INSTITUTO
UNIVERSITÁRIO
DE LISBOA

User Behaviour Identification based on location data

Miguel José Candeias Bento

Master in Information Systems Management

Supervisor:

PhD Luís Miguel Martins Nunes, Associate Professor
Iscte - Instituto Universitário de Lisboa

October, 2020

TECNOLOGIAS
E ARQUITETURA

Departamento de Ciências e Tecnologias da Informação (DCTI)

User Behaviour Identification based on location data

Miguel José Candeias Bento

Master in Information Systems Management

Supervisor:

PhD Luís Miguel Martins Nunes, Associate Professor
Iscte - Instituto Universitário de Lisboa

October, 2020

Acknowledgement

I am grateful to Prof. Dr. Luis Nunes for all his guidance, and support throughout this project. I would like to thank ISCTE – Instituto Universitário de Lisboa for making me feel truly at home. A huge thanks goes to a remarkable group of friends who emotionaly supported me and have made the last years of my life the best. In particular, I would like to thank all the members of 3 teams to which I belong, Malaikes do Sado, CMTV Corona News and Sado eSports. A special thanks goes out to Tesseract, Think Music, Breaking Benjamin, Asking Alexandria and Ariana Grande for supplying the ideal soundtrack. None of this work would have been possible, without the love and support of my family. They have always provided me with necessary help and a series of opportunities, for which I am forever grateful. They have been the pillars of all my accomplishments, and their endless belief and encouragement have inspired me in each and every aspect of my life.

Resumo

Ao longo dos anos tem-se verificado um aumento quase exponencial no que toca à utilização de novas tecnologias em vários sectores. Estas tecnologias têm como objetivo principal, melhorar ou facilitar o quotidiano.
O presente estudo vai incidir sobre uma destas tecnologias utilizada dentro de um tema que tem sido muito falado nos últimos anos, a utilização de dados pessoais de um grupo de indvíduos para identificar certos tipos de comportamentos. Mais concretamente, tem como objetivo utilizar os dados de GPS, guardados nas respectivas contas Google de nove voluntários, de modo a identificar os locais que estes mais frequentam - Pontos de Interesse. Os dados são utilizados também para identificar as trajectórias percorridas mais vezes por cada um dos voluntários.
Foi realizado um estudo com uma amostra de 9 participantes, enviando-lhes os respectivos mapas com POI e trajectórias obtendo assim a validação dos mesmos.
Desta forma foi possível concluir que que a melhor forma de identificar POI tem como base a utilização de clusters diários utilizando DBSCAN. Para o caso das trajectórias, o método Snap-to-Road foi o que originou melhores resultados.
Verificou-se que foi possível responder ao problema inicial, desta forma, foi encontrado um método que identifica a maior parte dos POI com sucesso, bem como algumas trajetórias. Com base neste trabalho, existe uma oportunidade para futuramente melhorar alguns dos algoritmos e processos que possuem algumas limitações de modo a desenvolver soluções mais eficazes.

Palavras-chave: Padrões, Inteligência Artificial, GPS

Abstract

Over the years there has been an almost exponential increase in the use of new technologies in various sectors. These technologies have as their main objective, to improve or facilitate our daily life.
This study will focus on one of these technologies used within a theme that has been widely talked about over the last few years, the use of personal data of various people to identify certain types of behavior. More specifically, this study aims primarily to use the GPS data stored in the respective Google accounts of nine volunteers in order to identify the places they frequent most, also known as Points of Interest. This same data will also be used to identify the trajectories covered more often by each of the same volunteers. A study was carried out with a sample of 9 participants, sending them their maps with POI and trajectories, thus obtaining their validation.
It was thus possible to conclude that the best way to identify POI is to use daily clusters using DBSCAN. In the case of trajectories, the Snap-to-Road method was the one that gave the best results.
It was found that it was possible to respond to the initial problem, and thus a method was found that identifies most of the POI successfully and also some trajectories.Based on this work, there is a great opportunity to improve some of the algorithms and processes that have some limitations in the future, and with this in mind it's possible to develop more effective solutions.

Keywords: Patterns, Artificial Intelligence, GPS

Table of Contents

Table List

Table of Figures

List of Abbreviations

AI – Artificial Intelligence

CLIQUE - Clustering in QUEst

DBRS - Density-Based Spatial Clustering Method with Random Sampling

DBSCAN - Density-based spatial clustering of applications with noise

DBSCAN-DLP - Multi-density DBSCAN Algorithm Based on Density Levels Partitioning

DD_DBSCAN - Density Difference DBSCAN

DDSC - Density Differentiated Spatial Clustering

DMDBSCAN – Dynamic Method DBSCAN

DVBSCAN - Density Variation Based Spatial Clustering of Applications with Noise

EDBSCAN - Enhanced Density Based Spatial Clustering of Applications with Noise

GADAC - The Genetic Algorithm with a Density-Based Approach for Clustering

GD-DBSCAN - Grid and Kd-tree for DBSCAN

GPS – Global Positioning System

Habit – Habit Analytics

HDBSCAN – Hierarchical DBSCAN

HDBSCAN - Hierarchical Density-based spatial clustering of applications with noise

IDBSCAN - Improved Sampling-Based DBSCAN

KDDClus - KD-tree data clustering

LDBSCAN - local-density based spatial clustering algorithm with noise

MDBSCAN – MST DBSCAN

MR-DBSCAN - MapReduce-based DBSCAN

OPTICS - Ordering points to identify the clustering structure

PDBSCAN – Parallel DBSCAN

POI - Points of Interest

ST-DBSCAN – Spatial-Temporal DBSCAN

VDBSCAN – Varied Density Based Spatial Clustering of Applications with Noise

1 Introduction

1.1 Research Theme

Over the years, there has been a great increase in technology in various sectors with the aim of improving people's lives.
One of the ways in which technology has substantially changed people's lives has been the implementation of GPS in both vehicles and smartphones, thus revolutionising the way we move today, making it possible to collect various location data that can be used for a multitude of everyday applications.
Some of these applications and platforms are using GPS location data from these devices together with Artificial Intelligence algorithms to study and analyze certain user behaviors and habits.
Some more specific cases of this type of user behavior studies can be exemplified through the use of mobile GPS location data for the purpose of identifying both general points of interest and user-only points of interest (POIs), and also quite important, the identification of regular user routes (Ashbrook & Starner, 2003b). This analysis and study of behavior makes it possible, for example, to improve the advertising audience for the benefit of the user, by recommending restaurants or other establishments in the most frequented areas (Yaqiong Liu & Seah, 2015). Widely recognized and also increasingly used is the implementation of this technology for warning of accidents on the road or routes where there is the most traffic at that time, so the application can thus present a faster alternative to the destination compared to the route originally planned (Hanan, Idris & Kaiwartya, 2017).
As mentioned above, this use of GPS location data for real life is becoming increasingly apparent, so it is almost mandatory that this type of algorithms and data analysis technologies continue to evolve proportionally.
Consequently, this project is based on a number of aspects mentioned above, including the identification of user-only points of interest and regular trajectories. The analysiss was carried out using GPS location data from smartphones, specifically, the data of the author and one more volunteer for training, and data of 9 volunteers for testing.
This dissertation will provide a review of the published methods used to solve the problem of how to identify POI and trajectories, the knowledge of how they can be used, what limitations and strengths they have and, if possible, will contribute to solve some of the problems presented by the published algorithms.
As mentioned above, the GPS data used comes from the author and volunteers, and considering all the analysis that will have to be performed using this data. The research questions raised by this challenge are: To what extent does the GPS data available in most Google accounts allows us to identify POI and trace trajectories? How can we evaluate the efficiency of the published algorithms used to solve similar problems? What are their limitations and strengths?
In the evaluation phase of the results, several maps containing the relative POI of each user are extracted from the algorithms as well as their trajectories, and then sent to them for evaluation and validation. In the end, it was possible to identify the algorithm with the best results, and to propose some changes to improve the results in the future.

It should be noted that the images in map format presented in this thesis come exclusively from the author's own data, all other information has been anonymized so that no user data can be identified.
This project was inspired by a challenge presented by a company (Habit). On the company side, the objective is to investigate which solutions to the problem are already published, and to understand in a general context how these processes are currently being executed. This vision of how the systems currently work will allow the company to develop a potentially better solution to the problem.

1.2 Objectives

For the success of this dissertation, especially the experimental and prototype part, it is necessary to define some objectives.
The first objective and one of the most important is to realize if the data coming from Google give us the necessary values to make the data analysis, such as coordinates and timestamp. Another objective is to obtain a sufficient number of datasets in order to be able to draw a conclusion about the approaches. It is also necessary to perform an efficient literature search in order to learn what has already been done and what can be improved. To make the prototype it is necessary to decide the approaches to be carried out and through tests to reach a solution.
The solution consists of presenting a map with the POI and several other maps with each of the regular routes.
This solution must be evaluated by several volunteers by sending them the result from their data. Finally, it is necessary to have an analysis and reflection of the results to conclude if all the objectives were successfully accomplished.

1.3 Outline of the Thesis

The following study is divided into five chapters:

- First chapter: introduces the theme of the research and its objectives as well as a brief description of the work structure.
- Second chapter: The Literature Review, reviews recent work on this subject.
- Third chapter: explains the theoretical context and the Methodology used in the process of data gathering and treatment as well as the methods of analysis applied.
- Fourth chapter: discusses the experiments made and its results.
- Fifth and last chapter presents the conclusions of this analysis as well as the recommendations, limitations, and future work.

2 Literature Review

In the world we live in, it is possible to observe various types of technology that we did not even imagine that would exist some years ago, such as smartphones, self-driving cars, smart homes, smart appliances, etc. All this technology can generate one of the most valuable things in existence now - data. But data, even in large quantities, is useless if it is not possible to extract knowledge from it, as (Brock and von Wangenheim 2019) said:

"We have access to a vast quantity of data but it's hard to extract meaningful information that helps us improve the quality of the care we provide."

This extraction of knowledge from data is increasingly possible due to the great evolution in the area of artificial intelligence which, according to (Haenlein and Kaplan 2019), can be defined by:

"a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation."

There are numerous examples where AI has proved to be of great value, for example in services such as UBER to estimate travel time considering factors such as traffic and time, and thus be able to decide which car will perform this service. (Ghahramani 2019).
Also widely used and currently already embedded in most smartphones and cars are the GPS applications with traffic monitoring. These applications are able to analyze GPS data and determine points of interest, both individual and from general population. Considering the destination location of each user, the application uses artificial intelligence algorithms to calculate the fastest route considering also traffic on the road (Herrera et al. 2010).

