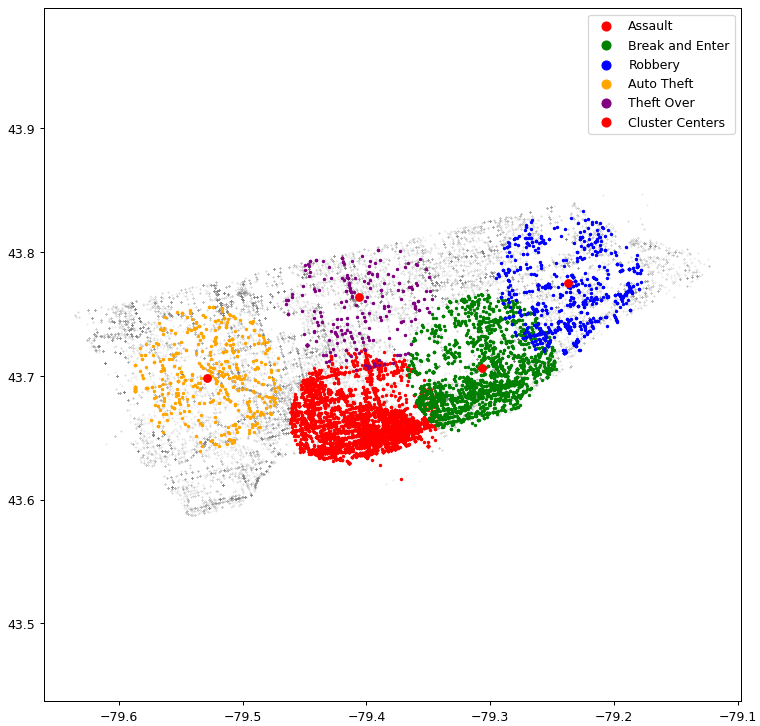
Crime Hotspot Prediction

Max Haviv



# Table Of Contents

[**Table Of Contents 2**](#_eb42a65p709j)

[**Project Introduction and Overview 3**](#_fxnlsvoyk6lc)

[**Literature Review 4**](#_mzs8q5tflzpq)

[**Methodologies 5**](#_lep23wxp54nf)

[K-Nearest-Neighbor 5](#_w5ks8y806fq6)

[K-Means Clustering 6](#_l44siwvki5bn)

[Density Based Clustering 7](#_mm09bcd7c4en)

[**Results 8**](#_r8ccgxlybl41)

[KMeans Clustering Classified by KNN 8](#_7w6zw5z22bsf)

[Density Based Clustering 12](#_1juahpze6rka)

[**Conclusion 14**](#_8a918t5ii5sf)

[**References 16**](#_6oec1md68spm)

[**Appendix 17**](#_ubvaf5v2ojms)

# Project Introduction and Overview

This project uses KMeans clustering and a modified KNN model to predict hotspots of crime given a dataset of crime. The dataset that is being used is the Toronto MCI from 2014 to 2019. The dataset has around 180,000 crimes reported with 5 different types of crimes being Break and Enter, Assault, Auto Theft, Robbery, and Theft Over. Theft over isn't very well defined but it is any theft that exceeds a certain amount of money which was not described in the dataset. Each crime also has many features ranging from the longitude and latitude where the crime was committed and the dates and the environment in which the crime occured.

For the clustering portion of the project a KMeans and density based model were created and fitted with five clusters on every crime in the dataset. Starting with the KMeans model first, after fitting the clusters a modified KNN was used to label each cluster with one of the crimes in the MCI. After all the clusters were successful labels with a crime then all of the crimes are plotted alongside with the clusters on a matplotlib plot. Then a radius is defined and each cluster within the defined radius will plot the crimes that were labeled with a color that is associated with the crime. This gives a visualization of Toronto as a whole and where the hotspots for crime are and what they are.

Lastly for the Density based clustering model. The model was fitted based on a random sample of 13,000 of the crimes for the dataset. This was done as the density based model is very ram intensive, and going over 13,000 samples would crash google colab. With this bump in the road, a decent model was still able to be formed. The model was forced to only create five clusters and each cluster would be classified by summing up each crime that was committed in the cluster and choosing the crime that was committed the most.

