] =pd.to_numeric(df_demographics['population_female_60_to_64'])
df_demographics['population_male_65_to_69'] =pd.to_numeric(df_demographics['population_male_65_to_69'])
df_demographics['population_female_65_to_69'] =pd.to_numeric(df_demographics['population_female_65_to_69'])
df_demographics['population_male_70_to_74'] =pd.to_numeric(df_demographics['population_male_70_to_74'])
df_demographics['population_female_70_to_74'] =pd.to_numeric(df_demographics['population_female_70_to_74'])
df_demographics['population_male_75_to_79'] =pd.to_numeric(df_demographics['population_male_75_to_79'])
df_demographics['population_female_75_to_79'] =pd.to_numeric(df_demographics['population_female_75_to_79'])
df_demographics['population_male_80_to_84'] =pd.to_numeric(df_demographics['population_male_80_to_84'])
df_demographics['population_female_80_to_84'] =pd.to_numeric(df_demographics['population_female_80_to_84'])
df_demographics['population_male_85_and_over'] =pd.to_numeric(df_demographics['population_male_85_and_over'])
df_demographics['population_female_85_and_over'] =pd.to_numeric(df_demographics['population_female_85_and_ove
r'])
df_demographics['one_person_male_household'] =pd.to_numeric(df_demographics['one_person_male_household'])
df_demographics['one_person_female_household'] =pd.to_numeric(df_demographics['one_person_female_household'])

```

```
df_demographics['multiperson_family_husband_wife'] =pd.to_numeric(df_demographics['multiperson_family_husband_wife'])
df_demographics['multiperson_husband_wife_family_with_own_children_under_18_years'] =pd.to_numeric(df_demographics['multiperson_husband_wife_family_with_own_children_under_18_years'])
df_demographics['multiperson_male_householder_no_wife_present'] =pd.to_numeric(df_demographics['multiperson_male_householder_no_wife_present'])
df_demographics['multiperson_male_householder_no_wife_present_with_own_children_under_18_years'] =pd.to_numeric(df_demographics['multiperson_male_householder_no_wife_present_with_own_children_under_18_years'])
df_demographics['multiperson_female_householder_no_husband_present'] =pd.to_numeric(df_demographics['multiperson_female_householder_no_husband_present'])
df_demographics['multiperson_female_householder_no_husband_present_with_own_children_under_18_years'] =pd.to_numeric(df_demographics['multiperson_female_householder_no_husband_present_with_own_children_under_18_years'])
df_demographics['multiperson_nonfamily_households'] =pd.to_numeric(df_demographics['multiperson_nonfamily_households'])
df_demographics['home_owned_with_mortgage_or_loan'] =pd.to_numeric(df_demographics['home_owned_with_mortgage_or_loan'])
df_demographics['home_owned_free_and_clear'] =pd.to_numeric(df_demographics['home_owned_free_and_clear'])
df_demographics['home_rented'] =pd.to_numeric(df_demographics['home_rented'])
```

Let's take a look to make sure the datatypes are right.

In [20]: `df_demographics.dtypes`

```
Out[20]: population_total int64
population_male_under_5 int64
population_female_Under_5 int64
population_male_5_to_9 int64
population_female_5_to_9 int64
population_male_10_to_14 int64
population_female_10_to_14 int64
population_male_15_to_17 int64
population_female_15_to_17 int64
population_male_18_to_19 int64
population_female_18_and_19 int64
population_male_20_to_24 int64
population_female_20_to_24 int64
population_male_25_to_29 int64
population_female_25_to_29 int64
population_male_30_to_34 int64
population_female_30_to_34 int64
population_male_35_to_39 int64
population_female_35_to_39 int64
population_male_40_to_44 int64
population_female_40_to_44 int64
population_male_45_to_49 int64
population_female_45_to_49 int64
population_male_50_to_54 int64
population_female_50_to_54 int64
population_male_55_to_59 int64
population_female_55_to_59 int64
population_male_60_to_64 int64
population_female_60_to_64 int64
population_male_65_to_69 int64
population_female_65_to_69 int64
population_male_70_to_74 int64
population_female_70_to_74 int64
population_male_75_to_79 int64
population_female_75_to_79 int64
population_male_80_to_84 int64
population_female_80_to_84 int64
population_male_85_and_over int64
population_female_85_and_over int64
one_person_male_household int64
one_person_female_household int64
multiperson_family_husband_wife int64
multiperson_husband_wife_family_with_own_children_under_18_years int64
```

multiperson_male_householder_no_wife_present	int64
multiperson_male_householder_no_wife_present_with_own_children_under_18_years	int64
multiperson_female_householder_no_husband_present	int64
multiperson_female_householder_no_husband_present_with_own_children_under_18_years	int64
multiperson_nonfamily_households	int64
home_owned_with_mortgage_or_loan	int64
home_owned_free_and_clear	int64
home_rented	int64
dtype: object	

Next we need to convert each of the columns to percentages. The reason we are interested in percentage over count is that neighborhoods are not of uniform size or populations.

If, for example, we were wanting to know how many copies a book would be needed to send one to every kids aged 5 - 9, the count would be useful. If we were trying to understand characteristically how "age 5-9" a particular neighborhood is, we need percentages.

First the age / gender ranges.

TIP Again, this appears to be a lot of code, but it can be expedited using Excel and string concatenation. There are doubtless more elegant ways to do this, but this code works and is very easy to understand

```
In [21]: df_demographics['population_male_under_5'] =df_demographics['population_male_under_5'] /df_demographics['population_total'] * 100
df_demographics['population_female_Under_5'] =df_demographics['population_female_Under_5'] /df_demographics['population_total'] * 100
df_demographics['population_male_5_to_9'] =df_demographics['population_male_5_to_9'] /df_demographics['population_total'] * 100
df_demographics['population_female_5_to_9'] =df_demographics['population_female_5_to_9'] /df_demographics['population_total'] * 100
df_demographics['population_male_10_to_14'] =df_demographics['population_male_10_to_14'] /df_demographics['population_total'] * 100
df_demographics['population_female_10_to_14'] =df_demographics['population_female_10_to_14'] /df_demographics['population_total'] * 100
df_demographics['population_male_15_to_17'] =df_demographics['population_male_15_to_17'] /df_demographics['population_total'] * 100
df_demographics['population_female_15_to_17'] =df_demographics['population_female_15_to_17'] /df_demographics['population_total'] * 100
df_demographics['population_male_18_to_19'] =df_demographics['population_male_18_to_19'] /df_demographics['population_total'] * 100
df_demographics['population_female_18_and_19'] =df_demographics['population_female_18_and_19'] /df_demographics['population_total'] * 100
df_demographics['population_male_20_to_24'] =df_demographics['population_male_20_to_24'] /df_demographics['population_total'] * 100
df_demographics['population_female_20_to_24'] =df_demographics['population_female_20_to_24'] /df_demographics['population_total'] * 100
df_demographics['population_male_25_to_29'] =df_demographics['population_male_25_to_29'] /df_demographics['population_total'] * 100
df_demographics['population_female_25_to_29'] =df_demographics['population_female_25_to_29'] /df_demographics['population_total'] * 100
df_demographics['population_male_30_to_34'] =df_demographics['population_male_30_to_34'] /df_demographics['population_total'] * 100
df_demographics['population_female_30_to_34'] =df_demographics['population_female_30_to_34'] /df_demographics['population_total'] * 100
df_demographics['population_male_35_to_39'] =df_demographics['population_male_35_to_39'] /df_demographics['population_total'] * 100
df_demographics['population_female_35_to_39'] =df_demographics['population_female_35_to_39'] /df_demographics['population_total'] * 100
df_demographics['population_male_40_to_44'] =df_demographics['population_male_40_to_44'] /df_demographics['population_total'] * 100
df_demographics['population_female_40_to_44'] =df_demographics['population_female_40_to_44'] /df_demographics['population_total'] * 100
df_demographics['population_male_45_to_49'] =df_demographics['population_male_45_to_49'] /df_demographics['population_total'] * 100
```

```

df_demographics['population_female_45_to_49'] =df_demographics['population_female_45_to_49'] /df_demographics
['population_total'] * 100
df_demographics['population_male_50_to_54'] =df_demographics['population_male_50_to_54'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_50_to_54'] =df_demographics['population_female_50_to_54'] /df_demographics
['population_total'] * 100
df_demographics['population_male_55_to_59'] =df_demographics['population_male_55_to_59'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_55_to_59'] =df_demographics['population_female_55_to_59'] /df_demographics
['population_total'] * 100
df_demographics['population_male_60_to_64'] =df_demographics['population_male_60_to_64'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_60_to_64'] =df_demographics['population_female_60_to_64'] /df_demographics
['population_total'] * 100
df_demographics['population_male_65_to_69'] =df_demographics['population_male_65_to_69'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_65_to_69'] =df_demographics['population_female_65_to_69'] /df_demographics
['population_total'] * 100
df_demographics['population_male_70_to_74'] =df_demographics['population_male_70_to_74'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_70_to_74'] =df_demographics['population_female_70_to_74'] /df_demographics
['population_total'] * 100
df_demographics['population_male_75_to_79'] =df_demographics['population_male_75_to_79'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_75_to_79'] =df_demographics['population_female_75_to_79'] /df_demographics
['population_total'] * 100
df_demographics['population_male_80_to_84'] =df_demographics['population_male_80_to_84'] /df_demographics['po
pulation_total'] * 100
df_demographics['population_female_80_to_84'] =df_demographics['population_female_80_to_84'] /df_demographics
['population_total'] * 100
df_demographics['population_male_85_and_over'] =df_demographics['population_male_85_and_over'] /df_demographi
cs['population_total'] * 100
df_demographics['population_female_85_and_over'] =df_demographics['population_female_85_and_over'] /df_demogr
aphics['population_total'] * 100

```

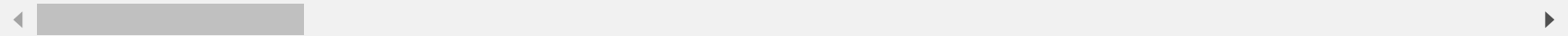
Let's see what that looks like:

In [22]: `df_demographics.head()`

Out[22]:

	population_total	population_male_under_5	population_female_Under_5	population_male_5_to_9	population_female_5_to_9
neighborhood					
ALAMEDA	5214	4.161872	3.433065	4.104334	3.509781
ARBOR LODGE	6153	2.811637	2.925402	2.697871	2.275313
ARDENWALD- JOHNSON CREEK	4748	2.969671	3.138163	3.011794	2.548441
ARGAY	6006	3.413253	2.980353	3.346653	2.913753
ARLINGTON HEIGHTS	718	4.596100	2.924791	3.621170	3.760446

5 rows × 51 columns



And let's double check that:

If the provided data adds up and the calculation is correct, this should result in a series containing numerous values of 100. If there are slight differences, than the issue is with numeric precision in Python. If there are big differences, the data provided is incomplete and we must consider than in our analysis