2.1 User Behavior, Points of Interest (POI) and Paths

As mentioned (Paola et al. 2019), understanding user's behaviors and habits is critical to the development of advanced services such as personalized recommendation and virtual assistance.
Therefore, this research project will focus mainly in this area, that is, how to successfully detect points of interest and usual routes using GPS data.
But what are points of interest? There are two types of POI, the global ones well described by Angkhawey and Muangsin as popular places that create activities and transport needs, places that should be on the map (Angkhawey and Muangsin 2018) and individual, points where a certain person stops regularly. These are the points on which the work is focused in order to associate them with activities to know which are the regular ones of the individual.
A study conducted by (Gonzalez, Hidalgo, and Barabasi 2008) proved that the usual human routes have a great regularity in terms of space and time, and, each person has a significant probability of returning to some of the most frequent locations.
In practical terms, how will these POI be computed? The GPS sensor will record coordinate data each time users move around, so does it make sense to use each of these points to find POI? According to (Fu et al. 2016) and (Montoliu, Blom, and Gatica-Perez 2013), although they use different terms, they consider that there are location points, stay points and stay regions (Figure 1):

- A location point is considered a measurement provided by the GPS sensor containing the coordinates, date and time of the day it was recorded.
- A stay point is considered a cluster of points of location of the same day that represent a geographic location where the user stayed some time.
- A stay region can be considered a synonym of POI by some authors, and consists of a cluster of stay points of several days.

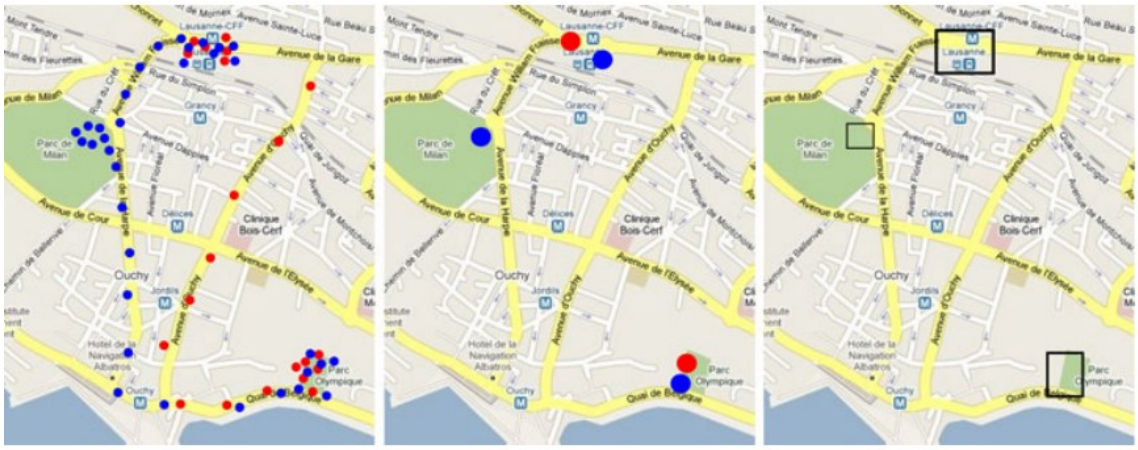

Figure 1-Types of location data by (Montoliu et al., 2013)

The authors (Fu et al. 2016) also define the concept of trajectory (or path), which consists of a list of daily GPS points containing the coordinates and temporal data (Figure 2).

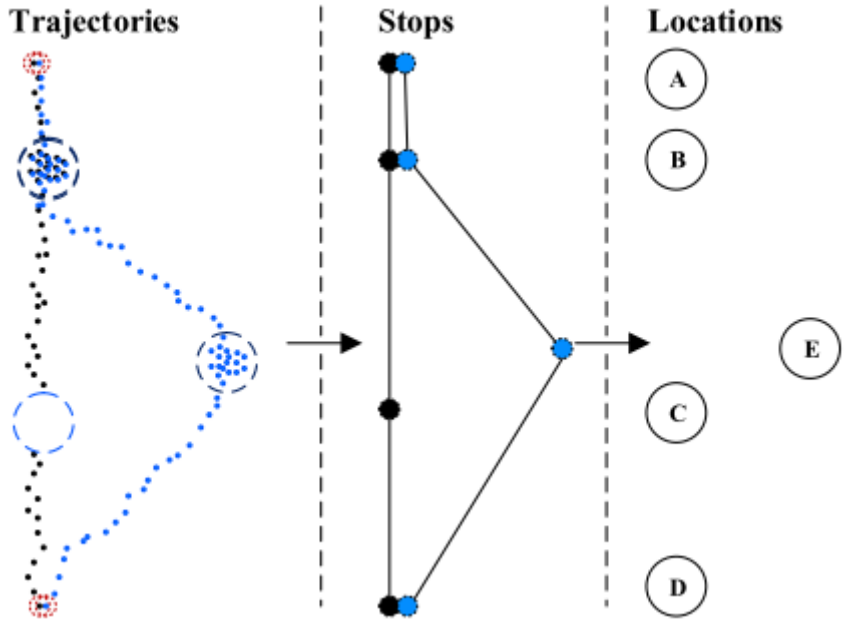

Figure 2-Trajectories, Stops and Locations by (Fu et al., 2016)

Through these clusters it is then possible to extract the locations where the user has spent considerable time and their routes using temporal data. (Angkhawey and Muangsin 2018;

Fu et al. 2016; Montoliu, Blom, and Gatica-Perez 2013). The clustering algorithms that are relevant for this research are briefly explained in Methodology chapter.
For some authors like Liu and Seah, POI can also be obtained separating clustering into three categories: partitioning clustering, time-based clustering and density-based clustering.
Although mapping known POIs through the user's position is a relatively simple task (Nishida et al. 2014; Shaw et al. 2013), projects that do not have a previous set of known POIs suffer from an initial cold-start problem.
According to (Montoliu, Blom, and Gatica-Perez 2013), there are two major methods for mapping locations using data: geometry-based and fingerprint-based.
The geometry-based method consists of using GPS location data to produce coordinates, circles, or polygons to identify locations that the user frequents. These algorithms use the user's location point history and look for locations where the user has stayed for significant periods of time using a clustering-based technique. The fingerprint-based method, in contrast to the previous one, does not use GPS data but rather communication with telephone towers and Wi-Fi points, so it will not be relevant for the development of this project.
In another study conducted by (Cao, Cong, and Jensen 2010), locations were identified using an improved version of the clustering algorithm OPTICS (Ordering points to identify the clustering structure) which basically performs several split-and-merge steps until each cluster represents only one location.
Another study by (Ashbrook and Starner 2003a) focused more on the time difference between consecutive GPS points, thus detecting a Stop point based on a defined time interval, i.e., a location. This happens because the GPS signal is lost inside buildings so a crash in the signal for some time means a location.
In this study, the authors had two problems in the detection of POI, location errors and the presence of high-density POI's that originate a great difficulty in the detection of POI already visited that does not always coincide with the POI where the user stopped for a while.
To combat the problem of Stop Point discovery, the authors present the approach of (Montoliu, Blom, and Gatica-Perez 2013) which identifies a Stop Point when there are 2 measurement points "Pa" and "Pb" that satisfy the following three premises:

$$SpaceDistance(Pa, Pb) < \delta d, \quad (1)$$

$$TimeDifference(Pa, Pb) > \delta t, \quad (2)$$

$$TimeDifference(Pk, Pk+1) < \Delta d, \forall k: a \leq k \leq b, \quad (3)$$

δd is the maximum distance a user can travel to be considered as still being in the same location, δt is the minimum visiting time for the user to be considered as a location, and Δd is the maximum time interval between two measurements to be considered part of the same cluster.
One disadvantage of this method is that it cannot handle signal losses (Paola et al. 2019), so the authors (Fu et al. 2016) proposed an approach to solve this problem. This approach consists in focusing on the premise of maximum time, and any point that violates this restriction is stored in a temporary buffer to be compared with the next Stop Point. If the distance to the previous point is less than the set value, these points are merged with all

those in the buffer. Even so, this approach generates too many Stop Points (Paola et al. 2019), so the authors propose to exceed this limitation by comparing the Stop Points with information saved in a database with known waypoints, so it is possible to join several Stop Points that correspond to the same waypoint.
To combat the problem of difficult POI detection already visited, it is possible to use the nearest neighbor method (Cao, Cong, and Jensen 2010; Nishida et al. 2014). The disadvantage of this method is that it is affected by GPS errors and by areas with high POI density, so to overcome this limitation, some methods have been proposed by several authors, (Shaw et al. 2013) proposes a spatial search system method composed of a phase in which a set of POI is extracted and another phase in which they are evaluated. The first phase selects the most popular points close to the user's location and to evaluate them a machine learning algorithm is used which is trained using user feedback.
Another approach taken by (Lian and Xie 2011) is to identify a set of POI just by checking that its distance to the Stop Point is less than a defined value. Nishida proposes to use a generative model to distinguish already visited POIs from unidentified Stop Points through a Bayesian network that represents the probabilistic dependence of different factors of POIs already visited, namely user preferences, duration, popularity and stop time(Nishida et al. 2014).
Other strategy adopted by (Paola et al. 2019) is to implement a Dynamic Bayesian Network (Parsons 2011) in order to include context, knowledge of past history in the POI detection algorithm and deal with noise from sensors and POI already visited (Figure 3).

