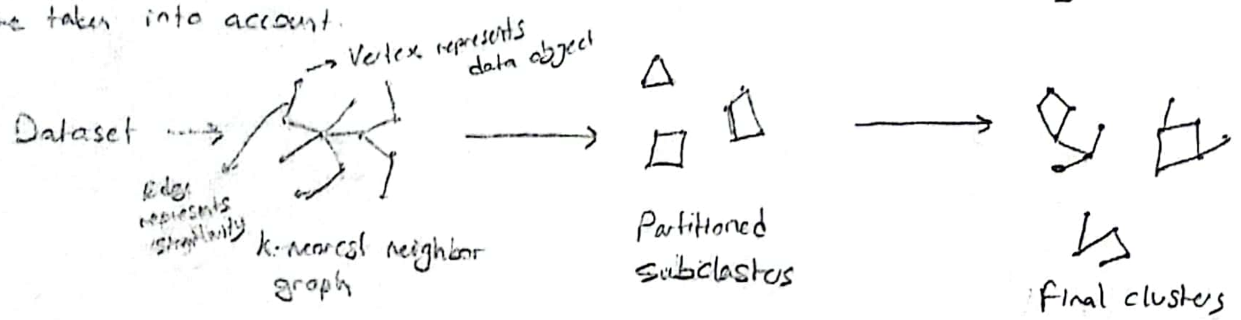
ASSIGNMENT 2

Çağrı Çaycı
1801042629

a) Chameleon is a hiererchical clustering algorithm that uses dynamic modeling to determine the similarity between pairs of clusters. It based on how well data points connected within a cluster and how close clusters are to each other. Two clustes are merged if they have high interconnectivity and they are close. So, chameleon does not rely on a fixed model provided by user; instead, it dynamically determines cluster similarity. It adapts to the internal characteristic of the clustes being merged. This adaptive approach helps discover natural and homogenous clusters.

It uses k-nearest neighbor graph method to create a sparse graph. Then, it employs a graph partitioning algorithm to divide k-nearest-neighbor graph into small subclusters. Subsequently, chameleon utiliez an agglomerative hiererchical clustering algorithm. It merges subclusters based on their similarity. To asseses the similarity between each pair of subclasters, their relative interconnectivity and relative closeness are taken into account.



Relative Interconnectivity of two clusters is calculated as;

$$RI(C_1, C_2) = \frac{|EC_{\{C_1,C_2\}}|}{\frac{1}{2}(|EC_{C_1}| + |EC_{C_2}|)}$$

where $EC\{C_1, C_2\}$ is the edge cut, EC_{C_x} is the minimum sum of the cut edges that partition C_x into two roughly equal parts.

Relative closeness of two cluster is calculated as;

$$RC(C_1, C_2) = \frac{\bar{S}_{EC_{\{C_1,C_2\}}}}{\frac{|C_1|}{|C_1|+|C_2|}\bar{S}_{EC_{C_1}} + \frac{|C_2|}{|C_1|+|C_2|}\bar{S}_{EC_{C_2}}}$$

where $\bar{S}_{EC_{\{C_1,C_2\}}}$ is the average weight of the edges that connect vetices in C_1 to vertices C_2,
$\bar{S}_{EC_{C_x}}$ is the average weight of the edges that belong to min-cut bisector of cluster C_x.

```
procedure ChameleonClustering(data_set, k, threshold)
    Graph ← K_Nearest_Neighbor(data_set, k) // Create a graph using knn.
    Clusters ← hMetis(G) // Partition the graph to clusters using any partition alg.
    return AgglomerativeClustering(Clusters, threshold) // Apply Agglomerative Clustering:
endprocedure

procedure K_Nearest_Neighbor(data_set, k):
    Graph ← { }
    for i=0 to number_of_points:
        Graph.addVertex(i) // Add each point to graph as vertex.
    endfor
    for i=0 to number_of_points:
        Distances ← { }
        for j=0 to number_of_points: // Calculate distance of each point with point i.
            Distances[j] ← euclidian_distance(data_set[i], data_set[j])
        endfor
        SortedDistances ← sort(Distances.values()) // Sort the distances.
        for vertex in SortedDistances[1:k+1]
            Graph.addEdge(i, vertex) // Create edge for closest k neighbor.
        endfor
    endfor
    return Graph
endprocedure.

procedure AgglomerativeClustering((Clusters, threshold):
    while (number_of_clusters > 1): // Continue until there is no clusters to merge.
        any_cluster ← false
        for i=0 to number_of_clusters:
            for j=0 to number_of_clusters: // Calculate each clusters similarities.
                if similarity(Clusters[i], Clusters[j]) > threshold: // if similarity bigger
                    Clusters.append(Clusters[i] + Clusters[j]) // than threshold
                    Clusters.remove((Clusters[i]) // merge the clusters
                    Clusters.remove((Clusters[j])
                    any_cluster ← true
                endif
            endfor
        endfor
        if any_cluster is false: // if there is no cluster to merge, break the loop.
            break
        endif
    endwhile
endprocedure

procedure similarity(x,y):
    return RelativeInterconnectivity(x,y) x RelativeCloseness(x,y)
endprocedure
```