```
In [23]: df_demographics['population_male_under_5'] + df_demographics['population_female_Under_5'] + df_demographics[
'population_male_5_to_9'] \
+ df_demographics['population_female_5_to_9'] + df_demographics['population_male_10_to_14'] + df_demogr
aphics['population_female_10_to_14'] \
+ df_demographics['population_male_15_to_17'] + df_demographics['population_female_15_to_17'] + df_demogr
aphics['population_male_18_to_19'] \
+ df_demographics['population_female_18_and_19'] + df_demographics['population_male_20_to_24'] + df_demog
raphics['population_female_20_to_24'] \
+ df_demographics['population_male_25_to_29'] + df_demographics['population_female_25_to_29'] + df_demogr
aphics['population_male_30_to_34'] \
+ df_demographics['population_female_30_to_34'] + df_demographics['population_male_35_to_39'] + df_demogr
aphics['population_female_35_to_39'] \
+ df_demographics['population_male_40_to_44'] + df_demographics['population_female_40_to_44'] + df_demogr
aphics['population_male_45_to_49'] \
+ df_demographics['population_female_45_to_49'] + df_demographics['population_male_50_to_54'] + df_demogr
aphics['population_female_50_to_54'] \
+ df_demographics['population_male_55_to_59'] + df_demographics['population_female_55_to_59'] + df_demogr
aphics['population_male_60_to_64'] \
+ df_demographics['population_female_60_to_64'] + df_demographics['population_male_65_to_69'] + df_demogr
aphics['population_female_65_to_69'] \
+ df_demographics['population_male_70_to_74'] + df_demographics['population_female_70_to_74'] + df_demogr
aphics['population_male_75_to_79'] \
+ df_demographics['population_female_75_to_79'] + df_demographics['population_male_80_to_84'] + df_demogr
aphics['population_female_80_to_84'] \
+ df_demographics['population_male_85_and_over'] + df_demographics['population_female_85_and_over']
```

```

Out[23]: neighborhood
          ALAMEDA                100.0
          ARBOR LODGE            100.0
          ARDENWALD-JOHNSON CREEK 100.0
          ARGAY                  100.0
          ARLINGTON HEIGHTS       100.0
          ARNOLD CREEK            100.0
          ASHCREEK                100.0
          BEAUMONT-WILSHIRE        100.0
          BOISE                   100.0
          BRENTWOOD-DARLINGTON    100.0
          BRIDGETON               100.0
          BRIDLEMILE              100.0
          BROOKLYN                100.0
          BUCKMAN                 100.0
          CATHEDRAL PARK           100.0
          CENTENNIAL              100.0
          COLLINS VIEW            100.0
          CONCORDIA               100.0
          CRESTON-KENILWORTH      100.0
          CRESTWOOD               100.0
          CULLY                   100.0
          DOWNTOWN                100.0
          EAST COLUMBIA           100.0
          EASTMORELAND            100.0
          ELIOT                   100.0
          FAR SOUTHWEST           100.0
          FOREST PARK             100.0
          FOSTER-POWELL           100.0
          GLENFAIR                100.0
          GOOSE HOLLOW            100.0
          ...
          PARKROSE HEIGHTS        100.0
          PEARL                   100.0
          PIEDMONT                100.0
          PLEASANT VALLEY         100.0
          PORTSMOUTH              100.0
          POWELLHURST-GILBERT     100.0
          REED                    100.0
          RICHMOND                100.0
          ROSE CITY PARK          100.0
          ROSEWAY                 100.0
          RUSSELL                 100.0

```

SABIN	100.0
SELLWOOD-MORELAND	100.0
SOUTH BURLINGAME	100.0
SOUTH PORTLAND	100.0
SOUTH TABOR	100.0
SOUTHWEST HILLS	100.0
ST. JOHNS	100.0
SULLIVAN'S GULCH	100.0
SUMNER	100.0
SUNDERLAND	100.0
SUNNYSIDE	100.0
SYLVAN-HIGHLANDS	100.0
UNIVERSITY PARK	100.0
VERNON	100.0
WEST PORTLAND PARK	100.0
WILKES	100.0
WOODLAND PARK	100.0
WOODLAWN	100.0
WOODSTOCK	100.0

Length: 95, dtype: float64

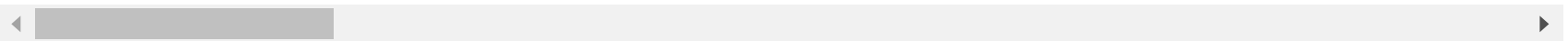
Perfect! So we now have the demographic data in percentage form. Next lets convert the household data to percentages. This is BARELY more complicated, because there is not a total household column, so we will want to start by creating it.

```
In [24]: df_demographics['all_one_person_household'] = df_demographics['one_person_male_household'] + df_demographics['one_person_female_household']
df_demographics['all_family_household'] = df_demographics['multiperson_family_husband_wife'] + df_demographics['multiperson_husband_wife_family_with_own_children_under_18_years'] + df_demographics['multiperson_male_householder_no_wife_present'] + df_demographics['multiperson_male_householder_no_wife_present_with_own_children_under_18_years'] + df_demographics['multiperson_female_householder_no_husband_present'] + df_demographics['multiperson_female_householder_no_husband_present_with_own_children_under_18_years'] + df_demographics['multiperson_nonfamily_households']
df_demographics['total_households'] = df_demographics['multiperson_nonfamily_households'] + df_demographics['all_one_person_household'] + df_demographics['all_family_household']
df_demographics.head()
```

Out[24]:

	population_total	population_male_under_5	population_female_Under_5	population_male_5_to_9	population_female_5_to_9
neighborhood					
ALAMEDA	5214	4.161872	3.433065	4.104334	3.509781
ARBOR LODGE	6153	2.811637	2.925402	2.697871	2.275313
ARDENWALD-JOHNSON CREEK	4748	2.969671	3.138163	3.011794	2.548441
ARGAY	6006	3.413253	2.980353	3.346653	2.913753
ARLINGTON HEIGHTS	718	4.596100	2.924791	3.621170	3.760446

5 rows × 54 columns



Let's also rename 'multiperson_nonfamily_households' for consistency

```
In [25]: df_demographics['all_non_family_household'] = df_demographics['multiperson_nonfamily_households']
```

Let's convert the counts to percentages

```
In [26]: df_demographics['all_one_person_household'] = df_demographics['all_one_person_household'] / df_demographics['total_households'] * 100
df_demographics['all_family_household'] = df_demographics['all_family_household'] / df_demographics['total_households'] * 100
df_demographics['all_non_family_household'] = df_demographics['all_non_family_household'] / df_demographics['total_households'] * 100
```

And finally let's drop the columns we aren't going to use

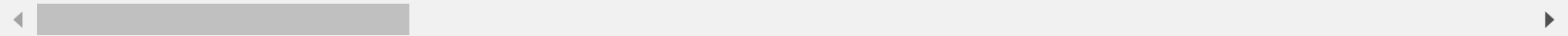
```
In [27]: df_demographics = df_demographics.drop(['one_person_male_household'], axis=1)
df_demographics = df_demographics.drop(['one_person_female_household'], axis=1)
df_demographics = df_demographics.drop(['multiperson_family_husband_wife'], axis=1)
df_demographics = df_demographics.drop(['multiperson_husband_wife_family_with_own_children_under_18_years'], axis=1)
df_demographics = df_demographics.drop(['multiperson_male_householder_no_wife_present'], axis=1)
df_demographics = df_demographics.drop(['multiperson_male_householder_no_wife_present_with_own_children_under_18_years'], axis=1)
df_demographics = df_demographics.drop(['multiperson_female_householder_no_husband_present'], axis=1)
df_demographics = df_demographics.drop(['multiperson_female_householder_no_husband_present_with_own_children_under_18_years'], axis=1)
```

In [28]: `df_demographics.head()`

Out[28]:

	population_total	population_male_under_5	population_female_Under_5	population_male_5_to_9	population_female_5_to_9
neighborhood					
ALAMEDA	5214	4.161872	3.433065	4.104334	3.509781
ARBOR LODGE	6153	2.811637	2.925402	2.697871	2.275313
ARDENWALD-JOHNSON CREEK	4748	2.969671	3.138163	3.011794	2.548441
ARGAY	6006	3.413253	2.980353	3.346653	2.913753
ARLINGTON HEIGHTS	718	4.596100	2.924791	3.621170	3.760446

5 rows × 47 columns



Walk Score Data Scraping

Next we need the data for walk score. The data is available from a website here: <https://www.pdxmonthly.com/articles/2018/3/27/portland-neighborhoods-by-the-numbers-2018-the-city> (<https://www.pdxmonthly.com/articles/2018/3/27/portland-neighborhoods-by-the-numbers-2018-the-city>).

The data is not available as a CSV, so it will need to be scraped. We will use BeautifulSoup to conduct the scraping.

In [29]: `!python -m pip install requests BeautifulSoup4`

```
Requirement already satisfied: requests in /opt/conda/envs/Python36/lib/python3.6/site-packages (2.21.0)
Requirement already satisfied: BeautifulSoup4 in /opt/conda/envs/Python36/lib/python3.6/site-packages (4.7.1)
Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from r
equests) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from requests) (3.0.4)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from requests) (1.24.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python36/lib/python3.6/site-packages (fr
om requests) (2019.6.16)
Requirement already satisfied: soupsieve>=1.2 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from B
eautifulSoup4) (1.7.1)
```

There are some libraries we will need

```
In [30]: import requests
from requests import get
from requests.exceptions import RequestException
from contextlib import closing
from bs4 import BeautifulSoup
import json
```

...and some functions we will use.

```
In [31]: def get_raw_xhtml(url):
        try:
            with closing(get(url, stream=True)) as resp:
                if is_xhtml(resp):
                    return resp.content
                else:
                    return None
        except RequestException as e:
            log_error('Error during requests to {0} : {1}'.format(url, str(e)))
            return None

def is_xhtml(resp):
    content_type = resp.headers['Content-Type'].lower()
    return (resp.status_code == 200
            and content_type is not None
            and content_type.find('html') > -1)
```

Next we create an object for the page.