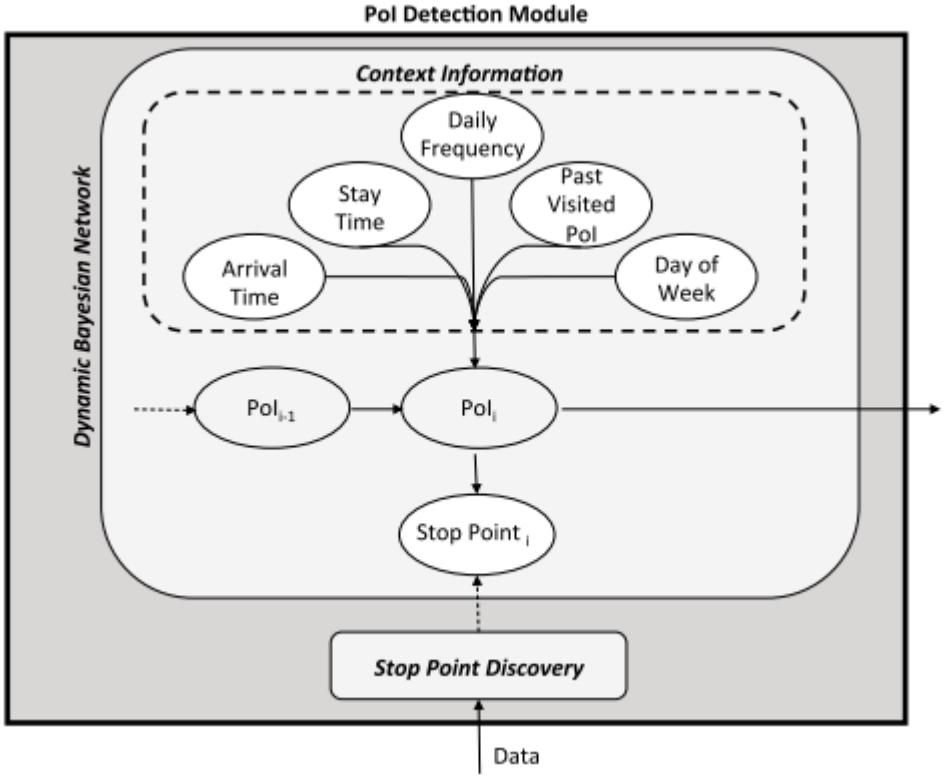

Figure 3-Block Diagram of the PoI Detection module by (Paola et al., 2019)

One of the most popular Clustering algorithms, mentioned by most of the clustering approaches to this problem, is DBSCAN (Density-Based Spatial Clustering of Application with Noise) first presented in 1996 (Ester et al. 1996).

Bearing in mind that this algorithm is quite sensitive to the arguments, the authors (Schubert et al. 2017) suggest values that make the initialization of the algorithm parameters more appropriate. Details on these parameters are given in chapter 3, when the algorithm is presented in more detail.
Taking this into consideration, this algorithm still does not work properly when the dataset has varying densities in different places because the parameters are not suitable for all clusters(Angkhawey and Muangsin 2018).To solve this problem, several variations of this algorithm were proposed by several authors, the most important ones can be seen in Table 1.

Table 1-Proposed Algorithms to solve DBSCAN varying densities problem, other variations found. in total 23 in Annex 6.1

Algorithms	Experiments	Results	References
AutoEpsDBSCAN	Pre select different Eps values for different densities according to a k-dist plot	AutoEpsDBSCAN is able to find clusters with varying densities, better result than DBSCAN , VDBSCAN and only requires one input parameter	(Gaonkar and Sawant 2013)
"Haversine" DBSCAN	Automatically determine the parameters of DBSCAN according to pick-up and drop-off locations of taxis in Bangkok	Better results than DMDBSCAN in all 3 types of areas (low, medium and high-density population)	(Angkhawey and Muangsin 2018)
OPTICS	Algorithm that creates an augmented ordering of the database representing its density-based clustering structure.	Suitable for interactively exploring the clustering structure, offering additional insights into the distribution and correlation of the data	(Ankerst et al. 1999)
HDBSCAN	Obtain a flat partition consisting of only the most significant clusters formulating the problem of maximizing the overall stability of selected clusters.	Experimental results on a wide variety of real-world data sets show better performance than other recent solutions to this problem.	(Campello, Moulavi, and Sander 2013)

DBSCAN's weakness lies in the delay of the neighbor-search function. Some authors presented suggestions for variations of the DBSCAN algorithm with better optimization in that aspect (Table 2).

Table 2- Proposed Algorithms to solve DBSCAN long search time problem

Algorithms	Experiments	Results	References
GD-DBSCAN	Improved DBSCAN algorithm which integrates partitioning method with kd-tree structure to improve its' computational performance	The experiment shows GD-DBSCAN has an improvement of at least 10% in performance compared with original fastest DBSCAN	(Zhang et al. 2016)
PDBSCAN	A parallel version of DBSCAN using the 'shared-nothing' architecture with multiple computers interconnected through a network.	The experiment evaluation shows that PDBSCAN has very good performance, speedup, scaleup and size up.	(Xu, Jäger, and Kriegel 1999)
MR-DBSCAN	Described in the previous table	Described in the previous table	(He et al. 2014)
IDBSCAN	Improved sampling-based DBSCAN which can cluster large-scale spatial databases effectively	Experimental results show that because of the sampling technique, the I/O cost and the memory requirement for clustering are reduced and a considerable amount of run-time is reduced	(B Borah and Bhattacharyya 2004)

Other authors decided to present GridBased clustering solutions due to its significant reduction of the computational complexity specially when datasets are large. Some of these are STING algorithm presented by(W. Wang, Yang, and Muntz 1997), DENCLUE for sparse datasets (Hinneburg and Keim 1998), WaveCluster for low dimensional data (Sheikholeslami, Chatterjee, and Zhang 1998), and CLIQUE for high dimensional data (Agrawal et al. 1998).

In summary, 5 approaches have been identified to address the problem.

- Spatial Clustering which consists of several found algorithms that cluster GPS location data.
- Spatial and Non-Spatial Clustering that also use time in the clustering process allowing the identification of the amount of time that the user remains in each region.
- Partitioning Clustering (K-means) which is fast but as it has a random component the results can be different and does not identify outliers.
- Grid Based Clustering with advantage in being faster especially in large datasets but sensitive to the initial parameters.
- Bayesian network that distinguishes known POI from unidentified POI, however this approach needs a prior POI database

3 Prototype

3.1 Data Analysis Strategy

As previously mentioned, the use of GPS data for various applications has been increasingly evident, especially in electronic devices used in everyday life or in services provided by companies and it is easy to obtain GPS data through google services on any android smartphone. This is the data that was used in this work to obtain information user's POI (points of interest) and regular trajectories.
In the previous chapter several solutions to this problem were identified, DBSCAN was selected for implementation as it was an algorithm mentioned by almost all the articles found as being the "basis" of all the solutions and achieving good results. Although most of the articles present a variation of DBSCAN to improve performance or to deal with varying densities, there is always the possibility that this variation will only be beneficial to the specific case of the author of the article, for this reason DBSCAN will be implemented in the first place. One of the DBSCAN variations will also be implemented in order to improve the problem of varying densities, HDBSCAN as it was one of the last to be published and with specific examples using location data available. In this chapter we will describe 4 algorithms, DBSCAN, HDBSCAN, KMEANS and CLIQUE used in the experiments and through (Figure 4), it is possible to understand in a simplified way, how the information is processed until it reaches the experimental stage. As mentioned above, the location data (symbolized by the google maps icon) is taken from google services and used in a Jupyter notebook, which will be processed and used together with the 4 algorithms to obtain the POI and the regular paths.

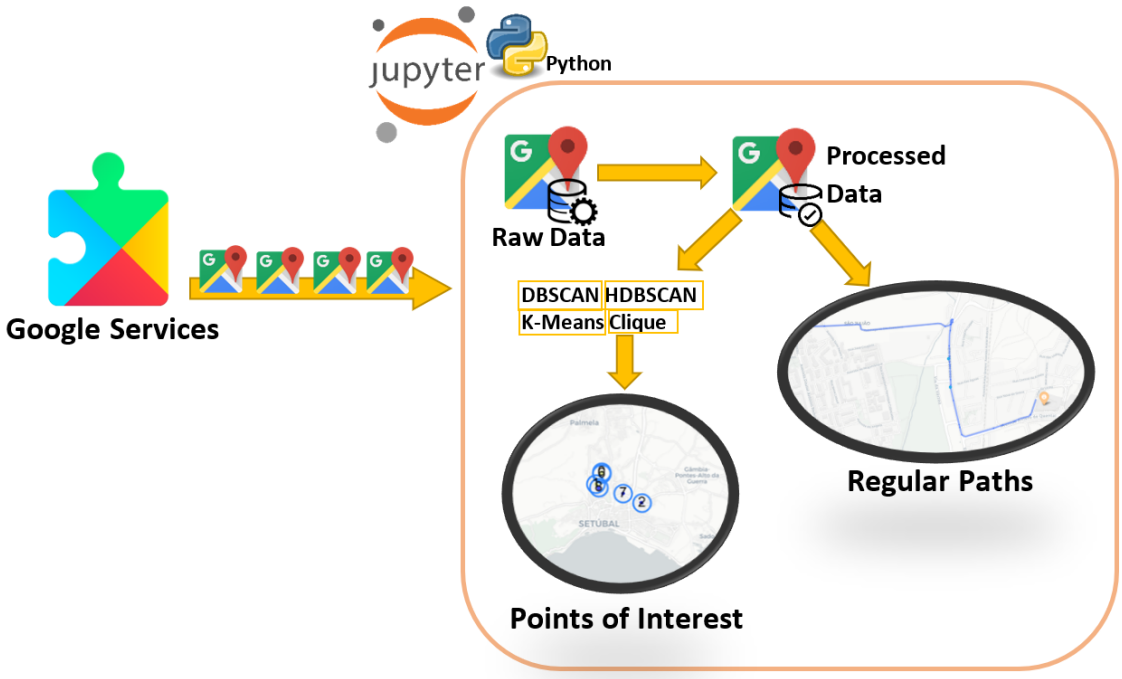

Figure 4- Simplified scheme of the process from data gathering to experiments

3.2 Clustering Algorithms

But what are clustering algorithms?
As explained in (Ashbrook and Starner 2003b), in a simplified way, what most clustering algorithms actually do is choosing a starting place and a radius, then mark all places within that radius and the average. This average becomes the central point and the process is repeated from the beginning again marking all locations within the radius, thus continuing until the average stops changing. When this happens, all points within the radius are placed in a cluster and removed from consideration, thus becoming a location. The process is repeated until there are no more places and only several locations, that is, several clusters.

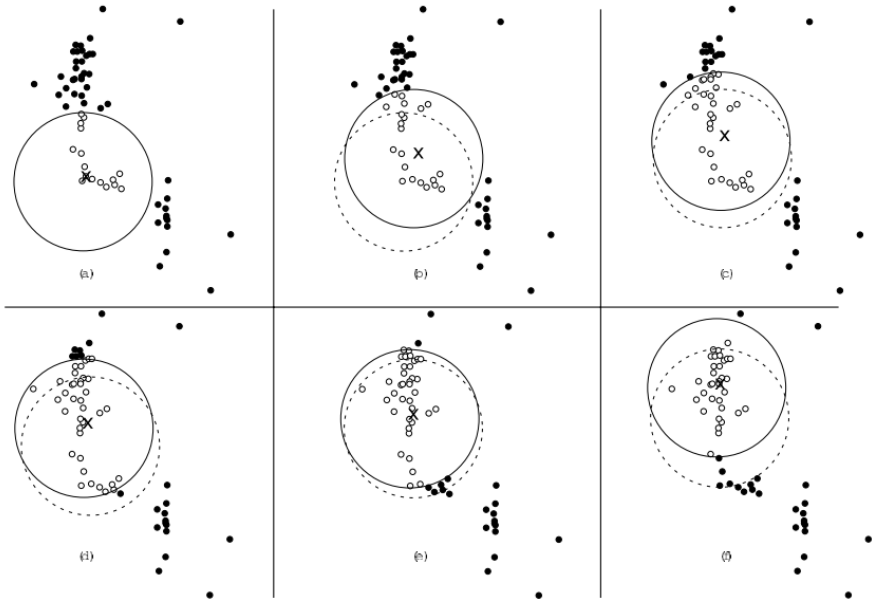

Figure 5-"Illustration of location clustering algorithm." by (Ashbrook & Starner, 2003b)

To better understand this process, the same authors presented an illustration (Figure 5) which is described as follows:" *The X denotes the center of the cluster. The white dots are the points within the cluster, and the dotted line shows the location of the cluster in the previous step. At step (e), the mean has stopped moving, so all of the white points will be part of this location.)"*(Ashbrook and Starner 2003b)

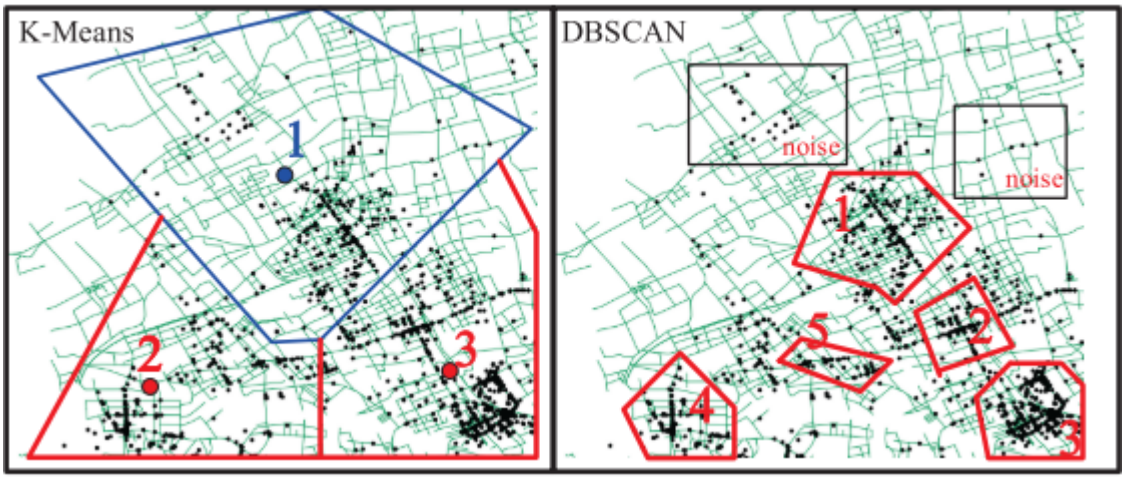

Figure 6-Comparison between K-Means and DBSCAN clustering results by (Zhang et al., 2016)

For Liu and Seah, (Yaqiong Liu and Seah 2015) to obtain POI through GPS data, they separated the clustering algorithms into three categories, partitioning clustering, time-based clustering and density-based clustering. Partitioning clustering is based on the K-means algorithm, which divides the total of points into several K-sets in order to minimize the quadratic sum of the distances from the points to the centroid of the cluster. Note that this algorithm does not solve the problem of noise and outliers, i.e. they remain in the results.
Figure 6 provides a comparison between K-means and DBSCAN (Zhang et al. 2016).
Time-based Clustering works by continuously using GPS coordinates and comparing them with the time intervals of that same data. Exemplifying, if the person stays in the same location longer than established, then the algorithm found a new location.
Density-based Clustering is a type of algorithm that has already been used by some authors, namely the DJ-Cluster algorithm (Zhou et al. 2007) and also DBSCAN (Ester et al. 1996; Milenova and Campos 2002; Zaïane et al. 2002) that will be explored next. This type of algorithms allows us to discover clusters with several formats and is very useful for problems that involve working with geographic data. It is also a good algorithm to deal with outliers and noise problems.
The first algorithm tested was the popular DBSCAN (Density-Based Spatial Clustering of Applications with Noise) first presented in 1996 and is one of the best known today (Ester et al. 1996). This algorithm focuses on point density by defining a cluster as a group of points that have their neighbors within a boundary distance as we can observe in Table 3 and Figure 7 (Peng, Dong, and Naijun 2007; Schubert et al. 2017). Each cluster must have central points with neighbors and points not belonging to any cluster are considered outliers.(Ester et al. 1996)

Table 3-Description of DBSCAN algorithm by steps (Peng et al., 2007)

DBSCAN algorithm
-Label all points as core, border or noise
-Eliminate noise points
-Put an edge between all core points that are within Eps of each other
-Make each group of connected core points into a separate cluster
-Assign each border point to one of the clusters with its associated core points

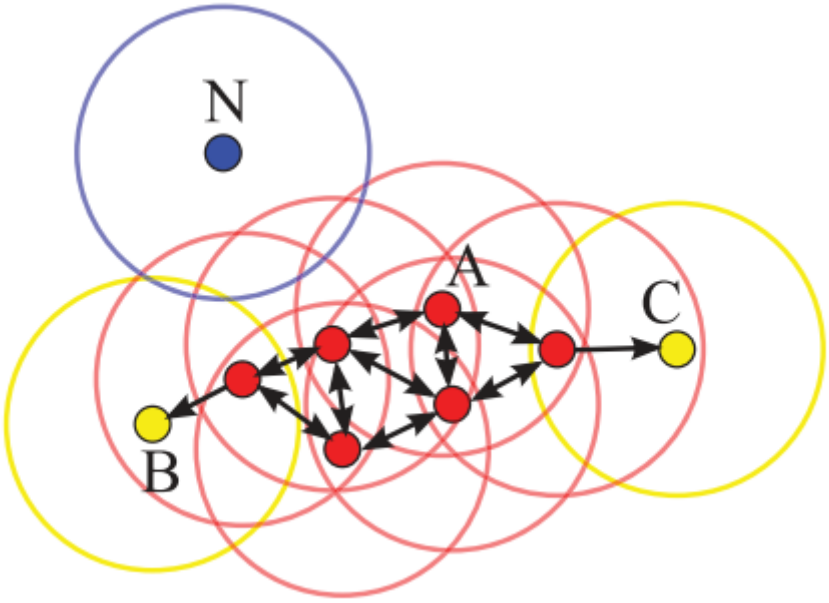

Figure 7-Illustration of the DBSCAN clustering model. "A" ,"B" and "C" belonging to the cluster and "N" as noise.

To use this algorithm two arguments are necessary, the distance between neighboring points (epsilon) and the minimum number of neighboring points surrounding central points to form a cluster (MinPts).
Therefore, in Figure 7 we can see how this algorithm identifies cluster points. In this case the MinPts parameter is 4 and we can see that the "A" points belong to the cluster because within its epsilon radius there are at least 4 points including the central point itself. The points "B" and "C", although they don't comply with this rule, are also part of the cluster as Border Points because they are within the epsilon radius. The "N" point is considered Noise as it does not comply with any of the rules described above.
The two main advantages of this algorithm are that it finds clusters with completely arbitrary formats and automatically excludes outliers.
As previously discussed, this algorithm is too sensitive to its parameters, namely Eps and Minpts, thus having difficulties in dealing with different data densities given this problem, a variation of this algorithm was chosen to be used as a test, the HDBSCAN, as is less sensitive to the arguments.
This algorithm was presented in 2013 by Campello, Moulavi, and Sander and is an extension of DBSCAN converting it into a hierarchical clustering algorithm.
Unlike DBSCAN, this algorithm only needs one argument, min_cluster_size that defines the minimum number of points that each cluster must have to be considered a cluster.

This algorithm can be represented in five steps:

1. The first step consists in transforming the space in view of its density and sparsity. (Figure 8)
2. The second step is calculating the mutual reachability distance between each point. (Figure 8)
3. Creation of the minimum spanning tree. (Figure 9)
4. Conversion of the obtained spanning tree into a hierarchy of connected components. (Figure 9)
5. Components condensed and the clusters are identified. (Figure 9)

In step 1 some points of the map with different neighbors are selected and their core distance is calculated, which according to the authors is characterized by the distance between these points and the (previously selected) k-furthest point found around them.
After these points of the map have been chosen, in step 2 the mutual reachability distance between each point is calculated. This distance, according to the authors, is the maximum distance between the core distances of the 2 points and the distance between them.

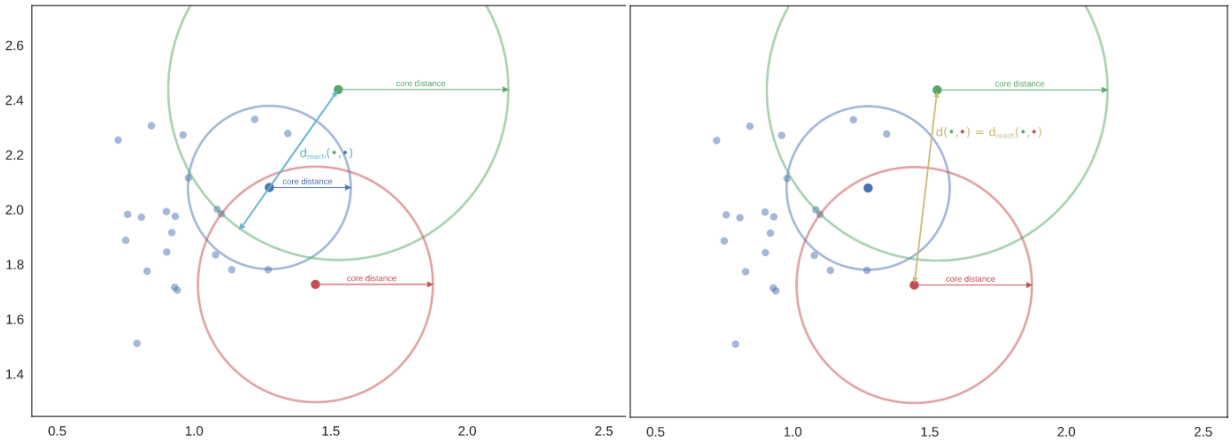

Figure 8- Illustration of HDBSCAN algorithm first steps ,adapted from https://hdbscan.readthedocs.io/en/latest/index.html

In the 3rd step, the minimum spanning tree is created using mutual reachability distance and Prim's Algorithm that adds one edge at a time, and always the one with the lowest cost.
Once the spanning tree is obtained, the 4th step is converting it into a hierarchy of connected components by organizing the edges of the tree by its distance and clustering them.
Finally, the 5th step, the cluster tree is condensed and together with the min_cluster_size argument the clusters are identified. This identification is done by choosing the nodes with the most colored area, remembering that choosing one node makes it impossible to choose another below that one.

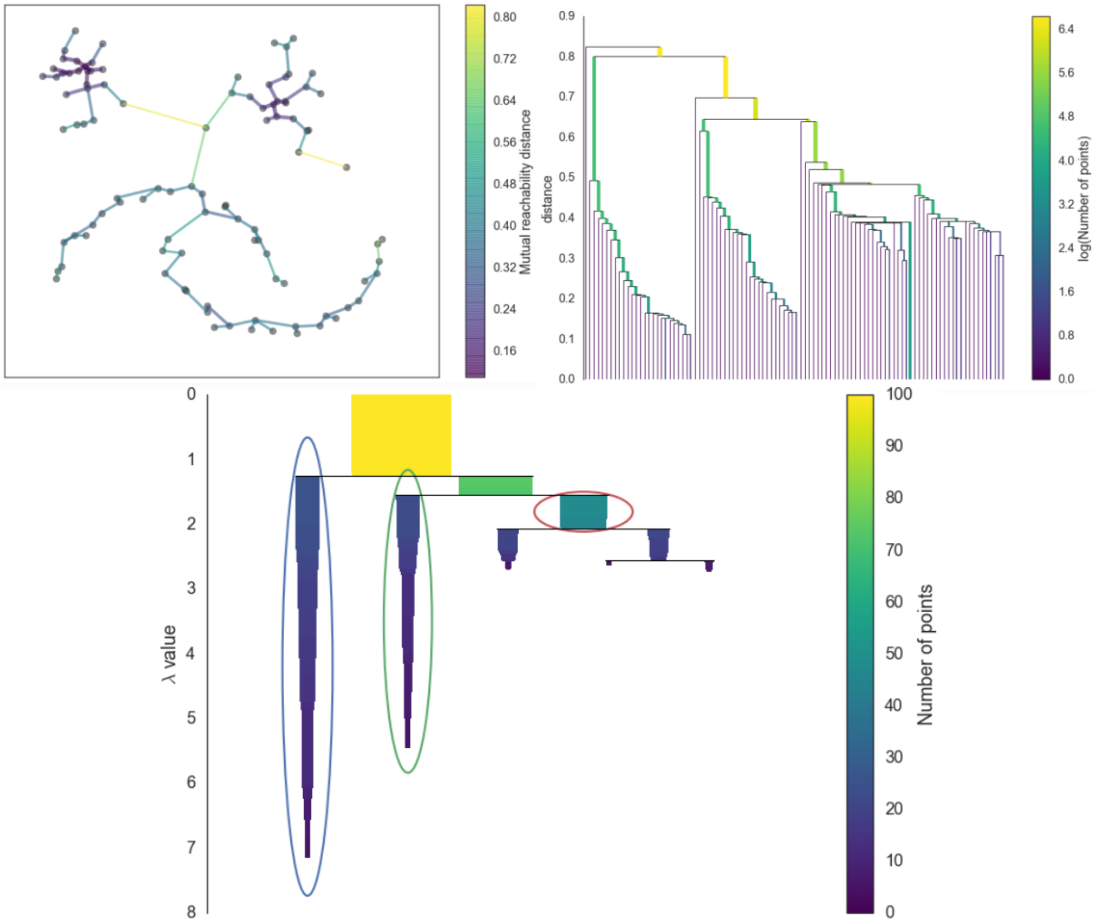

Figure 9- Illustration of HDBSCAN algorithm last steps , adapted from https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Another algorithm tested was K-Means because it is an easy-to-implement, fast and efficient algorithm in large amounts of data. Although it has these advantages, it is also an algorithm that needs some care in the selection of the number of clusters and does not identify any type of outliers.

This algorithm can be represented in four steps:

1. Select starting points on map
2. Assigning coordinates to the nearest points
3. Cluster average to get new points
4. Repeat points 2 and 3 until no changes are made

In step 1, this algorithm randomly selects on the map the number of points (clusters) initially established (Figure 10, b). In the next step, the closest coordinate points are assigned to these initial clusters to all points belonging to the nearest cluster (Figure 10, c), then the average of the points of each cluster is calculated and a new central point is assigned (Figure 10, d and step 3).The 4th and last step is assigning the points to the clusters and averaging is repeated until there are no changes, in the end the clusters are identified.(Figure 10, e and Figure 10, f)

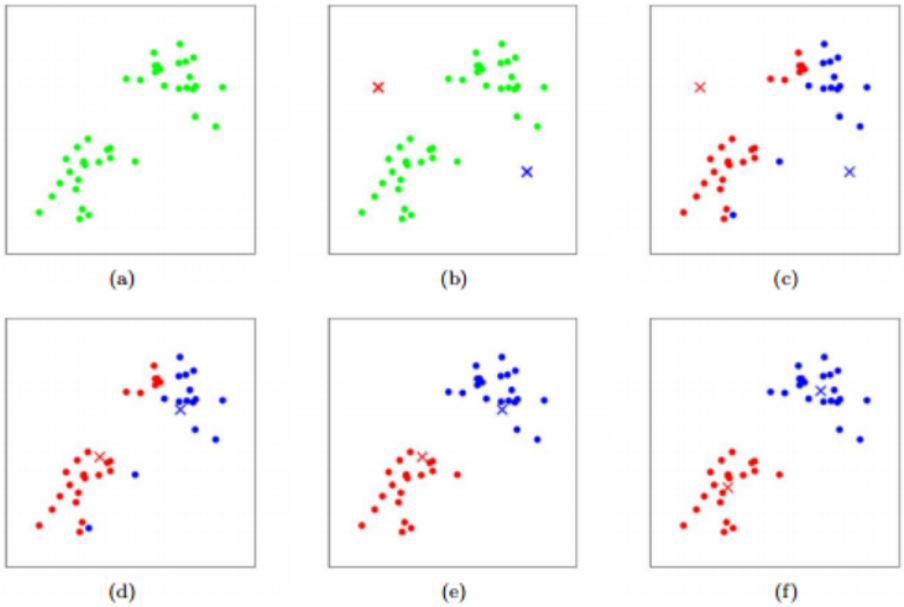

Figure 10- Illustration of K-Means algorithm, adapted from https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

To test a solution for the large processing time needed for large amounts of data, it was decided that a grid-based algorithm previously mentioned would also be tested, a grid-based algorithm, the CLIQUE.
This type of algorithm discretizes data by a grid and estimates density by the number of points in each cell of that grid.
In this particular case, CLIQUE is a subspace clustering algorithm that splits the data and counts the number of data points in each cell and then identifies the subspaces that contain clusters using the *a priori* principle (bottom-up approach).
This principle says that "if a collection of points S is a cluster in a k-dimensional space, then S is also part of a cluster in any (k-1)-dimensional projections of this space"(Agrawal et al. 1998).
Once the subspaces are identified, the clusters that intersect in the various dimensions are identified (Figure 11). The biggest disadvantage of this algorithm is that the final result depends a lot on the number and size of grid cells chosen

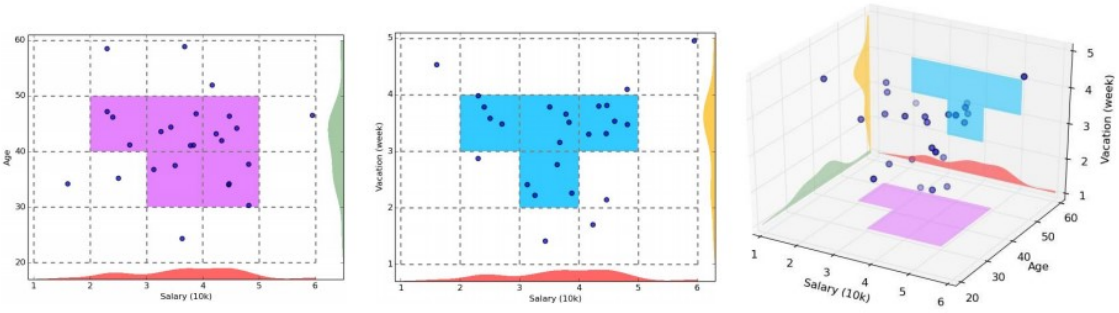

Figure 11- CLIQUE algorithm , adapted from https://list01.biologie.ens.fr/wws/d_read/machine_learning/SubspaceClustering/CLIQUE_algorithm_grid-based_subspace_clustering.pdf

3.3 POI identification

As a first approach, the 4 algorithms were tested on the GPS data without any kind of pre-processing.
As a second approach, an attempt was made to separate eventual POI from trajectories and then re-test the 4 algorithms, this separation was made by separating points that were at more than a certain speed, these being considered part of trajectories.
For the third approach, and as suggested in some articles, a daily data clustering was made in order to find stop points and to perform a last clustering on these points to find the final POI.
As a last approach and in order to try to assign a known location to each POI discovered (whenever it exists), a python (Overpy) wrapper was used to have access to the Overpass API which allowed to perform queries to the Open Street Maps database thereby allowing to attempt to assign the known location to the coordinates of each POI.

3.4 Trajectory identification

To identify recurring paths, two distinct approaches were taken.
The first approach was to analyze the data in order to identify trajectories.
To be considered a trajectory, it had to respect the following list of rules:

- Have at least 2 coordinate points;
- The points must correspond to a speed of at least 3.3 km/h (normal speed of a person walking);
- have a duration of at least 90 seconds.

Having already the trajectories, it was possible to identify the starting and ending points, and consequently the recurring trajectories. For each of the trajectories a snap-to-road API was used.
For the second approach we grouped all trajectories with equal start and finish, which occurred at least 10 times and finally tested 2 clustering algorithms on these data groups, hoping that the cluster was the sequence of points of several days for each trajectory. (DBSCAN and HDBSCAN)
At the end of next chapter several solutions to these problems were obtained which were subsequently properly tested on data from various users and the results were presented. These tests were based on processing the data of each user using the solutions, then the maps resulting from each solution were sent and an Excel file with each POI for each user to check if they are indeed correct. The results were then presented and the best solution was determined.

3.5 Data sample

The data referred to in this chapter is GPS coordinates donated by 9 volunteers. Most of the initial tests and training were done using only the author's data. All figures are from the author's own data, except where specifically mentioned.
The data presented is mostly located in the Setúbal and Lisbon regions. It was necessary to apply several steps of data cleaning, transformation and modelling before they were used.

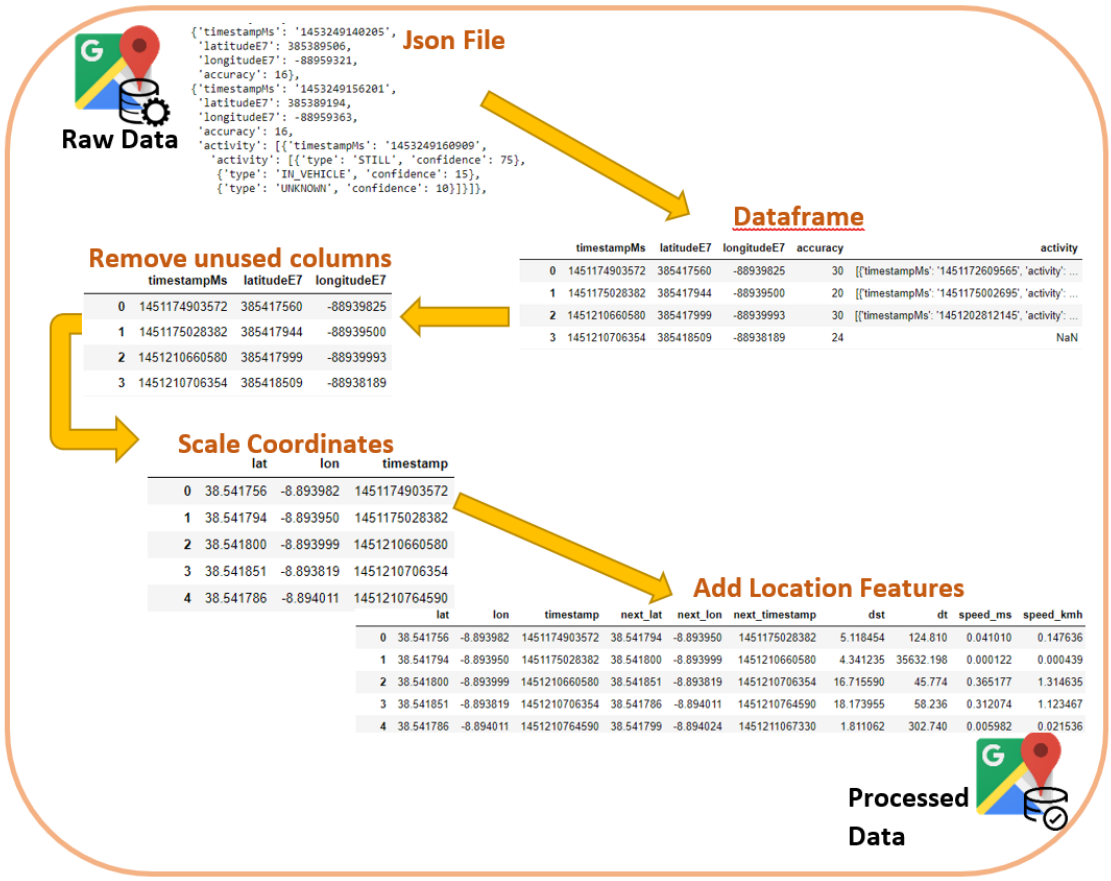

Figure 12- Simplified scheme of the Data Processing

Figure 12 shows how data is processed. In the first step, there is a sample of a Google GPS data set (in JSON format). Analyzing the figure more closely, it is possible to observe that in some cases there is additional information related to the type of activity that is being done, more specifically the way the user is moving, and, as one of the main objectives of this research is to draw all these types of conclusions, these extra information will not be taken into consideration since early experiments and previous, unpublished, work (private communication during supervision) has raised many doubts as to the accuracy of these labels.
This Json file is then converted into a Dataframe to make it easier to work with. The extra columns that will not be used in this project are then removed, for the reasons already described above, keeping only latitude, longitude and timestamp. The next step is to standardize the coordinates so that they can be used by some libraries in the experimental phase. Finally, in the last step several columns with important features for POI and trajectory identification algorithms are added to the Dataframe. These features consist of

the coordinates and timestamp of the next point to be able to also add the distance, time and speed between each of them.
This solution has been performed using 2 GPS data datasets from the author and one volunteer and tested using another 9 volunteers. One of the train datasets consists of 25293 location points between 1 January 2016 and 23 June 2020, and it's the dataset with the smallest amount of data. The other dataset consists of 426281 location points between 2 October 2011 and 26 September 2019. Due to the amount of data in the latter dataset and after testing data groups of different sizes, it was decided to limit (if necessary) the data of the volunteers to 60,000 because it was the number found that made it possible to obtain the desired results while also maintaining a reasonable processing time.
The number of points used and the time interval for each volunteer is shown in.Table 4

Table 4- Volunteers' Data Informations

Number of points of each volunteer	*Time interval*
60,000.00	1 Year
60,000.00	1 Year and 11 months
52,166.00	6 Years and 2 months
31,755.00	4 Years
60,000.00	1 Year and 7 months
8,843.00	2 Years and 2 months
6,347.00	2 Months
18,116.00	4 Years and 1 month
1,005.00	2 Weeks

3.6 Tools

One of the essential parts to perform all the tests and experiments to reach the final solution for the initial problem, was using some tools.
The python programming language was used for this research project because:

- Author's previous experience
- Versatility
- Many popular API's for machine learning and data science projects. (SlashData and Developer Economics 2020)

The Jupyter Notebook Platform was used due to the ease of integration between tests and notes.

4 Experiments

In this chapter we will explain in detail all the experiences made to reach the final solution to our initial problem. Therefore, we have two large groups of experiences, one for detecting POI and the other for identifying trajectories.
As seen in the previous chapter, the best existing solutions for POI detection are based on clustering algorithms, for this reason, 4 different clustering algorithms were chosen for the POI determination tests:

1. DBSCAN
2. HDBSAN
3. KMEANS
4. CLIQUE

As mentioned in chapter 3, the first algorithm chosen was DBSCAN because in most of the articles found, this algorithm was mentioned as giving good results and used by the authors as a basis for variations of it in order to overcome its limitations . As this algorithm has problems related to large variations in data density, HDBSCAN was chosen as the second algorithm because it is a variation of the first that deals with this problem.
As a third algorithm and also widely used, KMEANS was chosen to test a clustering partitioning algorithm. Finally, because it has advantages in terms of processing speed, a grid-based algorithm was chosen, the CLIQUE algorithm.

4.1 Data Understanding and Preparation

As previously described in the "data sample" section of chapter 3, for this data preparation phase, the Json file extracted from Google's account was initially read and converted to a Pandas dataframe where it will be processed in the next steps to be used as an argument to the algorithms mentioned above. An example of the structure of this dataframe can be found in Figure 13.

	timestampMs	latitudeE7	longitudeE7	accuracy	activity	velocity	heading	altitude	verticalAccuracy
0	1452779869698	385422003	-88905635	1799	NaN	NaN	NaN	NaN	NaN
1	1452779894607	385418368	-88940246	34	[{'timestampMs': '1452779890827', 'activity': ...	NaN	NaN	NaN	NaN
2	1452780014726	385418437	-88940404	33	[{'timestampMs': '1452780032430', 'activity': ...	NaN	NaN	NaN	NaN
3	1452780070110	385418408	-88940621	28	NaN	NaN	NaN	NaN	NaN
4	1452780170576	385418408	-88940621	178	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
25289	1580817054970	385417045	-88942936	16	NaN	NaN	NaN	NaN	NaN
25290	1580817967437	385416969	-88943021	16	NaN	NaN	NaN	72.0	2.0
25291	1580818913263	385417001	-88942867	16	NaN	NaN	NaN	72.0	11.0
25292	1580819828249	385417009	-88942941	16	NaN	NaN	NaN	NaN	NaN
25293	1580820730051	385417139	-88942896	16	NaN	NaN	NaN	NaN	NaN

Figure 13- Pandas dataframe containing Json data from Google services account

These Json files contain various data elements, such as accuracy, activity, velocity, heading, altitude and vertical accuracy, which will be of no interest for the development of this project and have therefore been removed from the dataframe. To facilitate the use of coordinates in the future they have been standardized to a scale of six decimal places, as this is the conventional representation and used by most systems, such as Google Maps and OpenStreetMap (example in Figure 14).

	lat	lon	timestamp
0	38.542200	-8.890564	1452779869698
1	38.541837	-8.894025	1452779894607
2	38.541844	-8.894040	1452780014726
3	38.541841	-8.894062	1452780070110
4	38.541841	-8.894062	1452780170576
...	...	...	...
25289	38.541705	-8.894294	1580817054970
25290	38.541697	-8.894302	1580817967437
25291	38.541700	-8.894287	1580818913263
25292	38.541701	-8.894294	1580819828249
25293	38.541714	-8.894290	1580820730051

Figure 14- Lat and Lon coordinates scaled

Then 3 more columns were added to the dataframe, the two coordinates and timestamp referring to the following point, this way each line of the dataframe contains one point and also the point bellow (Figure 15). This way it was possible to add 4 more columns, such as distance, time, and speed both in m/s and km/h between each of the two points on each row of the dataframe.
The column "dst" refers to the distance traveled between the 2 points of each line and was obtained by calculating the geodetic distance between the coordinates of the point and the next point of each line.
The column "dt" refers to the time spent between the same 2 points and was calculated by the difference between the timestamps of the 2 points divided by 1000 to transform from milliseconds to seconds.
The speed columns, both km/h and m/s, are calculated using the distance(dst) and time(dt) values previously obtained

	lat	lon	timestamp	next_lat	next_lon	next_timestamp	dst	dt	speed_ms	speed_kmh
0	38.542200	-8.890564	1452779869698	38.541837	-8.894025	1452779894607	304.431857	24.909	12.221761	43.998341
1	38.541837	-8.894025	1452779894607	38.541844	-8.894040	1452780014726	1.576111	120.119	0.013121	0.047236
2	38.541844	-8.894040	1452780014726	38.541841	-8.894062	1452780070110	1.919049	55.384	0.034650	0.124740
3	38.541841	-8.894062	1452780070110	38.541841	-8.894062	1452780170576	0.000000	100.466	0.000000	0.000000
4	38.541841	-8.894062	1452780170576	38.541800	-8.894050	1452780190330	4.615182	19.754	0.233633	0.841078
...	...	...	...	...	...	...	...	...	...	...
25288	38.541705	-8.894294	1580816154421	38.541705	-8.894294	1580817054970	0.000000	900.549	0.000000	0.000000
25289	38.541705	-8.894294	1580817054970	38.541697	-8.894302	1580817967437	1.122900	912.467	0.001231	0.004430
25290	38.541697	-8.894302	1580817967437	38.541700	-8.894287	1580818913263	1.388806	945.826	0.001468	0.005286
25291	38.541700	-8.894287	1580818913263	38.541701	-8.894294	1580819828249	0.651234	914.986	0.000712	0.002562
25292	38.541701	-8.894294	1580819828249	38.541714	-8.894290	1580820730051	1.495466	901.802	0.001658	0.005970

Figure 15- Dataframe containing Distance and Speed info

With these data present in the dataframe, it was possible to generate and present in Figure 16 a graph that gave us a general idea of how many points (X axis) each pre-defined range of km/h speeds had (Y axis).

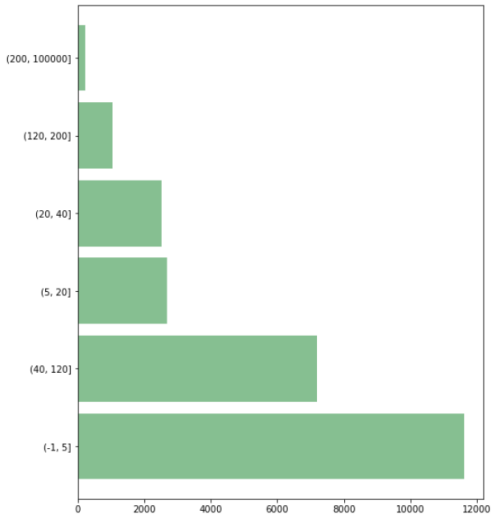

Figure 16- Speed intervals graph from dataframe

We can also observe in Figure 16 that there are unlikely and unrealistic speeds, so this data treatment was carried out afterwards in order to remove unreliable values.
To analyze the amount of data that existed per day, month and year, an analysis and graph were made which shows the regularity of data present in the dataset (Figure 18 and Figure 17).

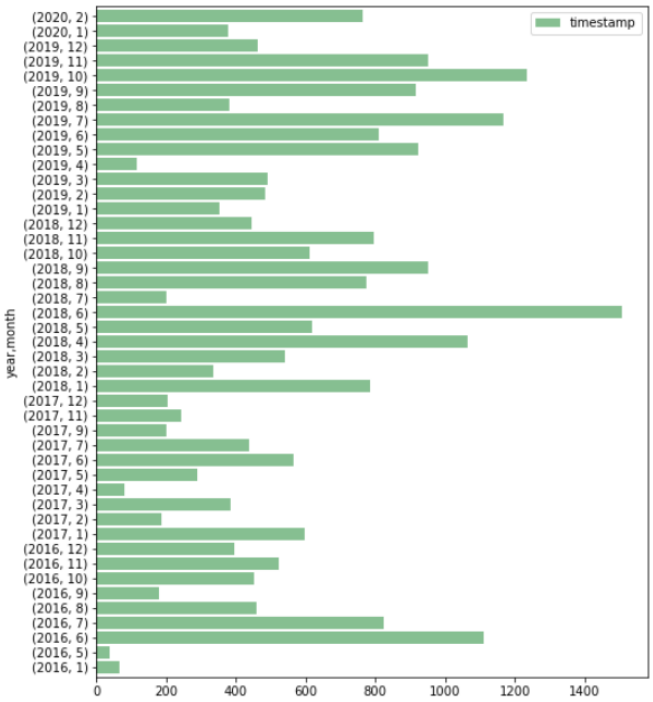

Figure 18- Data per month and year from dataframe

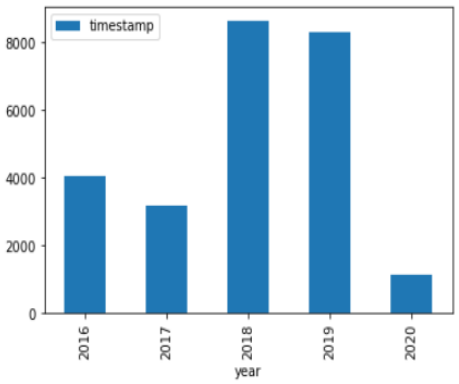

Figure 17- Data per year from dataframe

4.2 Raw Data Testing

For a first approach to find a solution to the POI problem, and to better understand how the algorithms work on the data, the 4 algorithms (DBSCAN, HDBSCAN, KMEANS and CLIQUE) were tested without any kind of data pre-processing.
Although it is somewhat inefficient for density clustering algorithms, especially those dealing with noise, for each result 3 different coefficients are also calculated to try to validate the consistency of the resulting clusters. (Yanchi Liu et al. 2010)
The Silhouette Coefficient is a measure of the cluster's cohesion, which means, how similar an object is to its own cluster, compared to its separation. Varies between -1 and 1 and is calculated using the mean intra-cluster distance and the mean nearest cluster distance. A value close to 1 corresponds to a high distance to a neighboring cluster, while points with a coefficient of -1 have likely been assigned to the wrong cluster. A value of zero is given to points which are equidistant to two clusters, and therefore have undetermined cluster membership. An algorithm yielding a high number of values with negative coefficients identifies an unsuccessful method (Rousseeuw 1987).
The Calinski and Harabasz index also known as the Variance Ratio Criterion is defined by the ratio between intra-cluster dispersion and inter-cluster dispersion. The larger the index, the denser and better separated the clusters are, and therefore the better the performance (Caliński and Harabasz 1974).
The Davies and Bouldin index measure the average similarity between clusters. This similarity is calculated by comparing the distance between clusters and their size. A low value indicates a better separation between clusters (Davies and Bouldin 1979).
All maps that will be shown during these experiments are based on actual data from the author except for a few exceptions where they are given by a volunteer who has given permission to do so.

4.2.1 DBSCAN

This algorithm requires 4 input parameters, *eps* that defines the maximum distance between 2 points to be considered a neighbor, *min_samples* that defines the minimum number of neighboring points to be considered a cluster, *metric* that defines how the distance between points in an array is calculated and *algorithm* that is used by the Nearest Neighbors module to find the closest neighbors, the first two are the most important to understand how this algorithm works.
As we are calculating distances of points in a sphere, the *metric* used in this algorithm was **Haversine** and the Ball Tree algorithm. After some tests with various values, for the distance between two points, 30 was the value that was producing the best results, more specifically, the resulting clusters were more often coincident with the correct POI, thus not obtaining very large clusters, nor a large number of small clusters. Due to the *metric* used (haversine), the *eps* value had to be introduced in radians and, in order to do this conversion, the 30 meters were divided by the earth radius (6371000 meters). Although the solutions were developed using a small dataset (of the author) and a denser dataset (of one volunteer), when they were tested in the data of the 9 volunteers it was possible to conclude that there are some datasets that have a high coordinate density in a short time frame, usually corresponding to people who have the smartphone's location on 24/7 or people whose smartphone is programmed to store location points more frequently. In these cases, if the parameter set is low, it will lead to too many POIs and lots of false positives. In datasets with lower coordinate density, in cases where the person only turns on the smartphone's location sometimes, if the *eps* parameter is too high, very few POI or

even none can be detected. Since the data we used was limited to 60,000 points we initially decided to assign a value of 5 for users whose data is less than 30,000 points and 10 for those with data greater than 30,000. Although not the perfect solution for all datasets, the results were good and better than just using a fixed value for them.
The resulting map of this test can be seen in Figure 19.

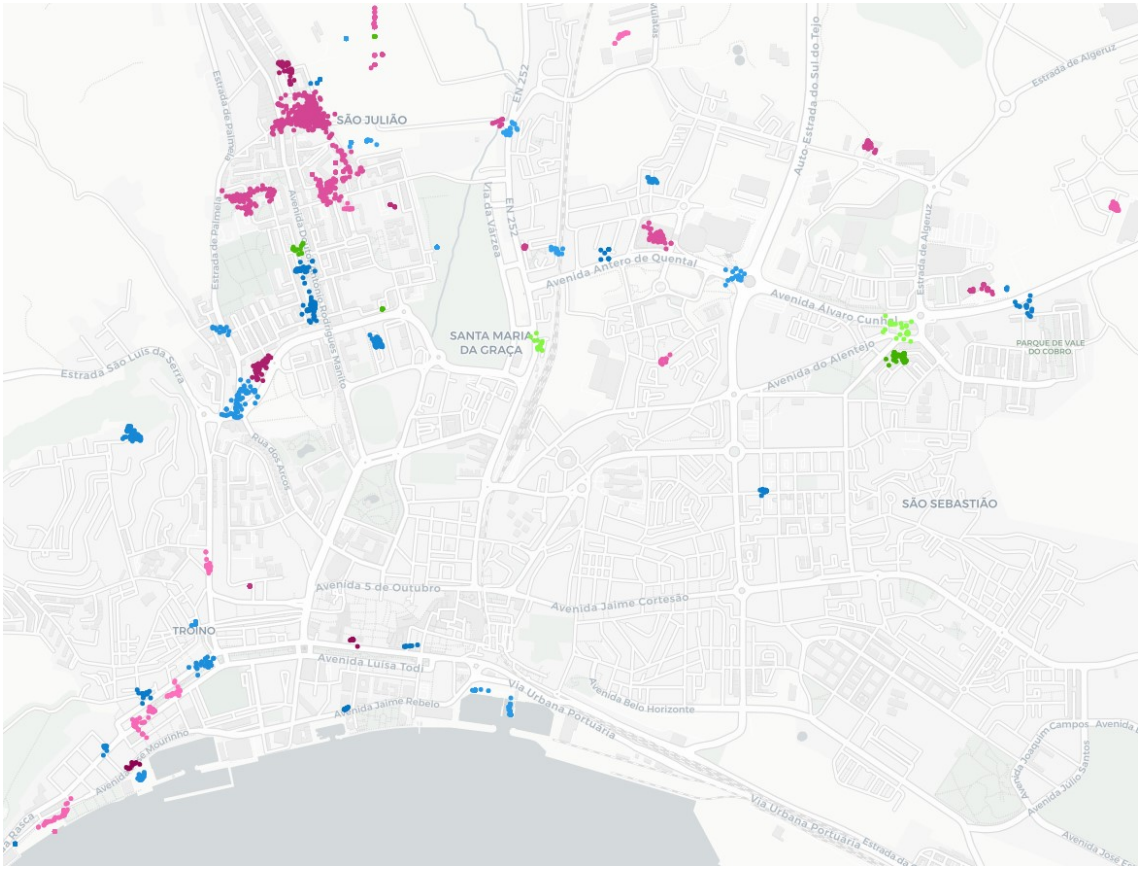

Figure 19- Resulting DBSCAN map using Raw Data

4.2.2 HDBSCAN

As this algorithm came to solve the limitation of DBSCAN densities, it only requires an input parameter, *min_cluster_size*. This value, as for DBSCAN, will depend very much on the type of dataset being used. After testing lots of values, for the low-density dataset the value 15 was used because lower values obtained too many clusters and higher values lower values resulted in few clusters. For the high sensitivity datasets the same testing was applied and the value 100 was used. Lower and higher values were tested and that was the one with the best results.
The resulting map can be seen in Figure 20.