# Literature Review

This project uses a K-nearest-neighbor model and a KMeans clustering model as well as multiple libraries like numpy, pandas, and matplotlib. The KNN model is a model that was learned in class but modified to never reuse a label. The KNN looks at the closest K neighbors and checks to see if the label was already predicted or not. If it has not then the closest neighbor is predicted, if it has been used the next closest neighbor is looked at and predicted. This continues until a neighbor is predicted.

The Next model is a KMeans clustering model that was learned from Turner Luke. Luke published a paper titled “Create a K-Means Clustering Algorithm from Scratch in Python '' and this article goes in detail of how a KMeans clustering algorithm works and how to code one using python. This article came in great use from implementing the KMeans model for the project.

The final model is the Density based model. Which clusters are based on a set of core points and its neighbors. Core points are set based on the number of samples within its radius. The number of samples and the radius is user defined when initializing the model. If a point satisfies the minimum number of samples within its radius it becomes a core point. And all the other points around it become neighboring points. Then the model goes through all neighboring points and checks to see if any of them satisfy the rules to become a core point, if they do they become a core point. This continues on until there are no longer any new core points and there are no neighbors that could become core points.

The libraries like pandas and numpy are used for preparing the data for models and math inside of the models. For example the pandas function factorize was used to factorize all of the data inside of the dataset so they can be used in the models easily.

# Methodologies

## K-Nearest-Neighbor

The KNN model is a simple model just like the one taught in class except for one difference; and that difference is that the KNN will not predict one of the same crimes twice. So if the KNN predicts a crime that has already been predicted it will then go to the next closest neighbor to predict.

This is achieved by keeping track of a list of labels that are removed as they are predicted. If the KNN predicts a label that is not inside of the list of available labels it will try to predict the next nearest neighbor. It will keep doing this until it predicts a label that is in the list of available labels. In the case that there are no labels left it will return a -1 meaning that no neighbor was found. This is avoided completely though by making sure that the model looks at enough neighbors that every label is represented and that the model does look past five iterations since there are only five labels to predict.

The rest of the KNN is just a simple algorithm just like the one that was taught in class. The model is first fitted with every crime in the dataset. The features are the Longitude and Latitude of where the crimes were committed and the target value is the actual crime that was committed at that location. The model uses the euclidean distance between each point for the evaluation of which points are closest. Then the distance from the point that is trying to be predicted is calculated against the k number nearest neighbors.

## K-Means Clustering

K-Means clustering is a version of clustering where k number of clusters are defined and plotted randomly and points in the dataset are assigned to their closest cluster. The clusters are then adjusted and shifted based on the mean of all the points belonging to it. This is repeated until the clusters no longer move.

The algorithm in this project is slightly different though, but only slightly. Instead of setting all initial clusters randomly, only one is set randomly and the rest are set based on calculated probabilities proportional to their distances.

Turner Luke explained in his article “Create a K-Means Clustering Algorithm from Scratch in Python '' to use a different version of K-Means clustering called K-Means++. In this version of K-Means clustering the first cluster is initialized in a random location. The rest of the clusters instead of being set completely randomly. They are now set based on a calculated value of the sum of all the distances between each data point and their associated cluster centers. Once that is calculated you select a new center with a probability proportional to the calculated value. And you then continue repeating these steps until all of the clusters are plotted.

The reason for this change is that Luke found that when initializing all of the cluster centers in completely random locations that if a center was plotted too far from any points that it would never move. He also noted that if the cluster centers are too close to each other then they would be unlikely to separate from each other. K-Means++ solves both of these problems since it ensures that the cluster centers will not be initialized too far or too close to each other.

## Density Based Clustering

This model is fairly simple and it works by finding initial core points. These points are found by finding all points that satisfy the rules for being a core point. This is done by having a user set radius and minimum number of samples. This radius sets how close each sample must be to the core point, and the minimum sample means that the minimum number of samples set must be found in the radius of the point. If the correct number of samples are found within the radius, that point is then labeled a core point. This process continues with all the neighbors of core points until there are no longer any core points to be made, and no more neighbors to look at.