b) Chameleon clustering offers several advantages that make it versatile algorithm for cluster analysis. Firstly, it is not sensitive to the shape and size of clusters, allowing it to effectively identify clusters with irregular shapes and varying sizes. Another notable advantage is its ability to autonomously determine the number of clusters without requiring user input, alleviating the need for prior knowledge about the dataset. Moreover, Chameleon incorporates both relative closeness and relative interconnectivity concurrently, enhancing its robustness against noise and outliers and contributing to the overall quality of the clusters.

However, Chameleon does come with several drawbacks. One significant disadvantage is its computational expense, as it sequentially applies the K-Nearest Neighbor, Partitioning Algorithm, and Agglomerative Clustering Algorithm, coupled with pairwise distance calculations. Additionally, finding optimal parameters such as the threshold for merging clusters and the value of k for the Nearest Neighbor Algorithm can be a non-trivial task, potentially requiring extensive experimentation. Lastly, the choice of distance measure method can significantly impact the success of the algorithm, adding an extra layer of complexity to its implementation and performance.

Algorithm	Time	Space	Performance
K-Means	$O(tknd)$	$O(kd)$	Works well for spherical shapes
DBSCAN	$O(n^2)$	$O(n)$	It is efficient for clusters with arbitrary shapes and sizes. It is sensitive to density of data points.
Hierarchical	$O(n^3)$	$O(n^2)$	It can capture clusters at different scalas. It is sensitive to noise and outliers.
BIRCH	$O(n)$		Fast algorithm with good quality of clustering. However, it does not produce natural clusters.
Chameleon	$O(n^2)$	$O(n^2)$	It can identify clusters with different shapes and sizes. It provides good quality clusters.

c) As we stated in question a, Chameleon Clustering consist of 3 main steps and its time complexity depends on these steps. We can calculate them separately to find overall time complexity.

Start with K-Nearest Neighbor procedure:

The most time consuming process in this procedure is calculating distance between each data point. Nested for loop takes $O(n^2)$ if we assume that calculating distance between two point has $O(1)$ time complexity.

Continue with hMetis procedure:

Its time complexity is $O(V+E)$

End with Agglomerative procedure:

Agglomerative clustering procedure contains 3 nested loops which run n time if partitioning algorithm partitioned until each cluster contains only one point. So, the overall time complexity of this procedure is $O(n^3)$, if we assume that appending an element to cluster or removing an element from it has $O(1)$ time complexity.

Exclude because we are not responsible

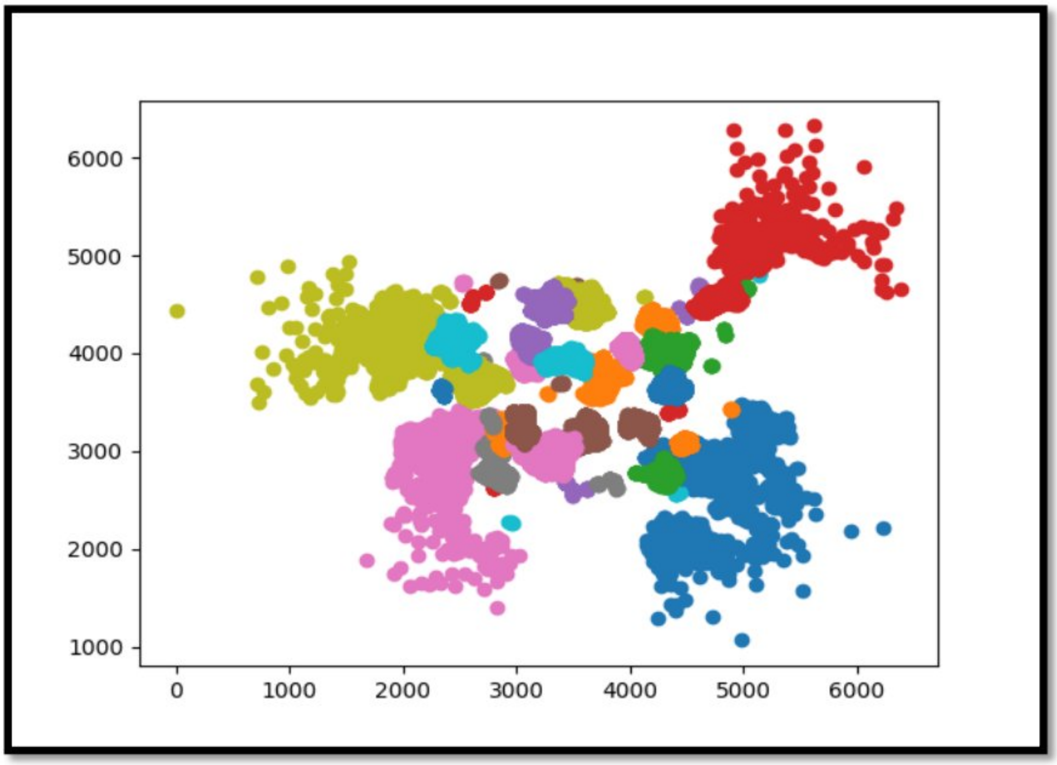
To sum up, Chameleon Clustering Algorithm has $O(n^2 + V+E + n^3)$ time complexity. However, n^3 dominates the time complexity. So, overall worst time complexity of this clustering algorithm is $O(n^3)$.

