Identification of the Best Location for a Medical Practice Based on Professional and Personal Data

Applied Data Science Capstone Project

Prepared By:

William Sanborn

July 3, 2019

Introduction

- Goal: Design a Model Based on Data Analysis That Will Advise a Doctor on Where He/She Might Want to Set Up a Medical Practice
- Model Will Consider the Professional Goals and Personal Interests of the Doctor
- Data Sources: Centers For Disease Control (CDC) 500 Cities Project, Foursquare and Other Publicly Available Information (See Appendix 1)
- Tools: Various Python Based Data Libraries (See Appendix 2)
- Assumptions:
  - The Assumed Doctor is a Cardiologist
  - The Doctor Enjoys Sushi, Wine, Museums, and Live Music
  - The Doctor is an Avid Golfer
  - The Doctor Has a Dog That Requires Regular Exercise

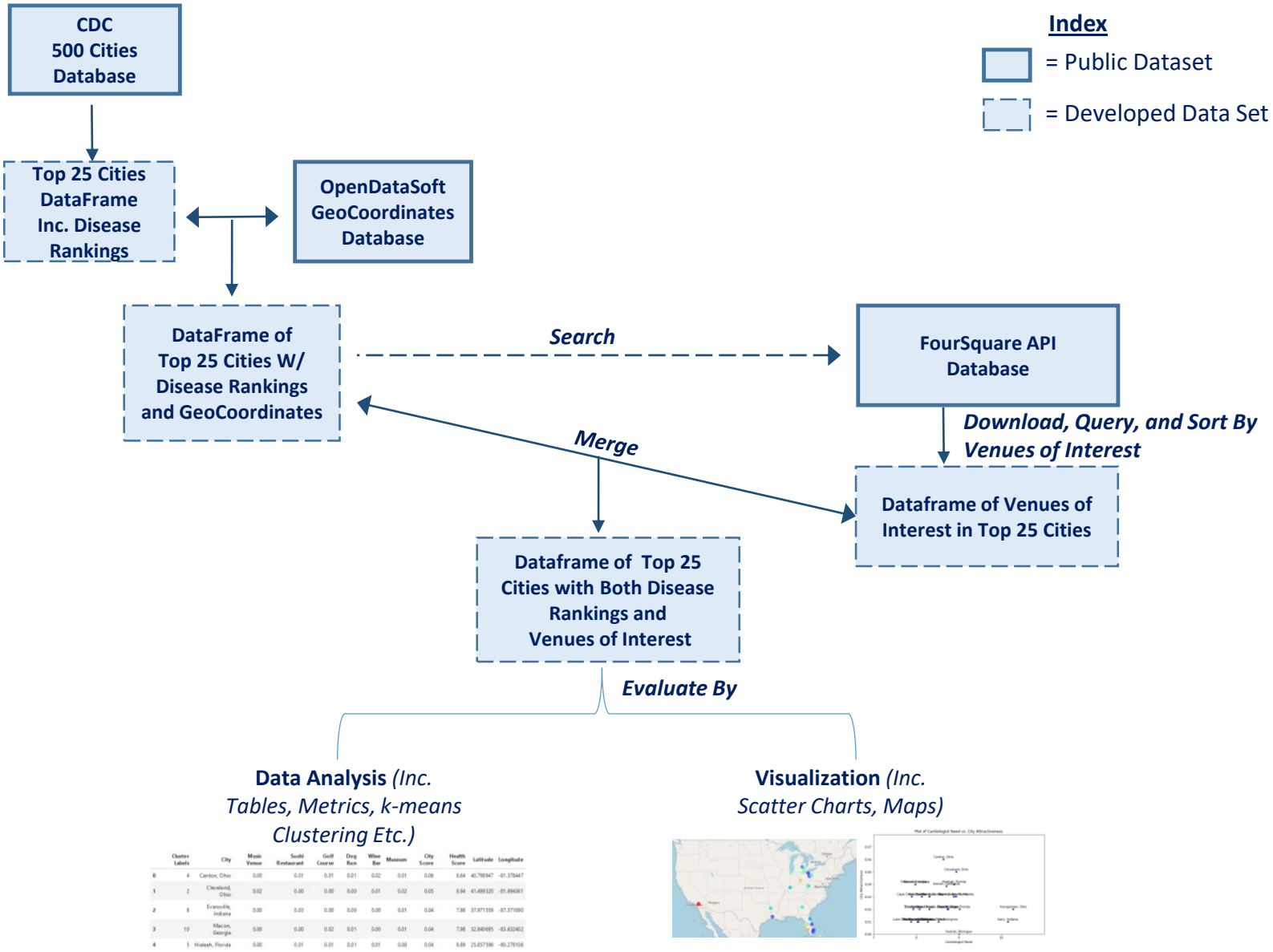
Goal: Answer the Following Question Through Data Analysis

What is the Best City for My Assumed Doctor to Start a Cardiology Practice?

Data

Source	Raw Data	Data Objective
CDC Centers for Disease Control and Prevention; 500 Cities: Local data for better health	Data on Levels of Heart Disease In Major US Cities	Dataframe of Top 25 US Cities With Highest Levels of Heart Disease
FOURSQUARE	Data on Venues in Each Requested City	Dataframe of Venues of Personal Interest fir Each of Top 25 Cities
OpenDataSoft	Data on GeoLocation Coordinates of 1000 US Cities	Dataframe of GeoLocation Coordinates for Each of Top 25 Cities (Note: Needed for Foursquare API)

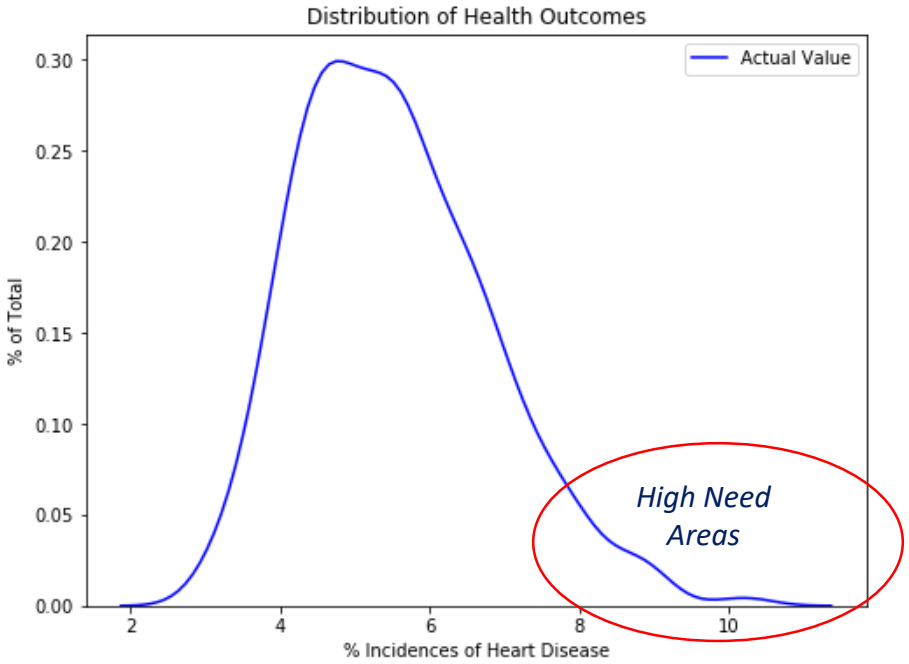
Project Methodology



Inferential Statistical Testing of CDC Data

Inferential Statistic Testing Difficult

- No Historical Information Available to Train and Test Data
- No Independent and Dependent Variables Identified
- Data Does Indicate the Presence of High Need Cities for Heart Related Medical Services




```
DB4['Data_Value'].describe()
```

```
count    500.000000
mean       5.535936
std        1.277269
min        2.975000
25%        4.560425
50%        5.411263
75%        6.365949
max       10.272727
Name: Data_Value, dtype: float64
```

Since Inferential Approaches Difficult, New Approach Needed to Identify Cities in Need of Heart Related Medical Services

CDC Data Process: “City Heart Health”

Start with ~28,000 Records in CDC Database

```
HD_DB['Data_Value'].describe()
```

```
count    28212.000000
mean         5.601081
std          2.092404
min          0.300000
25%          4.200000
50%          5.300000
75%          6.700000
max         35.800000
Name: Data_Value, dtype: float64
```

Steps:

- Group By City (Average ‘Data_Value’)
  - Average Data_Value=“Ave. Rate”
- Sort By Ave. Rate
  - Ascending = ‘False’
- Limit = Top 25
- Merge With GeoLocation Data
  - Coordinates: String➔Float

Final DataFrame: 25 Cities With Highest Levels of Heart Disease

```
# Create a new dataframe of the top 25 cities
limit = 25
map_db = city_db_sorted.iloc[0:limit, :]
map_db.reset_index(inplace=True)
map_db.head(limit)
```

	index	City	Ave. Rate	Latitude	Longitude
0	475	Youngstown, Ohio	10.27	41.099780	-80.649519
1	153	Gary, Indiana	10.15	41.593370	-87.346427
2	211	Largo, Florida	9.02	27.909467	-82.787324
3	114	Detroit, Michigan	9.00	42.331427	-83.045754
4	86	Cleveland, Ohio	8.94	41.499320	-81.694361
5	49	Boynton Beach, Florida	8.93	26.531787	-80.090547
6	171	Hialeah, Florida	8.89	25.857596	-80.278106
7	308	Palm Coast, Florida	8.88	29.584452	-81.207870
8	138	Flint, Michigan	8.75	43.012527	-83.687456
9	106	Dayton, Ohio	8.73	39.758948	-84.191607
10	168	Hemet, California	8.71	33.747520	-116.971968
11	62	Canton, Ohio	8.64	40.798947	-81.378447

Foursquare Process: “City Attractiveness”

Create URL Request

Send Request Using Request Library
- Top 25 City Lat/Long Coordinates
- Venues within 16,000 Meters from Coordinates
- Limit = 100

FOURSQUARE

JSON File of All Venues for Top 25 Cities

Normalize and Filter

Pandas DataFrame of All Venues for Top 25 Cities

Filter By
- 'Music Venue'
- 'Sushi Restaurant'
- 'Golf Course'
- 'Dog Run'
- 'Wine Bar'
- 'Museum'

Pandas DataFrame of Venues of Interest for Top 25 Cities

Calculate ‘Avg. Score’ /City Score Based on OneHot Coding

DataFrame of Top 25 Cities With “City Score”

	City	Music Venue	Sushi Restaurant	Golf Course	Dog Run	Wine Bar	Museum	Avg. Score
0	Boynton Beach, Florida	0.00	0.01	0.02	0.00	0.00	0.00	3.0
1	Canton, Ohio	0.00	0.01	0.01	0.01	0.02	0.01	6.0
2	Cape Coral, Florida	0.00	0.00	0.01	0.01	0.01	0.00	3.0
3	Charleston, West Virginia	0.01	0.00	0.00	0.00	0.00	0.00	1.0
4	Clearwater, Florida	0.00	0.01	0.00	0.00	0.01	0.00	2.0

Results: Consolidated Data Frame (Pandas)

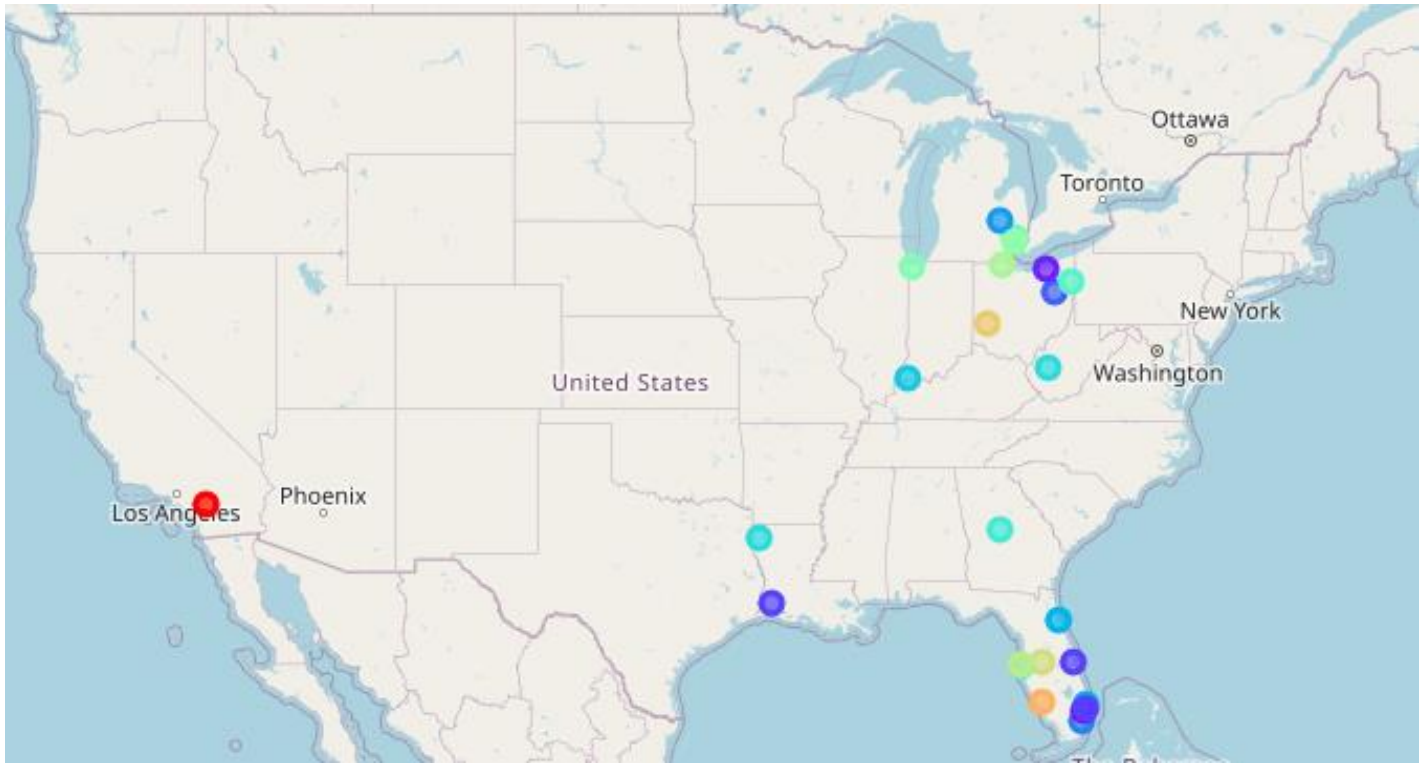
- Use Df.join to combine City Heart Health and City Attractiveness DataFrames
- Set Index on 'City'

	City	Latitude	Longitude	Cluster Label	Music Venue	Sushi Restaurant	Golf Course	Dog Run	Wine Bar	Museum	City Score	Health Score
0	Canton, Ohio	40.798947	-81.378447	3	0.00	0.01	0.01	0.01	0.02	0.01	6.0	8.64
1	Cleveland, Ohio	41.499320	-81.694361	5	0.02	0.00	0.00	0.00	0.01	0.02	5.0	8.94
2	Evansville, Indiana	37.971559	-87.571090	8	0.00	0.03	0.00	0.00	0.00	0.01	4.0	7.98
3	Macon, Georgia	32.840695	-83.632402	7	0.00	0.00	0.02	0.01	0.00	0.01	4.0	7.98
4	Hialeah, Florida	25.857596	-80.278106	13	0.00	0.01	0.01	0.01	0.01	0.00	4.0	8.89

Note: Df.Head() Only

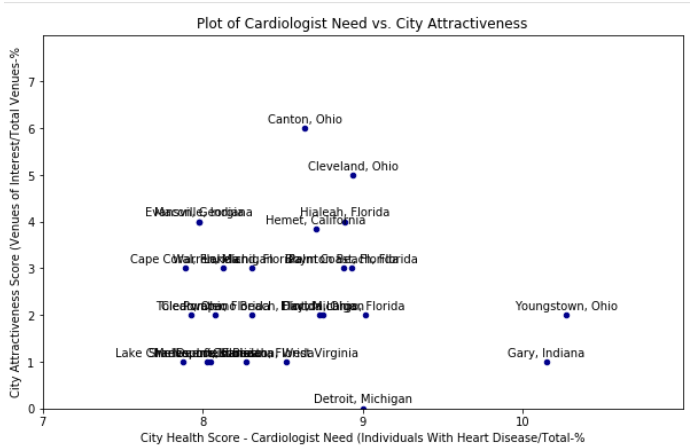
Final DataFrame Contains Information Needed for Analysis and Visualization

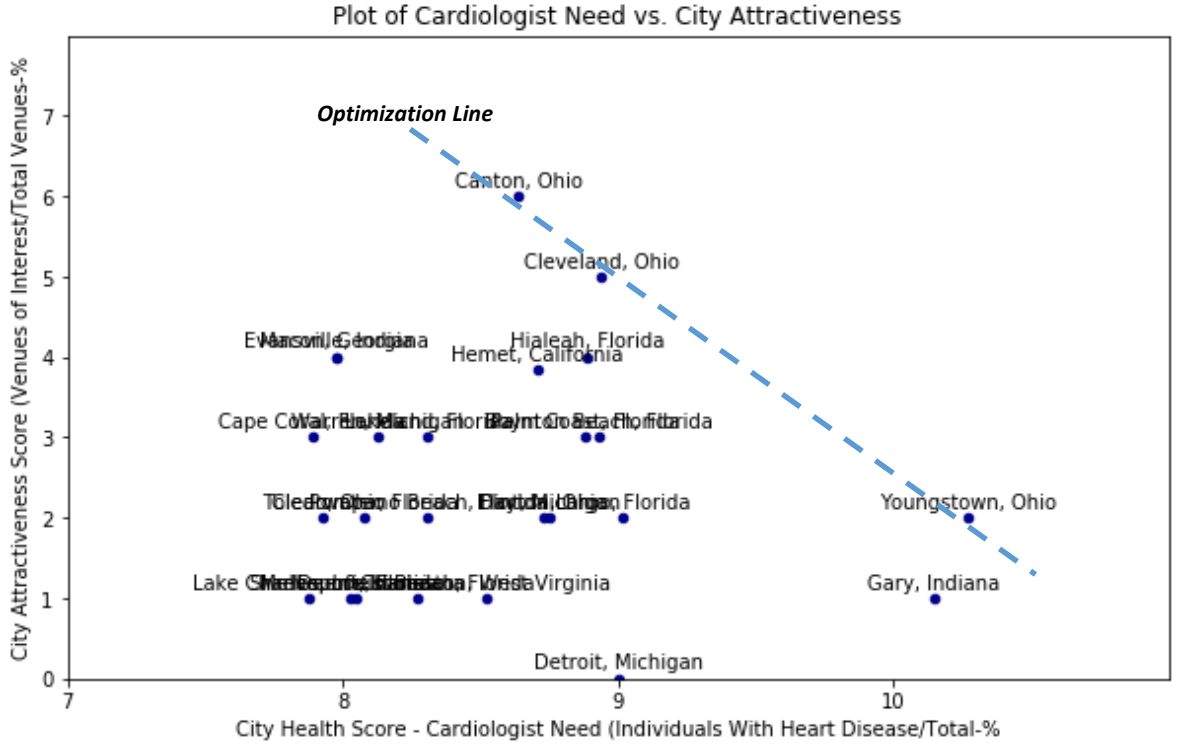
Results: Mapping Chart (SKLearn & Folium)



- Map of Kmeans Clusters
- Geographic Dispersity Makes Clustering Difficult
  - Only 20 Clusters Identified
- No Top Candidate(s) Stands Out so Additional Analysis Needed

Results: Scatter Chart (MatPlotLib)





- Scatter Charts Effectively Show How Cities Compare Based on Level of Heart Disease and City Attractiveness
- Optimization Line Provides Additional Clarification of Top Candidates

Discussion of Results

- Final DataFrame Contains All Attributes Needed for Analysis and Visualization.
  - However, the Data Alone Does Not Provide An Answer to the Posed Question ➔ Analytical Framework Needed
- Clustering Map Does Not Provide Enough Information to Make a Recommendation.
  - It Does Suggest that Doctor Should Target Midwest US and Southern Florida
- Scatter Chart Provides Good Framework to Make a Recommendation
  - Optimization Line Shows Best Candidates
  - Suggests: Canton, Cleveland, and Youngstown Ohio Should Be Recommended
- Issues
  - Determination of Doctor's Relative Preference of Personal vs. Professional Matters Could Provide Additional Refinement
  - Increasing Foursquare Limit From 100 Could Also Increase Likelihood of Correct Outcomes

Conclusion

- Model Successfully Uses Publicly Available Data (Such as Foursquare API) to Provide Recommendations that Align to an Individual's Personal and Professional Objectives
- For the Assumptions in the Cardiologist Test Case: Canton, Cleveland, and Youngstown Ohio are Best Options to Live
- Suggestions for Further Model Refinement
  - Gain Additional Insight Into Relative Weightings of the Cardiologist's Personal vs. Private Objectives
  - Increase Limit of Foursquare Venues Obtained to Gain Additional Insight Into Individual City Offerings

Recommendations to Cardiologist:

1) Canton, Ohio
2) Cleveland, Ohio
3) Youngstown, Ohio

Appendix 1: Sources

1) **Information on Heart Disease By City**: https://chronicdata.cdc.gov/500-Cities/500-Cities-Coronary-heart-disease-among-adults-age/cqcq-r6f8/data
   - Note: Measure: % Respondents aged ≥18 years who report ever having been told by a doctor, nurse, or other health professional that they had angina or coronary heart disease. https://www.cdc.gov/500cities/definitions/health-outcomes.htm

2) **Database of Geographic (Latitude/Longitude) Coordinates By City:** https://public.opendatasoft.com/explore/dataset/1000-largest-us-cities-by-population-with-geographic-coordinates/table/?sort=-rank

3) **Database(s) of City Venue Detail:** *FourSquare API:* https://developer.foursquare.com/

Appendix 2: Python Libraries Used

- **Folium:** *Visualization and mapping*
- **Geocoder/Nominatim:** *Generate and Read Location Data*
- **JSON:** *Analyze JSON files*
- **Matplotlib:** *Python plotting and graphing*
- **Numpy:** *Arrays and Data Set Functions*
- **Pandas:** *Misc. DataFrame Functions*
- **Requests:** *Generate API requests*
- **Seaborn:** *Statistical Data Visualization*
- **SkLearn:** *K-means clustering*