Density-Based Place Clustering Using Geo-Social Network Data

MACHINE LEARNING PROJECT REPORT

by

KEERTI CHAUDHARY (181CO226)
SHUMBUL ARIFA (181CO152)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY, KARNATAKA
SURATHKAL, MANGALORE - 575025

ACKNOWLEDGEMENT

We would like to express our gratitude to our professor Dr. M. Venkatesan, Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal for his continuous encouragement and support in carrying out my project work. He has continuously guided our work and gave fruitful suggestions, without which it would have been difficult to carry out this project.

We would like to express our deep gratitude to our mentor Swathi P madam, Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal for her guidance in carrying out our project work.

Last but not least, We take this opportunity to express our deepest gratitude and appreciation to all those who have helped us towards the successful completion of this project.

Contents

Abstract

Geosocial Network clustering deals with the unsupervised grouping of places into clusters and finds important applications in urban planning and marketing. It also permits people to communicate concerning their recent or current areas, and with the assistance of location tracking technology, monitoring of movements and locations of people has become simple through information on latitude and longitude. The information regarding the relation of people to the clustered sites is not considered by current spatial clustering models. The paradigm of density-based clustering needs to be improvised for its application on sites in which visitors are users of a geosocial network. In our project, we have used Bright Kite and Gowalla datasets to implement Geosocial Network data clustering using various clustering approaches, Partitioning approaches like K-means, Hierarchical approach like BIRCH, and Density-based approaches like DBSCAN, HDBSCAN, and OPTICS. Later we have also implemented Voronoi-based graphs. We have also evaluated the different clustering algorithms using evaluation metrics like the Silhouette score, Calinski Harabasz score, and Davies Bouldin score.

***Keywords* - Geo-Social Network, Density-based clustering, Voronoi diagrams**

1. Introduction

A geo-social network (GeoSN) is an online social network augmented by geographical information. Using a GeoSN application, usually, through mobile devices, users can check-in at places, find friends in the vicinity, etc. The Social Web is changing the way people create and use information. Every day millions of pieces of information are shared through the medium of several online social networks and online services with a social layer such as Facebook, Google+, Twitter, Foursquare, and so on.

People have discovered a new way to exploit their sociality: from work to entertainment, from new participatory journalism to religion, from global to local government, from disaster management to market advertisement, from personal status update to milestone family events, the trend is to be social. Information or content is shared by users through the web by posting images or videos, blogging or micro-blogging, surveying and updating geographic information, or playing geographic-based games. Considering the increase in mobile Internet access through smartphones and the number of available (geo)social media platforms, we can expect the amount of information to continuously grow soon. To understand the potential of this change it is worth noticing the amount of "geosocial information" produced during recent years to be a daily occurrence. The following are just a few examples.

In August 2006, Flickr introduced the geo-tagging feature; by 2007, more than 20 million geotagged photos were uploaded to Flickr. In August 2011, Flickr announced its 6 billionth photo, with an increase of 20% year on year, over the last 5 years. Similarly, Twitter was born in 2006. The most impressive performance indicator is the increasing rate of messages. In 2010, the average number of Tweets sent per day was 50 million while in March 2012 it had increased to 340 million. In 2010, the geotagging feature was added to Twitter.

Even considering that the amount of geo-enabled messages is only around 1 percent, this still means millions of geo-tagged messages per day. People can now be considered as sensors, producing signals on events they are directly involved in or have witnessed. Finding, visualizing, and making sense of vast amounts of georeferenced information will lead to a multi-resolution, multi-dimensional representation of the planet known as Digital Earth.

Social networks that also use and create geosocial information have grown in importance and popularity, adopting names such as location-based mobile social networks, or geographic social networks, or simply social networks with geographic features. In general, there exists several types of media with temporal and geographical references on the Internet: (1) geotagged photos on photo-sharing websites like Flickr, (2) geo-referenced videos on websites like Youtube, (3) geo-referenced web documents, like articles in Wikipedia and blogs in MySpace, (4) geo-referenced microblogging websites like Twitter and (5) "check-in" services (users can post their location at a venue and connect with friends) such as Foursquare. Most of these services publish unsupervised (geospatial) content. Their importance has grown in such a way as to also have several definitions such as crowdsourcing which consider users as sensors for gathering data; distributed intelligence where users are basic interpreters or preprocessors in transmitting information; participatory science when citizens participate in problem definition, data collection, and data interpretation.

2. Problem Statement

This section gives us an insight into the aim and use of this project. The existing algorithms use clustering based on a similarity measure that combines social similarity and spatial similarity. Their approaches are distinguished into two categories depending on which side the clustering performed is centered on. In the user-centric approach, the clustering algorithm is essentially applied on the user side, with the spatial similarity incorporated into the social similarity represented as the edge weight. On the other hand, in the location-centric approach, the clustering algorithm is essentially applied on the location side, with the social similarity incorporated into the spatial similarity measure. In our project, we aim to form geo-social clusters using the location-centric approach, by applying the clustering algorithms on attributes such as the latitude and longitude given in our dataset.

3. Literature Review

The problem of clustering presented in the paper is associated with various topics in research, consisting of Partitioning clustering approaches like K-means, Hierarchical clustering approach like BIRCH, and Density-based clustering approaches like DBSCAN, HDBSCAN, and OPTICS.

3.1 What is Clustering?

Clustering is a process that partitions a given data set into homogeneous groups based on given features such that similar objects are kept in a group whereas dissimilar objects are in different groups. It is the most important unsupervised learning problem. It deals with finding structure in a collection of unlabeled data.

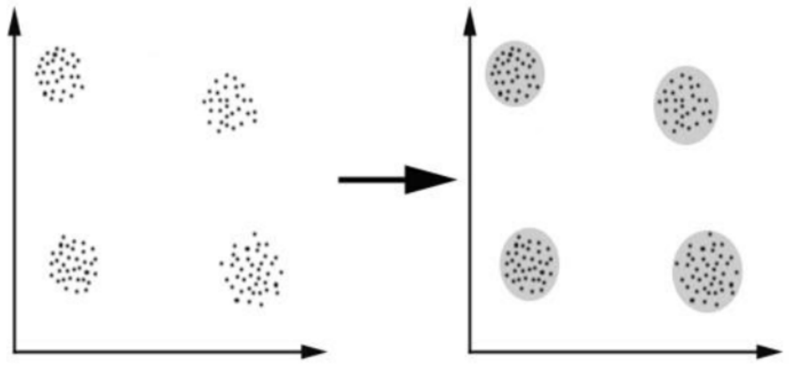


Fig 1: showing four clusters formed from the set of unlabeled data

For clustering algorithms to be advantageous and beneficial some of the conditions need to be satisfied.

1. Scalability - Data must be scalable otherwise we may get the wrong result.
2. The clustering algorithm must be able to deal with different types of attributes.
3. A clustering algorithm must be able to find clustered data with an arbitrary shape.
4. The clustering algorithm must be insensitive to noise and outliers.
5. Interpret-ability and Usability - Results obtained must be interpretable and usable so that maximum knowledge about the input parameters can be obtained.
6. Clustering algorithm must be able to deal with data set of high dimensionality
7. Clustering algorithms can be broadly classified into two categories:
    - ❖ Unsupervised linear clustering algorithms :
        1. k-means clustering algorithm
        2. The fuzzy c-means clustering algorithm
        3. Hierarchical clustering algorithm
        4. Gaussian(EM) clustering algorithm
        5. The quality threshold clustering algorithm
    - ❖ Unsupervised non-linear clustering algorithms:
        1. MST based clustering algorithms
        2. Kernel k-means clustering algorithm
        3. Density-based clustering algorithms

3.2 K-means clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed cunningly because different locations cause different results. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point, we need to re-calculate k new centroids as the barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop, we may notice that the k centers change their location step by step until no more changes are done or in other words, centers do not move anymore. Finally, this algorithm aims at minimizing an objective function known as the squared error function given by:

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

where '||*xi* - *vj*||' is the Euclidean distance between *xi* and *vj*.

'ci' is the number of data points in the *ith* cluster.

'c' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let X = {$x_1, x_2, x_3, \ldots\ldots, x_n$} be the set of data points and V = {$v_1, v_2, \ldots\ldots, v_c$} be the set of centers.

1. Randomly select *'c'* cluster centers.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster center whose distance from the cluster center is the minimum of all the cluster centers.
4. Recalculate the new cluster center using:

   $$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

   where *'ci'* represents the number of points in the *ith* cluster.
5. Recalculate the distance between each data point and newly obtained cluster centers.
6. If no data point was reassigned then stop, otherwise repeat from step 3).

Advantages

1) Fast, robust, and easier to understand.

2) Relatively efficient: O(tknd), where n is the number of objects, k is the number of

clusters, d is the number of dimensions of each object, and t is the number of iterations.

Disadvantages

1) The learning algorithm requires apriori (a popular frequency-determining , algorithm in data mining) specification of the number of cluster centers.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3.3 BIRCH algorithm

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm that can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible. This smaller summary is then clustered instead of clustering the larger dataset. BIRCH is often used to complement other clustering algorithms by creating a summary of the dataset that the other clustering algorithm can now use. However, BIRCH has one major drawback – it can only process metric attributes. A metric attribute is an attribute whose values can be represented in Euclidean space i.e., no categorical attributes should be present.

Clustering Feature (CF):

BIRCH summarizes large datasets into smaller, dense regions called Clustering Feature (CF) entries. Formally, a Clustering Feature entry is defined as an ordered triple, *(N, LS, SS)* where 'N' is the number of data points in the cluster, 'LS' is the linear sum of the data points and 'SS' is the squared sum of the data points in the cluster.

CF Tree:

The CF tree is the actual compact representation that we have been speaking of so far. A CF tree is a tree where each leaf node contains a sub-cluster. Every entry in a CF tree contains a pointer to a child node and a CF entry made up of the sum of CF entries in the child nodes. There is a maximum number of entries in each leaf node. This maximum number is called the *threshold*. We will learn more about what this threshold value is.