```
In [32]: full_page_url = 'https://www.pdxmonthly.com/articles/2018/3/27/portland-neighborhoods-by-the-numbers-2018-the-city'
        page_html = get_raw_xhtml(full_page_url)
        page_parsed_html = BeautifulSoup(page_html, 'html.parser')
        body_div = page_parsed_html.find('div', {"class": "c-body"})
```

From the body we extract the table and read it to a Pandas dataframe

```
In [33]: data_table = body_div.find_all("table", {"class": "dataTable"})[0]
        df_pdxmonthly = pd.read_html(str(data_table))[0]
```


The last row of the scraped dataframe is a total row.


```
In [34]: df_pdxmonthly.loc[df_pdxmonthly.Neighborhood == 'PORTLAND TOTAL**']
```

Out[34]:

	Neighborhood	Average home sale price (\$)	Median home sale price (\$)	Average cost per square foot (\$)	Days on market (avg.)	Homes sold in 2017 (#)	Condo sales (%)	1-year median price change (2016– 2017) (%)	5-year median price change (2013– 2017) (%)	Distressed property sales (%)	...	Commute by public transit (%)	Commute by bike (%)	Commute by walkin (%)
95	PORTLAND TOTAL**	460277	400350	248	31	10095	17	5	40	2	...	12.1	6.5	6.

1 rows × 57 columns



-----so lets delete it

```
In [35]: df_pdxmonthly = df_pdxmonthly.drop(df_pdxmonthly.index[95])
```

Finally let's append the 'WALK_SCORE' column to the df_demographics dataframe

```
In [36]: walk_scores = df_pdxmonthly['Walk score'].tolist()
```

```
In [37]: df_demographics['WALK_SCORE'] = walk_scores
```

Let's check the results

```
In [38]: df_demographics['WALK_SCORE'].head()
```

```
Out[38]: neighborhood
ALAMEDA                65
ARBOR LODGE            72
ARDENWALD-JOHNSON CREEK 54
ARGAY                  45
ARLINGTON HEIGHTS      40
Name: WALK_SCORE, dtype: int64
```

Venue data acquisition

Next we will acquire the data we need from FourSquare. This includes the list of venues for a particular neighborhood, the category for each of the venues, and some other basic information.

First we will need some libraries installed.

```
In [39]: from geopy.geocoders import Nominatim
         from geopy.extra.rate_limiter import RateLimiter
         import numpy as np
         import math
```

Next we will need some functions declared

```

In [40]: #this needs an address
def get_location_object(place):
    geolocator = Nominatim(user_agent="matthewgsullivan_week3_assignment")
    geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)
    location = None
    try:
        location = geolocator.geocode(place)
    except:
        log_error("error encountered looking up {}".format(place))
    return location

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']
    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)
    return row_categories_sorted.index.values[0:num_top_venues]

def get_nearby_venues(names, latitudes, longitudes, radius=500, limit=100):

    CLIENT_ID = # your Foursquare ID
    CLIENT_SECRET = # your Foursquare Secret
    VERSION = '20180605' # Foursquare API version
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        if(not math.isnan(lat) or not math.isnan(lng)):
            # 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"][0]['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 = ['Neighborhood',
                        'Neighborhood Latitude',
                        'Neighborhood Longitude',
                        'Venue',
                        'Venue Latitude',
                        'Venue Longitude',
                        'Venue Category']

return(nearby_venues)

def get_sorted_venues_dataframe(df_grouped, num_top_venues = 10):
    indicators = ['st', 'nd', 'rd']

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

    # create a new dataframe
    neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
    neighborhoods_venues_sorted['Neighborhood'] = df_grouped['Neighborhood']

    for ind in np.arange(df_grouped.shape[0]):

```

```

neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(df_grouped.iloc[ind, :], num_to
p_venues)

return neighborhoods_venues_sorted

def log_error(e):
    print(e)

```

Let's create the dataframe that will hold the venues data

```

In [41]: neighborhoods_venues_columns = ['Neighborhood', 'Latitude', 'Longitude']
neighborhoods_venues_df = pd.DataFrame(columns=neighborhoods_venues_columns)
for neighborhood_name in df_demographics.index:
    neighborhood_location = get_location_object("{}, Portland, OR".format(neighborhood_name))
    neighborhood_location_latitude = neighborhood_location_longitude = None
    if(neighborhood_location is not None):
        neighborhood_location_latitude = neighborhood_location[1][0]
        neighborhood_location_longitude = neighborhood_location[1][1]
        neighborhoods_venue_row_df = {'Neighborhood':neighborhood_name, 'Latitude':neighborhood_location_latitude, 'Longitude':neighborhood_location_longitude}
        neighborhoods_venues_df = neighborhoods_venues_df.append(neighborhoods_venue_row_df, ignore_index=True)
    print("Done")

```

Done

Let's look at some of the data

```
In [42]: neighborhoods_venues_df.head()
```

```
Out[42]:
```

	Neighborhood	Latitude	Longitude
0	ALAMEDA	45.548631	-122.636481
1	ARBOR LODGE	45.571794	-122.690152
2	ARDENWALD-JOHNSON CREEK	45.458516	-122.627539
3	ARGAY	45.552830	-122.523204
4	ARLINGTON HEIGHTS	45.519496	-122.710667

And let's check the shape

```
In [43]: neighborhoods_venues_df.shape
```

```
Out[43]: (95, 3)
```

Drop any empty rows without a neighborhood name, latitude or longitude. This is important because we don't want to add items to the map that can't be displayed, or otherwise cause failures. Also, we can't look up venues for locations without latitude and longitude.

```
In [44]: neighborhoods_venues_df = neighborhoods_venues_df[neighborhoods_venues_df.Neighborhood != ""]  
neighborhoods_venues_df = neighborhoods_venues_df.dropna(axis = 0, how = 'any')
```

And let's look at the shape again.

```
In [45]: neighborhoods_venues_df.shape
```

```
Out[45]: (94, 3)
```

Displaying the neighborhoods on a map

Let's create a map to display the neighborhood locations. First we will need to install folium

In [46]: `!conda install -c conda-forge folium=0.5.0 --yes`

Solving environment: done

Package Plan

environment location: /opt/conda/envs/Python36

added / updated specs:

- folium=0.5.0

The following packages will be downloaded:

package	build		
ca-certificates-2019.6.16	hecc5488_0	145 KB	conda-forge
branca-0.3.1	py_0	25 KB	conda-forge
openssl-1.1.1c	h516909a_0	2.1 MB	conda-forge
altair-3.2.0	py36_0	770 KB	conda-forge
folium-0.5.0	py_0	45 KB	conda-forge
vincent-0.4.4	py_1	28 KB	conda-forge
certifi-2019.6.16	py36_1	149 KB	conda-forge
Total:		3.3 MB	

The following NEW packages will be INSTALLED:

altair:	3.2.0-py36_0	conda-forge
branca:	0.3.1-py_0	conda-forge
folium:	0.5.0-py_0	conda-forge
vincent:	0.4.4-py_1	conda-forge

The following packages will be UPDATED:

ca-certificates:	2019.5.15-1	--> 2019.6.16-hecc5488_0	conda-forge
certifi:	2019.6.16-py36_1	--> 2019.6.16-py36_1	conda-forge

The following packages will be DOWNGRADED:

openssl:	1.1.1c-h7b6447c_1	--> 1.1.1c-h516909a_0	conda-forge
----------	-------------------	-----------------------	-------------

Downloading and Extracting Packages

ca-certificates-2019 | 145 KB | ##### | 100%

branca-0.3.1	25 KB	#####	100%
openssl-1.1.1c	2.1 MB	#####	100%
altair-3.2.0	770 KB	#####	100%
folium-0.5.0	45 KB	#####	100%
vincent-0.4.4	28 KB	#####	100%
certifi-2019.6.16	149 KB	#####	100%

Preparing transaction: done
 Verifying transaction: done
 Executing transaction: done

We will need to import libraries

```
In [47]: import folium
```

And we will need to create some functions

```
In [48]: def get_map(location, zoom_start = 12):
    #this expects a location object. Use get_location_object()
    map_object = folium.Map(location=[location.latitude, location.longitude], zoom_start=zoom_start)
    return map_object

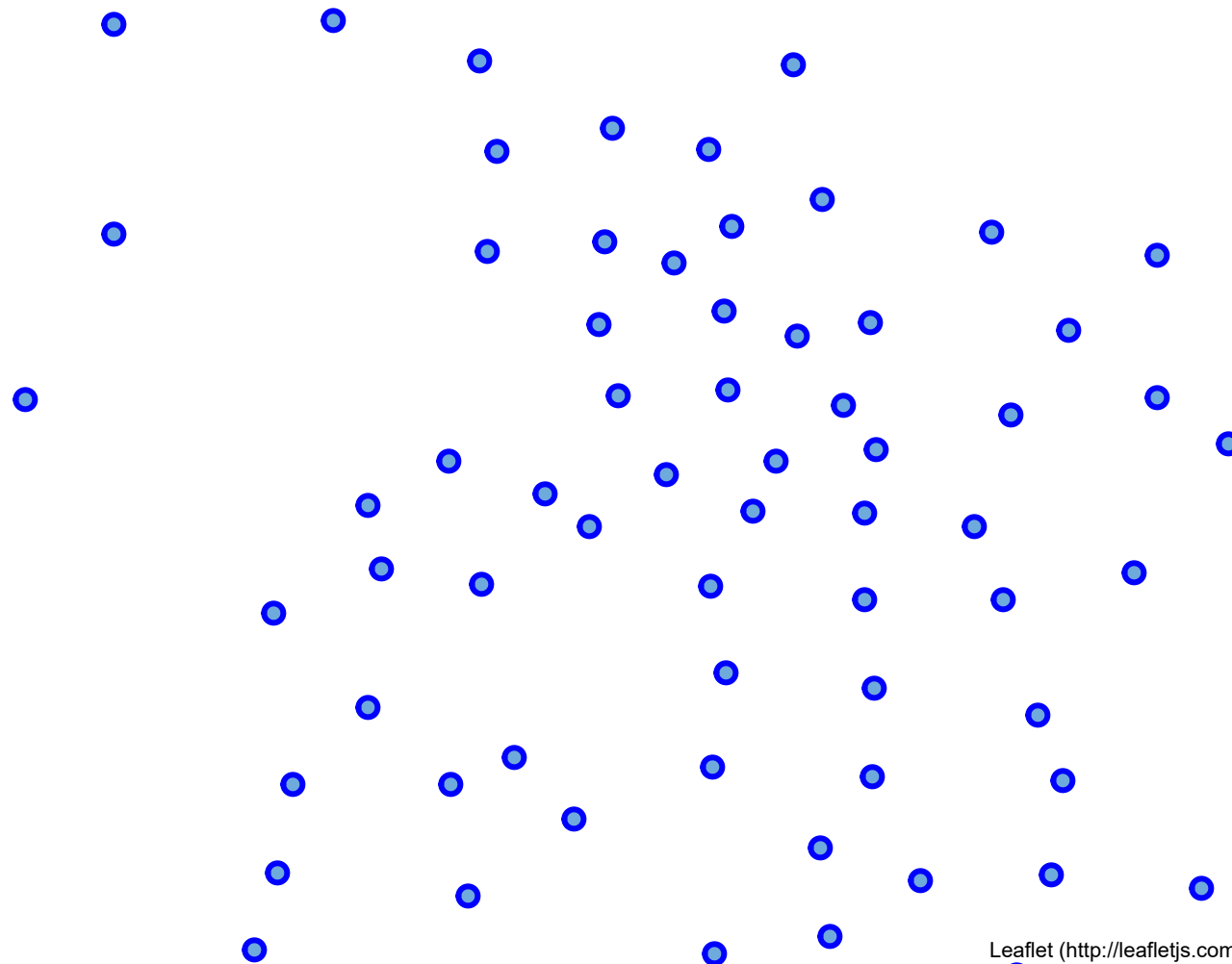
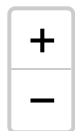
def add_neighborhood_markers_to_map(this_map, this_neighborhoods_dataframe):
    # add markers to map
    for lat, lng, neighborhood in zip(this_neighborhoods_dataframe['Latitude'], this_neighborhoods_dataframe[
'Longitude'], this_neighborhoods_dataframe['Neighborhood']):
        label = '{}'.format(neighborhood)
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='#3186cc',
            fill_opacity=0.7,
            parse_html=False).add_to(this_map)
```

Now let's add markers for the neighborhoods and display the map

```
In [49]: print("Portland data loaded. Reference variable neighborhoods_venues_df to view it")
print("generating Portland map")
portland_location = get_location_object('Portland, Oregon')
portland_map = get_map(portland_location)
print("Adding Portland neighborhood markers to map")
add_neighborhood_markers_to_map(portland_map, neighborhoods_venues_df)
portland_map
```

Portland data loaded. Reference variable neighborhoods_venues_df to view it
generating Portland map
Adding Portland neighborhood markers to map

Out[49]:



Leaflet (<http://leafletjs.com>)

Looking up the most common venues for each neighborhood

```
In [50]: portland_venues = get_nearby_venues(names=neighborhoods_venues_df['Neighborhood'], latitudes=neighborhoods_venues_df['Latitude'], longitudes=neighborhoods_venues_df['Longitude'], radius=500, limit=100)
print("Venues have been retrieved. reference variable portland_venues to view it.")

portland_onehot = pd.get_dummies(portland_venues[['Venue Category']], prefix="", prefix_sep="")
portland_onehot['Neighborhood'] = portland_venues['Neighborhood']
fixed_columns = [portland_onehot.columns[-1]] + list(portland_onehot.columns[:-1])
portland_onehot = portland_onehot[fixed_columns]
portland_grouped = portland_onehot.groupby('Neighborhood').mean().reset_index()
portland_venues_sorted = get_sorted_venues_dataframe(portland_grouped)
```

Venues have been retrieved. reference variable portland_venues to view it.

```
In [51]: portland_venues_sorted.head()
```

Out[51]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	ALAMEDA	Optical Shop	Garden Center	Pilates Studio	Coffee Shop	Italian Restaurant	Soccer Field	Event Service	Eastern European Restaurant	Electronics Store	Elementary School
1	ARBOR LODGE	Convenience Store	Playground	Park	Mexican Restaurant	Marijuana Dispensary	Sushi Restaurant	Bus Stop	Thai Restaurant	Gas Station	Flower Shop
2	ARDENWALD-JOHNSON CREEK	Convenience Store	Coffee Shop	Food & Drink Shop	Park	Grocery Store	Café	Fast Food Restaurant	Fish Market	Farmers Market	Electronics Store
3	ARGAY	Thai Restaurant	Mexican Restaurant	Convenience Store	Farm	Elementary School	Ethiopian Restaurant	Event Service	Event Space	Eye Doctor	Farmers Market
4	ARLINGTON HEIGHTS	Garden	Botanical Garden	Gift Shop	Trail	Park	Bus Station	Fountain	Café	Train Station	Tennis Court

In []:

Great, so that give us the data we need. The next task is to cluster them.

Clustering the data sets

In this section, we take the dataset that was gathered for most common venues, as well as the dataset for demographic data and determine clusters. By clusters, we mean items (neighborhoods) that are similar either in demographics or in predominant venues. Let's start with the venues.

Clustering the venue dataset

The code below determines clusters for the neighborhoods based on most commonly found venues (i.e. businesses or other types of places).

Important caveat

In a more complete analysis, some time and computation would be spent determining the optimal value of k (i.e. the number of clusters). For the purpose of this report, we will say hypothetically that number is 10.

First we need some libraries imported.

```
In [52]: import matplotlib.cm as cm  
import matplotlib.colors as colors  
from sklearn.cluster import KMeans
```

```
In [ ]:
```

Next, let's define some necessary functions.

```

In [53]: def get_clustering_df(grouped_df):
    if('Neighborhood' in grouped_df.columns):
        grouped_clustering_df = grouped_df.drop('Neighborhood', 1)
    else:
        grouped_clustering_df = grouped_df
    return grouped_clustering_df

def get_clusters(grouped_clustering_df, number_of_clusters = 10):
    kmeans = KMeans(n_clusters=number_of_clusters, random_state=0).fit(grouped_clustering_df)
    return kmeans

def get_clustered_neighborhoods_df_by_kmeans(kmeans, clusters_df, neighborhoods_df):
    if 'Cluster Labels' not in clusters_df.columns:
        clusters_df.insert(0, 'Cluster Labels', kmeans.labels_)
    neighborhoods_merged = neighborhoods_df
    neighborhoods_merged = neighborhoods_merged.join(clusters_df.set_index('Neighborhood'), on='Neighborhood'
)
    return neighborhoods_merged

def get_clustered_venue_df(grouped_df, sorted_venue_df, neighborhoods_df, number_of_clusters=10):
    if('Neighborhood' in grouped_df.columns):
        grouped_clustering_df = grouped_df.drop('Neighborhood', 1)
    else:
        grouped_clustering_df = grouped_df
    kmeans = KMeans(n_clusters=number_of_clusters, random_state=0).fit(grouped_clustering_df)
    if 'Cluster Labels' not in sorted_venue_df.columns:
        sorted_venue_df.insert(0, 'Cluster Labels', kmeans.labels_)
    neighborhoods_merged = neighborhoods_df
    neighborhoods_merged = neighborhoods_merged.join(sorted_venue_df.set_index('Neighborhood'), on='Neighborhood')
    return neighborhoods_merged

def create_clustered_map(location, kclusters, clustered_neighborhood_df):
    # create map
    map_clusters = folium.Map(location=[location.latitude, location.longitude], zoom_start=11)

    # set color scheme 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(clustered_neighborhood_df['Latitude'], clustered_neighborhood_df['Longitude'], clustered_neighborhood_df['Neighborhood'], clustered_neighborhood_df['Cluster Labels']):
    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)

return map_clusters

```

And now let's retrieve the clusters and add them to the venues dataframe