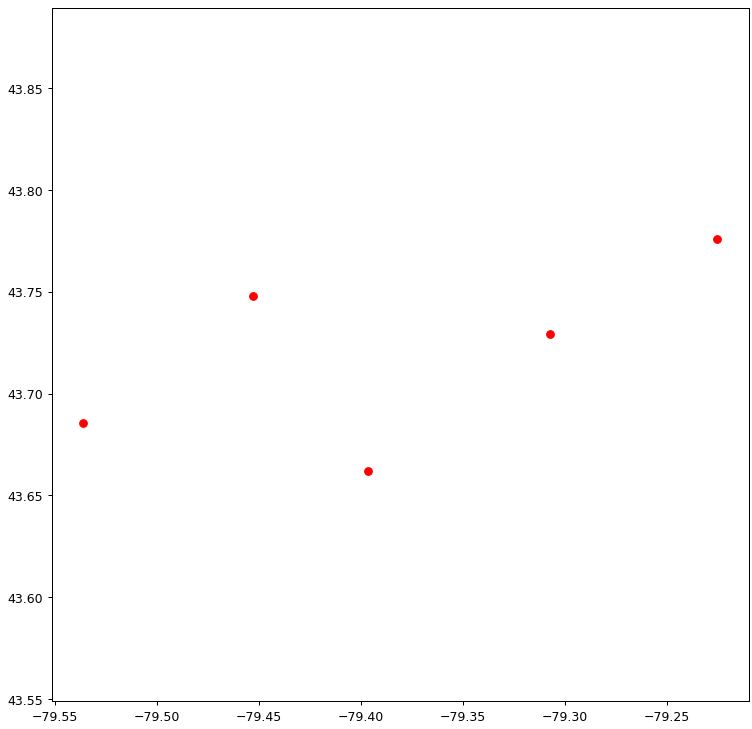
This process is unfortunately very RAM intensive and requires that the training data be cut down. And this was a significant cut. The number of samples was cut down from around 180,000 to 13,000. More samples and there would be a good chance that the program would crash. But continuing on the model would be fit with these 13.000 samples with three features instead of the two that the KMeans model would use. These would be the Longitude, Latitude, and the MCI. Adding the MCI added another layer that would help classify each cluster easier and better than the KNN. After the model was fit with these 13,000 samples and five clusters were made, each cluster would sum up their MCI’s and the crime that was committed the most would classify the cluster.

# 

# Results

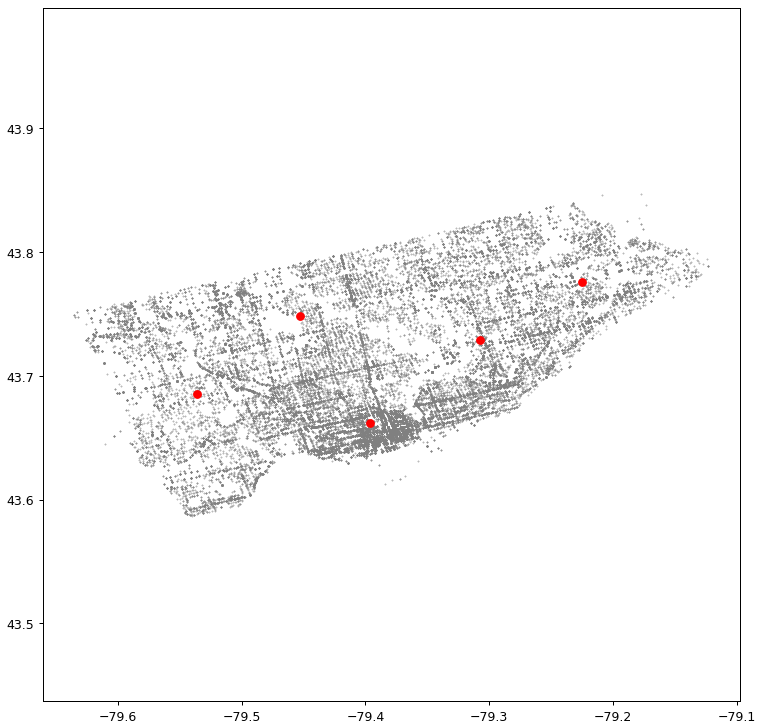
## KMeans Clustering Classified by KNN

This portion of the project predicts hotspots based on all crimes reported in Toronto from 2014-2019. These clusters have a slight deviation but after many test runs remain in the same general areas. Let ,,,, be cluster points in the City of Ontario. Each cluster point would reside roughly in the points,



