BATTLE OF ZIP CODES – CHARLOTTE

Coursera Data Science Captstone Project

Introduction

Problem Statement

- Provide a recommendation for our client on the best zip code to open a new restaurant in Charlotte NC
- Our client wants to open a high end Indian restaurant
- Profitability and success are the goals for the restaurant

Objective

- Develop a model to recommend best zip code to open a restaurant
- Keep the model flexible enough to tweak to provide new recommendation, if restaurant concept were to change
- Use multiple dimensions/features to provide better accuracy
- If needed, be able to provide the next best option

Data Used

- Demographics
  - *Zip code, Population, Population Density, median home value, median home income*
- Safety/Crime data
  - *Identify crime incidents by zip code to determine safety of each zip code*
- Economic activity in the area
  - *Measure economic activity/spending by zip code. High spending zip codes may lend themselves better for a restaurant*
- Competition
  - *Understand current competition and identify any sweet spots.*

Approach

- Collect data for each of the dimensions above
- Clean/wrangle data. Make appropriate assumptions where data is not available
- Normalize data in relation to population of the zip code if appropriate.
- Cluster each zip code using K-means Clustering on the four dimensions of data, where available
- Assess all clusters on the four dimensions to identify the best zip for the restaurant

Data Sources

- Demographics
  - *Sourced from US Zip Codes package in Python*
  - *Zip code, Population, Population Density, median home value, median home income*
- Safety/Crime data

  The City of Charlotte does not have all the crime data openly available to public. However, they do have three APIs/files for the three categories below
  - *CMPD Officer-Involved Shootings – Individuals here*
  - *CMPD Officer Involved Shootings – Officers here*
  - *CMPD Officer-Involved Shootings – Incidents here*
- Economic activity in the area
  - *Used Shopping Centers statistics as proxy*
  - *Data is sourced from here*
- Competition
  - *Sourced list of restaurants and categories of those restaurants from FourSquare API*

- Note on Visual representation of data
  - *Size of the circle on the map indicates importance of the dimension*
  - *Circles are centered on the zip codes*
  - *Color of the circle indicates our recommendation for that dimension*
    - Dark Green – Strong Yes, Green – Yes, Yellow – May be and Red - No

Demographic Clustering

- Pull necessary data from USZipcode package in python
- Discard zip codes that have less than 100 households for consideration
- For each zip get latitude and longitude to represent the zip on the map

```
#using Zipcode database, I am retrieving all the zipcodes and the associated demographics. This crucial demogrpahic data will be
 a major factor in
#recommending location for the restaurant. All this information is stored in a list then moved to a dataframe at the end.

search = SearchEngine(simple_zipcode=True)
zipcode = search.by_city_and_state("Charlotte", "NC", returns =50)

w, h = 9, 27;
zip_info = [[0 for x in range(w)] for y in range(h)]
i =0

for zipinfo in zipcode:
    if zipinfo.bounds_south:
        append_list = [zipinfo.zipcode,zipinfo.lat,zipinfo.lng,zipinfo.population,zipinfo.population_density, zipinfo.median_hom
e_value, zipinfo.median_household_income]
        zip_info[i] = append_list
        i = i+1
    else:
        pass

city_demographics_df = pd.DataFrame(zip_info, columns =['zipcode','latitude','longitude','population','population_density','medi
an_home_value','median_household_income'])
```

Demographic Clustering

Sample Demographics data before clustering

```
city_demographics_df
```

]:

	zipcode	latitude	longitude	population	population_density	median_home_value	median_household_income
0	28202	35.23	-80.84	11195	6213.0	251200.0	70300.0
1	28203	35.21	-80.86	11315	3411.0	367400.0	64604.0
2	28204	35.22	-80.83	4796	2774.0	304600.0	56286.0
3	28205	35.22	-80.79	43931	3716.0	160100.0	35310.0
4	28206	35.25	-80.82	11898	1686.0	86400.0	21087.0
5	28207	35.20	-80.82	9280	3686.0	743500.0	119063.0
6	28208	35.24	-80.91	34167	1553.0	86400.0	28435.0
7	28209	35.18	-80.85	20317	3705.0	268300.0	60180.0
8	28210	35.13	-80.85	42263	3327.0	242500.0	54915.0
9	28211	35.17	-80.79	28523	2647.0	366700.0	70403.0
10	28212	35.19	-80.75	38457	4162.0	114400.0	33781.0
11	28213	35.28	-80.73	37309	2700.0	136700.0	42405.0
12	28214	35.28	-80.96	34721	1060.0	126100.0	53527.0
13	28215	35.25	-80.69	53629	1757.0	127500.0	45983.0
14	28216	35.31	-80.90	47208	1579.0	123400.0	48491.0
15	28217	35.17	-80.92	24204	1634.0	107500.0	38832.0
16	28226	35.11	-80.83	37286	2500.0	281800.0	68291.0
17	28227	35.18	-80.64	49635	1283.0	149700.0	51527.0

Cluster Analysis for Demograpic data

]:

```
zip_clusters.groupby('demo_cluster').mean().reset_index()
```

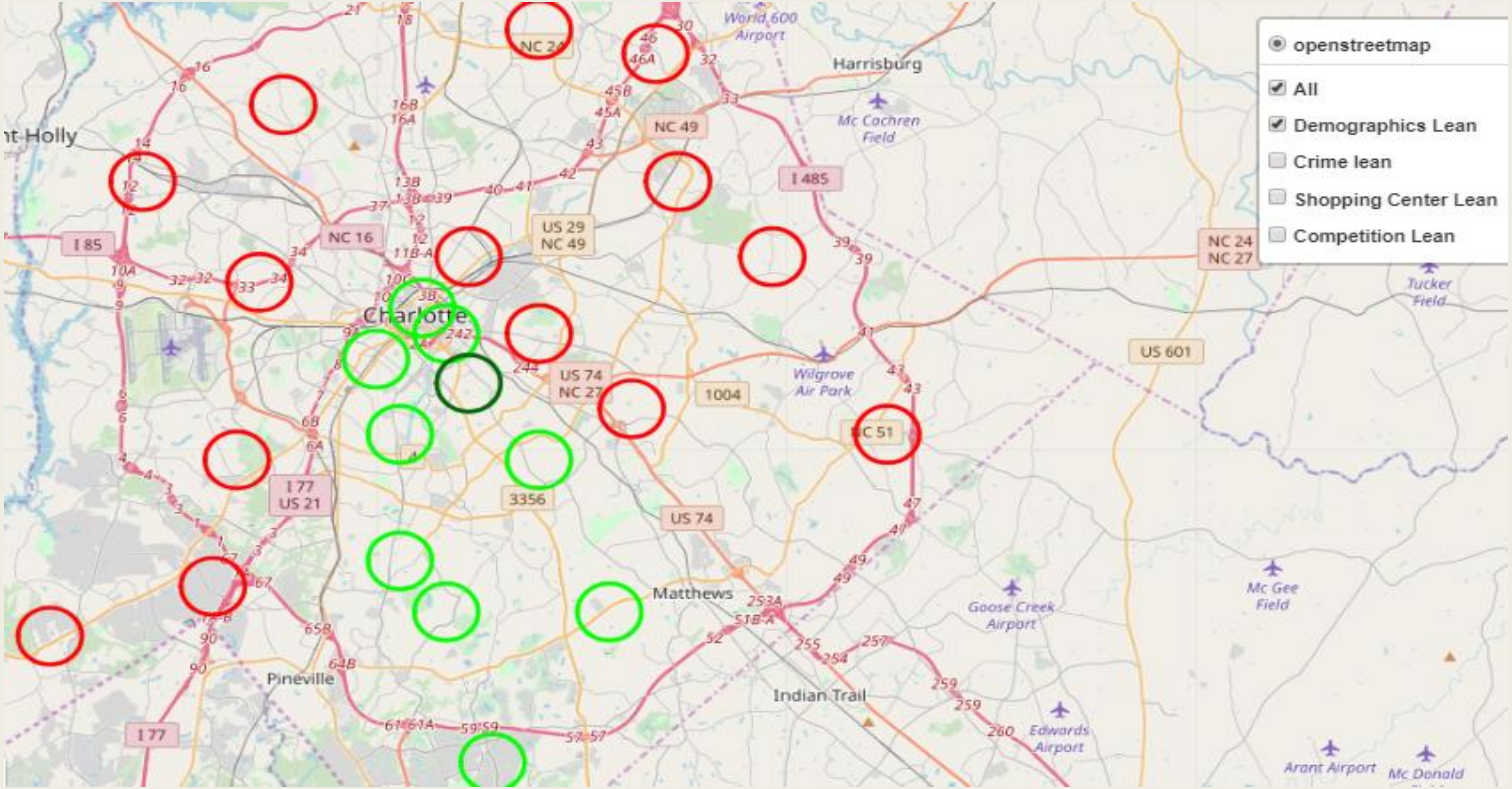
]:

	demo_cluster	population	population_density	median_home_value	median_household_income
0	0	38179.642857	1950.642857	134028.571429	45467.357143
1	1	9280.000000	3686.000000	743500.000000	119063.000000
2	2	27431.555556	3291.777778	292644.444444	69044.111111

- Cluster #0 - Low Income Low Density - Lean would be No
- Cluster 1 - High Density High Income - Lean is Strong Yes
- Cluster 2 - High Population Medium Income – Yes

Demographic Clustering

- Uptown (epicenter of the city) and South of Charlotte are favored
- 28207 got a Strong Yes (Cluster #1)
- See the Jupyter notebook for info on cluster labeling



Crime/Safety Clustering

- Used one function to get data from all 3 APIs
- Primary purpose is to calculate number of incidents by zip code

```
#this function takes in API info for crime data and the type of incident, and returns a Dataframe with the crime info from the A
PI. Mainly we are interested
#in the location of the crime. As this factors into our recommendation for the restaurant

def get_crime_data(url,type_crime):
    crime_results = requests.get(url).json()
    if crime_results:
        crime_df = json_normalize(crime_results['features'])
        crime_df.rename(index=str, columns={"attributes.ObjectID": "type","attributes.Longitude": "longitude","attributes.Latitu
de":"latitude","attributes.YR":"year" }, inplace=True)
        crime_df = crime_df[['latitude','longitude','year']]
        crime_df['type'] = type_crime
        return crime_df
    else:
        print ("problem with API call for", type_crime ," at", url)
```

Crime Clusters Analysis

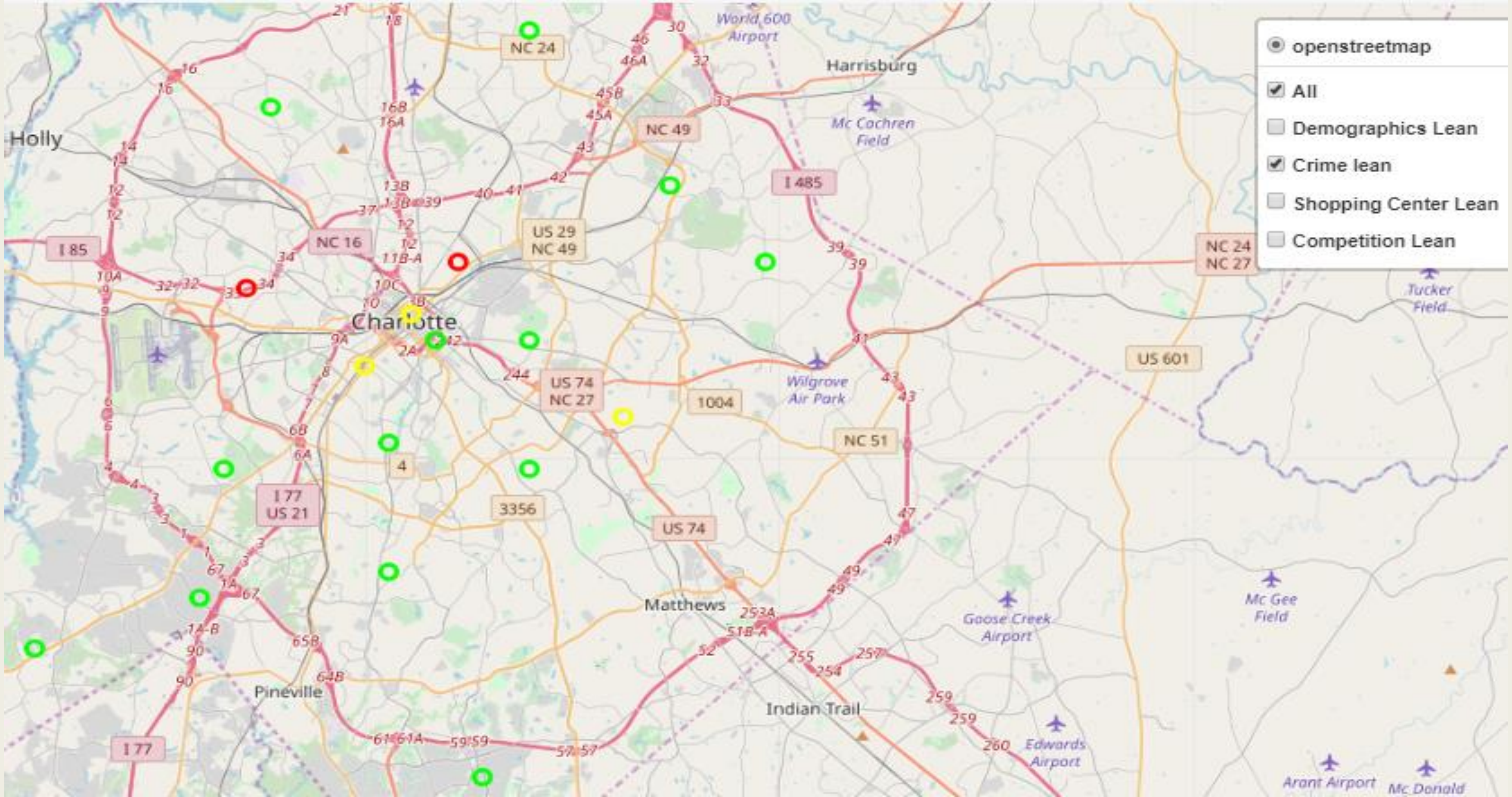
```
agg_crime_df.groupby('crime_cluster').mean().reset_index()
```

	crime_cluster	type
0	0	0.000286
1	1	0.003322
2	2	0.001592

- Cluster 0 - Low Crime - Yes
- Cluster 1 - High Crime – No
- Cluster 2 - Medium Crime - Maybe

Crime/Safety Clustering

- Crime rate is high or medium close to the epicenter of the city
- Looks pretty safe everywhere else
- Note that this is based on limited data available from Charlotte Open Data portal
- Size of circles on the map implies weightage for this particular aspect
- Colors indicate our recommendation disposition



Shopping Center Data Analysis

- We want to look at type of shopping centers and their size by zip code
- Since zip code is not available, we extract latitude, longitude from shape data in the API

```
shopping_center_df.head(2)
```

	latitude	longitude	type	size	polygon
0	-80.749288	35.306797	Regional	745951.0	[[[-80.74928773616489, 35.30679682239016], [-8...
1	-80.736158	35.201798	Neighborhood	81809.0	[[[-80.73615768708153, 35.20179771919721], [-8...

```
shopping_center_df.head(10)
```

	type	size	zipcode
0	Regional	745951.0	28262
1	Neighborhood	81809.0	28212
2	Regional	582651.0	28270
3	Neighborhood	113041.0	28226
4	Convenience	55761.0	28262
5	Community	181771.0	28217
6	Community	225534.0	28205
7	Community	238135.0	28105
8	Convenience	58105.0	28208
9	Neighborhood	106297.0	28208

Shopping Center Data Analysis

- Data is transposed before we standardize and cluster
- Shopping Centers are clustered into three
  - Cluster 0 - Less Desirable (Maybe)
  - Cluster 1 - Desirable (Yes)
  - Cluster 2 – Desirable (Yes)

```
shopping_center_df  = shopping_center_df.drop(['level_0'],axis=1, inplace = False)
shopping_center_df.head(5)
```

type	index	zipcode	Community	Convenience	Neighborhood	Regional	Super-Regional
0	0	28031	458404.0	531130.0	437427.0	1291376.0	0.0
1	1	28036	0.0	44017.0	0.0	0.0	0.0
2	2	28078	157961.0	57189.0	279598.0	0.0	0.0
3	3	28104	0.0	0.0	206489.0	0.0	0.0
4	4	28105	758467.0	41481.0	193602.0	1379701.0	0.0

Shopping Center Cluster Analysis

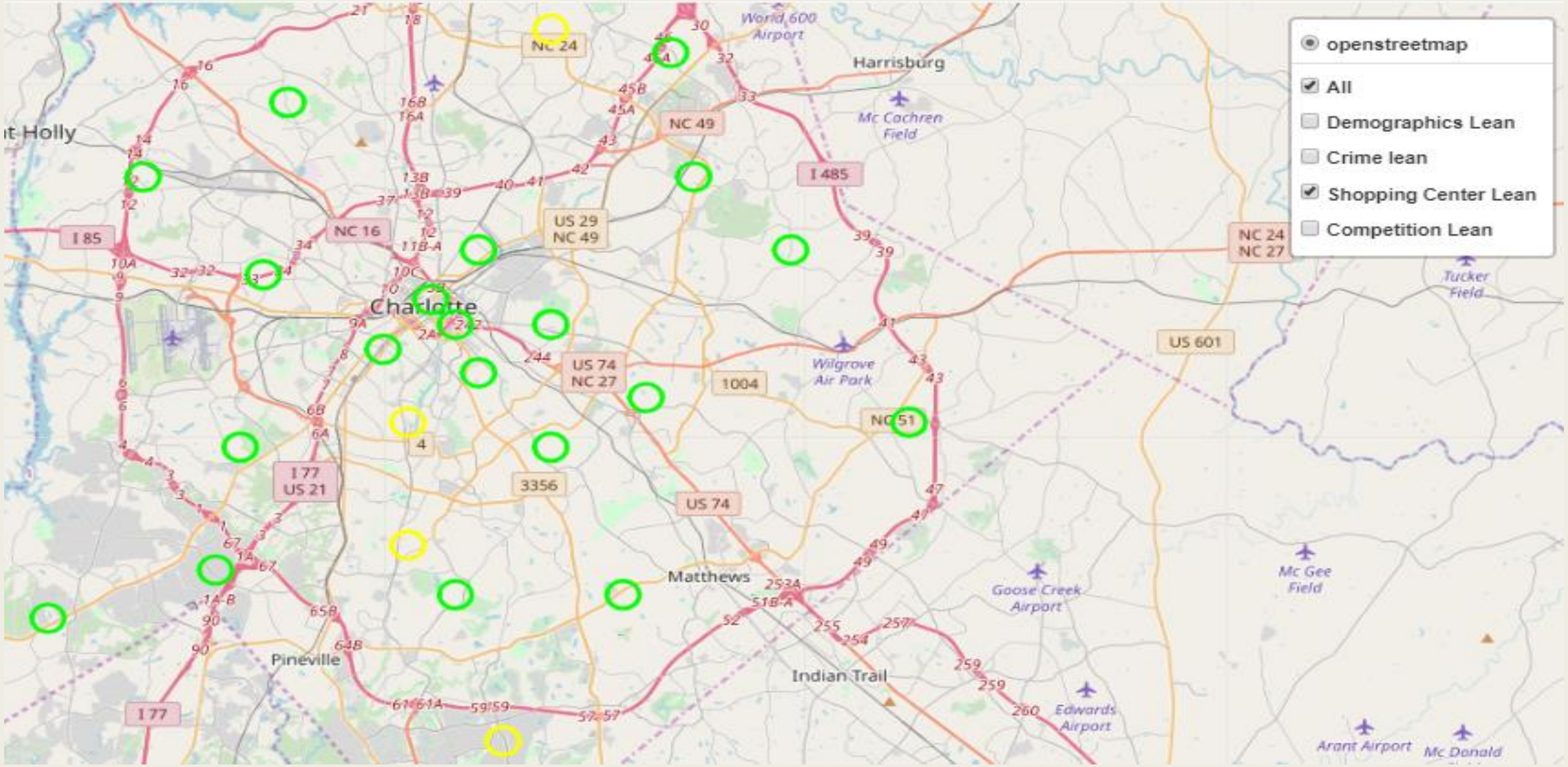
```
]: shopping_center_df.groupby('sc_cluster').sum()
```

]:

type	Community	Convenience	Neighborhood	Regional	Super-Regional
sc_cluster					
0	732369.0	720687.0	961719.0	2292697.0	0.0
1	3574551.0	2015678.0	2060457.0	4571237.0	890000.0
2	1838032.0	527384.0	1746960.0	3641737.0	4714003.0

Shopping Center Data Analysis

- Most zip codes where data is available are rated favorably except some in south and north of epicenter



Competition

- Use FourSquare to get restaurants and their category in all Zipcodes in Charlotte
- Condense 43 categories to 7 categories to interpret clustering meaningfully

```python
def get_search_results (query,latitude,longitude,radius=2000,limit=500):
    url ='https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    latitude,
    longitude,
    VERSION,
    query,
    radius,
    limit)

    results = requests.get(url).json()
    try:
        venues = results['response']['venues']
        venues_detail = json_normalize(venues)
        venues_detail['categories']=venues_detail.apply(get_category_type, axis=1)
        columns = ['categories','location.postalCode','name']
        venues_detail = venues_detail[columns]
        venues_detail.columns = ['category','zipcode','name']
        return venues_detail
    except:
        emptyframe = pd.DataFrame(columns=['category','zipcode','name'])
        return emptyframe
```

```python
# mapping 43 or so different kinds of restaurants down to a few so that I can make sense of the clusters.
category_dict={'Food':'Unknown','American Restaurant':'American','Italian Restaurant':'European','Mexican Restaurant':'Latin American','Fast Food Restaurant':'Unknown',
               'Restaurant':'Unknown','Greek Restaurant':'European','Chinese Restaurant':'Asian','New American Restaurant':'American','Southern / Soul Food Restaurant':'Amer
               'Japanese Restaurant':'Asian','Thai Restaurant':'Asian','Pub':'Drinking establishment','French Restaurant':'European','Ethiopian Restaurant':'African','Latin
               'Asian Restaurant':'Asian','Caribbean Restaurant':'African','Office':'Unknown','Hotel':'Unknown','Diner':'Unknown','Spanish Restaurant':'Latin American','Seaf
               'Bar':'Drinking establishment','Sushi Restaurant':'Asian','Breakfast Spot':'Unknown','Food Service':'Unknown','Cuban Restaurant':'Latin American','Indian Rest
               'Middle Eastern Restaurant':'Asian','Brewery':'Drinking establishment','Steakhouse':'Latin American','Argentinian Restaurant':'Latin American','Salad Place':'
               'Sports Bar':'American','Karaoke Bar':'Asian','Bowling Alley':'Unknown','General College & University':'Unknown','Theme Restaurant':'Unknown','Vietnamese Rest
               'Kitchen Supply Store':'Unknown','Peruvian Restaurant':'Latin American','Beer Garden':'American','Colombian Restaurant':'Latin American','None':'Unknown'}
```

Competition

- Aggregate restaurant information by Category
- Normalize number of restaurants to population before using standard scaler for clustering

```
restaurant_agg_df.drop(['level_0'],axis=1, inplace= True)
restaurant_agg_df  = restaurant_agg_df.fillna(0).reset_index()
restaurant_agg_df
```

category	index	zipcode	African	American	Asian	Drinking establishment	European	Latin American	Unknown
0	0	28202	0.0	35.0	3.0	7.0	4.0	5.0	24.0
1	1	28203	0.0	0.0	4.0	1.0	6.0	0.0	7.0
2	2	28204	3.0	7.0	3.0	0.0	7.0	0.0	12.0
3	3	28205	3.0	6.0	6.0	0.0	1.0	11.0	13.0
4	4	28206	0.0	3.0	0.0	0.0	0.0	1.0	6.0

```
# dividing each column by population from that zipcode so that clustering is done on standardized data across zipcodes
restaurant_onehot_df[['African','American','Asian','Drinking establishment','European','Latin American','Unknown']].div(restaurant_onehot_df.population, axis=0)
```

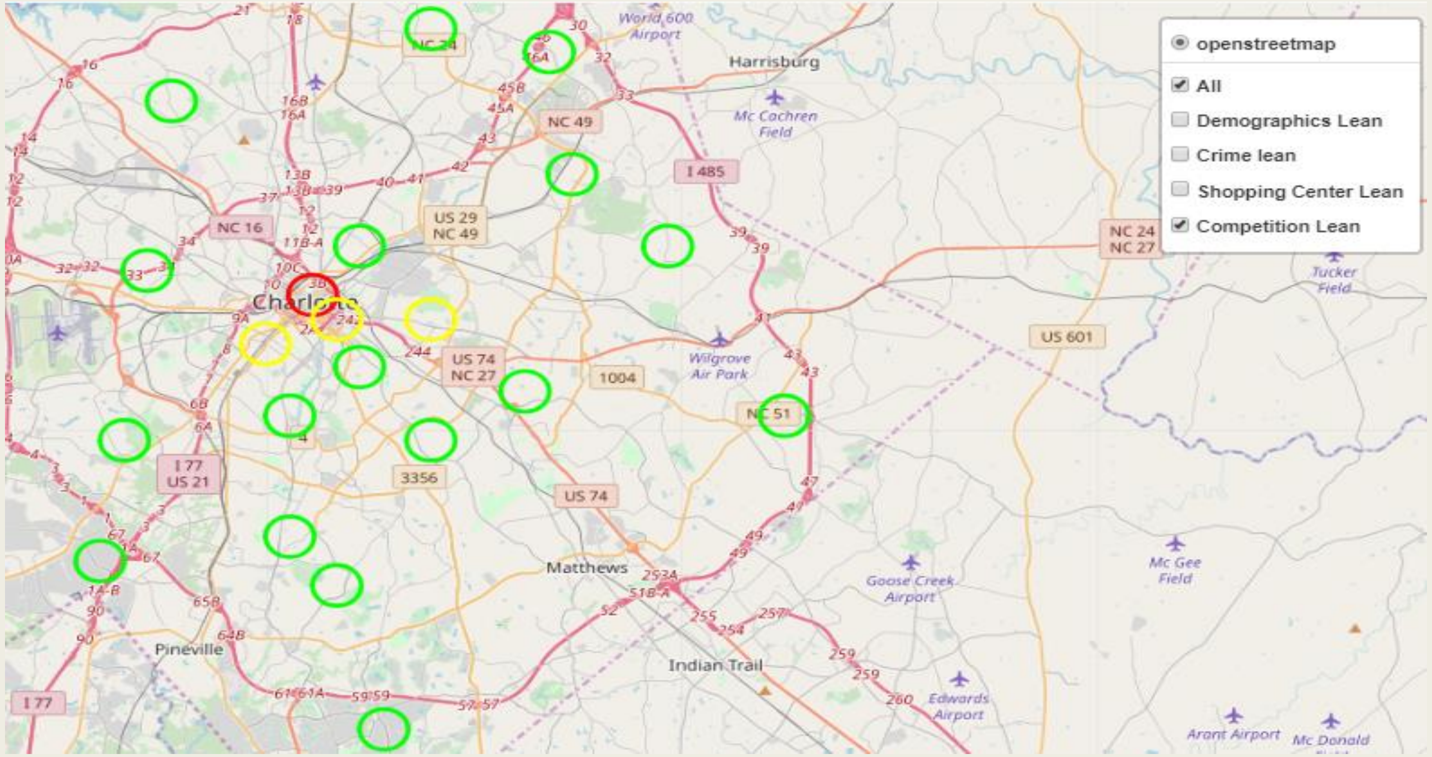
category	African	American	Asian	Drinking establishment	European	Latin American	Unknown
0	0.000000	0.003126	0.000268	0.000625	0.000357	0.000447	0.002144
1	0.000000	0.000000	0.000354	0.000088	0.000530	0.000000	0.000619
2	0.000626	0.001460	0.000626	0.000000	0.001460	0.000000	0.002502
3	0.000068	0.000137	0.000137	0.000000	0.000023	0.000250	0.000296
4	0.000000	0.000252	0.000000	0.000000	0.000000	0.000084	0.000504
5	0.000000	0.000431	0.000216	0.000000	0.000431	0.000000	0.000108
6	0.000000	0.000029	0.000059	0.000000	0.000000	0.000000	0.000234
7	0.000000	0.000049	0.000098	0.000098	0.000049	0.000000	0.000098

Competition

Restaurant Cluster Analysis

```
restaurant_agg_df.groupby('r_cluster').mean().reset_index()
```

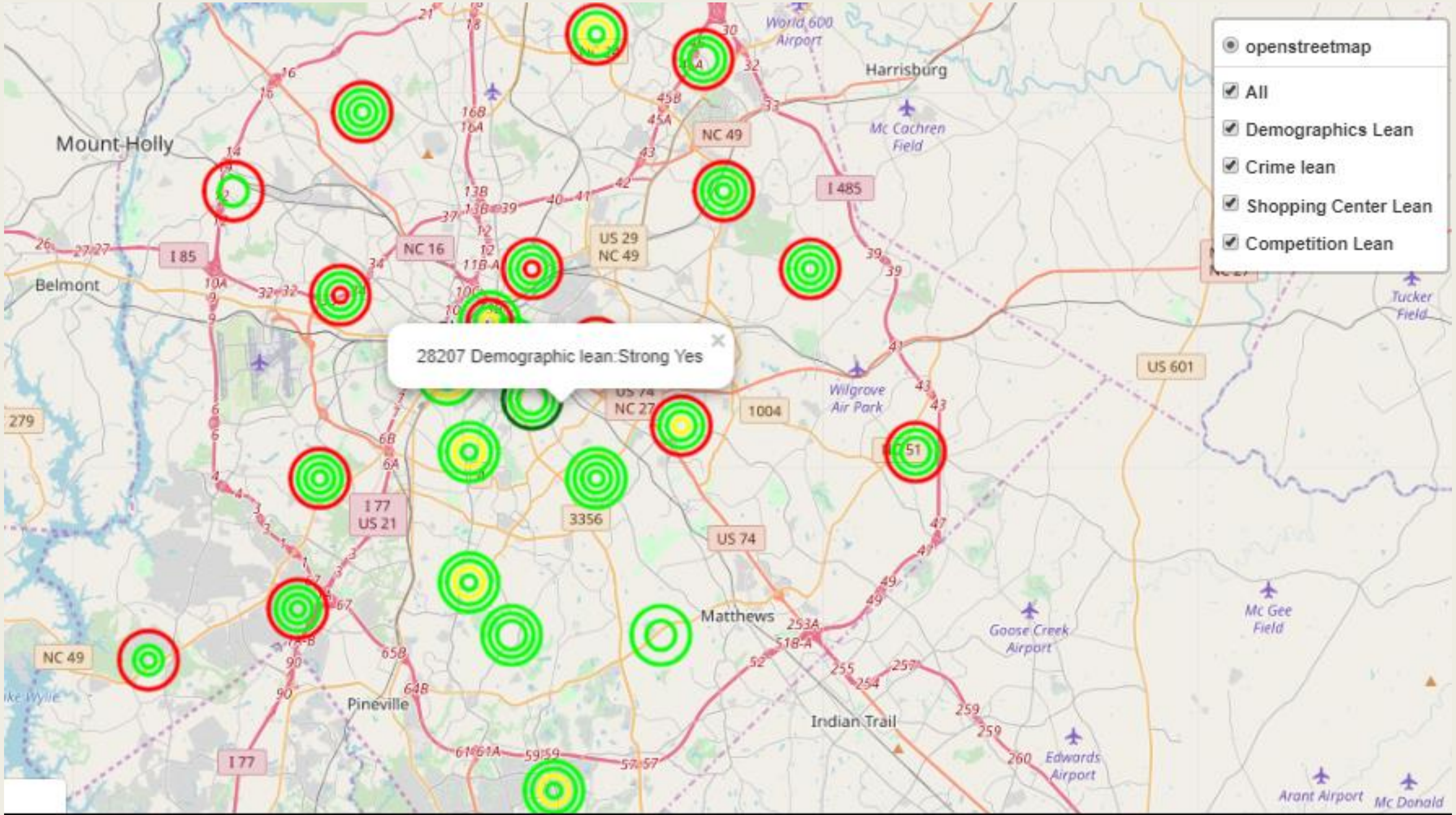
category	r_cluster	African	American	Asian	Drinking establishment	European	Latin American	Unknown
0	0	0.176471	1.000000	1.058824	0.117647	0.647059	0.823529	2.470588
1	1	0.000000	35.000000	3.000000	7.000000	4.000000	5.000000	24.000000
2	2	2.000000	4.333333	4.333333	0.333333	4.666667	3.666667	10.666667



Cluster 0 – Few Restaurants – Yes
Cluster 1 – Crowded – No
Cluster 2 – Too many Asian Restaurants - Maybe

Bringing it all Together

- There are three zipcodes for which all 4 dimensions are rated favorably
- 28207 takes the lead as demographic lean (the most important dimension) is rated most favorable among all
- 28211 and 28226 follow behind 28207
- Note that No crime data for these zip codes is positive (that means no incidents occurred in these zip codes)



Bringing it all Together – Tabular form

Final Tally - 28207 is the winner, 28211 and 28226 are close second.

```
final_tally_df.fillna('None')
```

	zipcode	demographic lean	safety lean	economic activity lean	competition lean
0	28202	Yes	Maybe	Yes	No
1	28203	Yes	Maybe	Yes	Maybe
2	28204	Yes	Yes	Yes	Maybe
3	28205	No	Yes	Yes	Maybe
4	28206	No	No	Yes	Yes
5	28207	Strong Yes	None	Yes	Yes
6	28208	No	No	Yes	Yes
7	28209	Yes	Yes	Maybe	Yes
8	28210	Yes	Yes	Maybe	Yes
9	28211	Yes	Yes	Yes	Yes
10	28212	No	Maybe	Yes	Yes
11	28213	No	Yes	Yes	Yes
12	28214	No	None	Yes	None
13	28215	No	Yes	Yes	Yes
14	28216	No	Yes	Yes	Yes
15	28217	No	Yes	Yes	Yes
16	28226	Yes	None	Yes	Yes
17	28227	No	None	Yes	Yes
18	28262	No	None	Yes	Yes
19	28269	No	Yes	Maybe	Yes
20	28270	Yes	None	Yes	None
21	28273	No	Yes	Yes	Yes
22	28277	Yes	Yes	Maybe	Yes
23	28278	No	Yes	Yes	None

Caveats

- We could use PAC (probably approximately correct) learning to condense number of variables we used in clustering. I know such thing existed, but did not have time to learn about it, yet.
- Population and Population density may not always tell the story. For example, uptown of the city is mostly office space that represents tremendous opportunity for a restaurant (as rightly reflected by the # of restaurants in the area). But our model tell us to avoid uptown. While this still could be correct, more research need to be done.
- Weightage for each dimension are subjective. So is disposition we assigned for each cluster
- It is entirely possible and probably true that this is a truly novice version and there are far better methods and models to solve this problem.

Appendix

- Link to the Jupyter Notebook to see the code and maps: https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/750ca25b-69e8-422b-8a80-2b5986f3acd2/view?access_token=619a8df1f4af52235d763ce442bcda894ea18137cb1b8dd3ae99c739112ec6aa

- Link to Github page: