1. Problem Definition:
   1. Goals and Objectives

Social Informatics is starting to become an area where Big Data and Data Science can have a big impact. One of the areas of Social Informatics that has the potential to impact almost anyone of us is crime prediction and prevention. Ever since the futuristic movie Minority Report came out many years ago, the idea of being to predict when and where crime will occur has been very intriguing. Although this project won’t be addressing all of the concepts introduced in Minority Report, it will explore the concept of being able to identify where crime is predicted to be most prevalent. Not only that, but a mechanism to identify the most dangerous and harmful crimes will be the catalyst to visualizing where highly intensive crime spots exist.

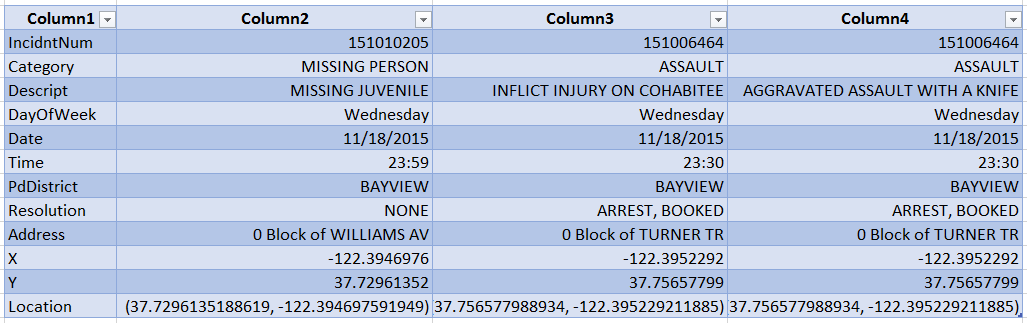
San Francisco is a city that is on the forefront of understanding how data can benefit the public at large. As such, it has made available an extensive amount of data regarding the city. Therefore, the San Francisco crime dataset for 2015 will be analyzed in order to identify the most dangerous crime areas within the city. Anyone that has visited San Francisco knows what a vibrant city it is. However, are there dangers lurking within the neighborhoods that can catch a tourist off guard? Worse yet, are there areas in the city where people are at a relatively high risk of being assaulted or harmed in an even more significant manner? I want to focus on crimes that can have a severe negative impact on the victims of the crime. Where are these types of crimes occurring? Are there certain areas that tourists should stay away from, or at least have knowledge that they should be on their guard?

Although San Francisco has made available a crime event dataset that goes even further back than the beginning of 2015, I have decided to focus on only the current year. Since I am not doing any trend analysis, the current year’s crime events should suffice. After all, it is the areas that are currently the most dangerous that should be of interest to people. This implies that in order to keep the results current, one would want to run the code created in this project every month with the latest crime data in order to capture any changes in violent crime hotspots.

* 1. Tasks to Complete Project Deliverables

Before this project was initiated, I had initially thought that I might have to enrich the data within the crime dataset. The crime dataset contains the following attributes: Incident Number, Category, Description, Day of Week, Date, Time Police Department District, Resolution, Address, Geocode X Coordinate, Geocode Y Coordinate, and Geocode Location. A sample of the dataset format and data is shown in Figure 1.

Figure 1. San Francisco 2015 Crime Dataset Format



In order to utilize only rows that were associated to the most dangerous crimes, I thought I would have to process the dataset to tag the rows that were associated with dangerous crimes. However, as I worked on the project, I discovered that Python has a very useful package called Pandas that allows one to create SQL statements against a file that is read into a Pandas dataframe. This allows the powerful capabilities of SQL logic to be utilized to group, filter, order and reformat data as needed. Therefore, a data transformation process was not necessary for this dataset. However, I did need to profile the dataset to gain an understanding of the different categories and category descriptions that were possible. This knowledge would be utilized later on to help build the Pandas SQL statement.

Another step that I had to go through was determine how I would identify the most dangerous crime clusters. My initial idea was to utilize the Scipy K-Means function in order to identify the K-Means centroids. In order to test my theory, I created a test Python program that would read in a subset of the data and plot the information on a graph to determine visually if the K-Means was the best function to reach my project goal.

Unfortunately, the initial results (see Figure 2) were not very promising. It became immediately obvious to me that utilizing the K-Means function would give me less than desirable results. As can be seen in Figure 2, the centroids are dispersed throughout the graph. The x and y axis represent the x and y values of the Geo-code for each crime incident. Since this was just a sample to determine the methodology that I would utilize, I did not bother to filter the crime incidents by the ones that were the most dangerous. Even so, it was clear that I needed a different solution for my problem. Just to verify, I added the data to a map plot in order to see the results on a map of San Francisco (see Figure 3). This too indicated that I needed to find a better solution. The main issue is that there are centroids created for even low density crime areas. For example, I would not want centroids for the low density red areas in Figure 2. Another example is that I would not want the Golden Gate Park low crime density are in Figure 3 to have a centroid.

Figure 2. Subset of Data Plotted Utilizing K-Means Function

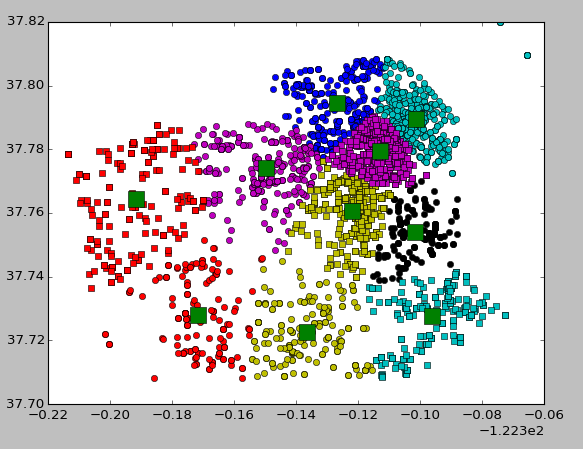
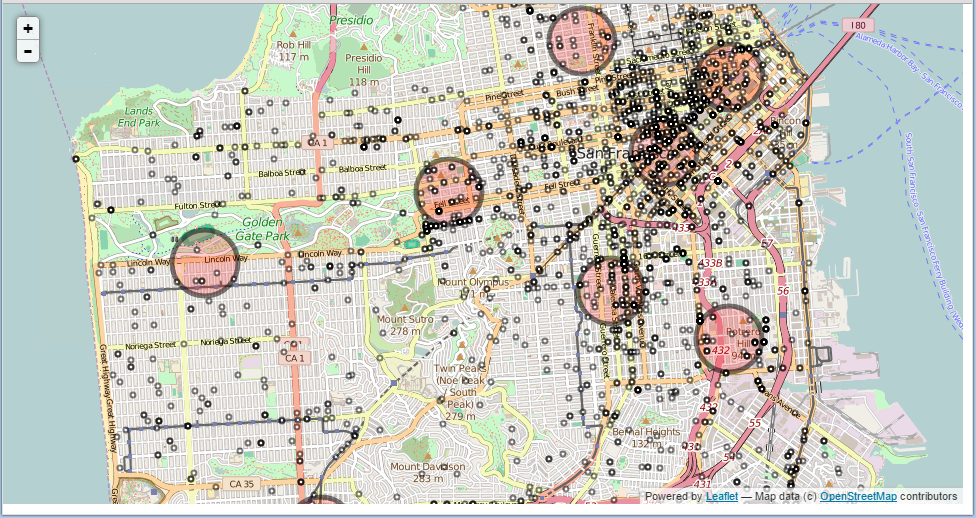


Figure 3. Map Plot of Data Subset Utilizing K-Means Function



1. Software Definition:
   1. Techniques Utilized

Based on my initial exploratory results, it appeared that I would need to utilize a different technique in order to attain my goal of visually identifying the most dangerous crime regions in San Francisco. I might have chosen the police department district, but I thought that this geographic region would be too broad. Since the Geo-code was available for each crime event, I could also have converted the Geo-code to a zip code, and then identify the most dangerous zip code. Once again though, I felt that this geographic area would be too broad and inflexible. Therefore, what would be the best solution given the type of data that I had, and the precision goals that I wanted to achieve?

I finally realized that the data itself presented me with a good solution. After all, the precision of the Geocode went to seven places for the x and y values. That precision was needed to identify the location to the address and/or block level. However, I did not want that level of precision. Therefore, I decided that rounding a group of Geo-code coordinates to the right level would give the right level of data grouping. I could round the values to the precision that would make the most sense. Besides that, the level of precision could be modified quickly without requiring extensive code changes.

As far as the mechanism to visualize the results, based on my initial experimentation with the sample data, I decided to utilize a map instead of any type of graph. To me, it made the most sense to plot the most dangerous areas on a map, since that would be most useful to anyone that wanted to take actions based on the results produced by this project. The highlighted areas would show to a tourist, for example, which areas have the highest concentration of dangerous crimes.

Comparing Figure 2 with Figure 3, one immediately gets the sense that the Figure 3 map provides richer content and usability when compared to the graph in Figure 2. Additionally, the fact that all of the crime incidents were tagged with the x and y Geo-codes makes the plotting of this information on a map very convenient. I discovered the open source Folium map application services which seemed to worked well when it was passed specific Geo-codes to plot. As a bonus, the results are given in an easy to use HTML output page which allows flexibility to zoom in and out as necessary. This would be a big benefit for anyone that wants to view the results at a street level.

* 1. Technical Aspects

In order to test my idea, I switched from using an advanced K-means technique, to a rather simple mathematical feature of rounding numbers. Not very exciting, but I believed it would give me the results I needed. Since I had discovered the elegant Pandas SQL feature, I was determine to leverage that to create the query that would do all of the mathematical and set functionality that was needed to generate the right information. I utilized my knowledge of SQL fully in order to accomplish this.

Let’s examine the SQL line by line:

This highlighted line shows the actual rounding of the Geo-code x and y values, along with the group by count (needed to determine density of the area).

select round(X,2) X, round(Y,2) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

group by round(X,2) || round(Y,2)

having count(\*) > 350

order by count(\*) desc;

This highlighted line shows the filtering of the crimes that will be designated as the most dangerous crimes in San Francisco. What is interesting is that everybody may have their own definition of what defines a dangerous crime. I chose crime categories that I believe have a high probability of a crime victim experience physical harm. To get an even more precise definition of dangerous crime, one could also utilize the description field as a filter. I chose not to do this for this exercise.

select round(X,2) X, round(Y,2) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

group by round(X,2) || round(Y,2)

having count(\*) > 350

order by count(\*) desc;

This highlighted line shows the SQL group by clause that allows us to group the incident counts by the desired Geo-code x and y area level.

select round(X,2) X, round(Y,2) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

group by round(X,2) || round(Y,2)

having count(\*) > 350

order by count(\*) desc;

This highlighted line is a filtering of the crime incident count for the x and y Geo-code grouping. It was somewhat difficult to decide where to break off the count, since it is somewhat subjective to say that one density is dense enough to highlight, whereas another density that is a little smaller should not be highlighted.

select round(X,2) X, round(Y,2) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

group by round(X,2) || round(Y,2)

having count(\*) > 350

order by count(\*) desc;

In the end, I decided to use a number that would have enough high crime areas to highlight without making the plots too busy, such that the meaning would be lost. In other words, the highlighted crimes are in the context of San Francisco, as opposed to a universal standard on what is and is not considered a dangerous crime area. Also, the areas identified are more akin to a Top-X categorization, as opposed to a more disciplined scientific categorization. The actual count number utilized (in this case 350), was arrived by trial and error to arrive at a cutoff point that was visually significant to highlight areas that had relatively more concentrations of dangerous crimes than other areas in San Francisco.

This highlighted line orders the incident counts so that the densest crime areas would show up first:

select round(X,2) X, round(Y,2) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

group by round(X,2) || round(Y,2)

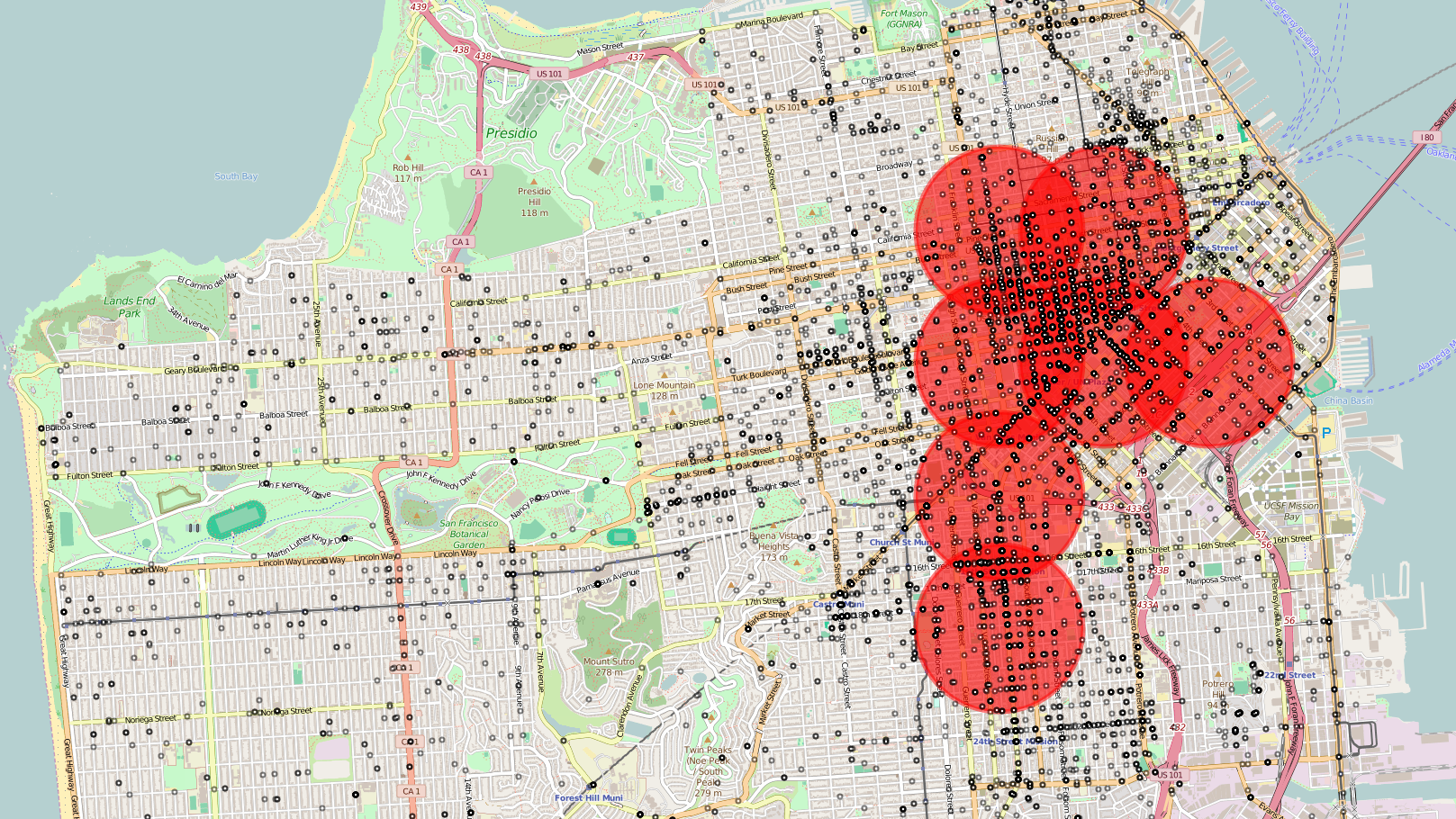
having count(\*) > 350

order by count(\*) desc;

1. Application of Analytical Techniques and Results:

The following figure shows the results of running the SQL as formatted above. As can be seen, the rounding of the Geo-code captures the areas with the highest density of dangerous crimes. I chose to represent the areas as circles, even though rectangles may have produced more accuracy in the visualization mapping to the actual data. Because of the fact that circles tend to cut off data at the corners, I had to make the circles larger in order to cover all of the incidents that would fall in the area. Since crime areas do not really have artificial geometric shapes that define the local crime incidents, choosing circle over other geometric shapes should suffice in this situation. As a method to verify accuracy, I included the individual incidents to visually compare with the actual results.

Figure 4. Map Plot of Dangerous Crime Areas with Geo-Code Rounded to 2 Decimal Places



The following next figure shows the results of running the SQL as formatted below.

select round(X,3) X, round(Y,3) Y, count(\*) from crimedata

where Category in ("ASSAULT","SEX OFFENSES FORCIBLE","KIDNAPPING")

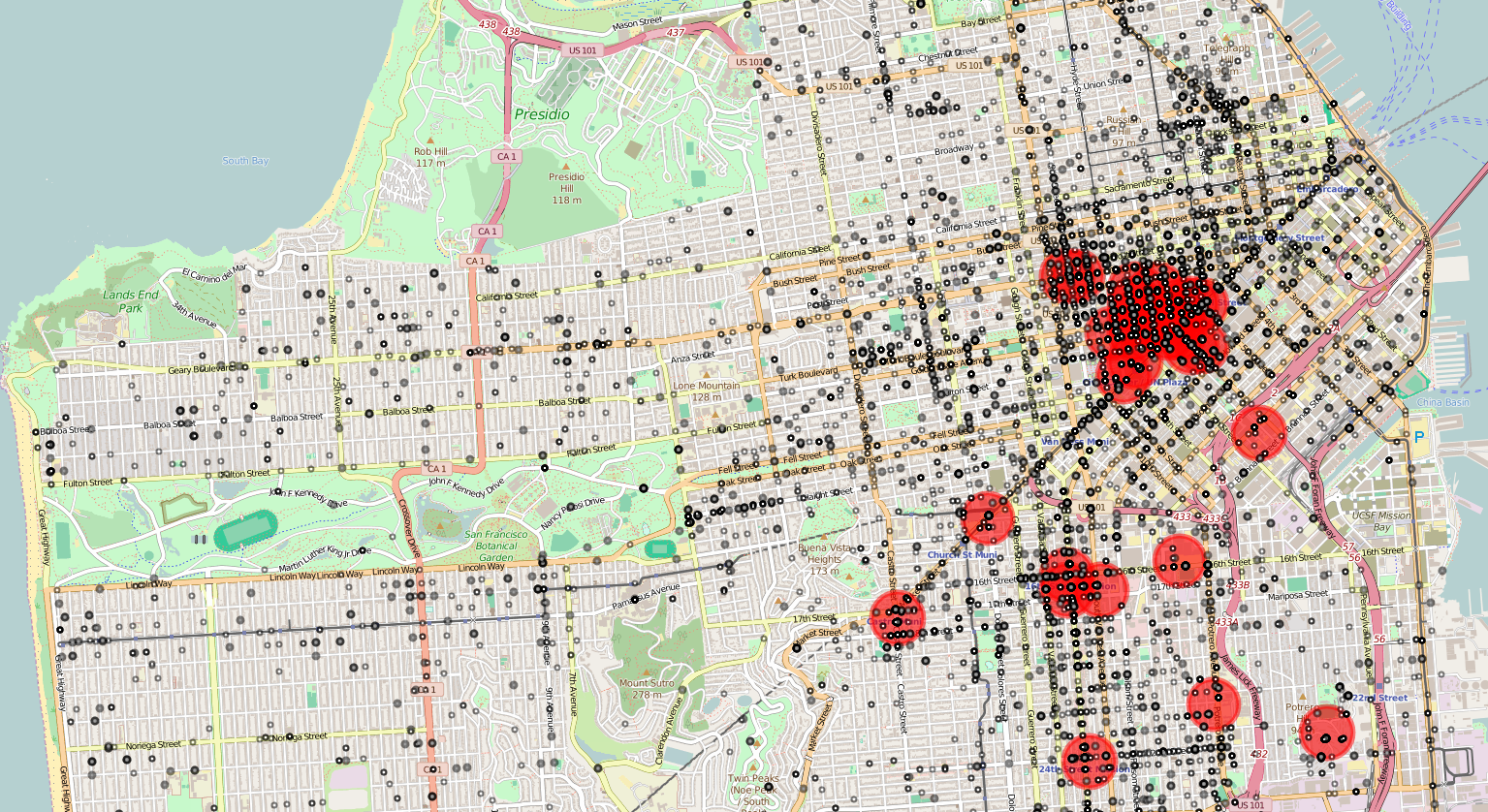
group by round(X,3) || round(Y,3)

having count(\*) > 28

order by count(\*) desc;

As can be seen, the more detailed rounding of the Geo-code captures the areas with the highest density of dangerous crimes in a more precise manner. All of a sudden, areas that were not identified in the previous SQL that utilize 2 decimal place rounding start appearing on the map. Notice that since the Geo-code area selected was more precise, I had to select a lower count number. Not only that, I also selected a number that would result in more areas being identified, since they were smaller than the previous implementation of the SQL code. The most dangerous crime areas are still in the same general vicinity, but there are now gaps between the areas, and the areas are more spread out. As far as determining the cut-off point for the grouping count, it was even more difficult than when three decimal places were utilized, since the number of incidents per grouping was much smaller. I arbitrarily cut the number at 28, but it is somewhat difficult to justify why I chose that number versus a different number (either higher or lower). As a method to verify accuracy, I included the individual dangerous crime incidents to visually compare with the actual results. One thing to note is that even though some of the dangerous areas may seem to have a lot less detail events than others, the same address or Geo-code coordinate may be utilized for different crime events. Therefore, any overlaps would make it appear that there are not as many crime events as there really are.

Figure 5. Map Plot of Dangerous Crime Areas with Geo-Code Rounded to 3 Decimal Places



So which of the two methods produced the most accurate and useful results? I selected the first process as my final results for the project. This is because there were no gaps between the most dangerous areas. This produced an aggregate view of a general area in San Francisco which is relatively very dangerous. The second process has gaps between the dangerous areas identified on the map, which might lead to someone having a false sense of security if they are not in the specific red zones identified. Based on the data observed visually, I could not justify the usage of the second map since the data seemed to dispersed in an arbitrary manner.

Even though I decided to utilize a grouping approach as opposed to utilizing a statistical method such as K-means, I decided to identify a method that might be used to perhaps improve upon the approach that was utilized in this project. One of the promising Python function that I came across was the DBSCAN function that takes a more Data Scientific approach to solving the problem introduced in this project. Even though my expertise is limited when it comes to the DBSCAN function, I decided to take an example of how to use DBSCAN and modify it to include the input data from this project. After adjusting the function parameters for DBSCAN through trial and error, I was able to produce the map displayed in Figure 6. DBSCAN takes a more elegant approach by combining the concepts introduced in this project (K-means and Cluster Density) in order to identify K-means clusters that have significant density. The other points are identified as outliers, since they are not concentrated enough near the cluster. The other benefit of using the DBSCAN function is that a sample of the data is selected. This means that for other applications that utilize large amounts of data, the processing time will be shortened because of the sampling methodology that is utilized.

Another nice feature of DBSCAN is that it allows the points farther away from the main concentration of points to be identified. Notice how larger circles were utilized to identify the area with the most dense crime incidents, both in Figure 6 and Figure 7. The first figure utilized the map plot that was used in the previous results. Additionally, a graph plot was produced for Figure 7 in order to more clearly see the cluster points versus the outliers (these points are non-red in color). Even though DBSCAN appears to give better results than other methods employed in the project, a more thorough understanding of how the function works is necessary in order to get results that are valid and reliable. I definitely plan on exploring this function in future projects, since it seems to be a very powerful tool for Data Scientists.

Figure 6. Map Plot of Dangerous Crime Areas Utilizing the DBScan Function

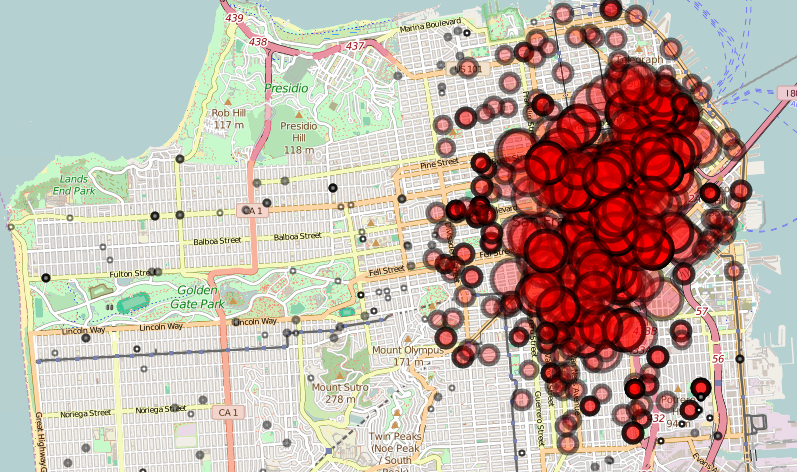
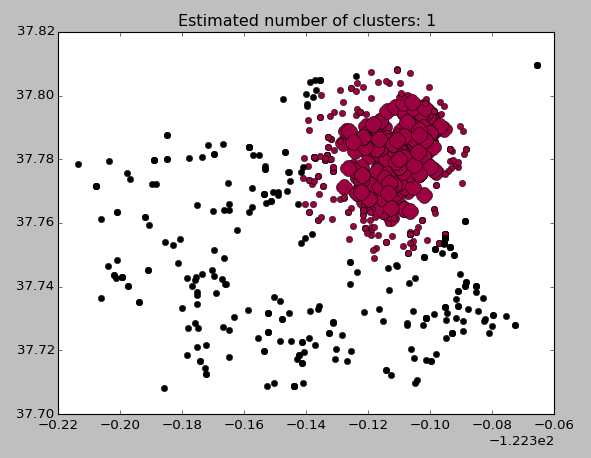


Figure 7. Graph of Dangerous Crime Areas Utilizing the DBScan Function



1. Reproducibility:

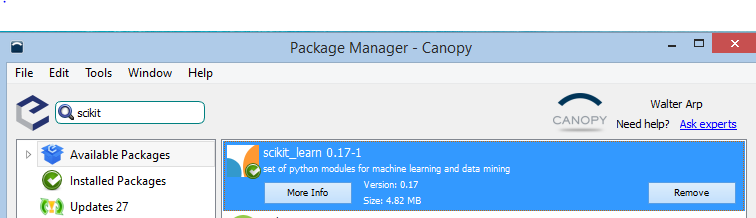
The various Python programs utilized in this project were run several time to verify that consistent results were obtained. One note is that running the K-means program produced different symbols and colors for the graph results, but the actual results of the data points and centroid were consistent between runs.

* 1. Environment, Installation and Datasets

The environment utilized was Canopy for Python. The only non-standard package utilized was Folium. The following installation script is necessary to install Folium into the Canopy environment.

* + 1. $pip install folium

The following package is necessary to utilize DBSCAN in the modified example module.

* + 1. Go to Canopy Package Manager and add scikit\_learn 0.17-1

The following dataset was utilized from the San Francisco Open Data website (note that if the actual dataset on the website is downloaded, it will be different than the dataset utilized, since the dataset on the website is constantly being updated). Therefore, the input file provided in GitHub should be utilized. The input file should be placed in the same directory as where the Python program is run:

SFPDIncidents2015.csv

This data was obtained at the following location (data changes daily):

https://data.sfgov.org/Public-Safety/SFPD-Incidents-Current-Year-2015-/ritf-b9ki

File Manifest:

Project Write-up:   
Project Walt Arp v1d.docx

Input File For all Software Runs:  
SFPDIncidents2015.csv

Exploratory code to determine methodology:   
Project Crime Kmeans.py

Exploratory code HTML map result:   
TestKmeanCrimeMap.html

Geo-Code Rounded to 3 decimal places:   
Project Crime Roundup3places v1.py

3 decimal places HTML map result:   
SFCrimeMapRound3Places.html

Geo-Code Rounded to 2 decimal places:   
Project Crime Roundup2places v1.py

2 decimal places HTML map result:   
SFCrimeMapRound2Places.html

Modified DBScan Sample Code:   
Project Crime DBScan.py

DBScan HTML map result:   
DBSCANCrimeMaps.html

**References:**

Public Safety - SF OpenData. (n.d.). Retrieved November 20, 2015, from https://data.sfgov.org/data?category=Public Safety

Folium: Python Data. Leaflet.js Maps. (n.d.). Retrieved November 20, 2015, from http://folium.readthedocs.org/en/latest/

Comparison with SQL. (n.d.). Retrieved November 27, 2015, from http://pandas.pydata.org/pandas-docs/stable/comparison\_with\_sql.html

Density-Based Clustering. (2015, September 9). Retrieved December 2, 2015, from http://blog.dominodatalab.com/topology-and-density-based-clustering/

Sklearn Cluster DBSCAN. (n.d.). Retrieved December 2, 2015, from http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Demo of DBSCAN clustering algorithm. (n.d.). Retrieved December 3, 2015, from http://scikit-learn.org/stable/auto\_examples/cluster/plot\_dbscan.html#example-cluster-plot-dbscan-py