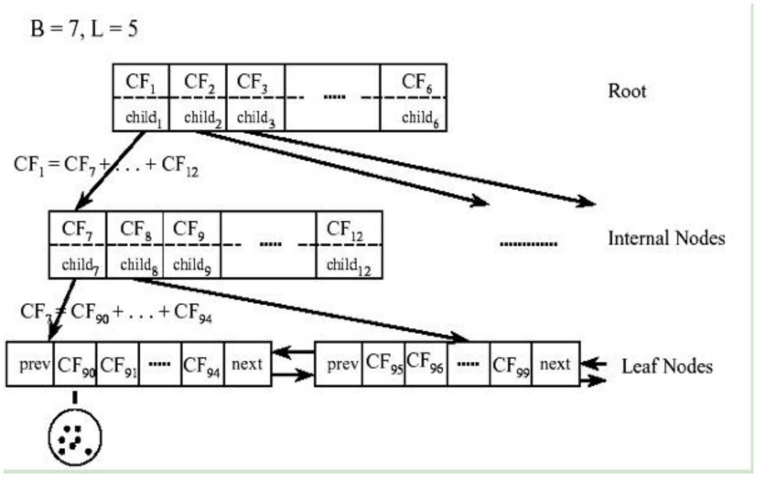

B = 7, L = 5
Root
CF1 CF2 CF3 ..... CF6
child1 child2 child3 child6
CF1 = CF7 + . . . + CF12
CF7 CF8 CF9 ..... CF12
child7 child8 child9 child12
Internal Nodes
CF7 = CF90 + . . . + CF94
prev CF90 CF91 ..... CF94 next
prev CF95 CF96 ..... CF99 next
Leaf Nodes

Fig 2: BIRCH Clustering

Algorithmic steps for BIRCH clustering

1) Building a clustering feature CFtree out of the data points, a height-balanced tree data structure.
2) The algorithm scans all the leaf entries in the initial CF tree to rebuild a smaller CF tree while removing outliers and grouping crowded subclusters into larger ones. This step is marked optional in the original presentation of BIRCH.
3) Here an agglomerative hierarchical clustering algorithm is applied directly to the subclusters represented by their CF vectors. It also allows the user to specify either the desired number of clusters or the desired diameter threshold for clusters. After this step, a set of clusters is obtained that captures major distribution patterns in the data.
4) The centroids of the clusters produced in step 3 are used as seeds and redistribute the data points to their closest seeds to obtain a new set of clusters.

Advantages

1) We do not need to know the expected number of clusters beforehand.

2) Without the computationally expensive final clustering, the fast BIRCH algorithm will become even faster.

Disadvantages

1) Handles only numeric data, and is sensitive to the order of the data record.

2) Favors only clusters with spherical shape and similar sizes, because it uses the notion of diameter to control the boundary of a cluster.

3.4 DBSCAN algorithm

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the most widely used density-based algorithm. It uses the concept of density reachability and density connectivity.

Density Reachability - A point "p" is said to be density reachable from a point "q" if point "p" is within ε distance from point "q" and "q" has a sufficient number of points in its neighbors which are within distance ε.

Density Connectivity - A point "p" and "q" are said to be density connected if there exists a point "r" which has a sufficient number of points in its neighbors and both the points "p" and "q" is within the ε distance. This is the chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" is the neighbor of "t" which in turn is neighbor of "p" implies that "q" is neighbor of "p".

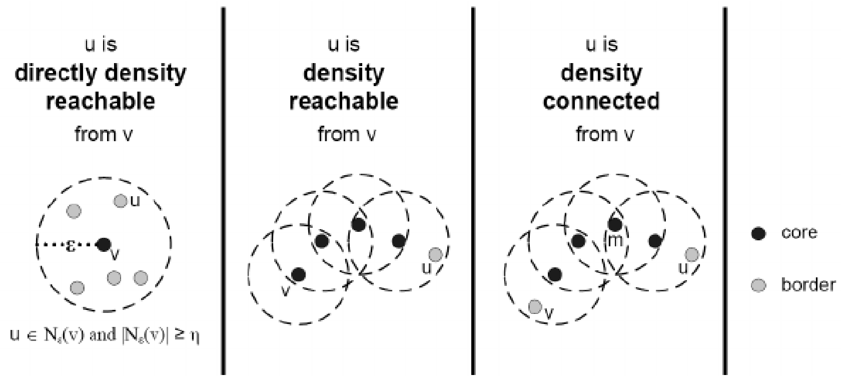

u is
directly density reachable
from v
ε
u
v
$u \in N_\varepsilon(v)$ and $|N_\varepsilon(v)| \geq \eta$
u is
density reachable
from v
v
u
u is
density connected
from v
m
u
v
core
border

Fig 3: Representation of Density Reachability and Density Connectivity

Algorithmic steps for DBSCAN clustering

Let X = {x_1, x_2, x_3, ..., x_n} be the set of data points. DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a cluster (minPts).

1. Start with an arbitrary starting point that has not been visited.
2. Extract the neighborhood of this point using ε (All points which are within the ε distance are neighborhood).
3. If there are sufficient neighborhoods around this point then the clustering process starts and the point is marked as visited else this point is labeled as noise (Later this point can become part of the cluster).
4. If a point is found to be a part of the cluster then its ε neighborhood is also the part of the cluster and the above procedure from step 2 is repeated for all ε neighborhood points. This is repeated until all points in the cluster are determined.
5. A new unvisited point is retrieved and processed, leading to further cluster or noise discovery.
6. This process continues until all points are marked as visited.

Advantages

1) Does not require apriori specification of the number of clusters.

2) Able to identify noise data while clustering.

Disadvantages

1) DBSCAN algorithm fails in the case of varying density clusters.

2) Fails in case of neck type of dataset.

3.5 HDBSCAN

HDBSCAN is a recent algorithm developed by some of the same people who write the original DBSCAN paper. Their goal was to allow varying density clusters. The algorithm starts much the same as DBSCAN: we transform the space according to density, exactly as DBSCAN does, and perform single linkage clustering on the transformed space. Instead of taking an epsilon value as a cut level for the dendrogram, however, a different approach is taken: the dendrogram is condensed by viewing splits that result in a small number of points splitting off as points 'falling out of a cluster. This results in a smaller tree with fewer clusters that 'lose points'. That tree can then be used to select the most stable or persistent clusters. This process allows the tree to be cut at varying heights, picking our varying density clusters based on cluster stability. The immediate advantage of this is that we can have varying density clusters; the second benefit is that we have eliminated the epsilon parameter as we no longer need it to choose a cut of the dendrogram. Instead, we have a new parameter min_cluster_size which is used to determine whether points are 'falling out of a cluster' or splitting to form two new clusters.

Algorithmic steps for HDBSCAN clustering

1. Construct the *k*-nearest-neighbors (*k*-NN) graph, with an undirected edge connecting each point *p* to the *k* most similar points to *p*. Use the *k*-NN information to define a new metric called the *mutual reachability distance* between all pairs of points in the data.
2. Find the minimum spanning tree (MST) connecting all points in the data according to the mutual reachability distance. This tree represents a hierarchy of clusters, and the individual clusters can be extracted using a simple heuristic.

Advantages

1) HDBScan is built for the real-world scenario of having data with varying density.

2) It is relatively faster.

Disadvantages

1) Two dense data partitions on the bottom are not separated from the large clusters.

2) For HDBSCAN, it's not clear how to use it only with a subset of the data.

3.6 OPTICS

Ordering Points To Identify Cluster Structure (OPTICS)) is a density-based clustering technique that allows partitioning data into groups with similar characteristics(clusters). Its addresses one of the DBSCAN's major weaknesses.

The problem of detecting meaningful clusters in data of varying density. In density-based clustering, clusters are defined as dense regions of data points separated by low-density regions. It adds two more terms to the concepts of DBSCAN clustering. They are:

1. **Core Distance:**
   The minimum value of radius is required to classify a given point as a core point. if the given point is not a Core point, then its Core Distance is undefined.

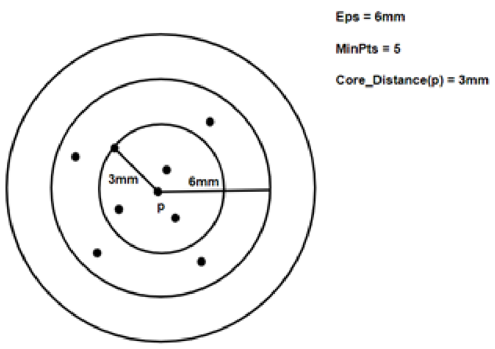


Fig 4: Core distance in OPTICS

2. **Reachability Distance:**
   It is defined concerning another data point 'q'. The Reachability distance between a point **p and q**. Note that The Reachability Distance is not defined if **'a** is not a Core point.

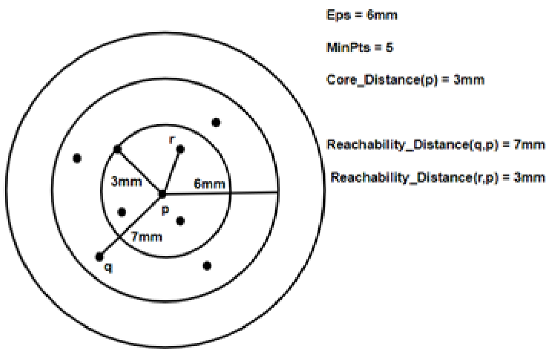

Eps = 6mm
MinPts = 5
Core_Distance(p) = 3mm
Reachability_Distance(q,p) = 7mm
Reachability_Distance(r,p) = 3mm
r
3mm
6mm
p
7mm
q

Fig 5: Reachability distance in OPTICS

Algorithmic steps for OPTICS clustering

1. Mark every point in the dataset as unclustered. Corresponding to each point mark both core distance and reachability distance is undefined.
2. Pick any point and evaluate the neighborhood of that point.

Advantages

1) OPTICS clustering doesn't require a predefined number of clusters in advance.

2) Clusters can be of any shape, including non-spherical ones.

Disadvantages

1) It fails if there are no density drops between clusters.

2) It is also sensitive to parameters that define density (radius and the minimum number of points) and proper parameter settings require domain knowledge.

3.7 Voronoi

A Voronoi diagram is a partition of a plane into regions close to each of a given set of objects. In the simplest case, these objects are just finitely many points in the plane (called seeds, sites, or generators). For each seed there is a corresponding region, called Voronoi cells, consisting of all points of the plane closer to that seed than to any other. The Voronoi diagram of a set of points is dual to its Delaunay triangulation.

Formally, the Voronoi diagram is simply the tuple of cells Voronoi region Rk, where k belongs to R. In principle, some of the sites can intersect and even coincide (an application is described below for sites representing shops), but usually, they are assumed to be disjoint. Besides, infinitely many sites are allowed in the definition (this setting has applications in the geometry of numbers and crystallography), but again, in many cases, only finitely many sites are considered.

Properties

1. The dual graph for a Voronoi diagram (in the case of Euclidean space with point sites) corresponds to the Delaunay triangulation for the same set of points.