Note: Time complexity of an algorithm can vary according to implementation. Above results is calculated as my pseudocode for this algorithm.

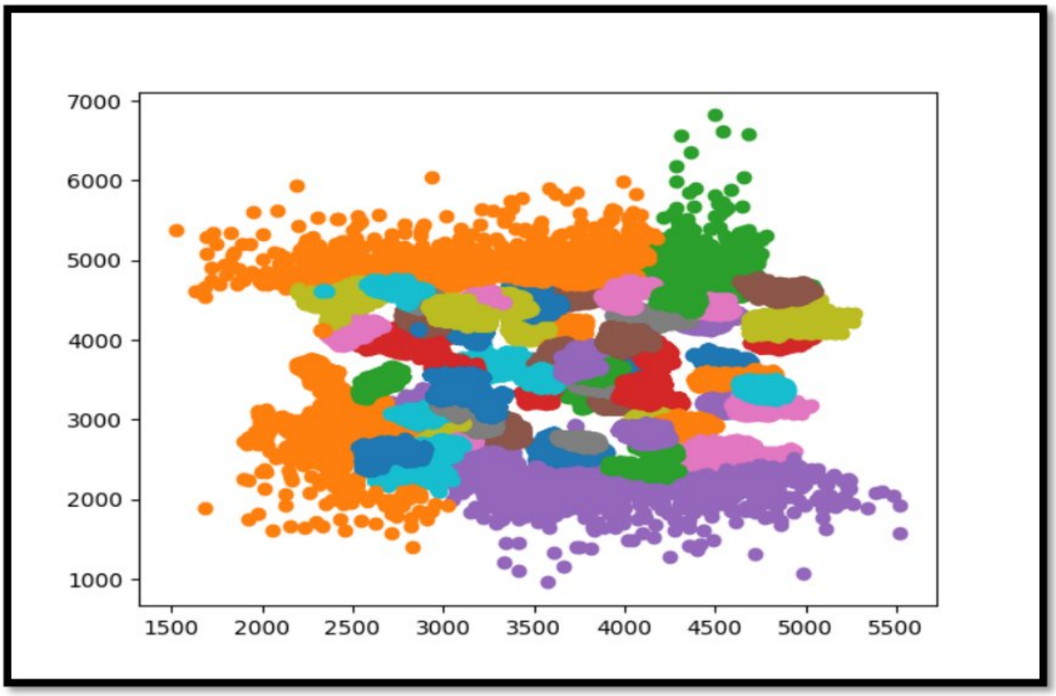
d) This is my video link: https://clipchamp.com/watch/Utk6BNyl2hF

e) I applied this euclidian tests to 105.000 2D points.

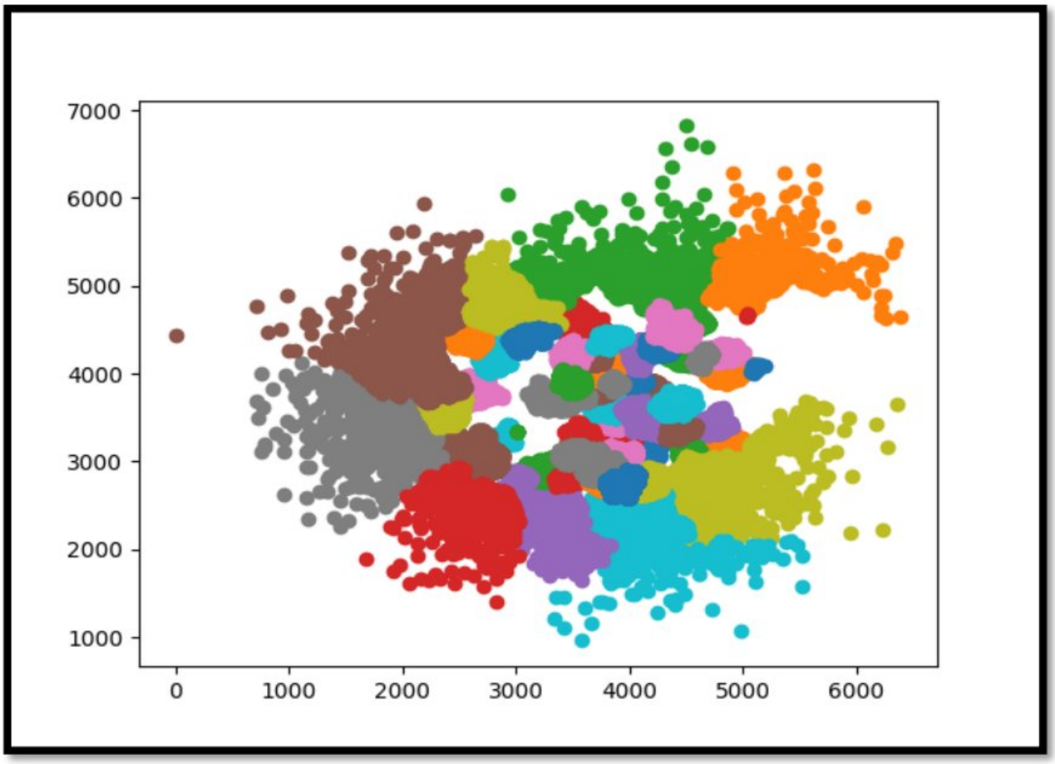
k = 3, threshold = 0.7



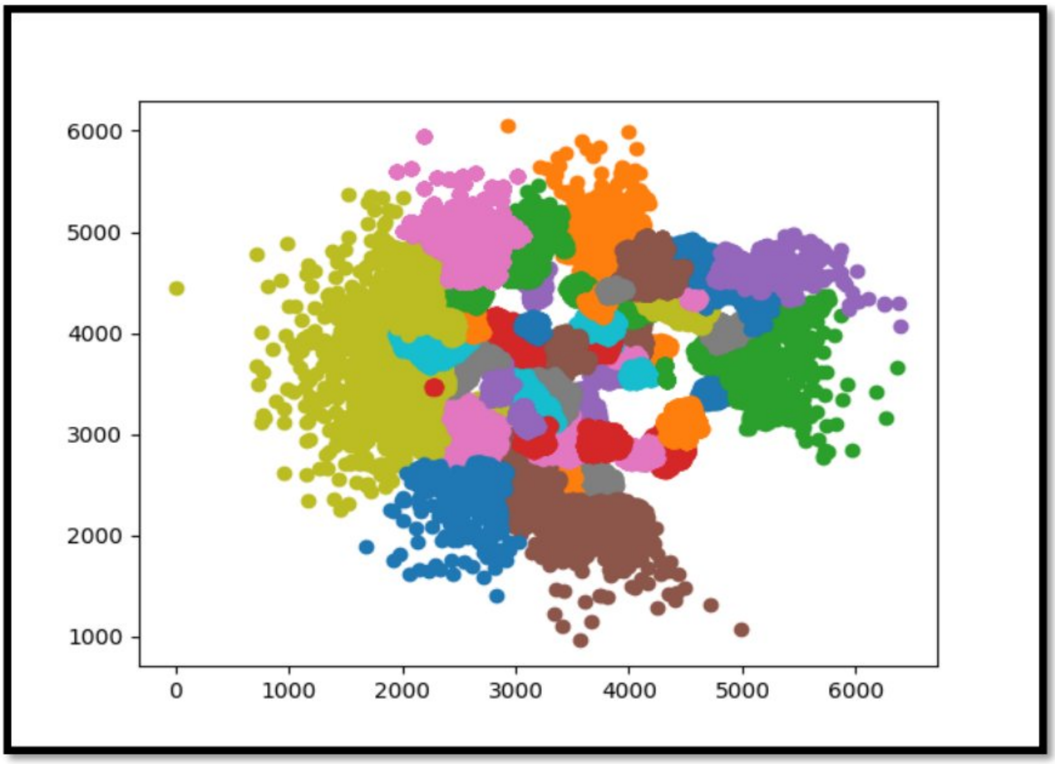
k = 4, threshold = 0.7



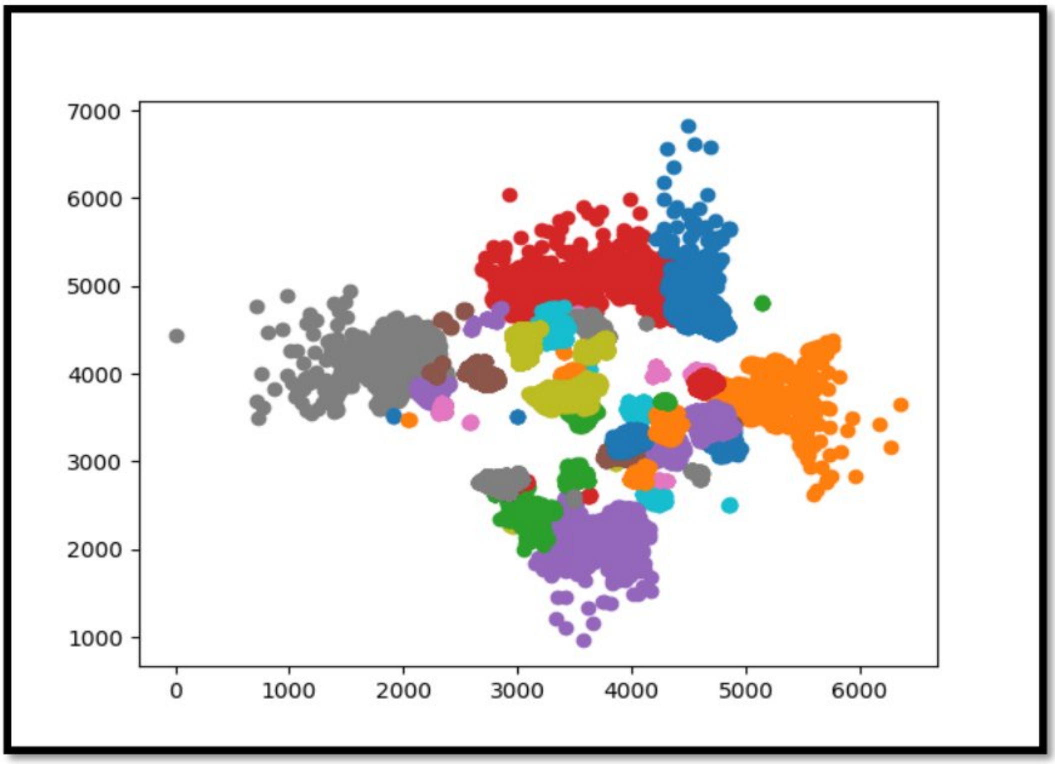
k = 5, threshold = 0.7



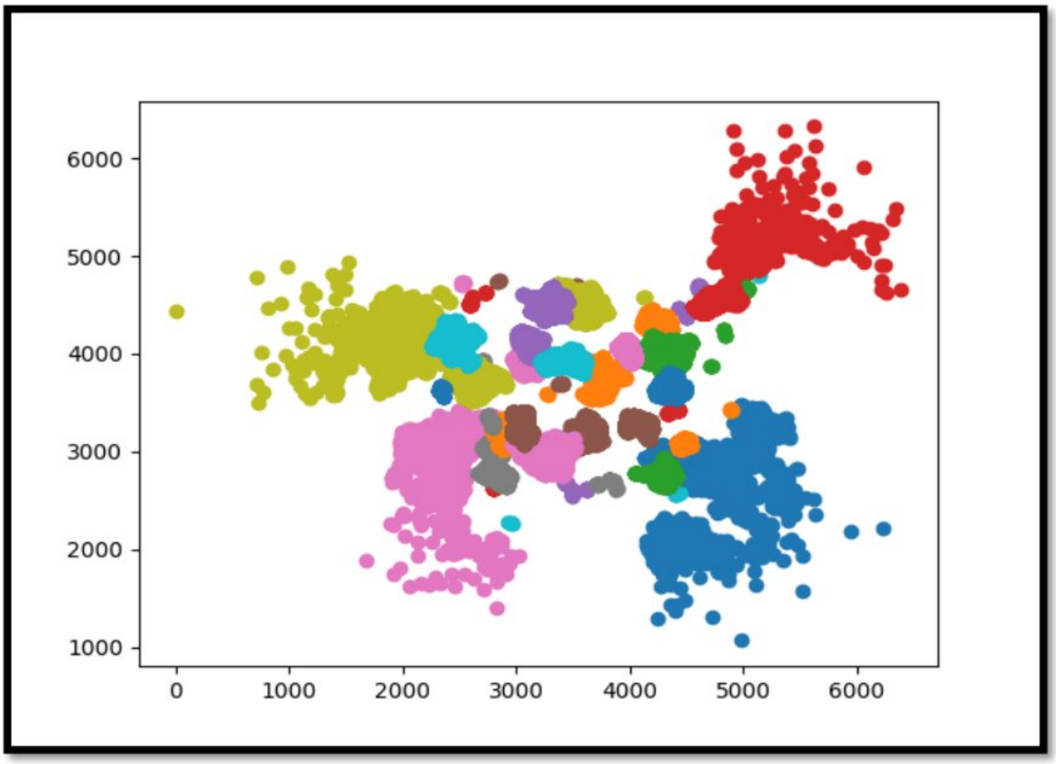
k = 6, threshold = 0.70



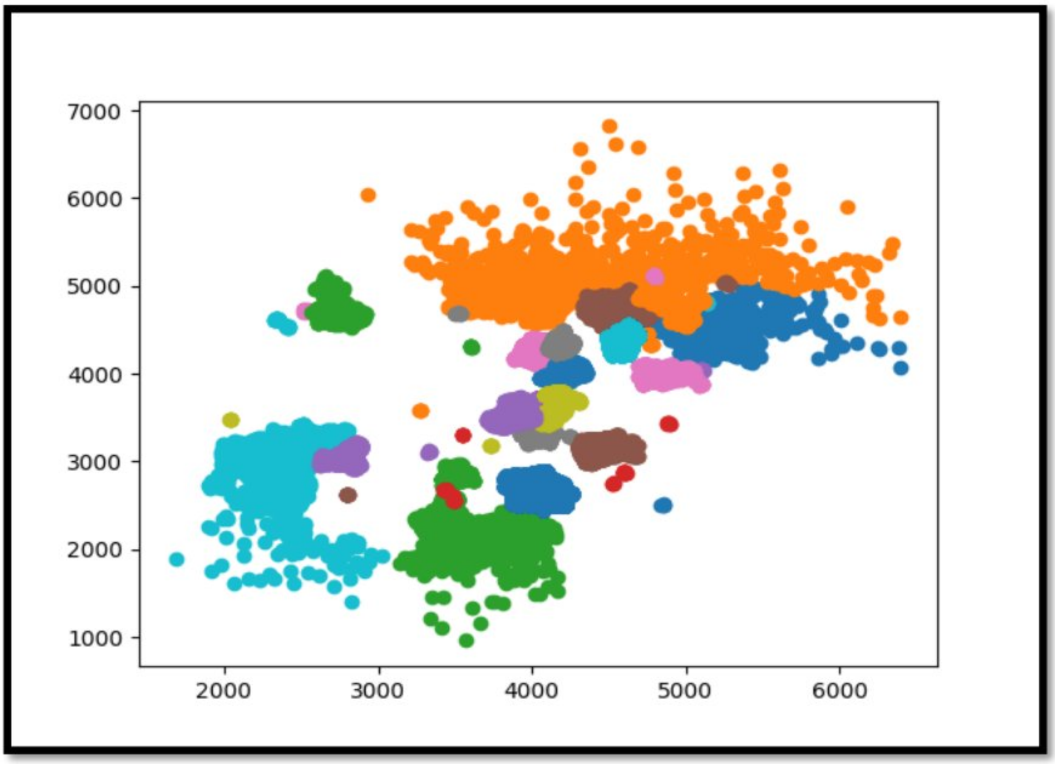
k = 3, threshold = 0.75



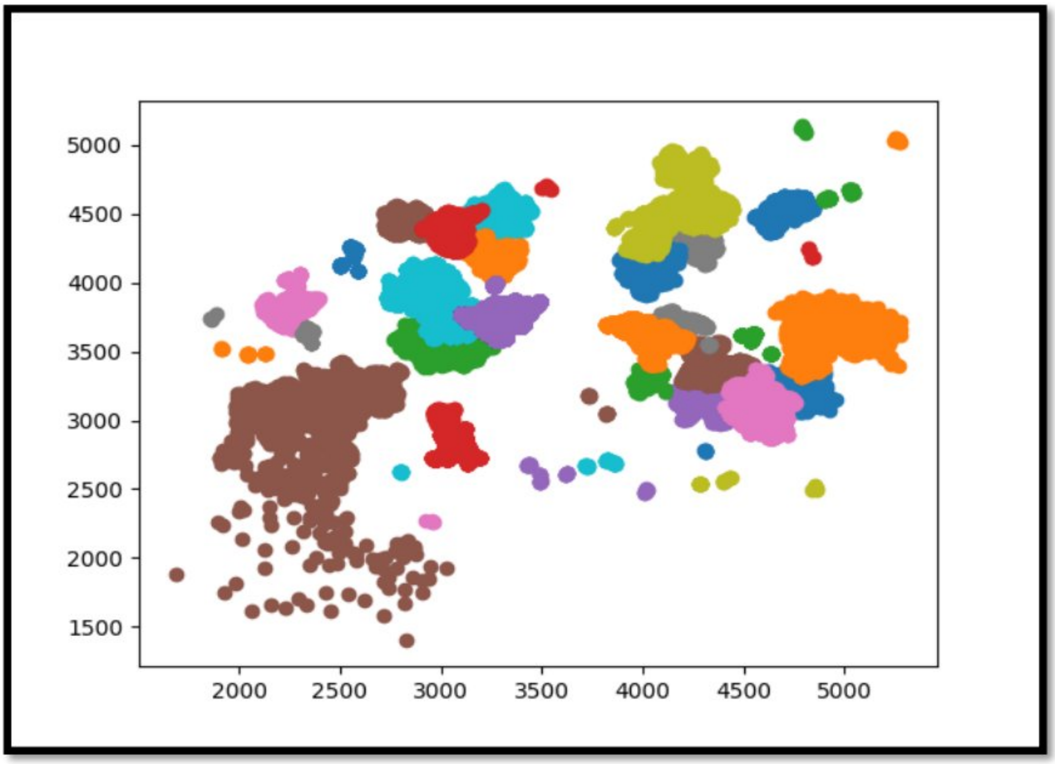
k = 3, threshold = 0.7



k = 3, threshold = 0.65

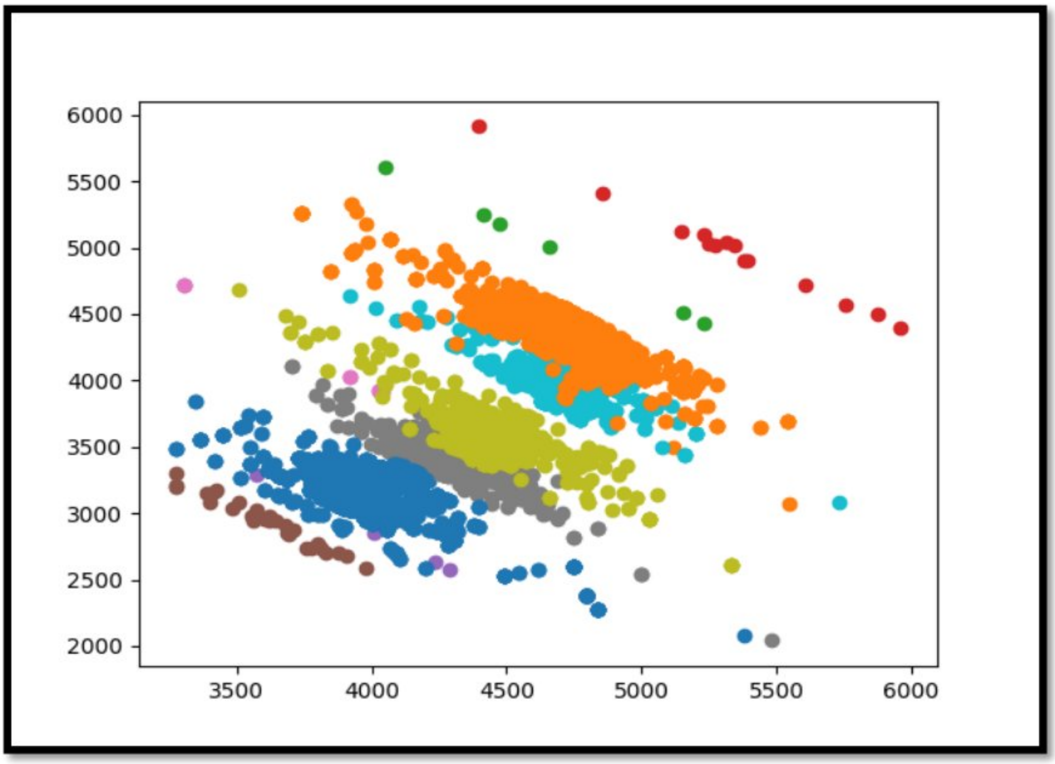


k = 3, threshold = 0.6

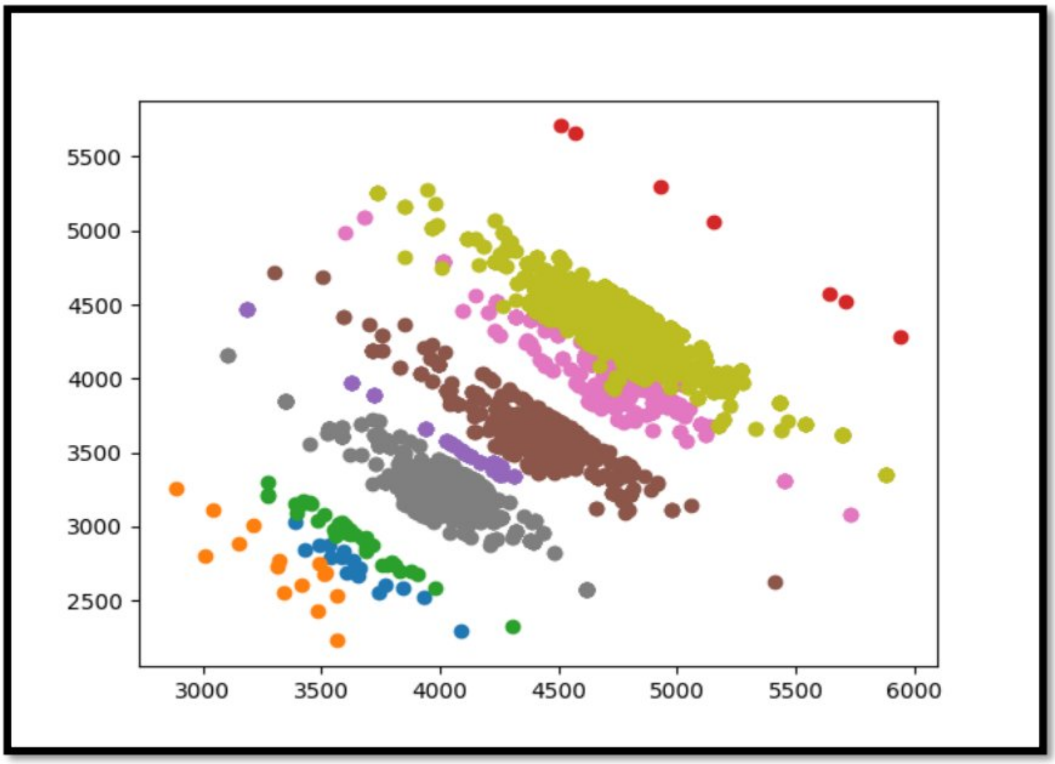


f) I applied this manhatten tests to 10.000 2D points.

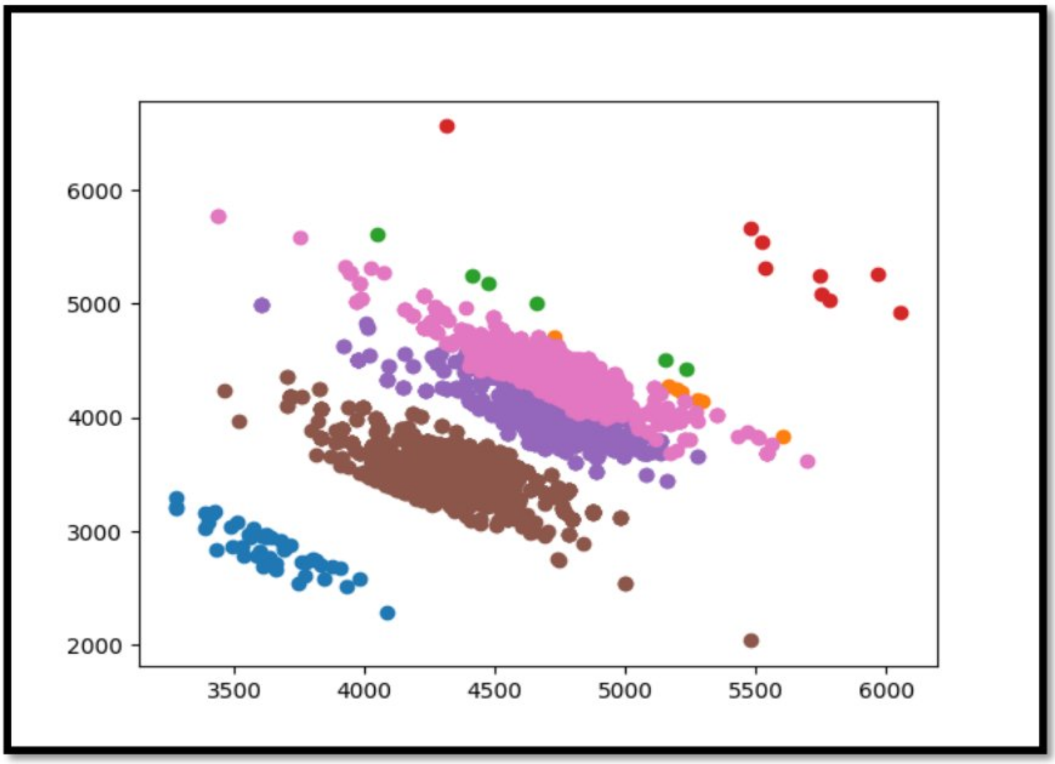
k = 3, threshold = 0.7



k = 5, threshold = 0.7



k = 3, threshold = 0.6



k = 4, threshold = 0.65