```

In [64]: kclusters = 10
print("Getting {} clusters".format(kclusters))
#this basically just drops the "Neighborhood" column, as a convenience method
portland_venue_clustering_df = get_clustering_df(portland_grouped)
portland_venues_kmeans = get_clusters(portland_venue_clustering_df, kclusters)
portland_clustered_venues_df = get_clustered_neighborhoods_df_by_kmeans(portland_venues_kmeans, portland_venues_sorted, neighborhoods_venues_df)
print("Clusters have been retrieved. reference variable portland_clustered_venues_df to view it.")

```

Getting 10 clusters

Clusters have been retrieved. reference variable portland_clustered_venues_df to view it.

Let's take a look at the clustered data

In [65]: portland_clustered_venues_df

Out[65]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
0	ALAMEDA	45.548631	-122.636481	7	Optical Shop	Garden Center	Pilates Studio	Coffee Shop	Italian Restaurant	Soccer Field
1	ARBOR LODGE	45.571794	-122.690152	7	Convenience Store	Playground	Park	Mexican Restaurant	Marijuana Dispensary	Sushi Restaurant
2	ARDENWALD-JOHNSON CREEK	45.458516	-122.627539	7	Convenience Store	Coffee Shop	Food & Drink Shop	Park	Grocery Store	Café
3	ARGAY	45.552830	-122.523204	7	Thai Restaurant	Mexican Restaurant	Convenience Store	Farm	Elementary School	Ethiopian Restaurant
4	ARLINGTON HEIGHTS	45.519496	-122.710667	7	Garden	Botanical Garden	Gift Shop	Trail	Park	Bus Station
5	ARNOLD CREEK	45.446565	-122.688535	9	Event Space	Trail	Pizza Place	Yoga Studio	Eastern European Restaurant	Electronics Store
6	ASHCREEK	45.461163	-122.733365	6	Event Service	Yoga Studio	Dry Cleaner	Food	Flower Shop	Flea Market
7	BEAUMONT-WILSHIRE	45.550391	-122.623694	7	Pizza Place	Bar	Park	Thai Restaurant	Breakfast Spot	Bagel Shop
8	BOISE	45.550159	-122.671878	7	Brewery	Food Truck	Coffee Shop	Pizza Place	Cocktail Bar	Bar
9	BRENTWOOD-DARLINGTON	45.468707	-122.597633	1	Park	Dog Run	Deli / Bodega	Bus Stop	Yoga Studio	Eye Doctor
10	BRIDGETON	45.602409	-122.668102	7	Harbor / Marina	American Restaurant	Grocery Store	Hawaiian Restaurant	Thai Restaurant	Fish Market
11	BRIDLEMILE	45.492559	-122.726693	1	Pool	Park	Tennis Court	Dry Cleaner	Electronics Store	Elementary School
12	BROOKLYN	45.494819	-122.651552	7	Bar	Bus Stop	Light Rail Station	Thrift / Vintage Store	Grocery Store	Convenience Store
13	BUCKMAN	45.517381	-122.651954	7	Bar	Coffee Shop	Brewery	Wine Bar	Mexican Restaurant	Vegetarian / Vegan Restaurant

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
14	CATHEDRAL PARK	45.587636	-122.758640	7	Coffee Shop	Brewery	Gift Shop	Yoga Studio	Camera Store	Fried Chicken Joint
15	CENTENNIAL	45.505595	-122.499711	7	Convenience Store	Food Truck	Sandwich Place	Electronics Store	Shipping Store	Clothing Store
16	COLLINS VIEW	45.457979	-122.681210	3	Photography Studio	Yoga Studio	Eye Doctor	Electronics Store	Elementary School	Ethiopian Restaurant
17	CONCORDIA	45.565866	-122.632216	7	Gift Shop	Hotel	Hotel Bar	Bar	Convenience Store	Restaurant
18	CRESTON-KENILWORTH	45.493679	-122.623111	7	Coffee Shop	Convenience Store	Bus Stop	Pizza Place	Bank	Pub
19	CRESTWOOD	45.450783	-122.736624	5	Playground	Yoga Studio	Farm	Electronics Store	Elementary School	Ethiopian Restaurant
20	CULLY	45.561546	-122.602080	7	Mexican Restaurant	Taco Place	Gas Station	Farm	Flea Market	Fish Market
21	DOWNTOWN	43.658442	-70.258430	7	Bar	Coffee Shop	Hotel	Ice Cream Shop	American Restaurant	Italian Restaurant
22	EAST COLUMBIA	45.593837	-122.663537	1	Casino	Park	Auto Dealership	Yoga Studio	Eastern European Restaurant	Elementary School
23	EASTMORELAND	45.473553	-122.630899	0	Park	Yoga Studio	Eye Doctor	Electronics Store	Elementary School	Ethiopian Restaurant
24	ELIOT	45.541219	-122.668437	7	Park	Brewery	Lounge	Yoga Studio	BBQ Joint	Convenience Store
25	FAR SOUTHWEST	45.439900	-122.735594	7	College Cafeteria	Soccer Field	Yoga Studio	Farm	Elementary School	Ethiopian Restaurant
26	FOREST PARK	45.561468	-122.758581	4	Forest	Yoga Studio	Farm	Electronics Store	Elementary School	Ethiopian Restaurant
27	FOSTER-POWELL	45.493188	-122.589231	7	Food	Flower Shop	Bus Stop	Furniture / Home Store	Yoga Studio	Eye Doctor
28	GLENFAIR	45.522719	-122.504133	1	Food Truck	Park	Yoga Studio	Farm	Elementary School	Ethiopian Restaurant

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
29	GOOSE HOLLOW	45.517749	-122.692819	7	Thai Restaurant	Pizza Place	Pub	Tunnel	Soccer Stadium	Coffee Shop
...
65	PARKROSE HEIGHTS	45.540156	-122.548360	1	Bookstore	Theater	Park	Yoga Studio	Eye Doctor	Elementary School
66	PEARL	45.529044	-122.681598	7	Coffee Shop	American Restaurant	Hotel	Café	Ice Cream Shop	Pizza Place
67	PIEDMONT	45.574552	-122.669405	7	Mexican Restaurant	Grocery Store	Coffee Shop	Eye Doctor	Electronics Store	Elementary School
68	PLEASANT VALLEY	45.471960	-122.507003	0	Park	Yoga Studio	Eye Doctor	Electronics Store	Elementary School	Ethiopian Restaurant
69	PORTSMOUTH	45.588021	-122.719441	1	Clothing Store	Park	Pizza Place	Farmers Market	Eye Doctor	Electronics Store
70	POWELLHURST-GILBERT	45.492568	-122.538696	7	Grocery Store	Convenience Store	Coffee Shop	Bus Station	Food Truck	Ice Cream Shop
71	REED	45.484746	-122.632438	7	Convenience Store	Farmers Market	Café	Performing Arts Venue	Scenic Lookout	Gym / Fitness Center
72	RICHMOND	45.504675	-122.622700	7	Thai Restaurant	Tea Room	Thrift / Vintage Store	Arts & Crafts Store	Beer Bar	Salon / Barbershop
73	ROSE CITY PARK	45.538786	-122.598634	1	BBQ Joint	Pub	Park	Furniture / Home Store	Event Space	Electronics Store
74	ROSEWAY	45.549485	-122.588352	7	Vietnamese Restaurant	Pharmacy	Sports Bar	Toy / Game Store	Coffee Shop	Donut Shop
75	RUSSELL	45.538804	-122.526761	1	Concert Hall	Park	Eye Doctor	Electronics Store	Elementary School	Ethiopian Restaurant
76	SABIN	45.551773	-122.649480	7	Bar	Café	Grocery Store	Marijuana Dispensary	Bakery	Cosmetics Shop
77	SELLWOOD-MORELAND	45.471488	-122.651430	7	Food Truck	Coffee Shop	Bar	Pizza Place	Pub	Farmers Market
78	SOUTH BURLINGAME	45.466659	-122.684434	7	Mexican Restaurant	Coffee Shop	Sports Bar	Butcher	Marijuana Dispensary	Grocery Store

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
79	SOUTH PORTLAND	45.488250	-122.676439	7	Coffee Shop	Deli / Bodega	Trail	Bus Stop	Hotel	Café
80	SOUTH TABOR	45.501381	-122.593759	7	Bar	Mexican Restaurant	Chinese Restaurant	Fish Market	Farm	Ethiopian Restaurant
81	SOUTHWEST HILLS	45.502274	-122.713288	7	Zoo Exhibit	Gas Station	Playground	Farm	Elementary School	Ethiopian Restaurant
82	ST. JOHNS	45.598078	-122.745471	7	Fruit & Vegetable Store	Massage Studio	Garden	Bus Station	Yoga Studio	Ethiopian Restaurant
83	SULLIVAN'S GULCH	45.532939	-122.640494	7	Coffee Shop	Bar	Video Store	Mexican Restaurant	Massage Studio	Gym
84	SUMNER	45.558740	-122.572495	7	Convenience Store	Deli / Bodega	Motel	Bar	Construction & Landscaping	Dance Studio
85	SUNDERLAND	45.582577	-122.637277	7	Diner	Golf Course	Baseball Field	Transportation Service	Eye Doctor	Elementary School
86	SUNNYSIDE	45.515774	-122.624528	7	Bar	Coffee Shop	Food Truck	Thai Restaurant	Breakfast Spot	Clothing Store
87	SYLVAN-HIGHLANDS	45.514134	-122.730008	7	Bistro	Pilates Studio	Insurance Office	Deli / Bodega	Gym	Yoga Studio
88	UNIVERSITY PARK	43.691719	-70.291867	7	Sandwich Place	Middle Eastern Restaurant	Fast Food Restaurant	Video Store	Italian Restaurant	Event Service
89	VERNON	45.562300	-122.648314	7	Coffee Shop	Wine Bar	Arts & Crafts Store	Gift Shop	Indian Restaurant	Furniture / Home Store
90	WEST PORTLAND PARK	45.447282	-122.720814	1	Park	Cosmetics Shop	Sandwich Place	Gym / Fitness Center	Fish Market	Fast Food Restaurant
91	WILKES	45.541190	-122.504702	2	Food Truck	Yoga Studio	Farm	Electronics Store	Elementary School	Ethiopian Restaurant
92	WOODLAND PARK	45.535226	-122.559577	7	Convenience Store	Clothing Store	Paper / Office Supplies Store	Liquor Store	Mexican Restaurant	Mobile Phone Shop

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
93	WOODLAWN	45.571983	-122.652380	7	Pizza Place	Brewery	Italian Restaurant	Deli / Bodega	Coffee Shop	Restaurant
94	WOODSTOCK	45.480705	-122.614549	7	Coffee Shop	ATM	Convenience Store	Grocery Store	Thai Restaurant	Vegetarian / Vegan Restaurant

94 rows × 14 columns



Let's drop any NaN rows, and set the data type for the cluster to integer

```
In [66]: portland_clustered_venues_df = portland_clustered_venues_df[np.isfinite(portland_clustered_venues_df['Cluster Labels'])]
```

In []:

Now let's create a map which shows the neighborhoods colored and labelled by cluster.

```
In [67]: print("Generating map")
portland_venues_clustered_map = create_clustered_map(portland_location, kclusters, portland_clustered_venues_df)
print("Map has been retrieved, reference variable portland_clustered_map to view it.")

print("finished with Portland")
```

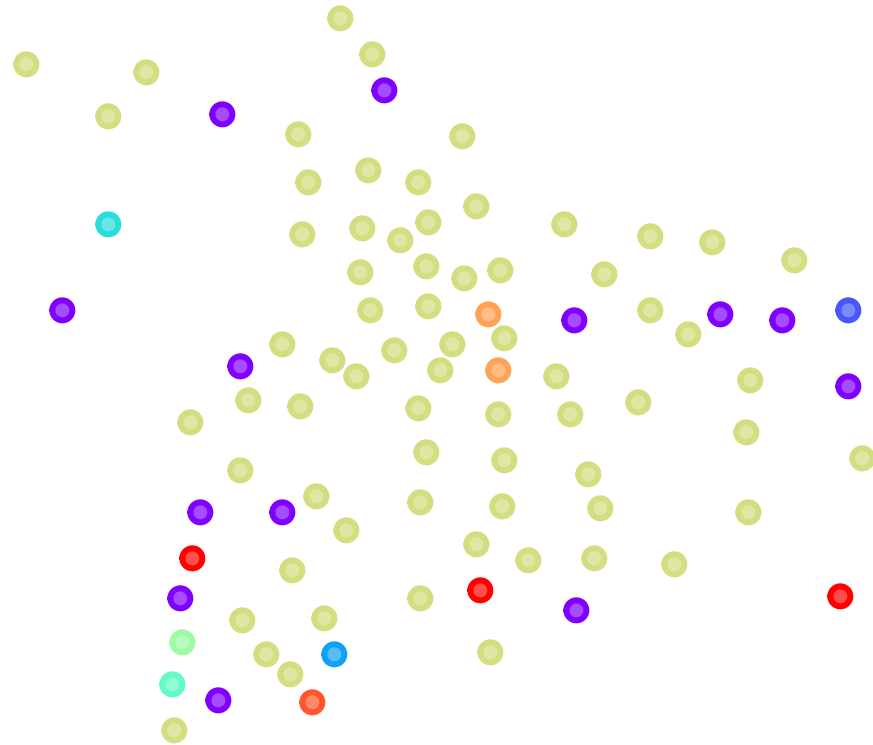
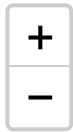
Generating map

Map has been retrieved, reference variable portland_clustered_map to view it.

finished with Portland

In [69]: portland_venues_clustered_map

Out[69]:



Leaflet (<http://leafletjs.com>)

Let's take a look to see which venue cluster Parkrose belongs to.

```
In [70]: portland_clustered_venues_df.loc[portland_clustered_venues_df['Neighborhood'] == 'PARKROSE']
```

Out[70]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
64	PARKROSE	45.557458	-122.550785	7	Mexican Restaurant	Hotel	Grocery Store	Market	Drugstore	Sandwich Place	German Restaurant	Chinese Restaurant

We can see that Parkrose belongs to cluster 4. Let's see what other neighborhoods belong to cluster 4

```
In [71]: portland_clustered_venues_df.loc[portland_clustered_venues_df['Cluster Labels'] == 7]
```


Out[71]:

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
0	ALAMEDA	45.548631	-122.636481	7	Optical Shop	Garden Center	Pilates Studio	Coffee Shop	Italian Restaurant	Soccer Field
1	ARBOR LODGE	45.571794	-122.690152	7	Convenience Store	Playground	Park	Mexican Restaurant	Marijuana Dispensary	Sushi Restaurant
2	ARDENWALD-JOHNSON CREEK	45.458516	-122.627539	7	Convenience Store	Coffee Shop	Food & Drink Shop	Park	Grocery Store	Café
3	ARGAY	45.552830	-122.523204	7	Thai Restaurant	Mexican Restaurant	Convenience Store	Farm	Elementary School	Ethiopian Restaurant
4	ARLINGTON HEIGHTS	45.519496	-122.710667	7	Garden	Botanical Garden	Gift Shop	Trail	Park	Bus Station
7	BEAUMONT-WILSHIRE	45.550391	-122.623694	7	Pizza Place	Bar	Park	Thai Restaurant	Breakfast Spot	Bagel Shop
8	BOISE	45.550159	-122.671878	7	Brewery	Food Truck	Coffee Shop	Pizza Place	Cocktail Bar	Bar
10	BRIDGETON	45.602409	-122.668102	7	Harbor / Marina	American Restaurant	Grocery Store	Hawaiian Restaurant	Thai Restaurant	Fish Market
12	BROOKLYN	45.494819	-122.651552	7	Bar	Bus Stop	Light Rail Station	Thrift / Vintage Store	Grocery Store	Convenience Store
13	BUCKMAN	45.517381	-122.651954	7	Bar	Coffee Shop	Brewery	Wine Bar	Mexican Restaurant	Vegetarian / Vegan Restaurant
14	CATHEDRAL PARK	45.587636	-122.758640	7	Coffee Shop	Brewery	Gift Shop	Yoga Studio	Camera Store	Fried Chicken Joint
15	CENTENNIAL	45.505595	-122.499711	7	Convenience Store	Food Truck	Sandwich Place	Electronics Store	Shipping Store	Clothing Store
17	CONCORDIA	45.565866	-122.632216	7	Gift Shop	Hotel	Hotel Bar	Bar	Convenience Store	Restaurant
18	CRESTON-KENILWORTH	45.493679	-122.623111	7	Coffee Shop	Convenience Store	Bus Stop	Pizza Place	Bank	Pub

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
20	CULLY	45.561546	-122.602080	7	Mexican Restaurant	Taco Place	Gas Station	Farm	Flea Market	Fish Market
21	DOWNTOWN	43.658442	-70.258430	7	Bar	Coffee Shop	Hotel	Ice Cream Shop	American Restaurant	Italian Restaurant
24	ELIOT	45.541219	-122.668437	7	Park	Brewery	Lounge	Yoga Studio	BBQ Joint	Convenience Store
25	FAR SOUTHWEST	45.439900	-122.735594	7	College Cafeteria	Soccer Field	Yoga Studio	Farm	Elementary School	Ethiopian Restaurant
27	FOSTER-POWELL	45.493188	-122.589231	7	Food	Flower Shop	Bus Stop	Furniture / Home Store	Yoga Studio	Eye Doctor
29	GOOSE HOLLOW	45.517749	-122.692819	7	Thai Restaurant	Pizza Place	Pub	Tunnel	Soccer Stadium	Coffee Shop
31	HAYDEN ISLAND	45.611070	-122.678742	7	Hotel	Fast Food Restaurant	American Restaurant	Coffee Shop	Harbor / Marina	Bar
33	HAZELWOOD	45.523934	-122.538339	7	Cosmetics Shop	Pharmacy	Convenience Store	Fast Food Restaurant	Liquor Store	Fried Chicken Joint
35	HILLSDALE	45.478606	-122.695296	7	Pizza Place	Brewery	Food Truck	Pilates Studio	Coffee Shop	Mexican Restaurant
37	HOLLYWOOD	45.534405	-122.622569	7	Pizza Place	Grocery Store	Furniture / Home Store	Café	Pharmacy	Sporting Goods Shop
38	HOMESTEAD	45.496029	-122.687134	7	Plaza	Coffee Shop	Café	Tram Station	Eye Doctor	Scenic Lookout
39	HOSFORD-ABERNETHY	45.506644	-122.649419	7	Bar	Coffee Shop	Mexican Restaurant	Breakfast Spot	Pizza Place	Italian Restaurant
40	HUMBOLDT	45.560506	-122.671030	7	Bar	Record Shop	Bookstore	Ethiopian Restaurant	Pub	Thrift / Vintage Store
41	IRVINGTON	45.541983	-122.648938	7	Yoga Studio	Tennis Court	Wine Bar	Bar	Bed & Breakfast	Eye Doctor
42	KENTON	45.583147	-122.693150	7	Coffee Shop	Flower Shop	Breakfast Spot	Thrift / Vintage Store	Pub	Record Shop

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
43	KERNS	45.526763	-122.644549	7	Bar	Coffee Shop	Cocktail Bar	Sandwich Place	Brewery	Building
...
56	MT. TABOR	45.515836	-122.599896	7	Cheese Shop	Women's Store	Playground	Basketball Court	American Restaurant	Lake
57	MULTNOMAH	45.466151	-122.712787	7	Café	Coffee Shop	Bar	Frozen Yogurt Shop	Bookstore	Taco Place
58	NORTH TABOR	45.524905	-122.604890	7	Coffee Shop	Convenience Store	Bridal Shop	Supermarket	Bus Stop	Light Rail Station
59	NORTHWEST DISTRICT	45.533013	-122.698845	7	Coffee Shop	Hotel	Boutique	New American Restaurant	Mexican Restaurant	Korean Restaurant
62	OLD TOWN-CHINATOWN	45.524934	-122.673516	7	Hotel	Sandwich Place	Bagel Shop	Food Truck	American Restaurant	Mexican Restaurant
63	OVERLOOK	45.559149	-122.691971	7	Bar	Sporting Goods Shop	Food Court	Café	Thai Restaurant	Sushi Restaurant
64	PARKROSE	45.557458	-122.550785	7	Mexican Restaurant	Hotel	Grocery Store	Market	Drugstore	Sandwich Place
66	PEARL	45.529044	-122.681598	7	Coffee Shop	American Restaurant	Hotel	Café	Ice Cream Shop	Pizza Place
67	PIEDMONT	45.574552	-122.669405	7	Mexican Restaurant	Grocery Store	Coffee Shop	Eye Doctor	Electronics Store	Elementary School
70	POWELLHURST-GILBERT	45.492568	-122.538696	7	Grocery Store	Convenience Store	Coffee Shop	Bus Station	Food Truck	Ice Cream Shop
71	REED	45.484746	-122.632438	7	Convenience Store	Farmers Market	Café	Performing Arts Venue	Scenic Lookout	Gym / Fitness Center
72	RICHMOND	45.504675	-122.622700	7	Thai Restaurant	Tea Room	Thrift / Vintage Store	Arts & Crafts Store	Beer Bar	Salon / Barbershop
74	ROSEWAY	45.549485	-122.588352	7	Vietnamese Restaurant	Pharmacy	Sports Bar	Toy / Game Store	Coffee Shop	Donut Shop
76	SABIN	45.551773	-122.649480	7	Bar	Café	Grocery Store	Marijuana Dispensary	Bakery	Cosmetics Shop

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
77	SELLWOOD-MORELAND	45.471488	-122.651430	7	Food Truck	Coffee Shop	Bar	Pizza Place	Pub	Farmers Market
78	SOUTH BURLINGAME	45.466659	-122.684434	7	Mexican Restaurant	Coffee Shop	Sports Bar	Butcher	Marijuana Dispensary	Grocery Store
79	SOUTH PORTLAND	45.488250	-122.676439	7	Coffee Shop	Deli / Bodega	Trail	Bus Stop	Hotel	Café
80	SOUTH TABOR	45.501381	-122.593759	7	Bar	Mexican Restaurant	Chinese Restaurant	Fish Market	Farm	Ethiopian Restaurant
81	SOUTHWEST HILLS	45.502274	-122.713288	7	Zoo Exhibit	Gas Station	Playground	Farm	Elementary School	Ethiopian Restaurant
82	ST. JOHNS	45.598078	-122.745471	7	Fruit & Vegetable Store	Massage Studio	Garden	Bus Station	Yoga Studio	Ethiopian Restaurant
83	SULLIVAN'S GULCH	45.532939	-122.640494	7	Coffee Shop	Bar	Video Store	Mexican Restaurant	Massage Studio	Gym
84	SUMNER	45.558740	-122.572495	7	Convenience Store	Deli / Bodega	Motel	Bar	Construction & Landscaping	Dance Studio
85	SUNDERLAND	45.582577	-122.637277	7	Diner	Golf Course	Baseball Field	Transportation Service	Eye Doctor	Elementary School
86	SUNNYSIDE	45.515774	-122.624528	7	Bar	Coffee Shop	Food Truck	Thai Restaurant	Breakfast Spot	Clothing Store
87	SYLVAN-HIGHLANDS	45.514134	-122.730008	7	Bistro	Pilates Studio	Insurance Office	Deli / Bodega	Gym	Yoga Studio
88	UNIVERSITY PARK	43.691719	-70.291867	7	Sandwich Place	Middle Eastern Restaurant	Fast Food Restaurant	Video Store	Italian Restaurant	Event Service
89	VERNON	45.562300	-122.648314	7	Coffee Shop	Wine Bar	Arts & Crafts Store	Gift Shop	Indian Restaurant	Furniture / Home Store
92	WOODLAND PARK	45.535226	-122.559577	7	Convenience Store	Clothing Store	Paper / Office Supplies Store	Liquor Store	Mexican Restaurant	Mobile Phone Shop
93	WOODLAWN	45.571983	-122.652380	7	Pizza Place	Brewery	Italian Restaurant	Deli / Bodega	Coffee Shop	Restaurant

	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
94	WOODSTOCK	45.480705	-122.614549	7	Coffee Shop	ATM	Convenience Store	Grocery Store	Thai Restaurant	Vegetarian / Vegan Restaurant

70 rows × 14 columns



Clustering the demographic data

Next, let's cluster the demographic data. First we will need to create a dataframe for clustering which only includes the normalized data.

```
In [72]: df_demographics_clustering = df_demographics[['population_female_Under_5', 'population_male_5_to_9',
'population_female_5_to_9', 'population_male_10_to_14',
'population_female_10_to_14', 'population_male_15_to_17',
'population_female_15_to_17', 'population_male_18_to_19',
'population_female_18_and_19', 'population_male_20_to_24',
'population_female_20_to_24', 'population_male_25_to_29',
'population_female_25_to_29', 'population_male_30_to_34',
'population_female_30_to_34', 'population_male_35_to_39',
'population_female_35_to_39', 'population_male_40_to_44',
'population_female_40_to_44', 'population_male_45_to_49',
'population_female_45_to_49', 'population_male_50_to_54',
'population_female_50_to_54', 'population_male_55_to_59',
'population_female_55_to_59', 'population_male_60_to_64',
'population_female_60_to_64', 'population_male_65_to_69',
'population_female_65_to_69', 'population_male_70_to_74',
'population_female_70_to_74', 'population_male_75_to_79',
'population_female_75_to_79', 'population_male_80_to_84',
'population_female_80_to_84', 'population_male_85_and_over',
'population_female_85_and_over', 'multiperson_nonfamily_households',
'home_owned_with_mortgage_or_loan', 'home_owned_free_and_clear',
'home_rented', 'all_one_person_household', 'all_family_household',
'total_households', 'all_non_family_household']].copy()

df_demographics_clustering.reset_index(inplace = True)
df_demographics_clustering = df_demographics_clustering.drop(['neighborhood'], axis=1)
df_demographics_clustering = df_demographics_clustering.dropna(axis = 0, how = 'any')
```

With that, we should be ready to cluster the data, and create a merged dataframe

```
In [77]: portland_demographics_kmeans = get_clusters(df_demographics_clustering, kclusters)
venues_merging_df = portland_clustered_venues_df[['Neighborhood', 'Latitude', 'Longitude']].copy()
demographics_neighborhoods_list = df_demographics.index.tolist()
demographics_merging_df = pd.DataFrame(demographics_neighborhoods_list, columns = ['Neighborhood'])
walk_scores = df_demographics['WALK_SCORE'].to_list()
demographics_merging_df['Walk Score'] = walk_scores
demographics_merging_df['Cluster Labels'] = portland_demographics_kmeans.labels_
demographics_neighborhoods_df = pd.merge( demographics_merging_df, venues_merging_df, on=['Neighborhood'])
```

Let's see what that data looks like.

```
In [78]: demographics_neighborhoods_df.head()
```

Out[78]:

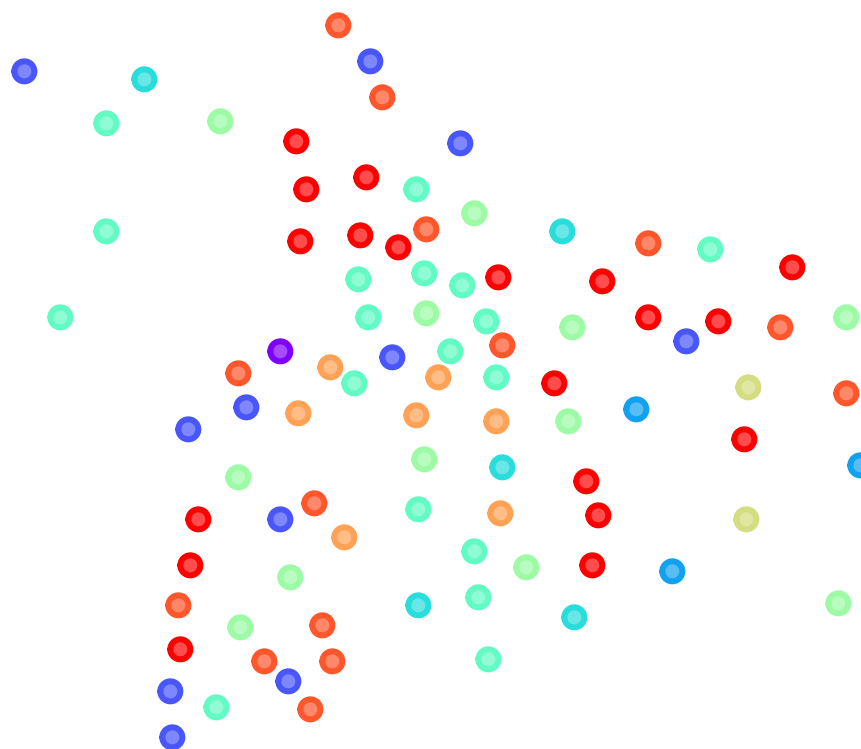
	Neighborhood	Walk Score	Cluster Labels	Latitude	Longitude
0	ALAMEDA	65	5	45.548631	-122.636481
1	ARBOR LODGE	72	0	45.571794	-122.690152
2	ARDENWALD-JOHNSON CREEK	54	5	45.458516	-122.627539
3	ARGAY	45	0	45.552830	-122.523204
4	ARLINGTON HEIGHTS	40	2	45.519496	-122.710667

Let's create another map which displays the neighborhoods with demographic clustering

```
In [82]: print("Generating map")
portland_demographics_clustered_map = create_clustered_map(portland_location, kclusters, demographics_neighbo
rhods_df)
portland_demographics_clustered_map
```

Generating map

Out[82]:



Leaflet (<http://leafletjs.com>)

Creating a dataset which combines clusters

Next we will need a dataframe which combines the cluster labels from venues along with cluster labels from demographics, and allows for a comparison

```
In [97]: merge_venue_df = portland_clustered_venues_df.copy()
merge_venue_df = merge_venue_df.rename(columns = {'Cluster Labels':'Venue Cluster'})
merge_demographic_df = demographics_neighborhoods_df.copy()
merge_demographic_df = merge_demographic_df.rename(columns = {'Cluster Labels':'Demographic Cluster'})
# We only need one set of Latitude, Longitude
merge_demographic_df = merge_demographic_df.drop(['Latitude', 'Longitude'], axis = 1)
combined_df = pd.merge(merge_venue_df, merge_demographic_df, how='inner', on='Neighborhood')
```

Let's take a look at what the data looks like

```
In [114]: combined_df[['Neighborhood', 'Venue Cluster', 'Demographic Cluster']].head()
```

Out[114]:

	Neighborhood	Venue Cluster	Demographic Cluster
0	ALAMEDA	7	5
1	ARBOR LODGE	7	0
2	ARDENWALD-JOHNSON CREEK	7	5
3	ARGAY	7	0
4	ARLINGTON HEIGHTS	7	2

Section 3 - Results

The objective of this report was to answer these questions:

1. Which Portland neighborhoods are classified as similar to Parkrose based demographic clustering
2. How does Parkrose compare in walkability to the other neighborhoods in that cluster
3. What are the most common non-residential location categories in neighborhoods of the same cluster, and how does Parkrose compare
4. What are the most common non-residential location categories in neighborhoods with higher walk \ bike \ transit scores

Below are the results of the report and how they help answer the questions

1. Which Portland neighborhoods are classified as similar to Parkrose based demographic clustering

Below is the demographic cluster that Parkrose was determined to be.

```
In [102]: demographics_neighborhoods_df.loc[demographics_neighborhoods_df['Neighborhood'] == 'PARKROSE']
```

Out[102]:

	Neighborhood	Walk Score	Cluster Labels	Latitude	Longitude
63	PARKROSE	57	5	45.557458	-122.550785

Here are the other neighborhoods in the same cluster

```
In [104]: demographics_neighborhoods_df.loc[demographics_neighborhoods_df['Cluster Labels'] == 5]
```

```
Out[104]:
```

	Neighborhood	Walk Score	Cluster Labels	Latitude	Longitude
0	ALAMEDA	65	5	45.548631	-122.636481
2	ARDENWALD-JOHNSON CREEK	54	5	45.458516	-122.627539
8	BOISE	92	5	45.550159	-122.671878
12	BROOKLYN	79	5	45.494819	-122.651552
14	CATHEDRAL PARK	79	5	45.587636	-122.758640
23	EASTMORELAND	24	5	45.473553	-122.630899
24	ELIOT	49	5	45.541219	-122.668437
26	FOREST PARK	39	5	45.561468	-122.758581
30	GRANT PARK	93	5	45.540070	-122.628226
45	LAURELHURST	90	5	45.526512	-122.624468
60	NORTHWEST HEIGHTS	93	5	45.540806	-122.774354
61	OLD TOWN-CHINATOWN	96	5	45.524934	-122.673516
63	PARKROSE	57	5	45.557458	-122.550785
70	REED	62	5	45.484746	-122.632438
75	SABIN	83	5	45.551773	-122.649480
82	SULLIVAN'S GULCH	87	5	45.532939	-122.640494
87	UNIVERSITY PARK	48	5	43.691719	-70.291867
89	WEST PORTLAND PARK	62	5	45.447282	-122.720814
92	WOODLAWN	72	5	45.571983	-122.652380

2. How does Parkrose compare in walkability to the other neighborhoods in that cluster

Here we can see the cluster sorted by walk score. As you can see, Parkrose is in the lower half of the list. Also, there is an amount of variance here that would not be explained by demographic data alone.

```
In [105]: demographics_neighborhoods_df.loc[demographics_neighborhoods_df['Cluster Labels'] == 5].sort_values("Walk Score", axis = 0, ascending = False, inplace = False, na_position = 'last')
```

Out[105]:

	Neighborhood	Walk Score	Cluster Labels	Latitude	Longitude
61	OLD TOWN-CHINATOWN	96	5	45.524934	-122.673516
60	NORTHWEST HEIGHTS	93	5	45.540806	-122.774354
30	GRANT PARK	93	5	45.540070	-122.628226
8	BOISE	92	5	45.550159	-122.671878
45	LAURELHURST	90	5	45.526512	-122.624468
82	SULLIVAN'S GULCH	87	5	45.532939	-122.640494
75	SABIN	83	5	45.551773	-122.649480
12	BROOKLYN	79	5	45.494819	-122.651552
14	CATHEDRAL PARK	79	5	45.587636	-122.758640
92	WOODLAWN	72	5	45.571983	-122.652380
0	ALAMEDA	65	5	45.548631	-122.636481
70	REED	62	5	45.484746	-122.632438
89	WEST PORTLAND PARK	62	5	45.447282	-122.720814
63	PARKROSE	57	5	45.557458	-122.550785
2	ARDENWALD-JOHNSON CREEK	54	5	45.458516	-122.627539
24	ELIOT	49	5	45.541219	-122.668437
87	UNIVERSITY PARK	48	5	43.691719	-70.291867
26	FOREST PARK	39	5	45.561468	-122.758581
23	EASTMORELAND	24	5	45.473553	-122.630899

3. What are the most common non-residential location categories in neighborhoods of the same cluster, and how does Parkrose compare

```
In [109]: combined_df.loc[combined_df['Demographic Cluster'] == 5].sort_values("Walk Score", axis = 0, ascending = False, inplace = False, na_position = 'last')
```

Out[109]:

	Neighborhood	Latitude	Longitude	Venue Cluster	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue
61	OLD TOWN-CHINATOWN	45.524934	-122.673516	7	Hotel	Sandwich Place	Bagel Shop	Food Truck	American Restaurant	Mexican Restaurant	Art Gallery
60	NORTHWEST HEIGHTS	45.540806	-122.774354	1	Park	Business Service	Soccer Field	Yoga Studio	Event Space	Elementary School	Ethiopian Restaurant
30	GRANT PARK	45.540070	-122.628226	8	Park	Bus Stop	Yoga Studio	Eye Doctor	Elementary School	Ethiopian Restaurant	Event Space
8	BOISE	45.550159	-122.671878	7	Brewery	Food Truck	Coffee Shop	Pizza Place	Cocktail Bar	Bar	Yoga Studio
45	LAURELHURST	45.526512	-122.624468	8	Bus Stop	Bus Station	Sculpture Garden	Elementary School	Yoga Studio	Ethiopian Restaurant	Event Space
82	SULLIVAN'S GULCH	45.532939	-122.640494	7	Coffee Shop	Bar	Video Store	Mexican Restaurant	Massage Studio	Gym	Bar
75	SABIN	45.551773	-122.649480	7	Bar	Café	Grocery Store	Marijuana Dispensary	Bakery	Cosmetics Shop	Coffee Shop
12	BROOKLYN	45.494819	-122.651552	7	Bar	Bus Stop	Light Rail Station	Thrift / Vintage Store	Grocery Store	Convenience Store	Soccer Field
14	CATHEDRAL PARK	45.587636	-122.758640	7	Coffee Shop	Brewery	Gift Shop	Yoga Studio	Camera Store	Fried Chicken Joint	Breakfast Shop
92	WOODLAWN	45.571983	-122.652380	7	Pizza Place	Brewery	Italian Restaurant	Deli / Bodega	Coffee Shop	Restaurant	Park
0	ALAMEDA	45.548631	-122.636481	7	Optical Shop	Garden Center	Pilates Studio	Coffee Shop	Italian Restaurant	Soccer Field	Event Space
70	REED	45.484746	-122.632438	7	Convenience Store	Farmers Market	Café	Performing Arts Venue	Scenic Lookout	Gym / Fitness Center	Farmers Market
89	WEST PORTLAND PARK	45.447282	-122.720814	1	Park	Cosmetics Shop	Sandwich Place	Gym / Fitness Center	Fish Market	Fast Food Restaurant	Farmers Market

	Neighborhood	Latitude	Longitude	Venue Cluster	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue
63	PARKROSE	45.557458	-122.550785	7	Mexican Restaurant	Hotel	Grocery Store	Market	Drugstore	Sandwich Place	Germ Restaurant
2	ARDENWALD-JOHNSON CREEK	45.458516	-122.627539	7	Convenience Store	Coffee Shop	Food & Drink Shop	Park	Grocery Store	Café	Fast Food Restaurant
24	ELIOT	45.541219	-122.668437	7	Park	Brewery	Lounge	Yoga Studio	BBQ Joint	Convenience Store	Dive Bar
87	UNIVERSITY PARK	43.691719	-70.291867	7	Sandwich Place	Middle Eastern Restaurant	Fast Food Restaurant	Video Store	Italian Restaurant	Event Service	Clean Bar
26	FOREST PARK	45.561468	-122.758581	4	Forest	Yoga Studio	Farm	Electronics Store	Elementary School	Ethiopian Restaurant	Event Service
23	EASTMORELAND	45.473553	-122.630899	0	Park	Yoga Studio	Eye Doctor	Electronics Store	Elementary School	Ethiopian Restaurant	Event Service

4. What are the most common non-residential location categories in neighborhoods with higher walk scores


```
In [111]: combined_df.sort_values("Walk Score", axis = 0, ascending = False, inplace = False, na_position = 'last').head(10)
```

Out[111]:

	Neighborhood	Latitude	Longitude	Venue Cluster	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Common Venue
65	PEARL	45.529044	-122.681598	7	Coffee Shop	American Restaurant	Hotel	Café	Ice Cream Shop	Pizza Place	Clothing Store	
61	OLD TOWN-CHINATOWN	45.524934	-122.673516	7	Hotel	Sandwich Place	Bagel Shop	Food Truck	American Restaurant	Mexican Restaurant	Art Gallery	
21	DOWNTOWN	43.658442	-70.258430	7	Bar	Coffee Shop	Hotel	Ice Cream Shop	American Restaurant	Italian Restaurant	Sandwich Place	
38	HOMESTEAD	45.496029	-122.687134	7	Plaza	Coffee Shop	Café	Tram Station	Eye Doctor	Scenic Lookout	Breakfast Spot	Fas Rest
30	GRANT PARK	45.540070	-122.628226	8	Park	Bus Stop	Yoga Studio	Eye Doctor	Elementary School	Ethiopian Restaurant	Event Service	
60	NORTHWEST HEIGHTS	45.540806	-122.774354	1	Park	Business Service	Soccer Field	Yoga Studio	Event Space	Elementary School	Ethiopian Restaurant	S
8	BOISE	45.550159	-122.671878	7	Brewery	Food Truck	Coffee Shop	Pizza Place	Cocktail Bar	Bar	Yoga Studio	V
85	SUNNYSIDE	45.515774	-122.624528	7	Bar	Coffee Shop	Food Truck	Thai Restaurant	Breakfast Spot	Clothing Store	Yoga Studio	F
40	HUMBOLDT	45.560506	-122.671030	7	Bar	Record Shop	Bookstore	Ethiopian Restaurant	Pub	Thrift / Vintage Store	Thai Restaurant	I
13	BUCKMAN	45.517381	-122.651954	7	Bar	Coffee Shop	Brewery	Wine Bar	Mexican Restaurant	Vegetarian / Vegan Restaurant	Beer Store	E

Section 4 - Discussion

In reviewing the results above, there are some observations from the findings.

1. Overall, the venue types among neighborhoods in the same demographic cluster as Parkrose are fairly similar.
2. While the venues are fairly similar among other neighborhoods of the same demographic clusters, there are some venue types frequently found in higher walk score neighborhoods missing from Parkrose. These include coffee shops, bus stops, elementary schools, farmers markets, food carts and and yoga studios
3. Some of the common venues missing from demographically similar neighborhoods (such as elementary school and yoga studio) are also found in lower walk score neighborhoods, so there doesn't necessarily appear to be a causative relationship with them.

There are some observations which indicate that some important data is not being considered

1. Population density is not being considered and this could skew data from outliers
2. A lot of the neighborhoods which are demographically similar are much wealthier. But wealthier neighborhoods are both higher and lower in walk score.
3. There is a wide variance of demographic clusters in the top walk score neighborhoods, but many of them are also in the same region of the city. The geographic location should be more carefully considered

Section 5 - Conclusions

Based on the information gathered in this report, there are a number of conclusions

1. Different types of studies are needed for comparison. There is not a strong correlation found between common venue types and walk score, but that does not mean one does not exist. Generally, most neighborhoods of all venue clusters share common venue types: restaurants, small retail spaces, hotels etc. But we are looking at the most common venues by total. We are not looking at the size of the venue, or how often it is visited. A Walmart and a food truck would both count as one venue
2. More analysis and time is needed. We do not know that the right number of clusters were created, either for venues or for demographics. More time to analyze and refine the results would provide more precision
3. More data sources could provide for more meaningful information. It is not clear that Foursquare was entirely useful in this analysis
4. It is not clear that increased walk score would lead to higher income demographics, or for that matter improved quality of life. This was presumed, and that presumption may not be valid

If we were to only rely upon the data as included in this report, the following recommendations would be given

1. Attracting businesses may not be the best avenue for increasing the walk score. Better infrastructure and increased population density seem to be more closely related
2. More parks and bus stops may have a higher impact on walk score
3. coffee shops, bars and food trucks lead to higher walk scores
4. Farmers markets are also found in neighborhoods with higher walk scores
5. The Parkrose Neighborhood Association and the Parkrose Business Council may find the most feasible and actionable improvements by convincing Trimet (the regional transit authority) to increase bus service. This is rather hard, but feasible. Secondly, increasing the number of parks may also have a high impact and this is also something they can work towards. Starting a farmers market would also have a lot of impact, as well as a food truck plaza.
6. There are some vacant properties in the Parkrose area that could immediately converted into food cart pods (places where multiple food trucks are allowed to park long term. There is also a large empty property that could host a farmers market. These would be the immediate recommendations
7. If business grants were to be prioritized, coffee shops would be a good option that would increase the quality of life because they (presumably) do not impact crime rate as much as bars do. MORE STUDY IS NEEDED ON THIS!!

There is a saying in the software business (perhaps others as well). "You can have it fast. You can have it cheap. You can have it high quality. You might be able to have 2 of them, but you cannot have all three". This would seem to be the same in data science. More study and time is needed on this and this has an expense. But skipping this results in less reliable data.

In []:

In []: