Gun Violence in U.S. Cities: A Clustering Exercise with DBSCAN

Justin Prachar
Data Mining: Final Project (Cpts 315)
Washington State University
May 7, 2020

Introduction

Data with proper analyzation and visualization serves, otherwise, hidden knowledge to its viewer. The goal of this project is to meet those demands while making it as automated and effortless as possible.

There is no knowledge more important than that which can keep you living and safe. In the United States violent gun crimes present a real threat to people's safety. In 2016 there were 35 thousand hospitalizations due to gun violence[1], with each visit costing an average of $95k[2]. Nearly all of this occurs in cities. Through data mining we can find the patterns within the data to determine the exact locations within cities that present the most danger. This was done through density clusters.

My personal motivation for doing this project has to do with my love for travel and my desire to stay safe. I want to see as many U.S. cities as I can in my life and having the information of which neighborhoods demand caution is invaluable. Additionally, I wanted to incorporate maps whatever project I did, and this is a powerful way to do that.

Our results are in the form of density maps, where colored points are clusters and black are outliers. By changing the minPts we are able to adjust the density of the responding clusters.

Data Mining Task

Input data was 239,677 incidents. What areas of each city have the highest number of incidents?

Can we map the gun violence as a density?

We use DBSCAN because it is capable of making non uniform clusters. Where k-means and k- nearest neighbor are not as adept at dealing with this kind of special data with lots of noise.

Big challenges are how do we verify the data? How can we set epsilon ((radius) without human input? How will we get k-distance to k nearest neighbor? How efficient is the algorithm (runtime)? How many sets do we need to convey the information?

Technical Approach

Defined a city name and the min number of points for a cluster (density). Pass these into the function.

[1] https://www.rand.org/pubs/tools/TLA243-3.html
[2] https://doi.org/10.1377/hlthaff.2017.0625

Read in all the data from the csv. Keep only the data from the city that we care about. Are all the data really from the city we want?

Example: there is a Miami, OH and when mapping Miami, FL I would get a very skewed graph because of these massive outliers.

Solution: Use geopy to find the coordinates of cities. Check each data point against the city coordinates, if outside the tolerance, then drop.

Limitation: Only big cities should be used as geopy is given only a city name so it returns the coordinates of the most common city with that name.

Read in the shape file of the subject city. One with streets is most useful.

Plot all the data points on to this map using geopandas.

Using standardScaler normalize the coordinate data.

Given the minPts input data calculate the distance to the minPts-nearest neighbor for each point.

Arrange this list in descending order. MinMax scale it to between 0 and 1.

When the change between the values of this list is less than 1% (acknowledgement paper #2) for minPts values that is our threshold position and where the clusters start to form. The previous points are noise.

Plot these points and mark the threshold point.

Use the threshold position to find the unscaled length for DBSCAN's eps.

Run the DBSCAN algorithm from sklearn store the labels with the points.

Generate as many random colors as there are clusters.

Run the Silhouette Score evaluation for unlabeled clusters using the predicted clusters and the normalized position array.

Plot the points with their colors based on their predicted label.

Run time is O(n). Chicago with n = 11k takes about 20 seconds.

Here's the DBSCAN algorthmn (taken from original paper):

```
DBSCAN (SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point,
            ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

Evaluation Methodology

I used the silhouette score method from sklearn.

Found here: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

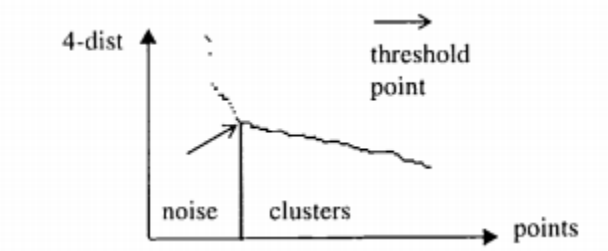
This method takes the mean distance from each point to each point (a) within the same label and the average distance to the nearest cluster (b). Then it calculates a coefficient: (b – a) / max(a, b). a value of 1 would be best -1 the worst.

Because real life data that I'm working on doesn't form pretty clusters I'm pretty happy with getting a positive value approaching .5

This is a very useful way to check on cluster data when no true labels are known.

Otherwise, visual inspection and domain knowledge are important.

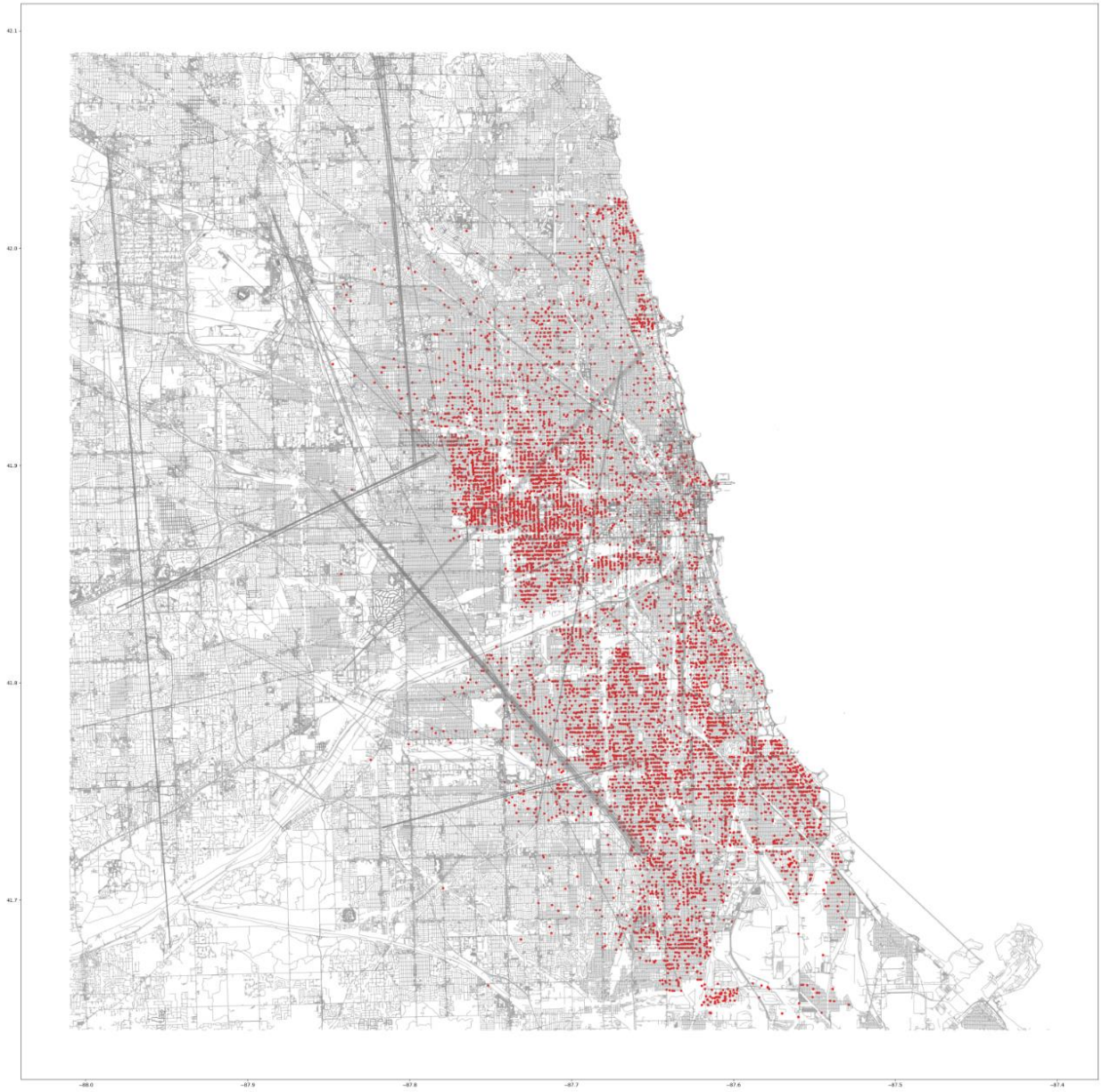
When using DBSCAN eps (1 of the two input variables) is normal entered by the user and is there for subject to human error. I have taken the DBSCAN authors advice and written into my program a way to automatically find this threshold point automatically through derivatives.



Results and Discussion

Chicago	10814
Baltimore	3943
Washington	3279
New Orleans	3071
Philadelphia	2963
Houston	2501
Saint Louis	2501
Milwaukee	2487
Jacksonville	2448
Memphis	2386
Columbus	2252
Indianapolis	1920
Detroit	1834
Cleveland	1784
Springfield	1755
Boston	1737
San Antonio	1628
Oakland	1501
Louisville	1425
Brooklyn	1418
Kansas City	1381
Jackson	1367
Atlanta	1358
Las Vegas	1348
Nashville	1329
Richmond	1317
Wilmington	1296
Charlotte	1291
Tulsa	1260
Dallas	1179
Columbia	1114
Cincinnati	1108
Los Angeles	1066
Fresno	1065
Dayton	1036

The top 25 cities when sorted by number of violent gun incidences from 1/2013 to 3/2018. I knew things were bad in Chicago but not this bad. At nearly 3 times higher incidents rates than the next highest city.



Here are those ~11k data points plotted onto a street map of Chicago.

41.8859/-87.6251	39
41.8781/-87.6298	27
41.7894/-87.7416	25
41.881/-87.6247	19
41.8666/-87.6902	16
41.8866/-87.6643	13
41.8797/-87.7355	13
41.8736/-87.7108	12
41.9009/-87.7238	10
41.802/-87.6194	9
41.7956/-87.6747	9
41.7639/-87.6249	9
41.8808/-87.7529	9
41.8674/-87.7229	9
41.899/-87.6343	9
41.8662/-87.7201	9
41.8847/-87.7111	9
41.9023/-87.7509	9
41.7783/-87.616	9
41.7741/-87.7202	8
41.8804/-87.7536	8
41.8642/-87.7105	8
41.8753/-87.7428	8
41.7792/-87.6849	8
41.8785/-87.755	8
41.8899/-87.7235	8
41.8569/-87.7326	8
41.8948/-87.7678	8
41.8813/-87.7039	8
41.7338/-87.6682	7
41.8939/-87.714	7
41.7733/-87.5839	7
41.7802/-87.6063	7
41.89/-87.6585	7
41.8783/-87.6867	7

According to the data the Chris Tower Apartments had 39-gun incidents over the 5 year period. The number is not too high for 11K cases.

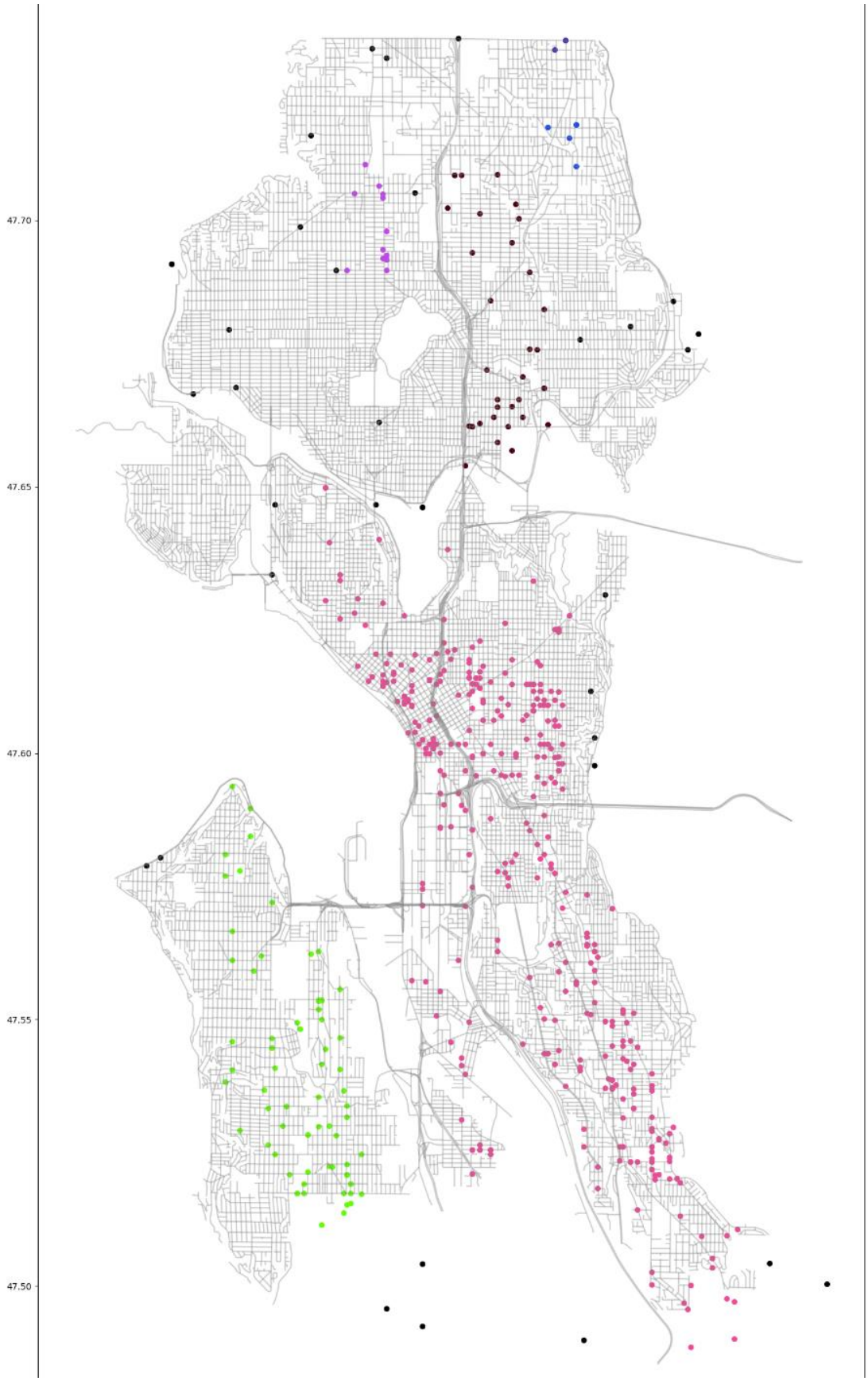
This is a street view of those apartments. Not sure if I believe that data. The next highest were the IRS building, Midway International Airport.

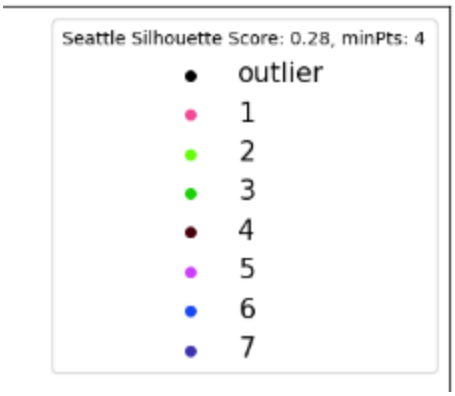
For a clearer example of why this location data is not to be trusted. Below is the most popular locations for Seattle:

47.4435/-122.296	88
47.6138/-122.332	33
47.4502/-122.309	9
47.6107/-122.339	5
47.6062/-122.332	5
47.6026/-122.334	4
47.608/-122.303	4
47.5448/-122.274	4
47.6129/-122.303	4
47.5208/-122.269	3
47.5628/-122.286	3
47.5376/-122.27	3
47.6009/-122.333	3
47.5233/-122.276	3
47.5993/-122.302	3
47.6062/-122.317	2
47.6141/-122.319	2
47.5262/-122.27	2
47.5261/-122.278	2
47.5256/-122.315	2
47.5256/-122.318	2
47.5256/-122.32	2
47.6126/-122.345	2
47.6123/-122.318	2
47.5242/-122.265	2
47.6685/-122.3	2
47.5235/-122.279	2
47.5497/-122.283	2
47.5233/-122.27	2
47.6144/-122.348	2
47.6131/-122.32	2
47.5284/-122.366	2
47.6613/-122.32	2
47.5291/-122.27	2
47.6499/-122.361	2

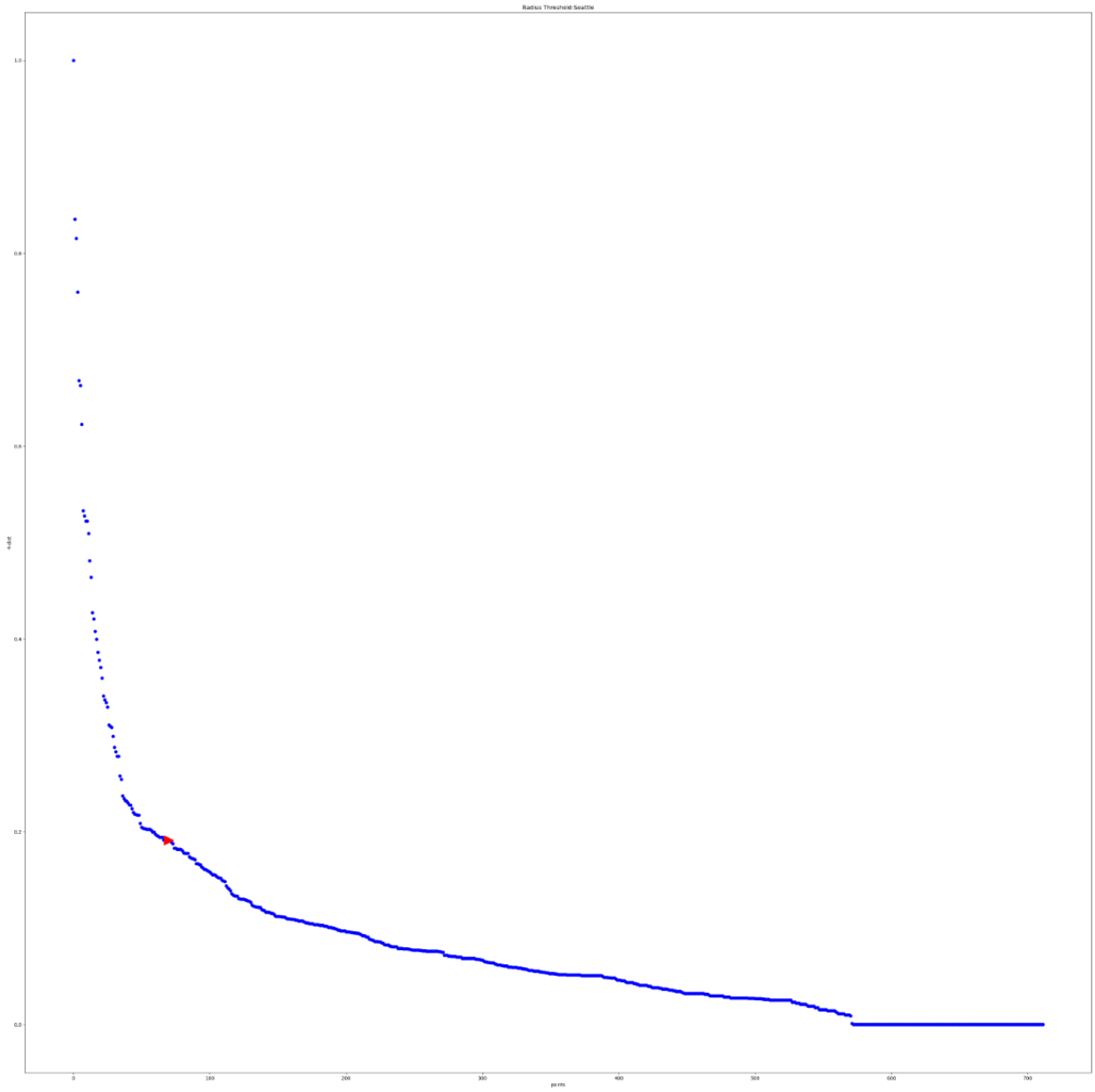
Wow! 88 incidents at the exact same coordinate, which turns out to be Sea-Tac airport. Given that Seattle only had 712 incidents, this makes up a significant 12%. Further investigating should be done as if there is a problem in the Seattle Police Department's data entry skill. 33 happened in a huge empty construction lot:

On to the cluster data:


47.70
47.65
47.60
47.55
47.50

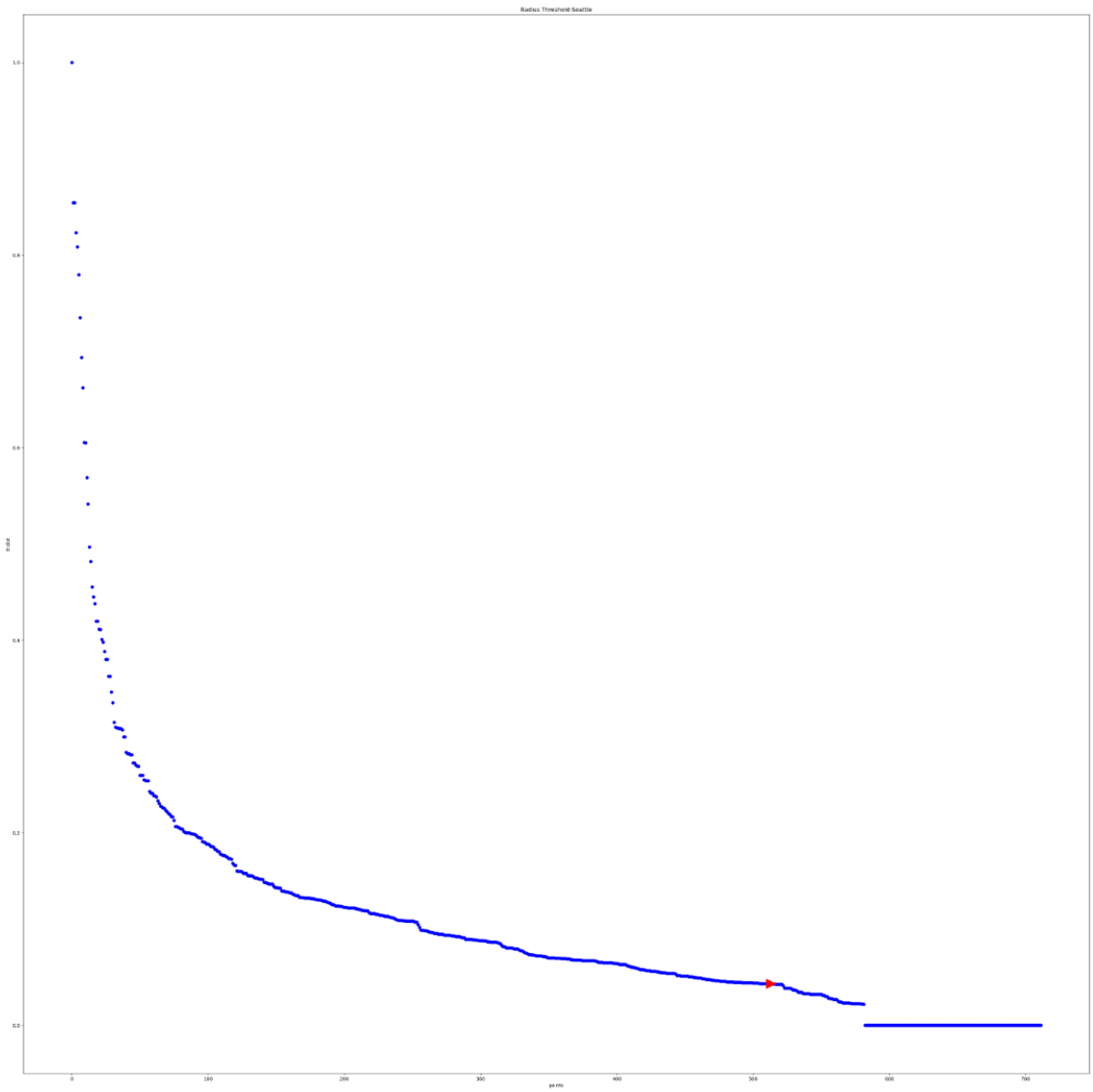


At 4 minPts, this is a low cluster density threshold. The colored areas represent the corresponding density, while black data points are outliers. Here the silhouette score is 0.28 in a range of [-1,1], where -1 means data is placed into the wrong clusters, 0 there are overlapping clusters, and 1 no overlapping clusters. Here is the threshold knee chosen by the program:

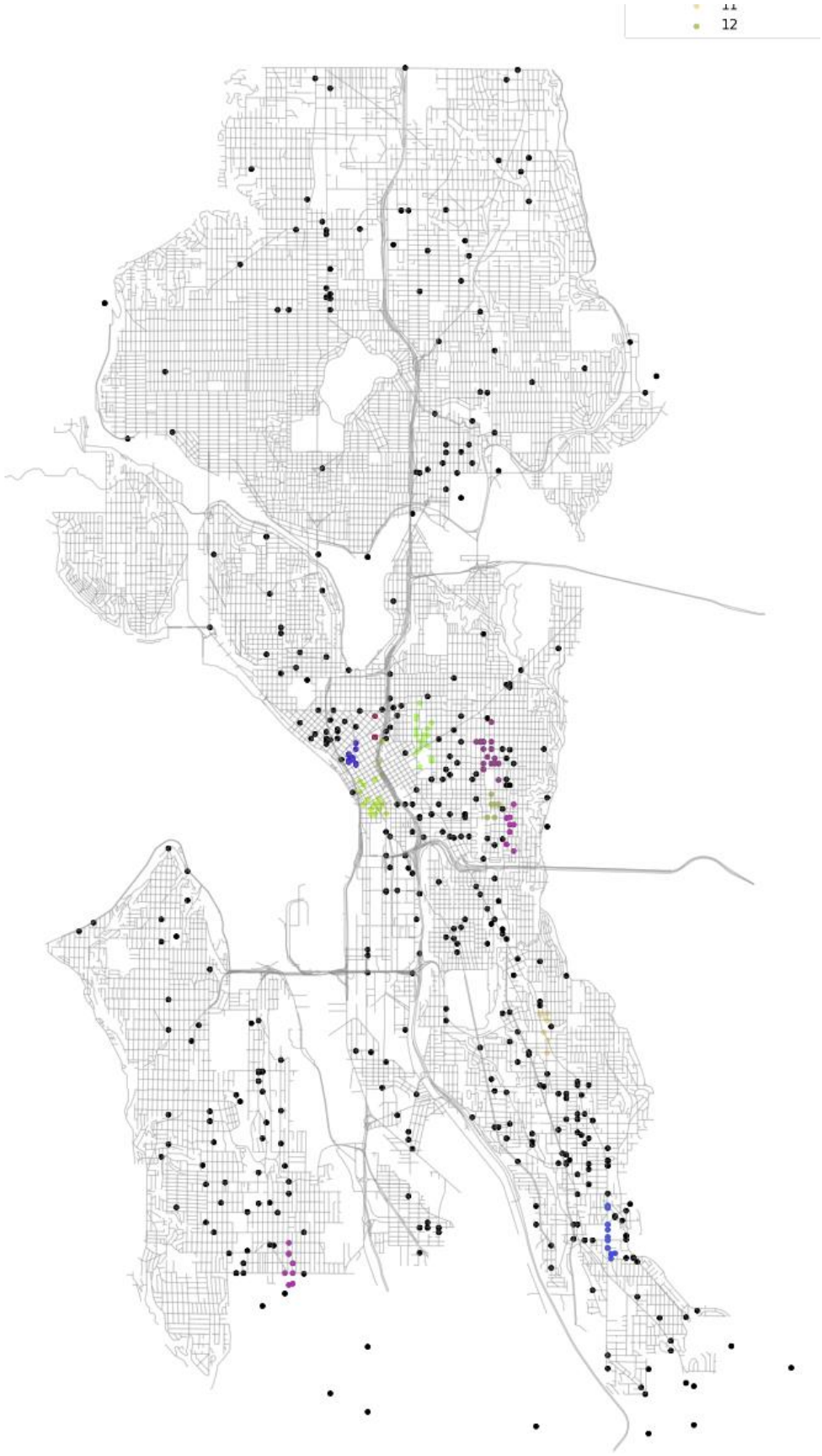


Everything to the left of the red arrow is noise.

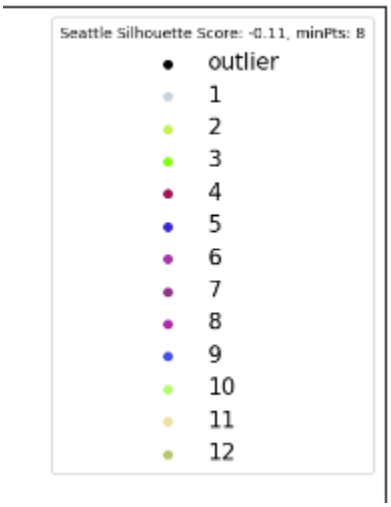
By increasing the minPts to 8 we get the high-density clusters. Observe how the threshold moves:



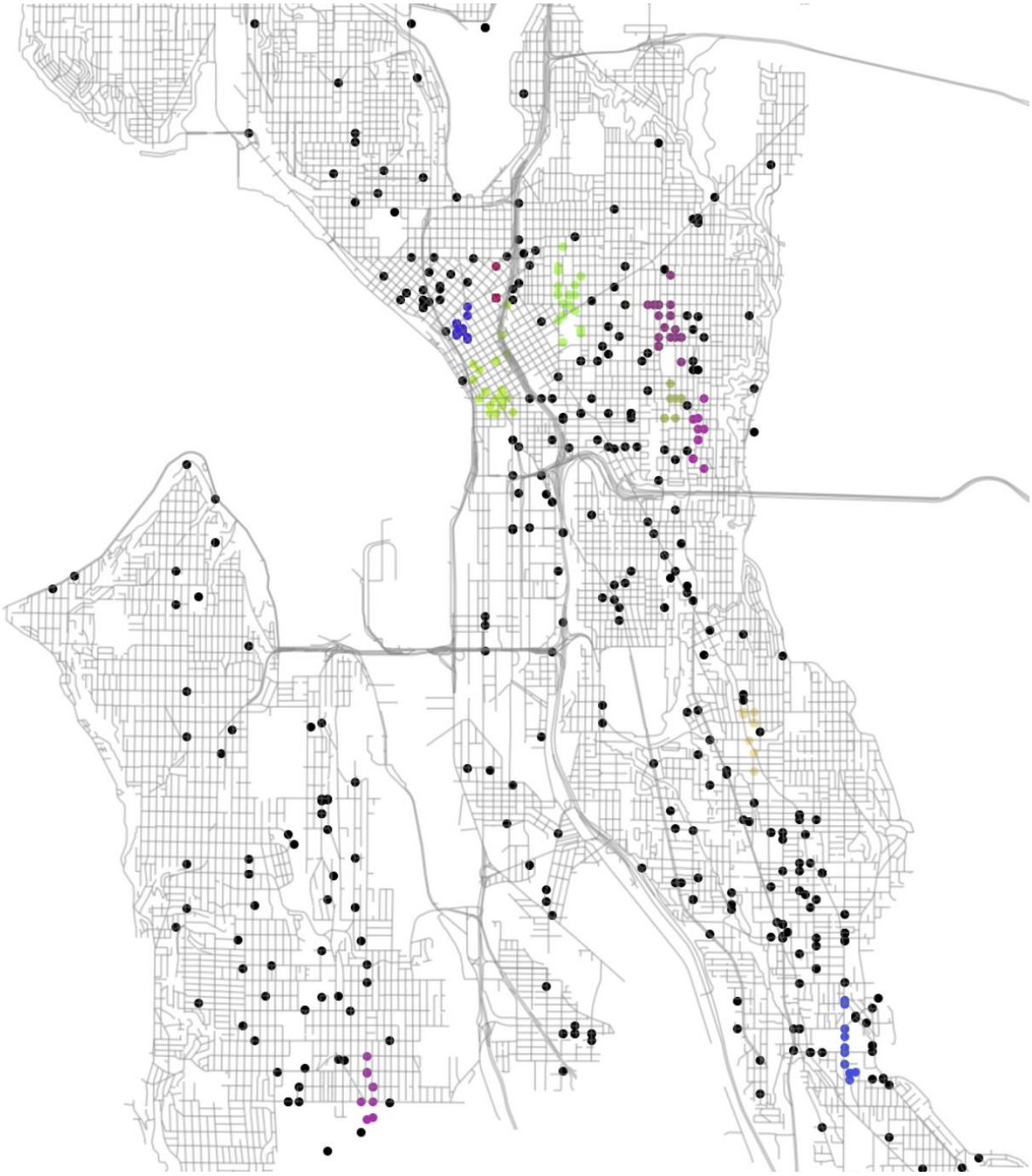
8 points with less than a 1% change in a row are now needed to qualify as a cluster. Now our map of Seattle shows the area you statistically wouldn't want to be at night:


11
12

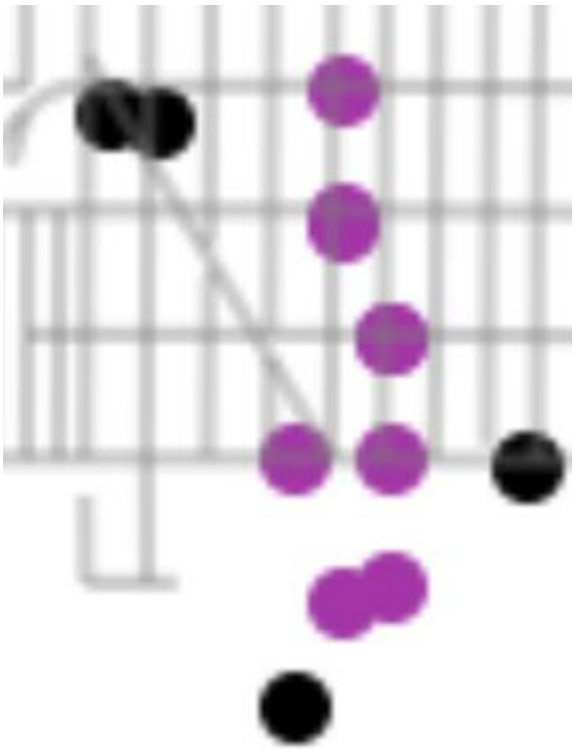
Also note the number of clusters has increased from 7 to 12:

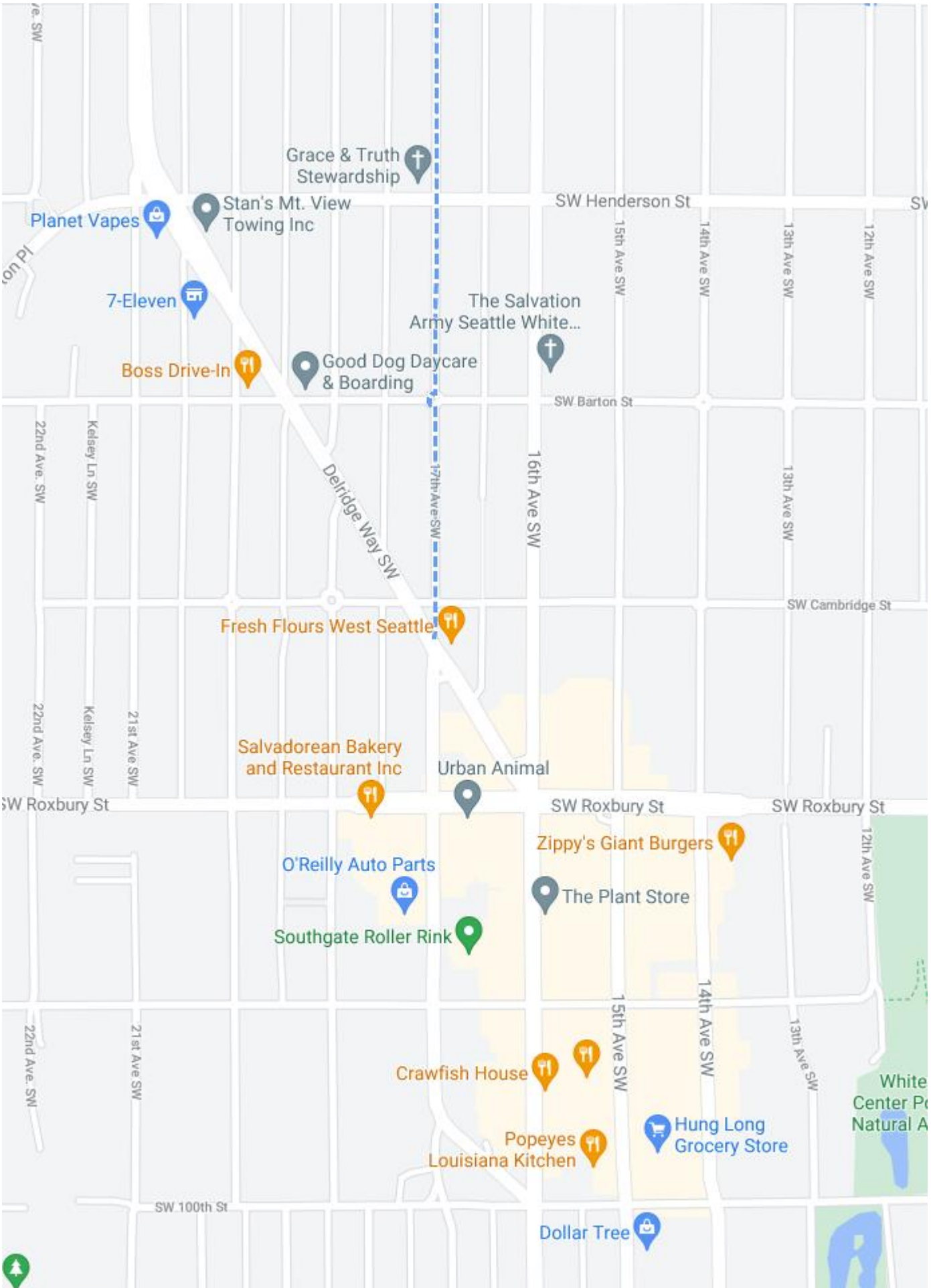


Closer view:



Zooming in on the south west purple cluster:




Grace & Truth Stewardship
Planet Vapes
Stan's Mt. View Towing Inc
SW Henderson St
15th Ave SW
14th Ave SW
13th Ave SW
12th Ave SW
7-Eleven
The Salvation Army Seattle White...
Boss Drive-In
Good Dog Daycare & Boarding
SW Barton St
22nd Ave. SW
Kelsey Ln SW
Delridge Way SW
17th Ave SW
16th Ave SW
13th Ave SW
SW Cambridge St
Fresh Flours West Seattle
22nd Ave. SW
Kelsey Ln SW
21st Ave SW
Salvadorean Bakery and Restaurant Inc
Urban Animal
SW Roxbury St
SW Roxbury St
SW Roxbury St
Zippy's Giant Burgers
12th Ave SW
O'Reilly Auto Parts
The Plant Store
Southgate Roller Rink
22nd Ave. SW
21st Ave SW
15th Ave SW
14th Ave SW
13th Ave SW
Crawfish House
Hung Long Grocery Store
Popeyes Louisiana Kitchen
SW 100th St
Dollar Tree

These are matched up. 16th ave SW seems to be one of our hotspots.

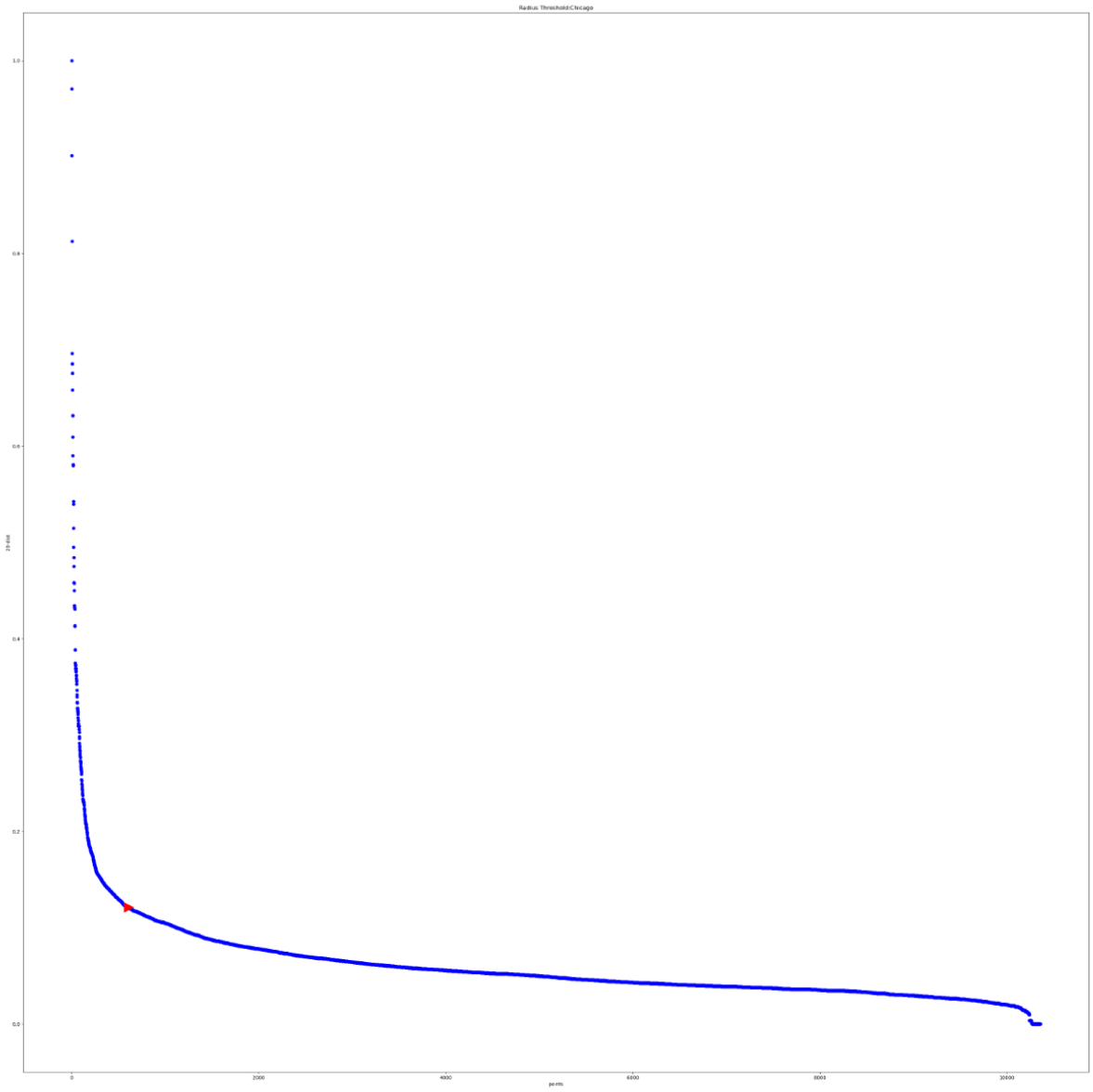
Another street would be this one in the south east:

This is Rainier Ave South, which my domain knowledge of Seattle supports as a dangerous place.

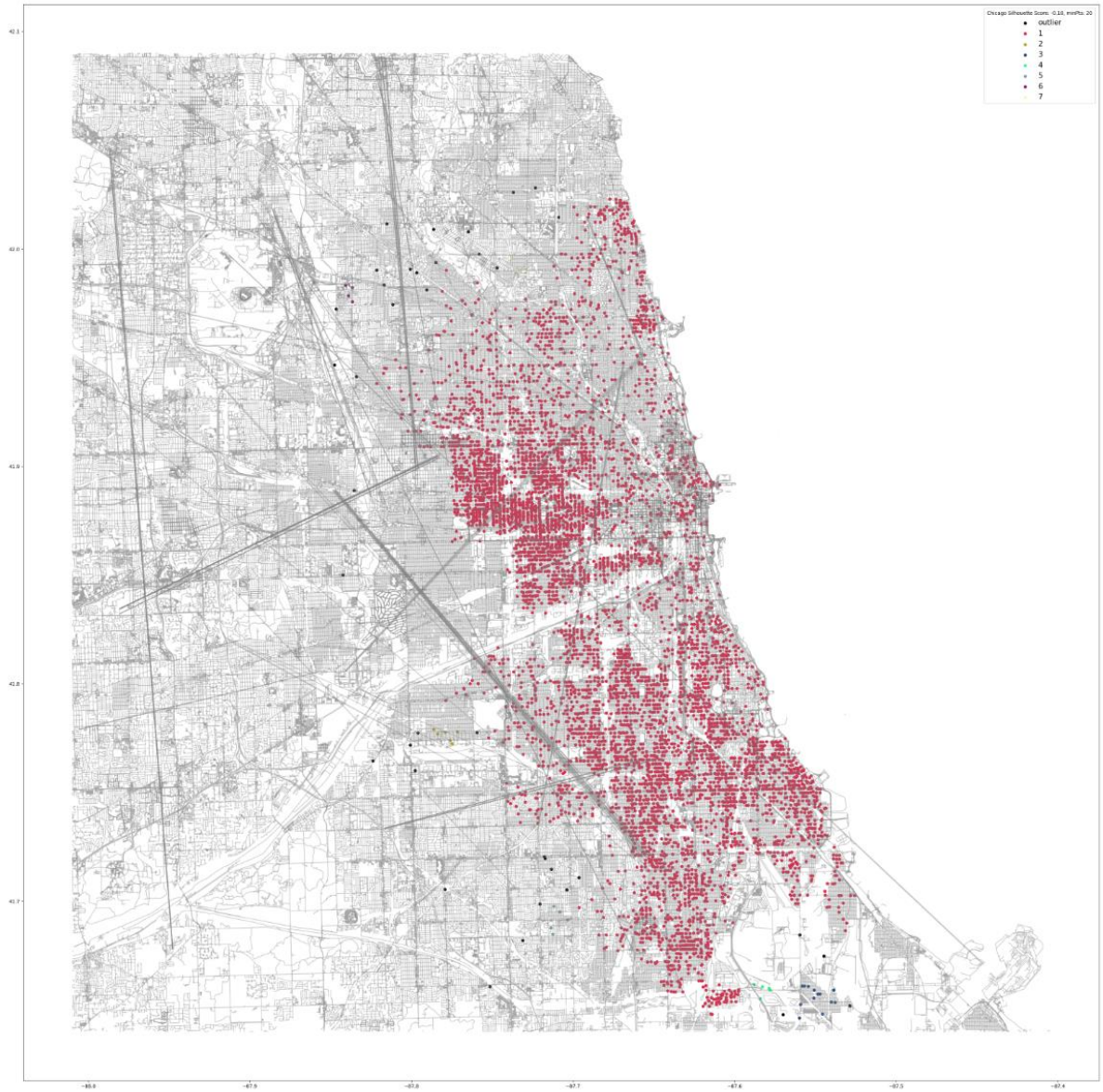
Then in proper Seattle:

Notably: Pioneer Square is the left most green, the middle green is just above the hospital region (pike and pine), and central Seattle on the right. This strongly aligns with my domain knowledge of high violent crime rates in Seattle.

Now let’s look at Chicago:

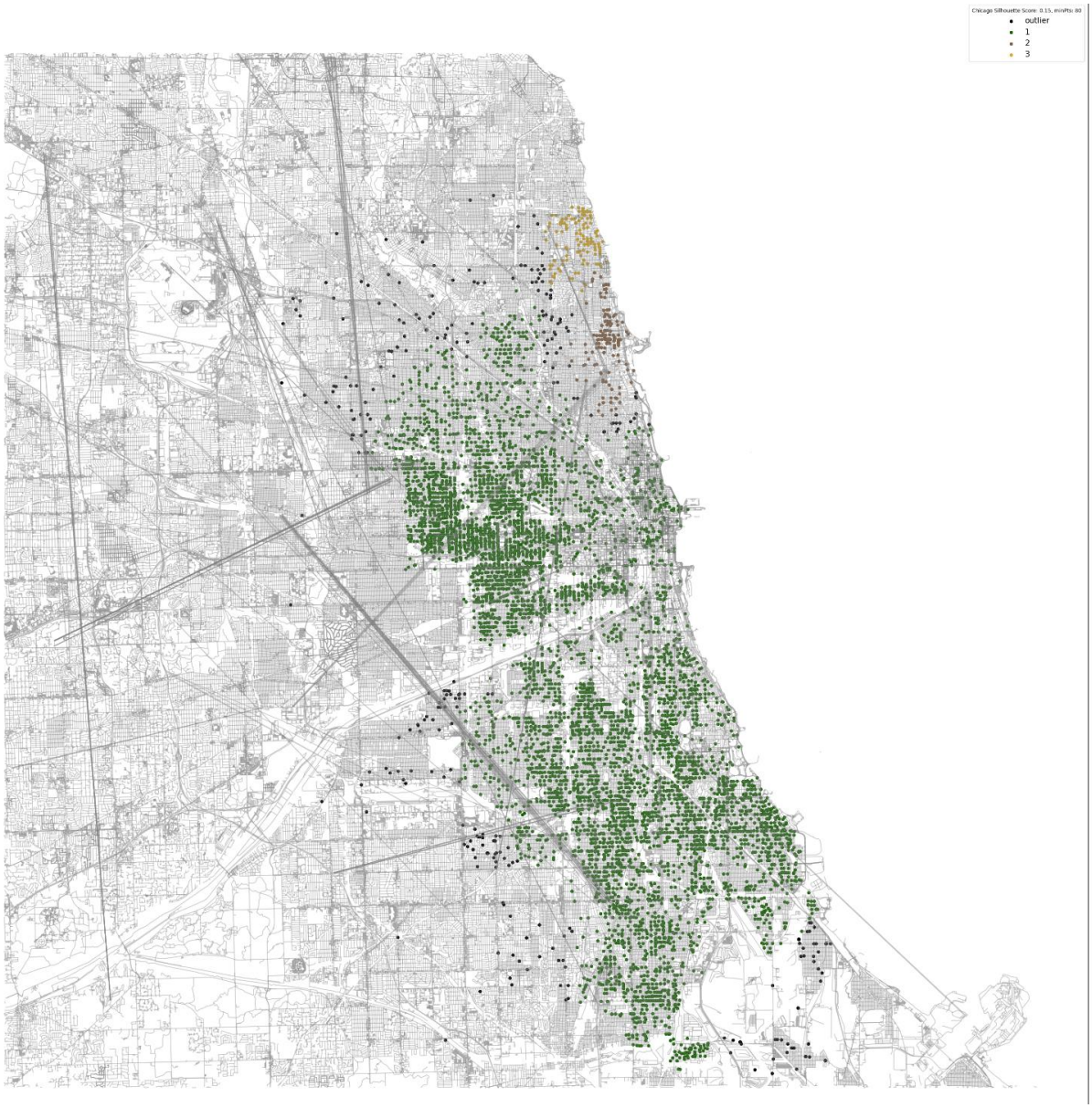


At 20 minPts the computer has a great spot for the threshold.

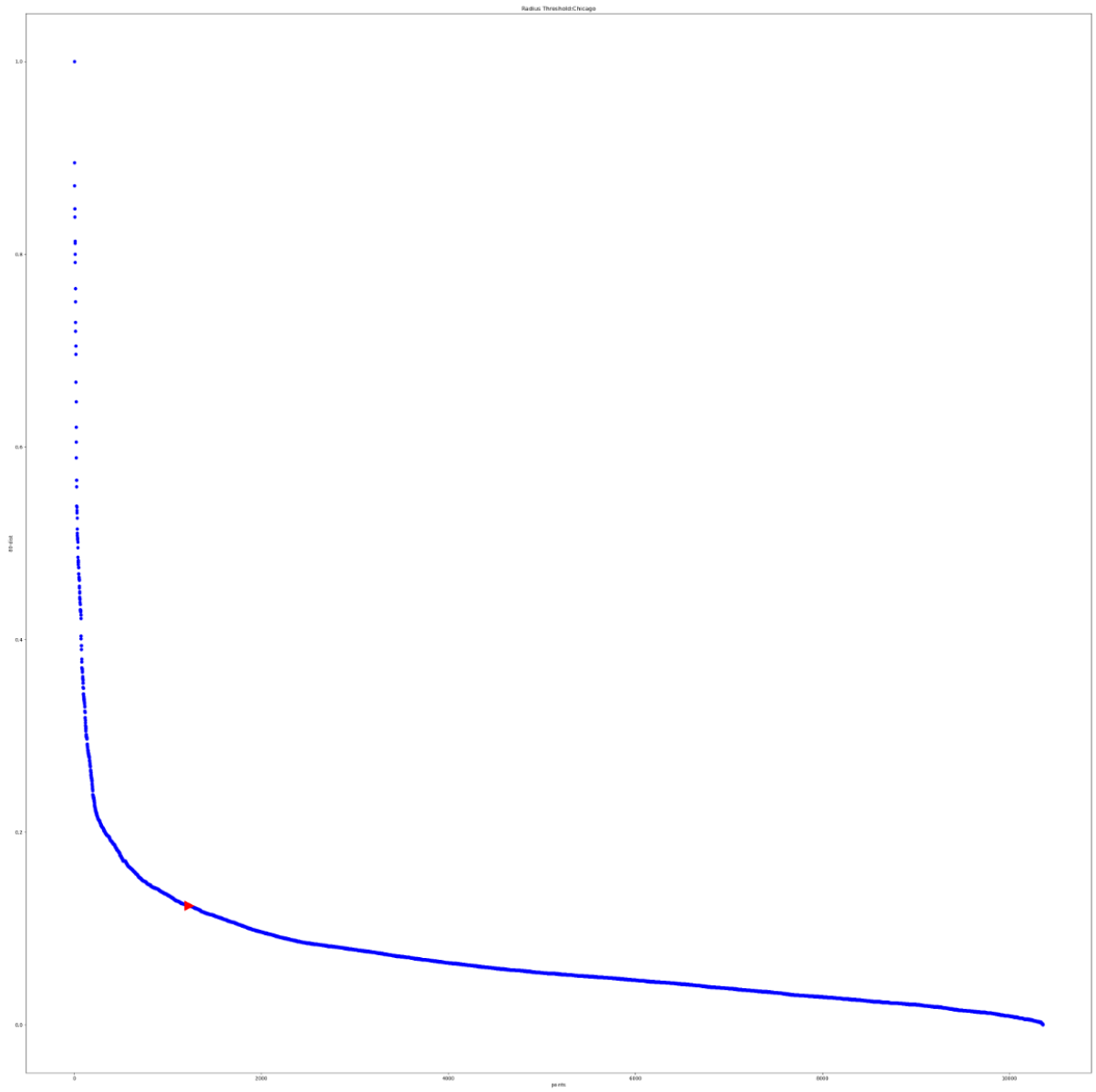


But here's the graph. One big cluster basically. Not that useful and with a silhouette score of -.18 its not a very good data set to be clustering.

Okay lets bump the minPts to 80:

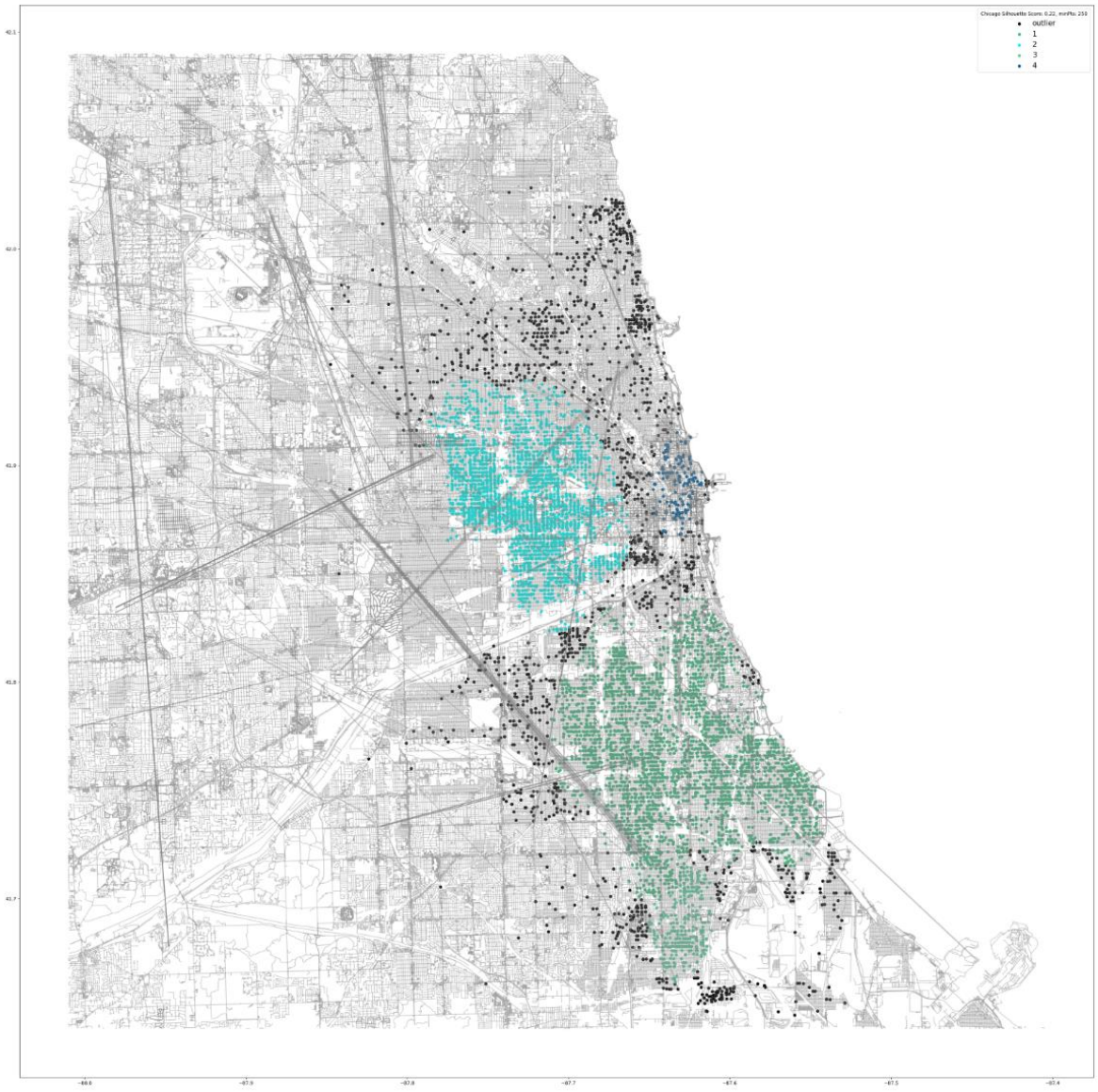


Still not very useful but we are getting some clusters and the silhouette score improved to 0.15. Here’s the threshold graph:

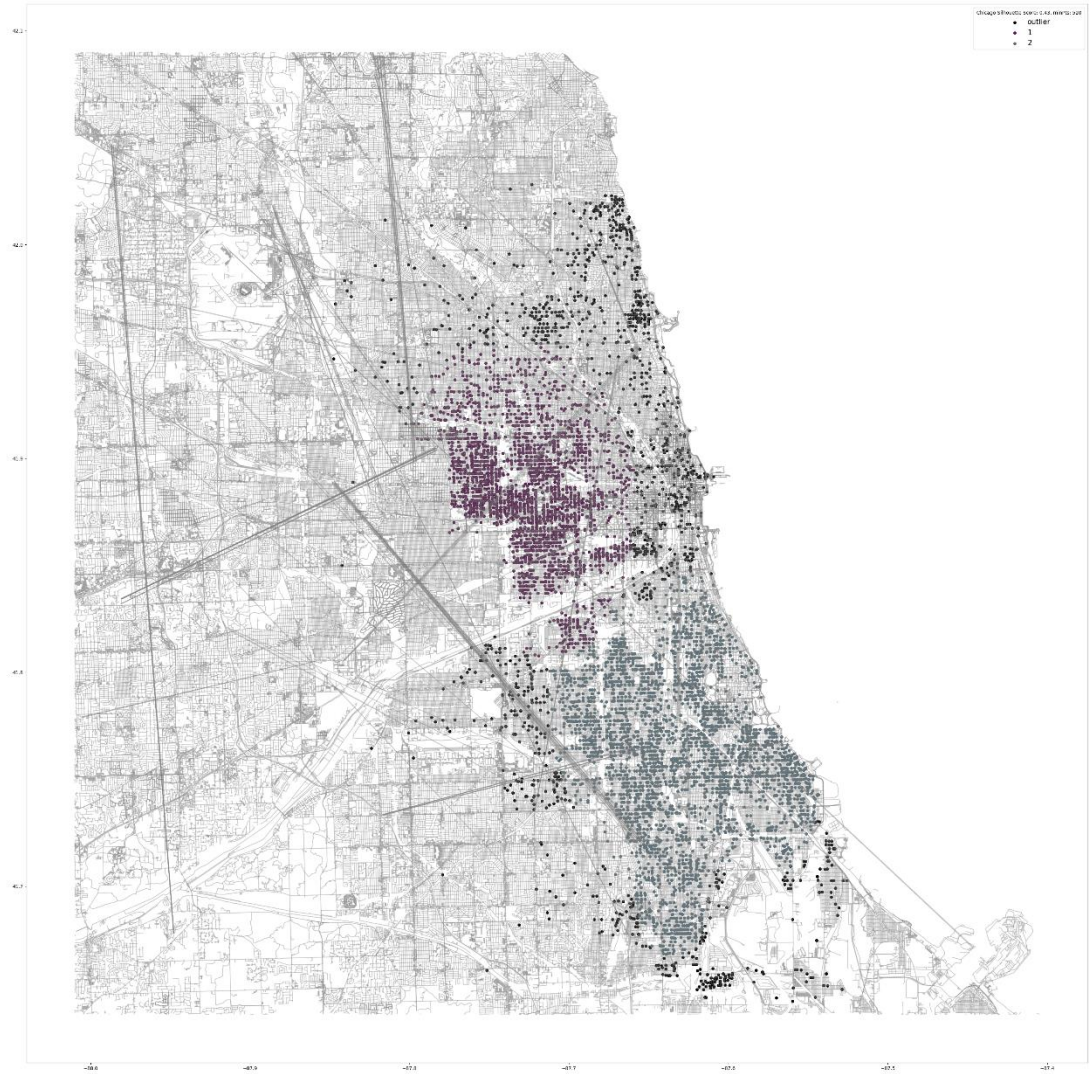


Still a long way to go to get high density. I would consider all of Chicago low density for gun violence.

Let’s try minPts = 250:



520:



This is right at the maximum density value I can get at my current derivative threshold. The silhouette score is an acceptable 0.43.

These results are quite alarming.

For the sake of scrolling I've only included these Seattle and Chicago. All the user of my program would have to do is pass in the name of a city (Unique or large just so geopy can find it; must be in data base; must have supporting shape file in project directory) and the density coefficient (minPts). Cities I've added the shape files to: Philadelphia, Miami, Brooklyn, Berkeley.

Lessons Learned

100% reliable data probably doesn't exist. Given these were police records and I know the locations were incorrect quite often I don't think there can be a real-world big data set that comes from human entry that can be trusted 100%.

Hindsight: Only use latitude and longitude coordinates not city name. I'm not sure how many data points may have been under a different name. Also this is why I did not do NYC (too many boroughs).

Hindsight #2: Do something to represent stacked data. I didn't think it would be so prevalent and even though its false it should show up in the results.

It can be hard to cluster real life data and even harder to evaluate it.

Hindsight: Automatic result running and subsequent graphing of the different density regions would make the maps easier to read and more impactful.

Start projects earlier: I really under estimated the amount of time this would take me. And there is still so much that I could continue to work on it with.

Always create new environments! I took me forever to get all the libraries installed because I had used pip on my root envo and so I would up uninstalling everything and that worked. When using conda especially.

Acknowledgements

Three papers formed the basis of the project:

1. https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. 1996.

Provided information on how to use DBSCAN

2. https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

Nadia Rahmah and Imas Sukaesih Sitanggang 2016 IOP Conf. Ser.: Earth Environ. Sci. 31 012012

Example of using the derivate of the k-dist graph to automatically find optimal epsilon (radius) value.

3. https://www.dbs.ifi.lmu.de/Publikationen/Papers/OPTICS.pdf

Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander. OPTICS: Ordering Points to Identify the Clustering Structure. 1999

Cluster analysis and evaluation.

Input Data

https://www.gunviolencearchive.org/

https://www.kaggle.com/jameslko/gun-violence-data

gun-violence-data_01-2013_03-2018.csv

Wolfram Schneider & Slaven Rezić. https://www.bbbike.org/

https://extract.bbbike.org/

This website allowed me to create my own shapefiles of anywhere on earth.

Python Libraries:

Geopandas, pandas, matplotlib, sklearn, numpy, geopy