2. The closest pair of points correspond to two adjacent cells in the Voronoi diagram.
3. Assume the setting is the Euclidean plane and a group of different points is given. Then two points are adjacent on the convex hull if and only if their Voronoi cells share an infinitely long side.
4. If space is a normed space and the distance to each site is attained (e.g., when a site is a compact set or a closed ball), then each Voronoi cell can be represented as a union of line segments emanating from the sites. As shown there, this property does not necessarily hold when the distance is not attained.
5. Under relatively general conditions (space is a possibly infinite-dimensional uniformly convex space, there can be infinitely many sites of a general form, etc.)

Applications

1. In aviation, they are used to identify the nearest airport in case of diversions.
2. In networking, Voronoi diagrams can be used in derivations of the capacity of a wireless network.
3. In biology, Voronoi diagrams are used to model several different biological structures, including cells and bone microarchitecture. Indeed, Voronoi tessellations work as a geometrical tool to understand the physical constraints that drive the organization of biological tissues.
4. In hydrology, Voronoi diagrams are used to calculate the rainfall of an area, based on a series of point measurements. In this usage, they are generally referred to as Thiessen polygons.
5. In ecology, Voronoi diagrams are used to study the growth patterns of forests and forest canopies, and may also help develop predictive models for forest fires.
6. In computational chemistry, ligand-binding sites are transformed into Voronoi diagrams for machine learning applications.
7. In computational fluid dynamics, the Voronoi tessellation of a set of points can be used to define the computational domains used in finite volume methods.
8. In computational physics, Voronoi diagrams are used to calculate profiles of an object with Shadowgraph and proton radiography in High energy density physics.
9. In mining, Voronoi polygons are used to estimate the reserves of valuable materials, minerals, or other resources. Exploratory drill holes are used as the set of points in the Voronoi polygons.
10. In computer graphics, Voronoi diagrams are used to calculate 3D shattering/fracturing geometry patterns.
11. In autonomous robot navigation, Voronoi diagrams are used to find clear routes. If the points are obstacles, then the edges of the graph will be the routes furthest from obstacles (and theoretically any collisions).
12. In machine learning, Voronoi diagrams are used to do 1-NN classifications.

13. In user interface development, Voronoi patterns can be used to compute the best hover state for a given point.
14. In Melbourne, government school students are always eligible to attend the nearest primary school or high school to where they live, as measured by a straight-line distance. The map of school zones is therefore a Voronoi diagram.

A Voronoi diagram we plotted in our project is below:

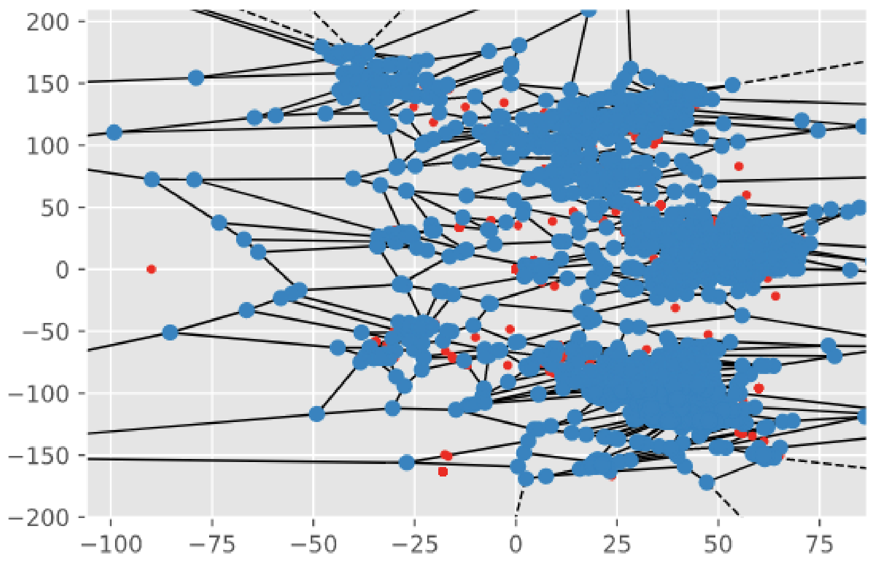


Fig. Spatial Voronoi Diagram for Gowalla dataset

3.8 Evaluation metrics for Clustering Algorithms

Clusters are evaluated based on some similarity or dissimilarity measures such as the distance between cluster points. If the clustering algorithm separates dissimilar observations apart and similar observations together, then it has performed well. The metrics we've used to evaluate clustering in our model are the Silhouette score, Davies–Bouldin index, and Calinski-Harabasz index.

3.8.1 Silhouette score

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The technique provides a succinct graphical representation of how well each object has been classified.

The silhouette value is a measure of how similar an object is to its cluster (cohesion) compared to other clusters (separation). The silhouette ranges from,−1 to +1, where a high value indicates that the object is well matched to its cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance. The calculation of the Silhouette score can be done by using the following formula

$$\textit{Silhouette score}=(p-q)/\max(p,q)$$

Here, p = mean distance to the points in the nearest cluster, and

q = mean intra-cluster distance to all the points.

3.8.2 Davies–Bouldin index

The Davies–Bouldin index (DBI), introduced by David L. Davies and Donald W. Bouldin in 1979, is a metric for evaluating clustering algorithms. This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. This has a drawback that a good value reported by this method does not imply the best information retrieval.
We can calculate the DB index with the help of the following formula −

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{j\neq i}(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)})$$

Here, n = number of clusters

σi = average distance of all points in cluster *i* from the cluster centroid ci

The less the DB index, the better the clustering model is.

3.8.3 Calinski-Harabasz index

Calinski-Harabasz (CH) Index can be used to evaluate the model when ground truth labels are not known where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. The CH Index (also known as Variance ratio criterion) is a measure of how similar an object is to its cluster (cohesion) compared to other clusters (separation). Here cohesion is estimated based on the distances from the data points in a cluster to its cluster centroid and separation is based on the distance of the cluster centroids from the global centroid. The CH index for K number of clusters on a dataset D =[d1, d2, d3, ... dN] is defined as,

$$CH = \left[\frac{\sum_{k=1}^{K} n_k \left\| c_k - c \right\|^2}{K-1} \right] / \frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| d_i - c_k \right\|^2}{N-K} \right]$$

where *nk* and *ck* are the no. of points and centroid of the *kth* cluster respectively, c is the global centroid, N is the total no. of data points.

The higher value of the CH index means the clusters are dense and well separated, although there is no “acceptable” cut-off value. We need to choose that solution that gives a peak or at least an abrupt elbow on the line plot of CH indices.

4. Implementation

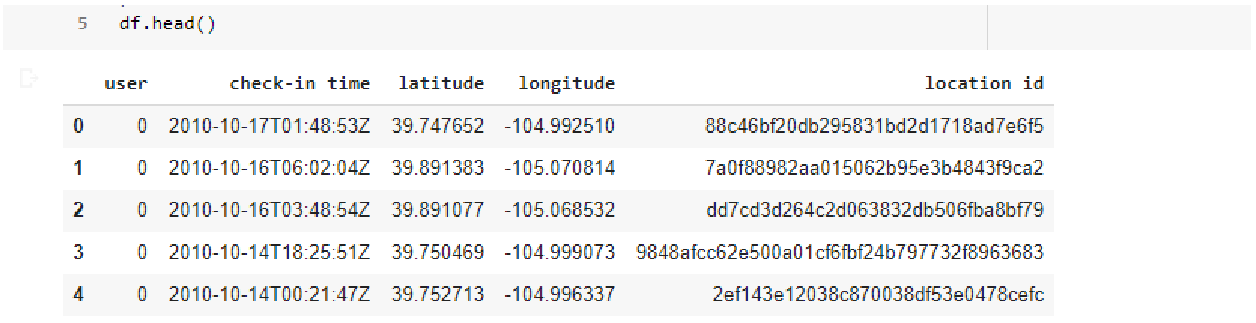
Dataset used

Two openly accessible datasets from past geosocial networking data are utilized. Gowalla incorporates |Y| = 196;591 users in the geo-social network. It has |Ck| = 6;442;892 check-ins performed by those users on |L| = 1;280;969 locations over the span of February 2009 - October 2010. The other dataset, Brightkite contains |Y| = 58;228 users in the geo-social network. There are |Ck| = 4;491;143 check-ins on |L| = 772;783 locations accumulated over the span of April 2008 - October 2010.

Code Implementation

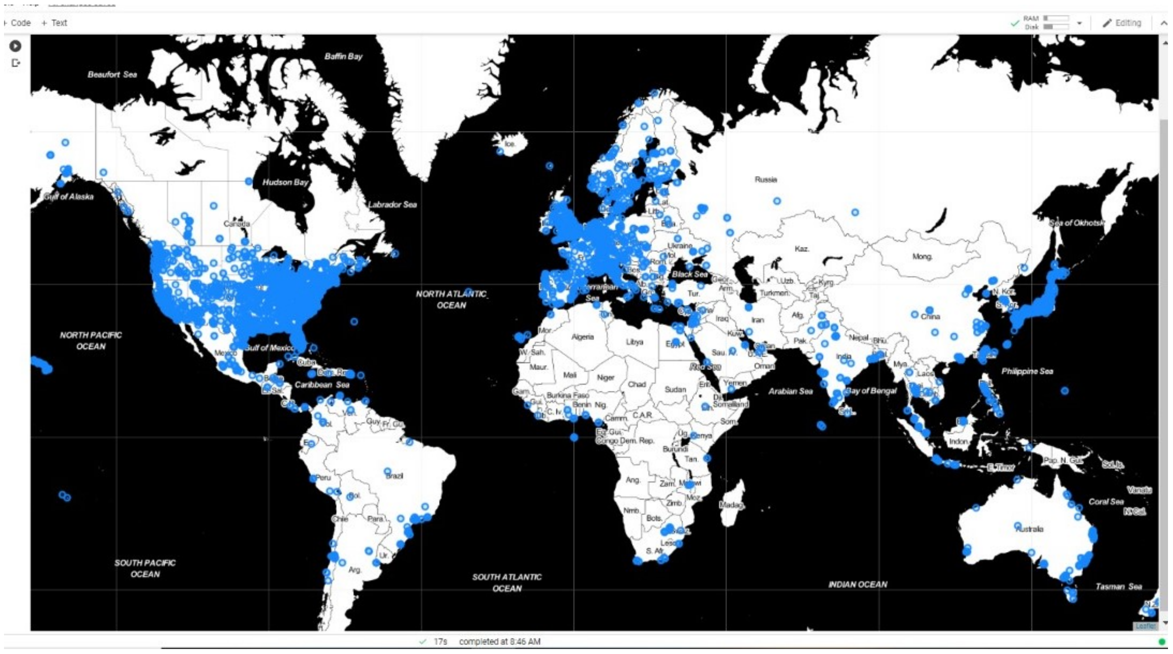
- **Loading Bright Kite Dataset**

- **Plotting Locations on Map using Folium library**

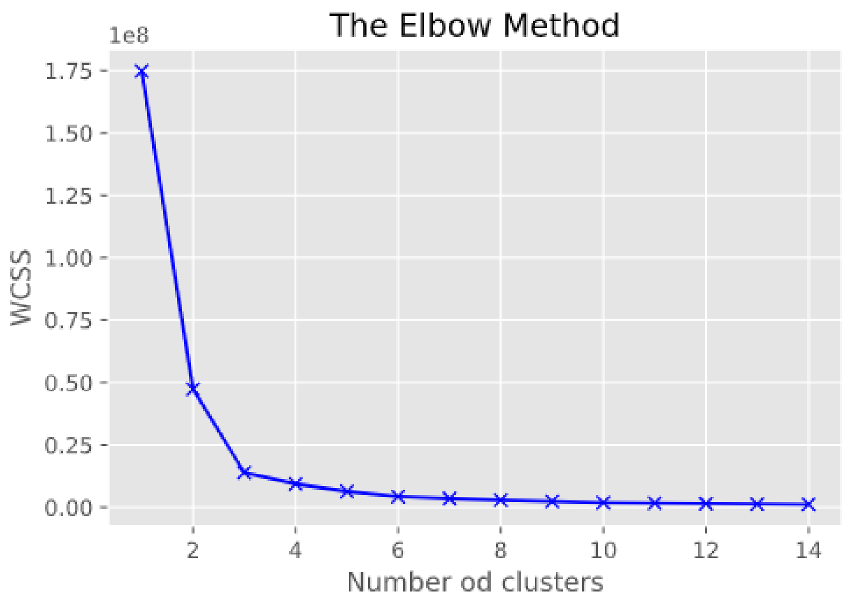


```
df.head()
```

```
   user      check-in time   latitude   longitude                         location id
0     0  2010-10-17T01:48:53Z  39.747652  -104.992510    88c46bf20db295831bd2d1718ad7e6f5
1     0  2010-10-16T06:02:04Z  39.891383  -105.070814    7a0f88982aa015062b95e3b4843f9ca2
2     0  2010-10-16T03:48:54Z  39.891077  -105.068532    dd7cd3d264c2d063832db506fba8bf79
3     0  2010-10-14T18:25:51Z  39.750469  -104.999073  9848afcc62e500a01cf6fbf24b797732f8963683
4     0  2010-10-14T00:21:47Z  39.752713  -104.996337    2ef143e12038c870038df53e0478cefc
```

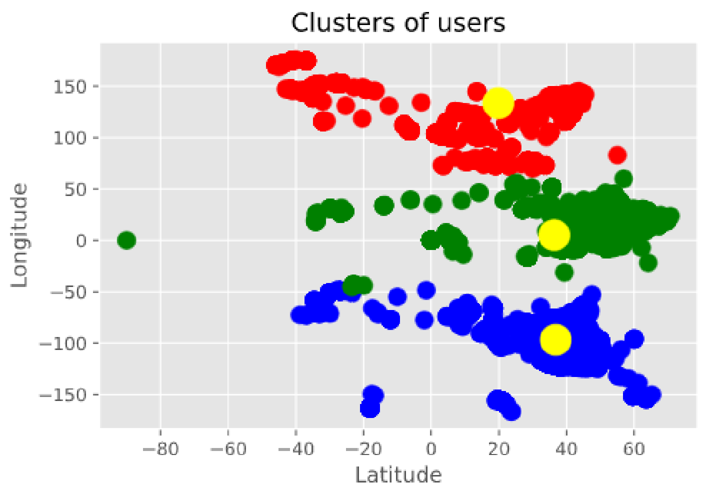


- **Finding an Appropriate number of the cluster for K-means using the Elbow method, which comes to be 3**

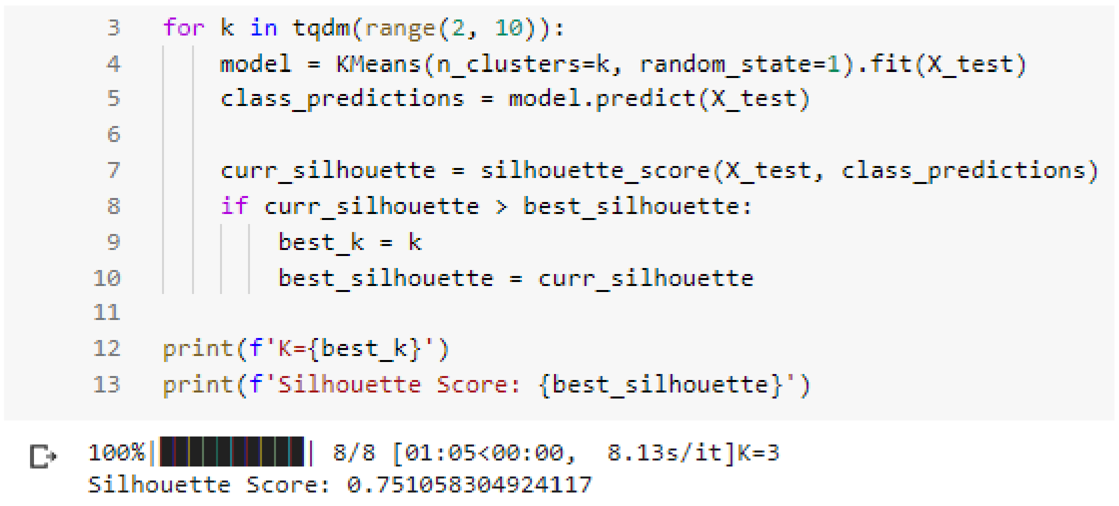


- **Scatterplot of clusters in K-means**

```
plt.scatter(X_test[y_kmeans==0,0],X_test[y_kmeans==0,1],s=100,c='red',label='cluster 1')
plt.scatter(X_test[y_kmeans==1,0],X_test[y_kmeans==1,1],s=100,c='blue',label='cluster 2')
plt.scatter(X_test[y_kmeans==2,0],X_test[y_kmeans==2,1],s=100,c='green',label='cluster 3')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='yellow',label='centroids')
plt.title('Clusters of users')
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.show()
```



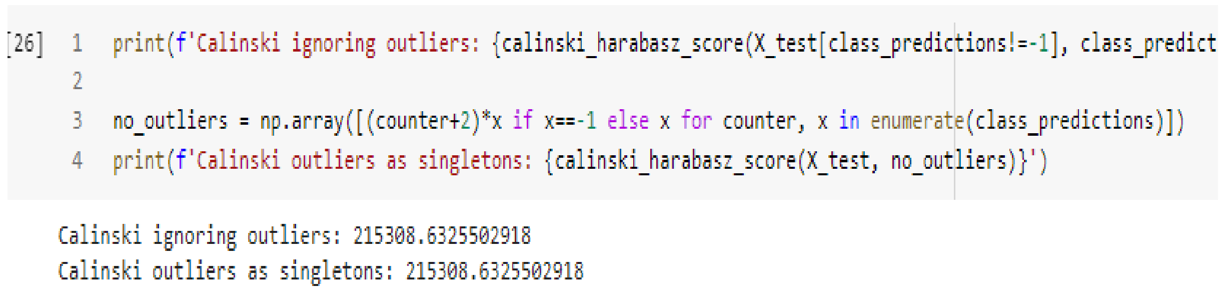
- **Calculation of Silhouette score of clusters in K-means**



```
for k in tqdm(range(2, 10)):
    model = KMeans(n_clusters=k, random_state=1).fit(X_test)
    class_predictions = model.predict(X_test)

    curr_silhouette = silhouette_score(X_test, class_predictions)
    if curr_silhouette > best_silhouette:
        best_k = k
        best_silhouette = curr_silhouette

print(f'K={best_k}')
print(f'Silhouette Score: {best_silhouette}')
```

```
100%|██████████| 8/8 [01:05<00:00,  8.13s/it]K=3
Silhouette Score: 0.751058304924117
```

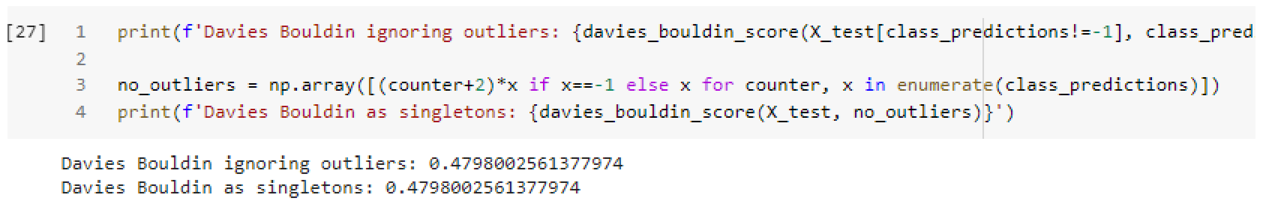
- **Calculation of Calinski Harabasz score of K-means**



```
[26]
print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions!=-1], class_predict

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictions)])
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers)}')
```

```
Calinski ignoring outliers: 215308.6325502918
Calinski outliers as singletons: 215308.6325502918
```

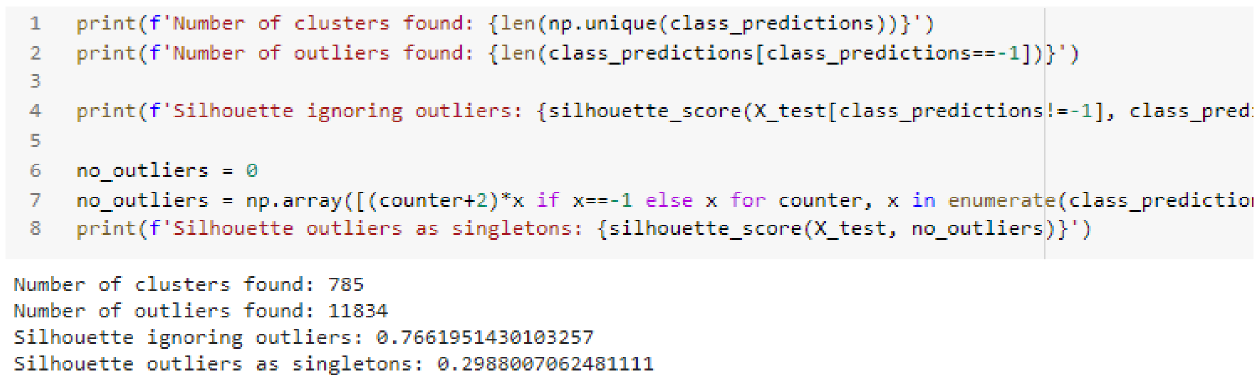
- **Calculation of Davies Bouldin score of K-means**



```
[27]
print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predictions!=-1], class_pred

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictions)])
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Davies Bouldin ignoring outliers: 0.4798002561377974
Davies Bouldin as singletons: 0.4798002561377974
```

- **Calculation of number of clusters and Silhouette score of clusters in DBSCAN**



```
print(f'Number of clusters found: {len(np.unique(class_predictions))}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1], class_pred

no_outliers = 0
no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictio
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')
```

```
Number of clusters found: 785
Number of outliers found: 11834
Silhouette ignoring outliers: 0.7661951430103257
Silhouette outliers as singletons: 0.2988007062481111
```

- **Calculation of Calinski Harabasz score of DBSCAN**

```
print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions!=-1], class_predi

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictions)])
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers)}')
```

```
Calinski ignoring outliers: 577981985.2748314
Calinski outliers as singletons: 68803410.67455058
```

- **Calculation of Davies Bouldin score of DBSCAN**

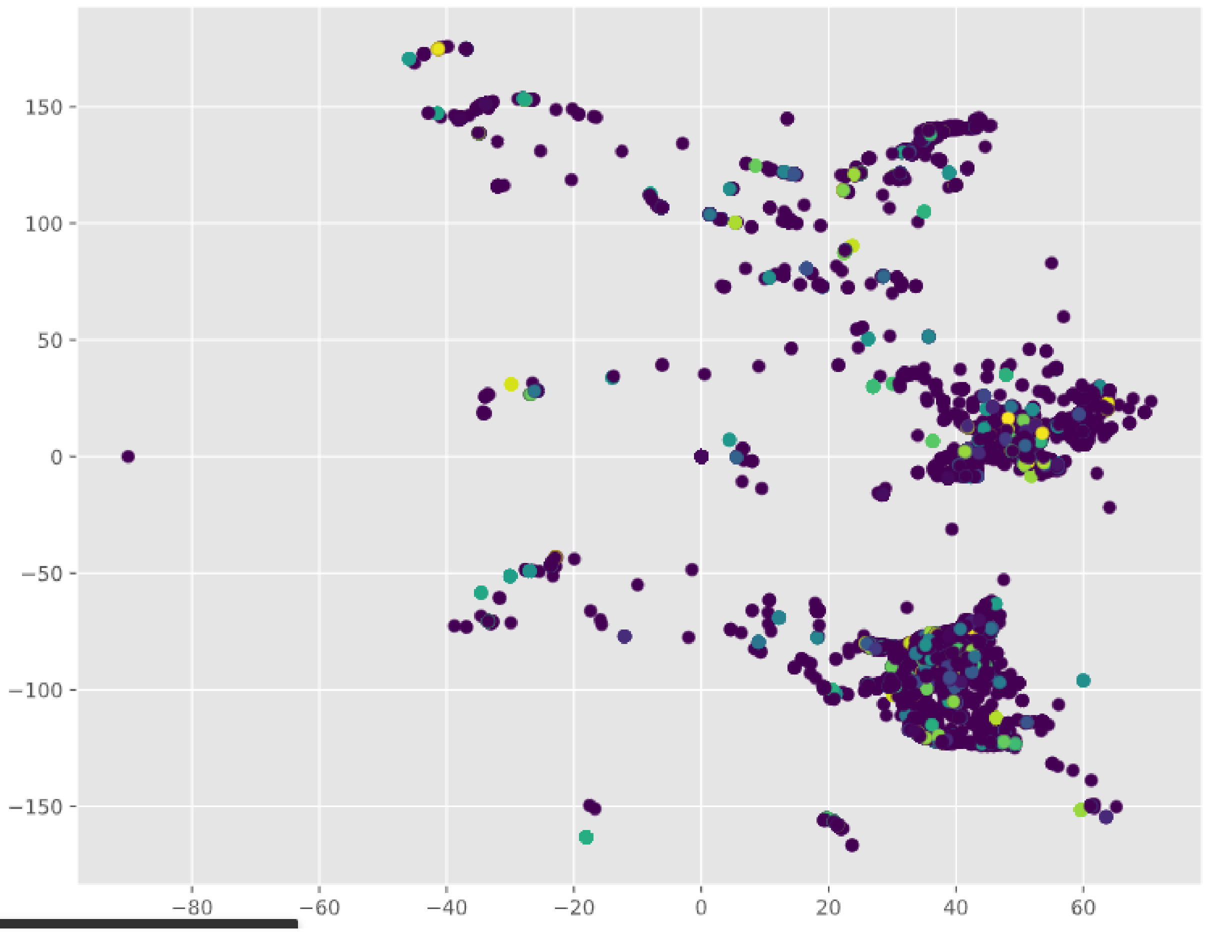
```
print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predictions!=-1], class_pr

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictions)])
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Davies Bouldin ignoring outliers: 0.14653707772832836
Davies Bouldin as singletons: 0.12403104775502434
```

- **Scatterplot of clusters in DBSCAN**

```
plt.figure(figsize=(10, 8))
plt.scatter(X_test[:,0], X_test[:,1], c=model.labels_.astype(float))
```

```
<matplotlib.collections.PathCollection at 0x7f7fc5e85cd0>
```



- **Calculation of number of clusters and Silhouette score of clusters in HDBSCAN**

```
print(f'Number of clusters found: {len(np.unique(class_predictions))-1}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')
```

```
Number of clusters found: 1485
Number of outliers found: 3801
Silhouette ignoring outliers: 0.6548991258646143
Silhouette outliers as singletons: 0.4596338355071577
```

- **Calculation of Calinski Harabasz score of HDBSCAN**

```
print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions!=-1],

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_pre
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers)}')
```

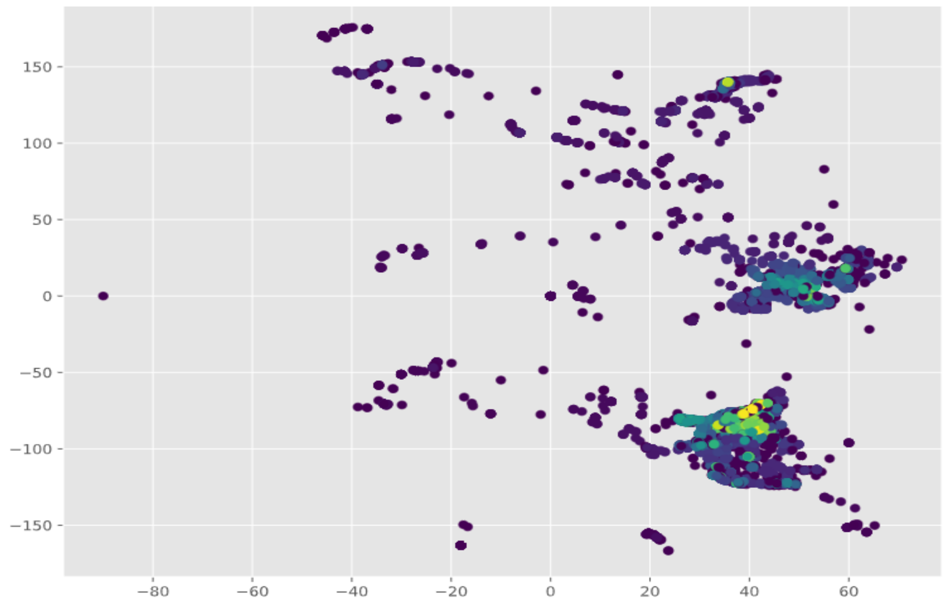
```
Calinski ignoring outliers: 456389.3477635357
Calinski outliers as singletons: 151607.63399134102
```

- **Calculation of Davies Bouldin score of HDBSCAN**

```
print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predict

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Davies Bouldin ignoring outliers: 0.4204648261224274
Davies Bouldin as singletons: 0.3548516344098311
```

- **Scatterplots of clusters in HDBSCAN**



- **Calculation of number of clusters and Silhouette score of clusters in BIRCH**

```
print(f'Number of clusters found: {len(np.unique(class_predictions))-1}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1],

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(clas
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')
```

```
Number of clusters found: 875
Number of outliers found: 0
Silhouette ignoring outliers: 0.6563531735169894
Silhouette outliers as singletons: 0.6563531735169894
```

- **Calculation of Calinski Harabasz score of BIRCH**

```
print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions!=-1]

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_pr
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers)}')
```

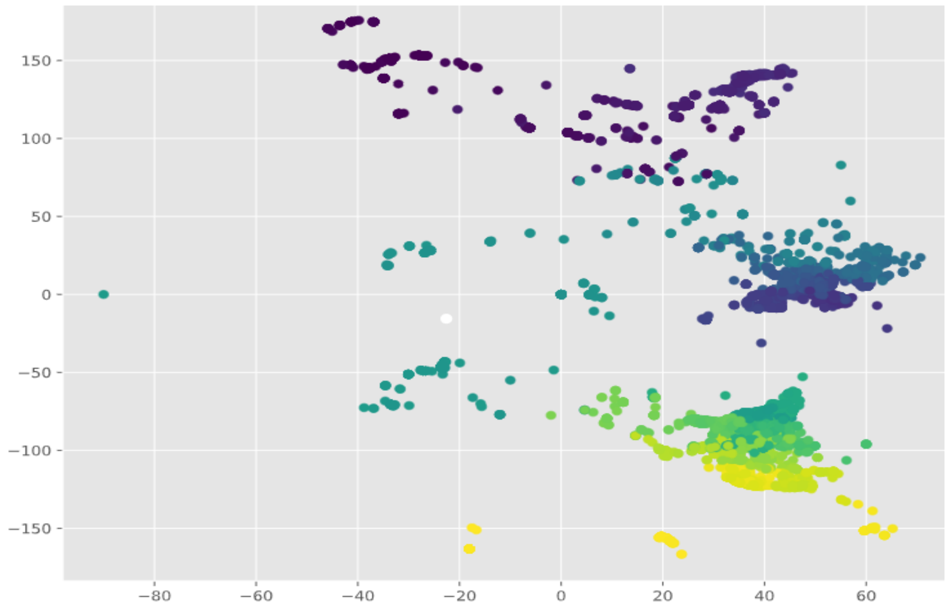
```
Calinski ignoring outliers: 2869745.220027992
Calinski outliers as singletons: 2869745.220027992
```

- **Calculation of Davies Bouldin score of BIRCH**

```
]  print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predict

   no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
   print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Davies Bouldin ignoring outliers: 0.38315825945478854
Davies Bouldin as singletons: 0.38315825945478854
```

- **Scatterplots of clusters in BIRCH**



- **Calculation of number of clusters and Silhouette score of clusters in OPTICS**

```
print(f'Number of clusters found: {len(np.unique(class_predictions))-1}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')
```

```
Number of clusters found: 5463
Number of outliers found: 3076
Silhouette ignoring outliers: 0.791322364163566
Silhouette outliers as singletons: 0.6747560455330907
```

- **Calculation of Calinski Harabasz score of OPTICS**

```
print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions!=-1],

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_pre
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers)}')
```

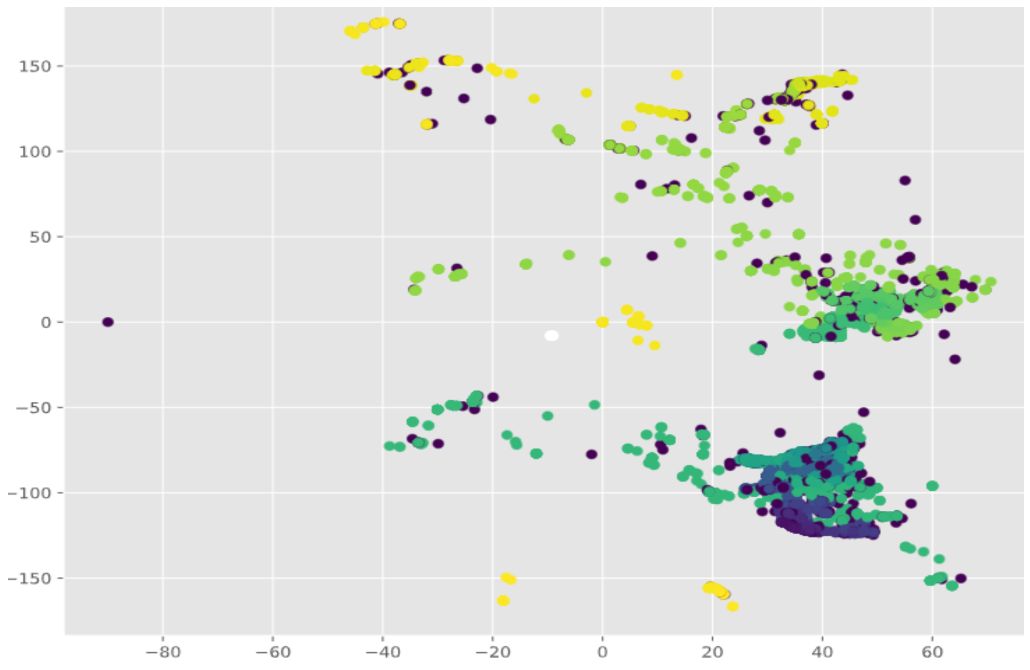
```
Calinski ignoring outliers: 971943.1093159069
Calinski outliers as singletons: 726856.1953551086
```

- **Calculation of Davies Bouldin score of OPTICS**

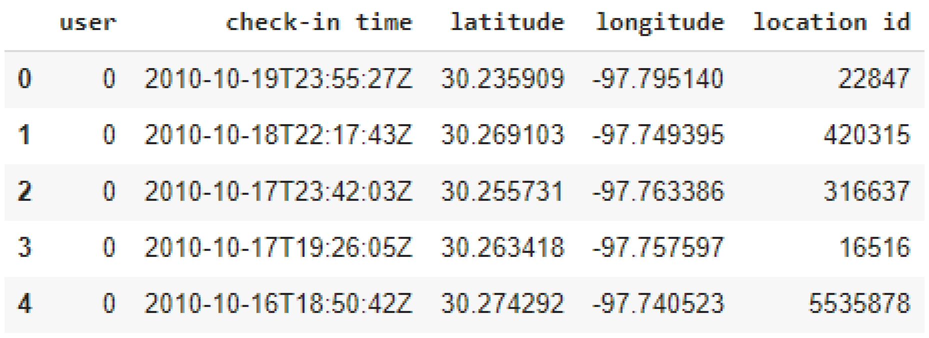
```
print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predict

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Davies Bouldin ignoring outliers: 0.33043150147184736
Davies Bouldin as singletons: 0.29516973191501505
```

- **Scatterplots of clusters in OPTICS**

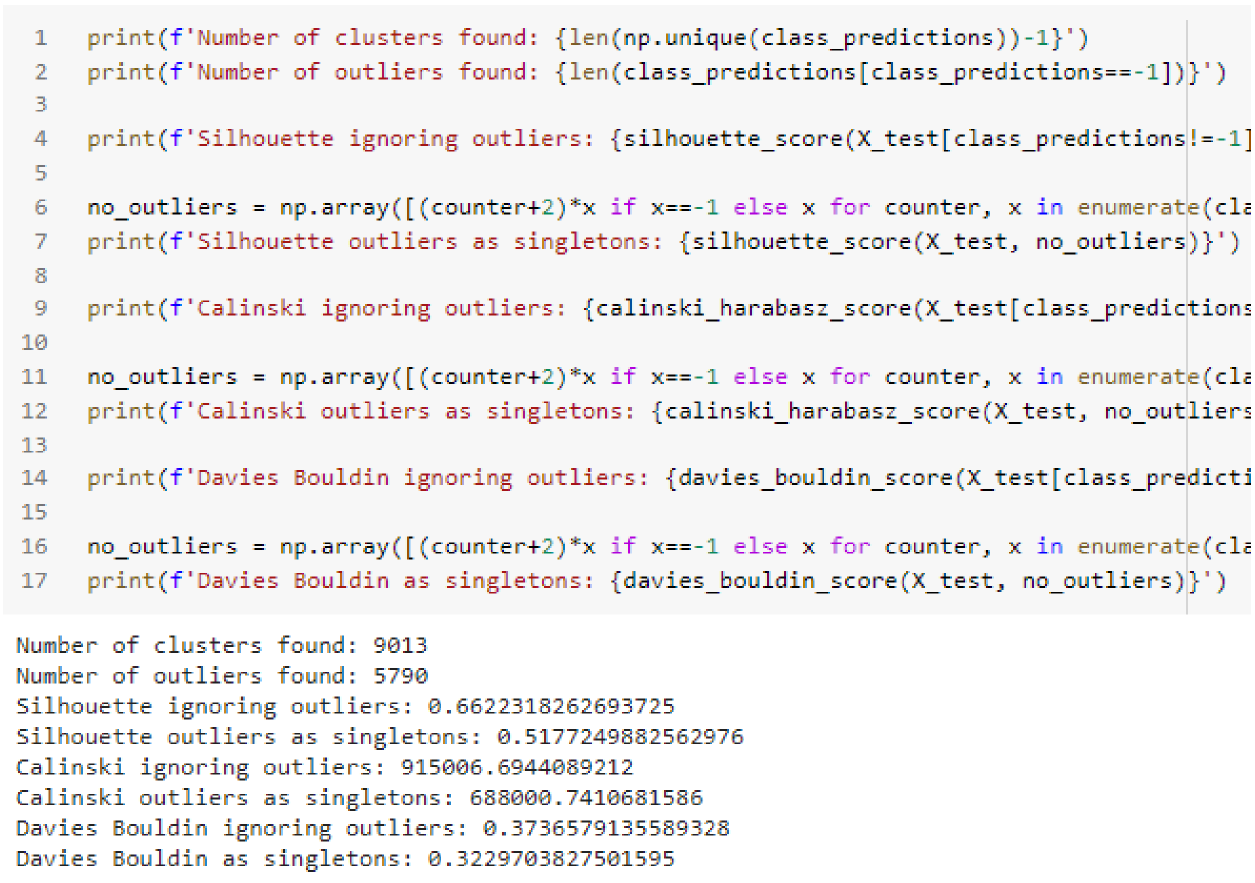


- **Loading Gowalla dataset**



	user	check-in time	latitude	longitude	location id
0	0	2010-10-19T23:55:27Z	30.235909	-97.795140	22847
1	0	2010-10-18T22:17:43Z	30.269103	-97.749395	420315
2	0	2010-10-17T23:42:03Z	30.255731	-97.763386	316637
3	0	2010-10-17T19:26:05Z	30.263418	-97.757597	16516
4	0	2010-10-16T18:50:42Z	30.274292	-97.740523	5535878

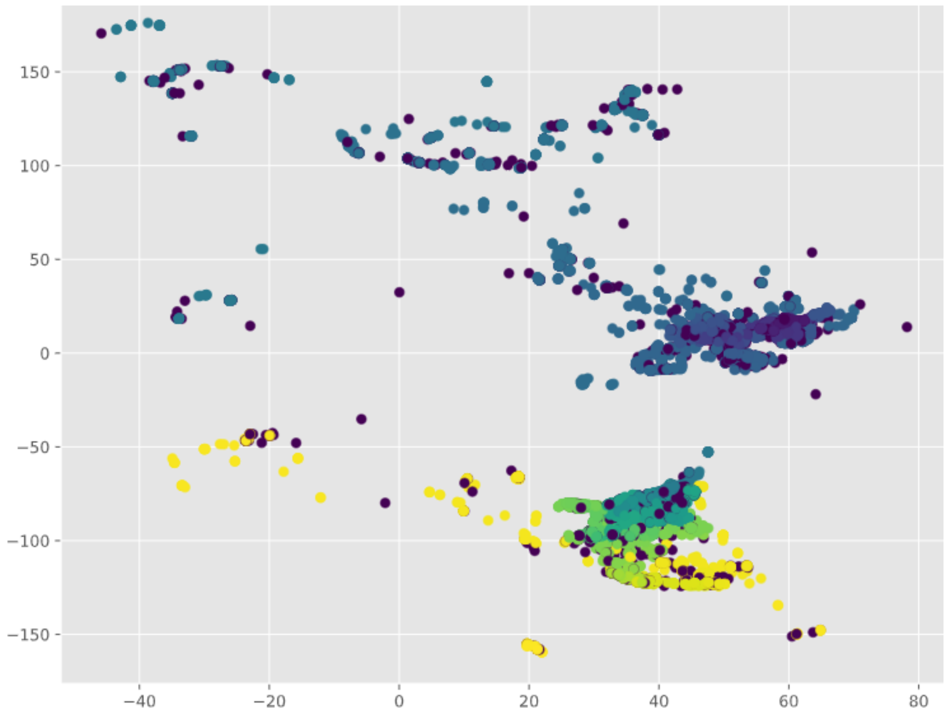
- **Calculation of number of clusters and Silhouette score, Calinski Harabasz score, and Davies Bouldin score of clusters in OPTICS**



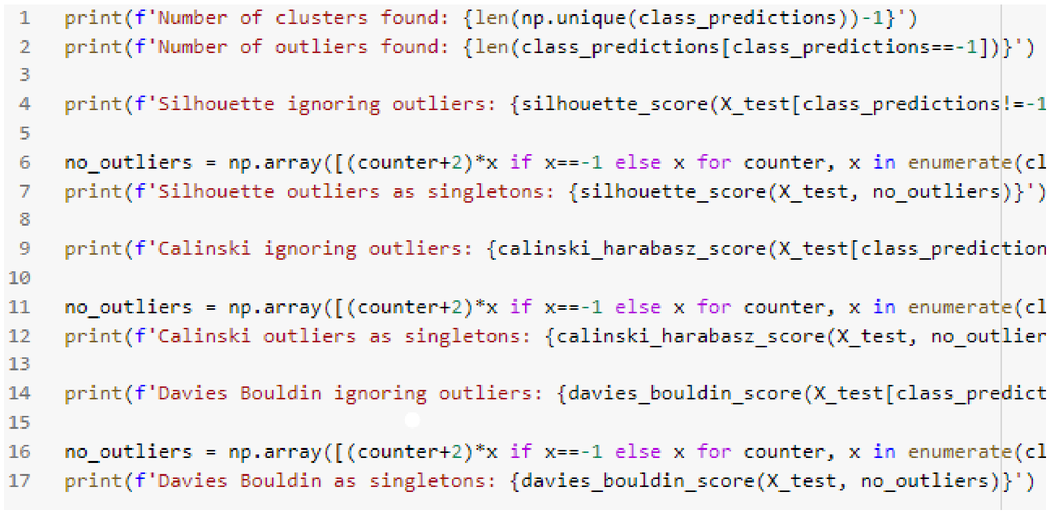
```
print(f'Number of clusters found: {len(np.unique(class_predictions))-1}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1]

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cla
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')

print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_predictions

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cla
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outliers

print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predicti

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cla
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Number of clusters found: 9013
Number of outliers found: 5790
Silhouette ignoring outliers: 0.6622318262693725
Silhouette outliers as singletons: 0.5177249882562976
Calinski ignoring outliers: 915006.6944089212
Calinski outliers as singletons: 688000.7410681586
Davies Bouldin ignoring outliers: 0.3736579135589328
Davies Bouldin as singletons: 0.3229703827501595
```

- **Scatterplots of clusters in OPTICS (Gowalla dataset)**



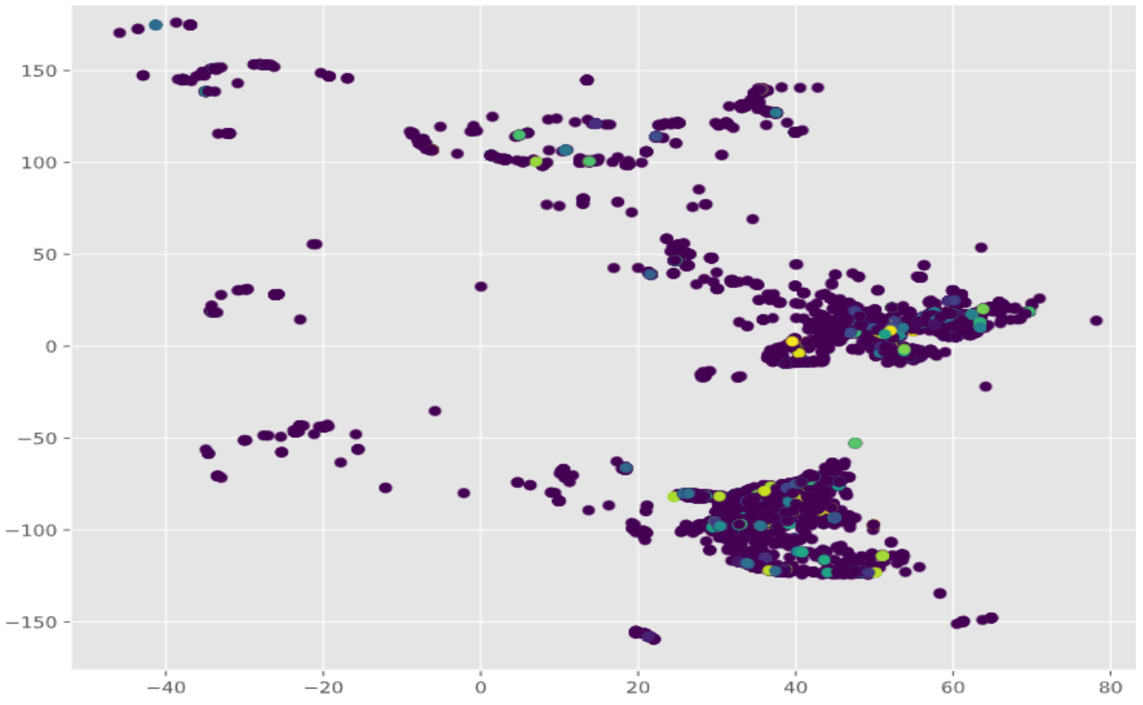
- **Calculation of number of clusters and Silhouette score, Calinski Harabasz score, and Davies Bouldin score of clusters in DBSCAN**



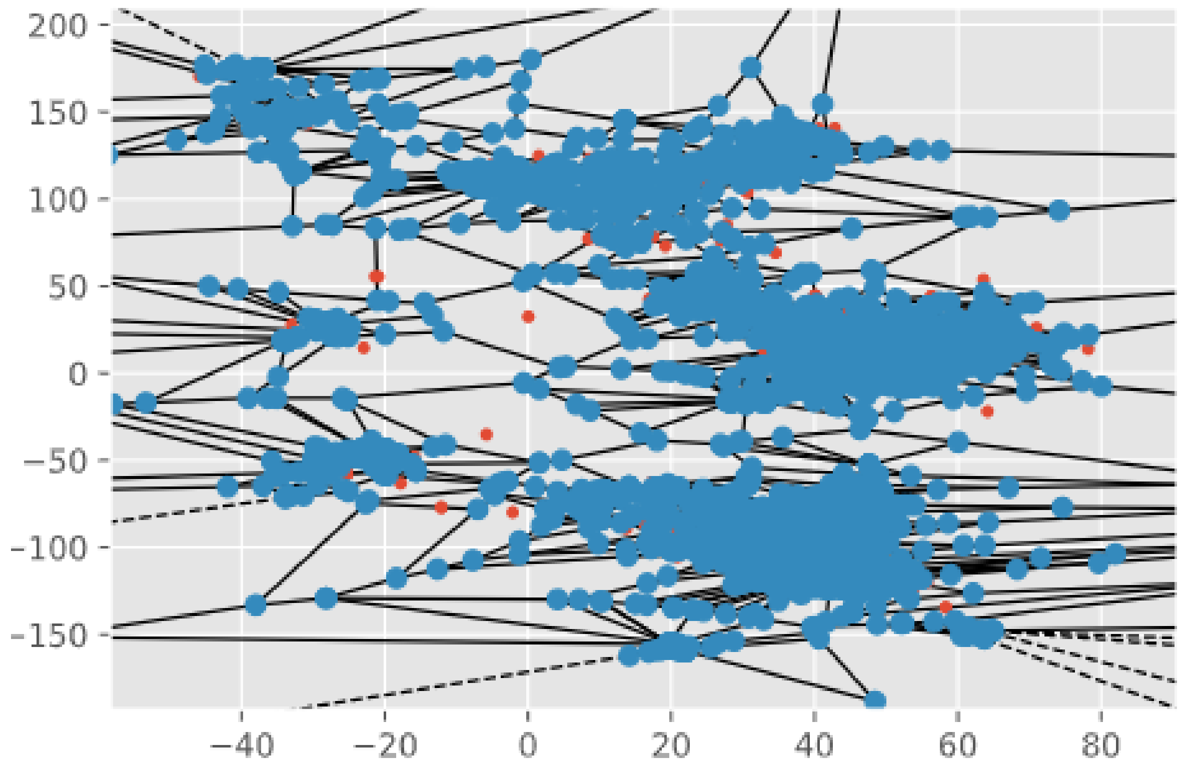
```
print(f'Number of clusters found: {len(np.unique(class_predictions))-1}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(X_test[class_predictions!=-1

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Silhouette outliers as singletons: {silhouette_score(X_test, no_outliers)}')

print(f'Calinski ignoring outliers: {calinski_harabasz_score(X_test[class_prediction

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Calinski outliers as singletons: {calinski_harabasz_score(X_test, no_outlier

print(f'Davies Bouldin ignoring outliers: {davies_bouldin_score(X_test[class_predict

no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(cl
print(f'Davies Bouldin as singletons: {davies_bouldin_score(X_test, no_outliers)}')
```

```
Number of clusters found: 770
Number of outliers found: 16560
Silhouette ignoring outliers: 0.5952262243383366
Silhouette outliers as singletons: 0.13146048379479877
Calinski ignoring outliers: 229765407.89558798
Calinski outliers as singletons: 22426541.654199768
Davies Bouldin ignoring outliers: 0.24912855423904723
Davies Bouldin as singletons: 0.11772929911226311
```

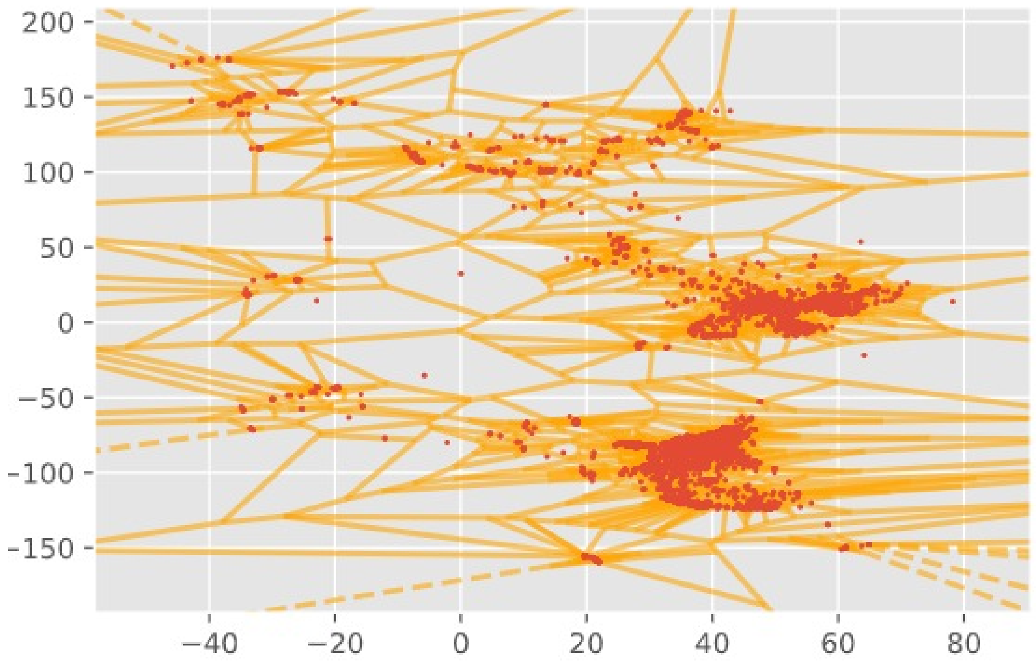
- **Scatterplots of clusters in DBSCAN (Gowalla dataset)**



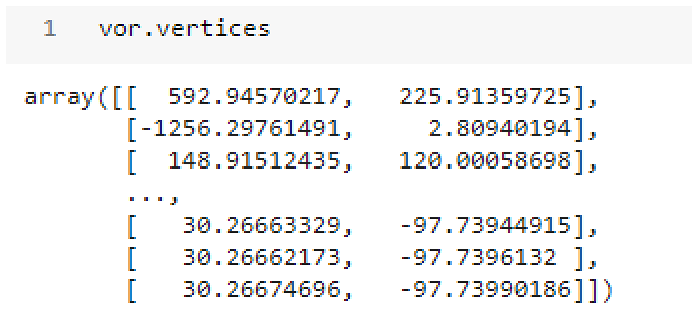
- **Spatial Voronoi diagrams for Gowalla dataset**



- **Spatial Voronoi diagrams for Gowalla dataset**



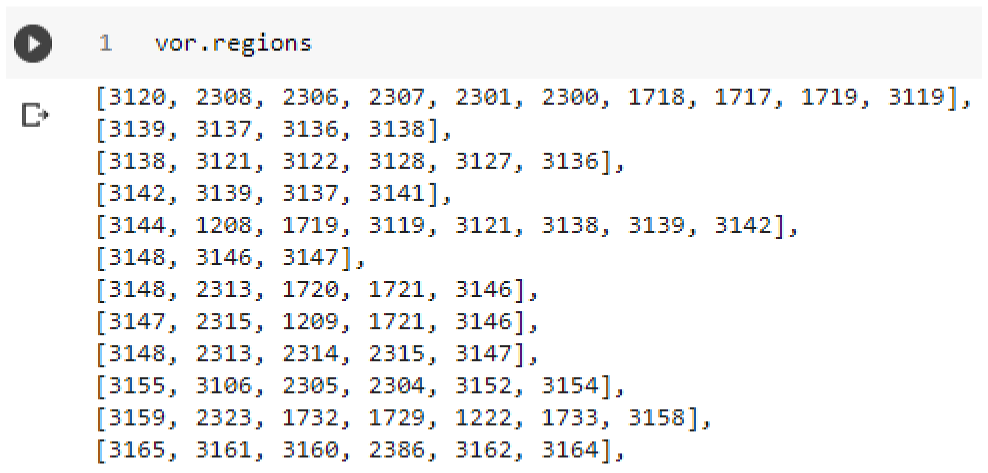
- **Spatial Voronoi vertices for Gowalla dataset**



```
vor.vertices
```

```
array([[  592.94570217,   225.91359725],
       [-1256.29761491,     2.80940194],
       [  148.91512435,   120.00058698],
       ...,
       [   30.26663329,   -97.73944915],
       [   30.26662173,   -97.7396132 ],
       [   30.26674696,   -97.73990186]])
```

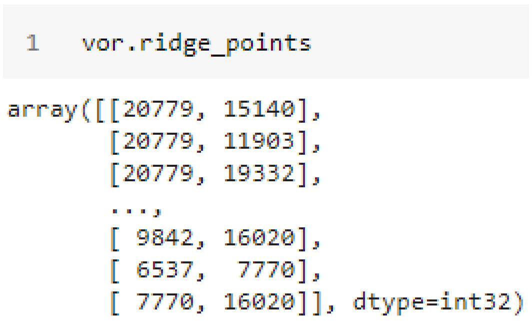
- **Spatial Voronoi regions for Gowalla dataset**



```
vor.regions
```

```
[3120, 2308, 2306, 2307, 2301, 2300, 1718, 1717, 1719, 3119],
[3139, 3137, 3136, 3138],
[3138, 3121, 3122, 3128, 3127, 3136],
[3142, 3139, 3137, 3141],
[3144, 1208, 1719, 3119, 3121, 3138, 3139, 3142],
[3148, 3146, 3147],
[3148, 2313, 1720, 1721, 3146],
[3147, 2315, 1209, 1721, 3146],
[3148, 2313, 2314, 2315, 3147],
[3155, 3106, 2305, 2304, 3152, 3154],
[3159, 2323, 1732, 1729, 1222, 1733, 3158],
[3165, 3161, 3160, 2386, 3162, 3164],
```

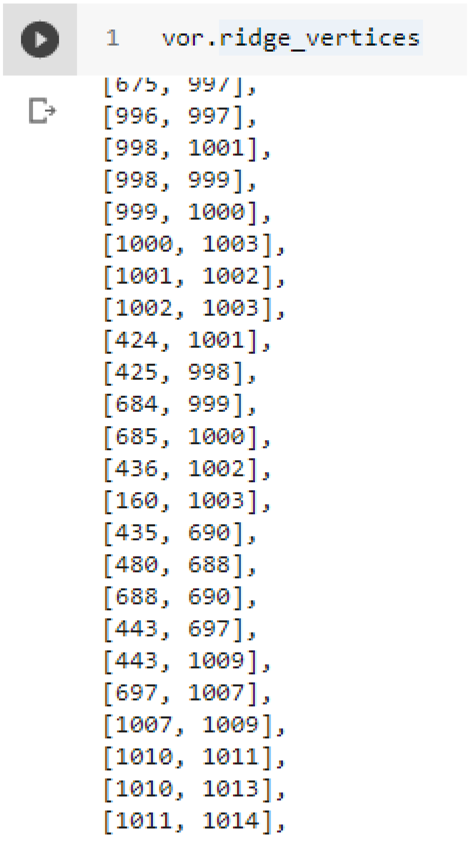
- **Spatial Voronoi ridge points for Gowalla dataset**



```
vor.ridge_points
```

```
array([[20779, 15140],
       [20779, 11903],
       [20779, 19332],
       ...,
       [ 9842, 16020],
       [ 6537,  7770],
       [ 7770, 16020]], dtype=int32)
```

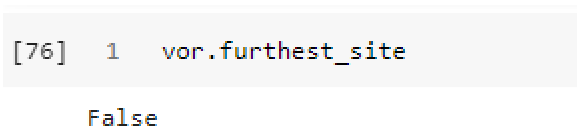
- **Spatial Voronoi ridge vertices for Gowalla dataset**



```
vor.ridge_vertices
```

```
[675, 997],
[996, 997],
[998, 1001],
[998, 999],
[999, 1000],
[1000, 1003],
[1001, 1002],
[1002, 1003],
[424, 1001],
[425, 998],
[684, 999],
[685, 1000],
[436, 1002],
[160, 1003],
[435, 690],
[480, 688],
[688, 690],
[443, 697],
[443, 1009],
[697, 1007],
[1007, 1009],
[1010, 1011],
[1010, 1013],
[1011, 1014],
```

- **Spatial Voronoi furthest site for Gowalla dataset**



```
[76]  vor.furthest_site
```

```
False
```

5. Conclusion

According to implementation done using python, the following observations are made for the Bright Kite dataset.

	K-MEANS	BIRCH	HDBSCAN	DBSCAN	OPTICS
No. of clusters	3	875	1485	785	5463
No. of outliers	0	0	3801	11834	3076
Silhouette outliers as singletons	0.75105830	0.65635	0.45963	0.298800	0.6747
Silhouette score (ignoring outliers)	0.75105830	0.65635	0.6548991	0.7661951	0.79132
C.H. score outliers as singletons	215308.632	2869745	151607.63	68803410	726856.1
C.H. score (ignoring outliers)	215308.632	2869745	456389.347	577981985	971943.109
DB index outliers as singletons	0.4798002	0.38315	0.3548516	0.1240310	0.29516
DB index (ignoring outliers)	0.4798002	0.38315	0.42046	0.1465370	0.33043

According to the above observation

1. Algorithm with the highest number of clusters = OPTICS
2. Algorithm with lowest numbers of clusters = K-means
3. Algorithm with highest Silhouette score = OPTICS
4. Algorithm with lowest Silhouette score = HDBSCAN
5. Algorithm with highest Calinski Harabasz score = DBSCAN

6. Algorithm with lowest Calinski Harabasz score = K-means
7. Algorithm with highest Davies Bouldin index = K-means
8. Algorithm with lowest Davies Bouldin index = DBSCAN

Based on the above observations on the Bright Kite dataset we can conclude that,

- Based on the Silhouette score OPTICS Algorithm performs best
- Based on Calinski Harabasz score DBSCAN Algorithm performs the best.
- Based on Davies Bouldin score DBSCAN Algorithm performs the best.

Observations are made for the Gowalla dataset.

	DBSCAN	**OPTICS**
No. of clusters	770	9013
No. of outliers	16560	5790
Silhouette outliers as singletons	0.13146	0.5177
Silhouette score (ignoring outliers)	0.59522	0.66223
C.H. score outliers as singletons	22426541	688000
C.H. score (ignoring outliers)	229765407	915006
DB index outliers as singletons	0.11772	0.32297
DB index (ignoring outliers)	0.24912	0.37365

Based on the above observations on the Gowalla dataset we can conclude that,

- Based on the Silhouette score OPTICS Algorithm performs best
- Based on Calinski Harabasz score DBSCAN Algorithm performs the best.
- Based on Davies Bouldin score DBSCAN Algorithm performs the best.

6. References

1. Kim, Jungeun & Lee, Jae-Gil & Lee, Byung & Liu, Jiajun. (2020). Geosocial Co-Clustering: A Novel Framework for Geosocial Community Detection. ACM Transactions on Intelligent Systems and Technology. 11. 1-26. 10.1145/3391708.
2. D. Wu, J. Shi, and N. Mamoulis, "Density-Based Place Clustering Using Geo-Social Network Data," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 838-851, 1 May 2018, DOI: 10.1109/TKDE.2017.2782256.
3. L. McInnes, J. Healy, S. Astels, *hdbscan: Hierarchical density-based clustering* In Journal of Open Source Software, The Open Journal, volume 2, number 11. 2017
4. Wikipedia contributors. "OPTICS algorithm." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 1 Feb. 2021. Web. 24 Apr. 2021.
5. Wikipedia contributors. "Voronoi diagram" *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 23 April 2021.
6. Wikipedia contributors. "BIRCH." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 25 Feb. 2021. Web. 25 Apr. 2021.