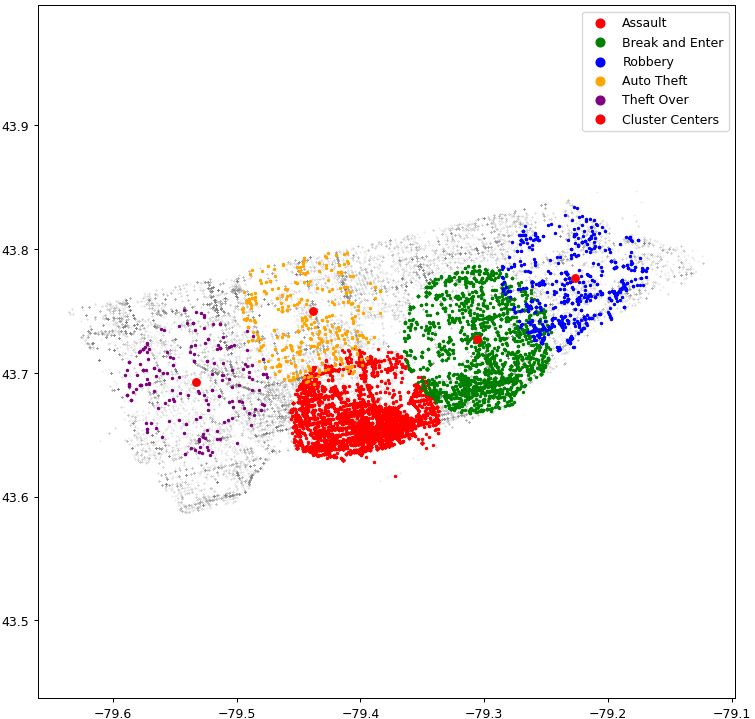
*Figure 2. Plotted Cluster Centers.*

Overlaying these points with every crime that is in the dataset, gives us a visual of where these cluster points are in relation to the city of Toronto.



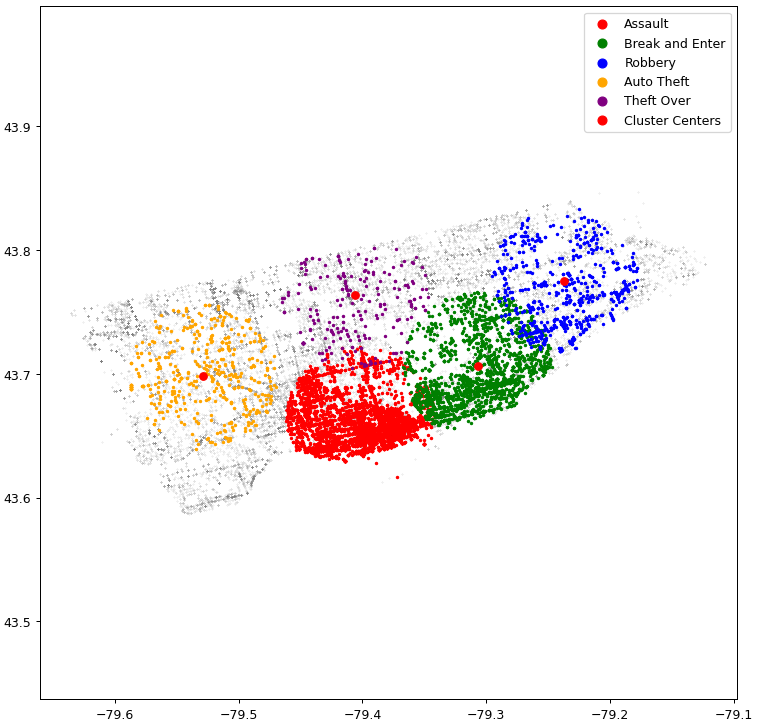
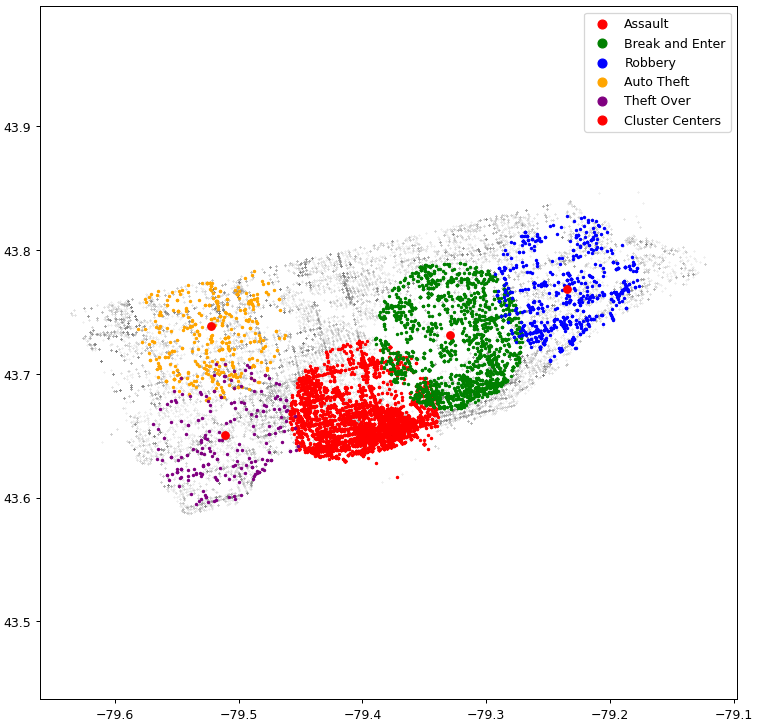
*Figure 3. Plotted Cluster Points and Plotted Crimes.*

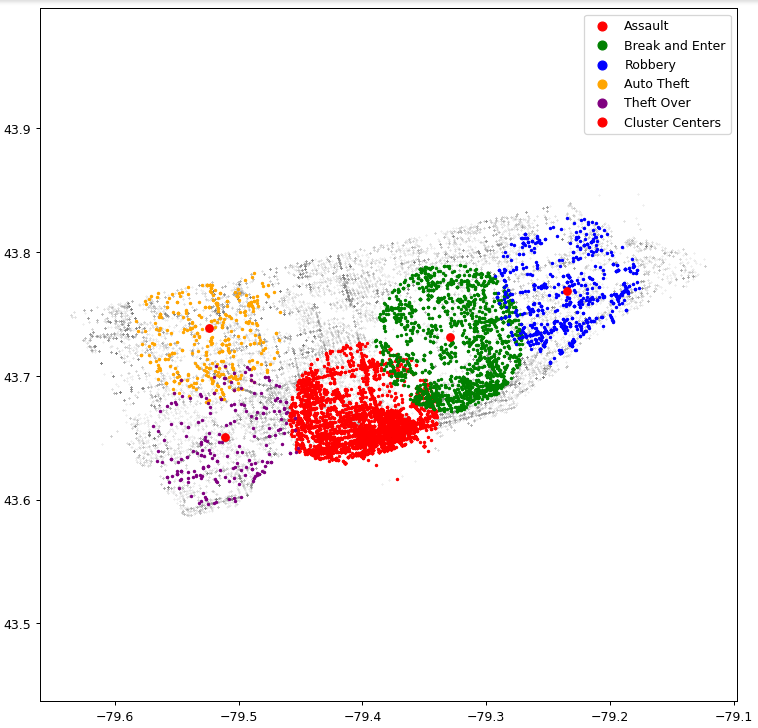
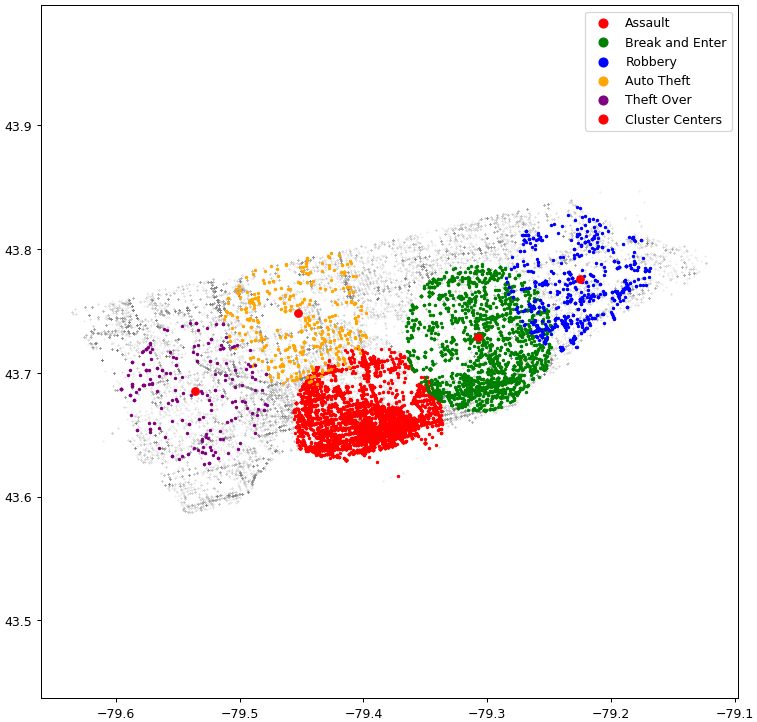
Next step is to now use the KNN to predict a crime for each cluster. The clusters are fed into the KNN in order of clusters with most crimes. This is done to make sure that the hotspots with the most crimes are represented first. If this was not done, then the random order in which the cluster centers would be fed into the KNN would skew the results. As they are predicted they are added to an array keeping track of cluster centers and the labels for their predicted crime. Once all clusters have been predicted and labeled by the KNN a radius is defined. The radius is arbitrary and was set to 0.06 for the best result. This radius allowed for the biggest spread without overlapping over other hotspots. Every crime is then given its own color to be distinct when graphing. Assault is red, Break and Enter is green, Robbery is blue, Theft Over is purple, and Auto Theft is orange. Then all crimes within the radius of their hotstop’s cluster center are graphed with their specified color. The following image is one example run with all the features put together.



*Figure 4. Sample Run.*

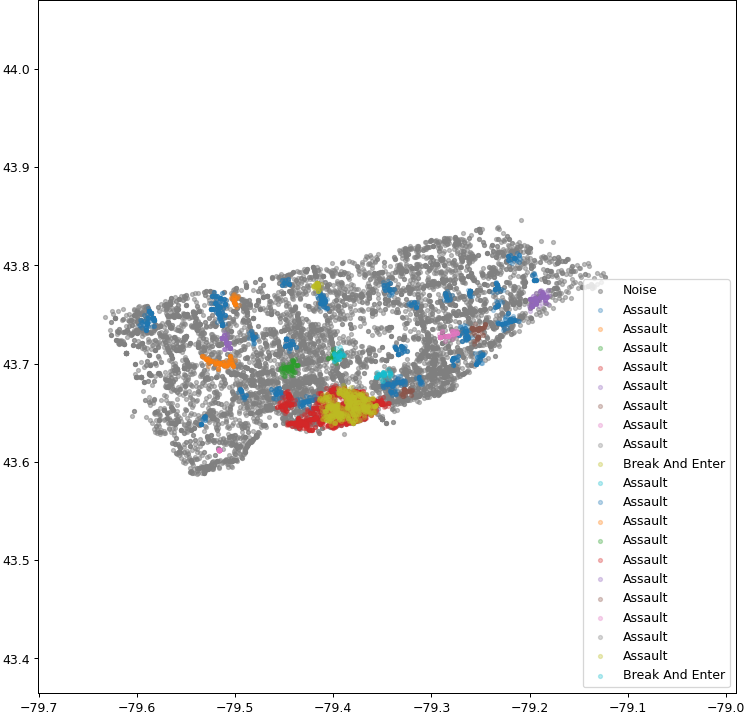
As you can see the program will plot all crimes within the hotspot’s radius that belong to its respective crime in its respective color. You can also see that even within a hotspot there are still gray points. Those gray points are crimes that are within the hotspot, but are not of the hotspot’s crime. The program is fairly consistent with the red cluster (Assault), the green cluster (Break and Enter), and the blue cluster (Robbery) staying in the same general area through multiple test runs. Auto Theft and Theft Over however do move around between multiple test runs. I believe this is because as the clusters in the top left move around and these being the least reported crimes it leaves them scrambling for a spot. This is especially true for Theft Over. Thet Over will move all around the top of Toronto while Auto Theft remains to the top left corner. This can be visualized through multiple test runs shown below.





*Figure 8. Array of four test runs .*

## Density Based Clustering

Finally for the density based clustering model. This model was restricted to only create five clusters not just because there are five crimes to predict, but because the model was very volatile. The model would go from predicting a single cluster with having the entire city as one cluster, to predicting thousands of clusters. Overall though changing the number of clusters did not make much of a difference as more clusters being allowed just meant that there would be more clusters labeled about the same had there only been five. For example the following figures show the difference between labeling five clusters and twenty.

As you can see, the difference between twenty and five clusters is insignificant, and not setting a limit to the clusters only results in a major headache ranging from hundreds of clusters all the way to thousands.

Each cluster is classified by summing up each crime that was committed within its cluster and then the crime that was committed the most is the label for that cluster. This seems to work well with the only issue being that Assault is a super popular choice to be labeled followed by Break and Enter and then rarely Robbery. Auto Theft and Theft Over are very rarely classified. But this makes sense when you see the dataset has very few reports of both. This was the same issue that the KMeans model had except more magnified heat as this model does not force a classification like the KMeans model does. I would argue that this makes the density model stronger since it won't spread misleading information like the KMeans model can. On the other hand, with this model it's hard to tell where a hotspot for Auto Theft or Theft Over would be since it would be very hard for it to be classified.

# 

# Conclusion

To summarize and conclude this report, this project aimed to provide a Crime Hotspot Prediction program using KMeans clustering and K-Nearest-Neighbor machine learning models to cluster and classify five different crime hotspots in the city of Toronto. First a set of five clusters are created through a KMeans model that is fitted with all crimes committed. Then cluster centers are sorted based on how many crimes are committed in their respective radius. Then from largest sum of crimes to the least they are fed into the KNN which will not predict duplicate crimes to classify each hotspot with their crime.

The accuracy of the KMeans clustering model seems to be fairly good. It's not easy to get a numerical answer for the accuracy of this model, but since four of the hotspots remain in the same places with the same crime classifications; it can be assumed that the accuracy is fairly good. The only issue being that two of the hotspots, Auto Theft and Theft Over seem to move around a bit more, especially Theft over. I do believe that this issue is caused by a combination of two things. First the way the KMeans model and the KNN work together, specifically when the clusters are ordered based on the sum of crimes and fed into the KNN. I believe that since the KNN doesn't repeat it forces a prediction and that will be what is left of the unpredicted crimes. This is made worse by the dataset as well. This isn't really a fault of the dataset but more that some crimes are just not as represented as others. Theft Over is not reported very often with only 1883 reports in the dataset. Compare this with Assault which has 26951, it's easy to see why Theft Over has issues being classified.

Lastly there is a Density Based Clustering model that clusters based on core points and its surrounding neighbors. It has a user defined radius and minimum number of samples to help define what a core point is. And each cluster is then classified by summing up the MCI’s that belong to the cluster and the crime that was committed the most is classified. This algorithm is fairly accurate but has the issue of its RAM usage forcing the data to be broken down to a random subset of only 13,000 samples. And since the data in the dataset is low on reports of Auto Theft and Theft Over as mentioned before, this model too suffers from that issue. This model will almost never classify Auto Theft or Theft Over as a crime hotspot because of that. Overall though this model does provide a more accurate representation of what the hotspots of crime look like and how they should be classified when compared to the KMeans model that is classified with a KNN.

# 

# References

Luke, Turner. “Create a K-Means Clustering Algorithm from Scratch in Python.” *Medium*, Towards Data Science, 3 May 2022, towardsdatascience.com/create-your-own-k-means-clustering-algorithm-in-python-d7d4c9077670.

Ali, Moosa. “DBSCAN Clustering Algorithm Implementation from Scratch: Python.” *Medium*, Becoming Human: Artificial Intelligence Magazine, 29 Nov. 2021, becominghuman.ai/dbscan-clustering-algorithm-implementation-from-scratch-python-9950af5eed97.

# Appendix

Google Colab Link:

[CrimePrediction.ipynb](https://colab.research.google.com/drive/18fw9VYmnwSMHZjwuwUSsFUlL0A3O1u4M?usp=sharing)

Data Set Google Drive Link:

https://drive.google.com/file/d/1Vl8hOINWaYPZlc4D9DX0nD4qdgo1z\_f4/view?usp=sharing