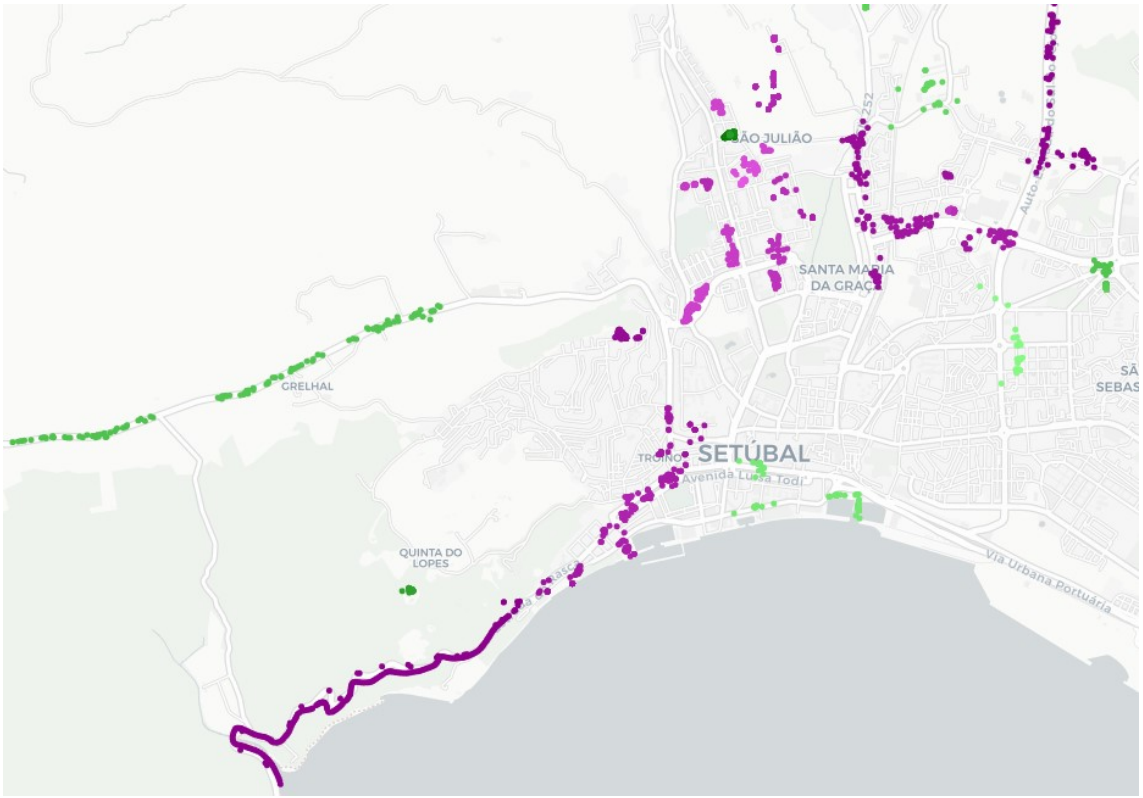

Figure 20- Resulting HDBSCAN map using Raw Data

4.2.3 KMEANS

To test this algorithm, it was only necessary to enter as an argument the number of clusters desired. In case of KMeans, this number could most easily be found using the Elbow Method (Figure 21) which basically iterates over a range of k clusters and uses the sum of the quadratic distances of each sample to the centroid of the cluster (inertial). The k-point chosen is the point after which inertia starts to decrease linearly on the graph.

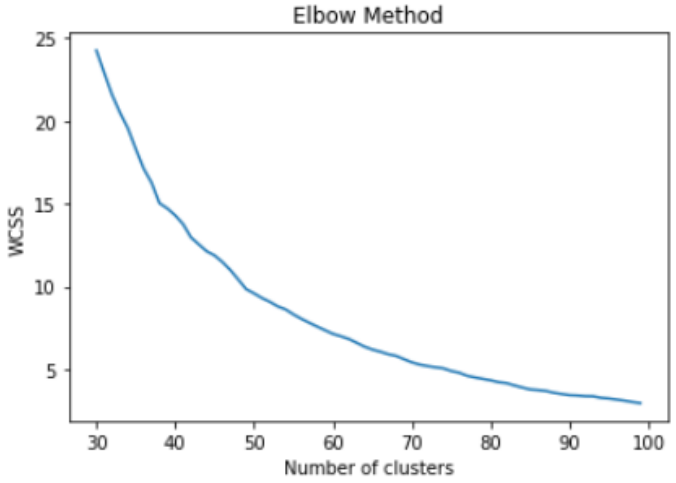

Figure 21- KMEANS Elbow method from Raw Data

In some cases, observing the graph it is not possible to easily extract a value from k clusters, for this reason it was also used a KneeLocator that does this task (Figure 22).

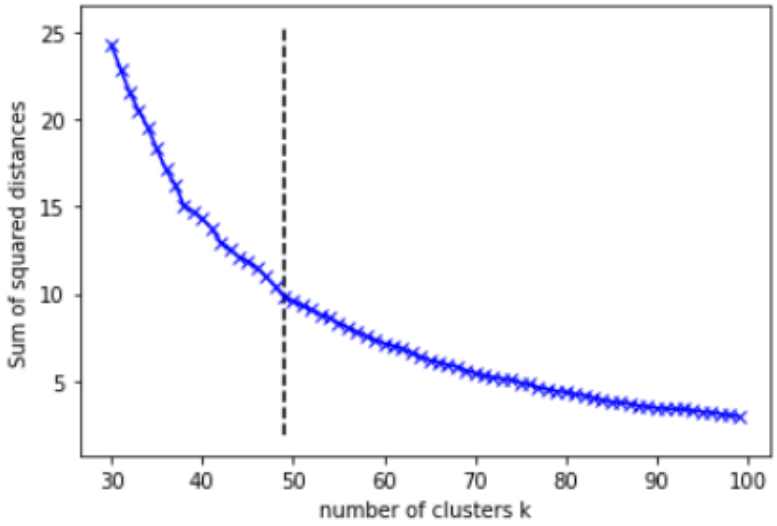

Figure 22- Elbow method knee locator from Raw Data

After finding the value (49) it was only necessary to enter it as a parameter to test the algorithm.
The resulting map can be seen in Figure 23.

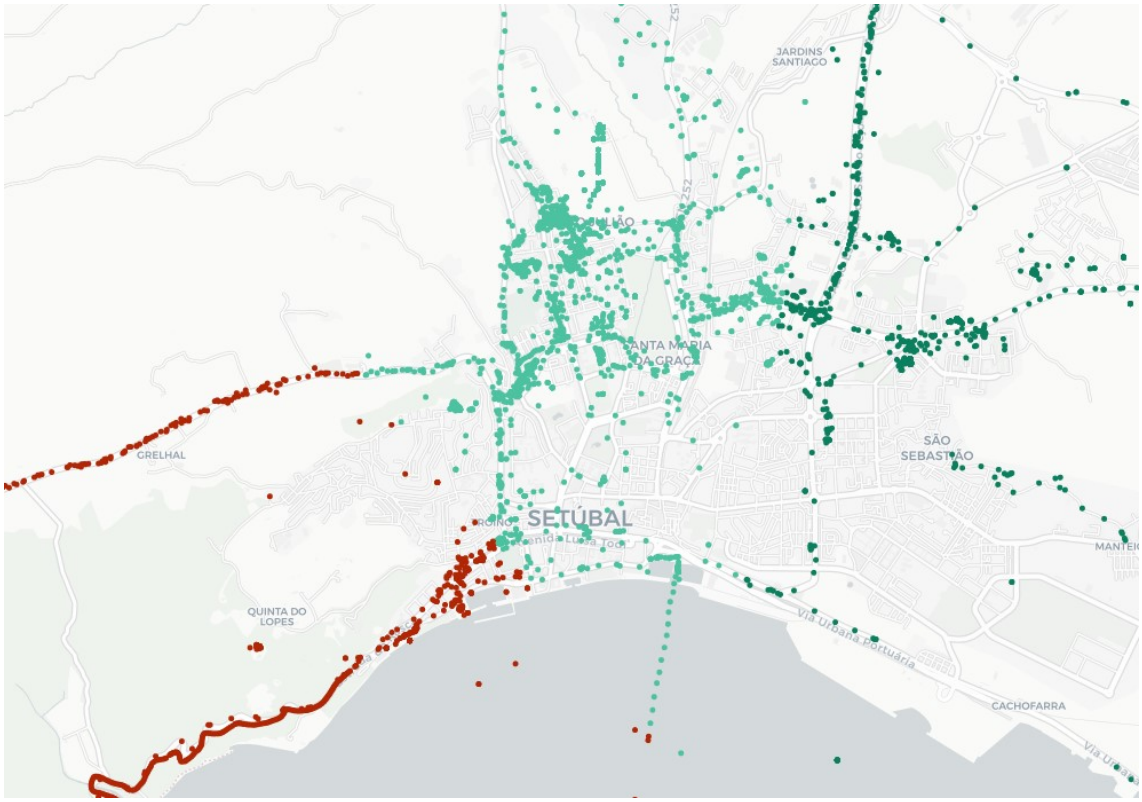

Figure 23- Resulting KMEANS map wusing Raw Data

4.2.4 CLIQUE

To test this algorithm, as it is grid based, two input parameters were necessary, to define the number of cells on the side of the grid, and the threshold that defines the minimum number of points that must be in a cell in order not to be considered outliers.
As with DBSCAN and HDBSCAN, in this algorithm the value of side cells will depend on the dataset used.
This dependency exists because the size of the grid will correspond to the distance between the two most distant points on the map. To better understand this situation, we can first imagine a grid with 100 side cells in a dataset that only has points in Portugal. If you then apply the same grid to a dataset that besides having points in Portugal, will also have points in Australia, we can see that each of the 100 cells will have a considerably different size in both cases. Considering this, for datasets where the points are more dispersed, a larger number of side cells is needed.
For this test we used 100 side cells and the value 5 as a threshold.
The resulting map can be seen in Figure 24.

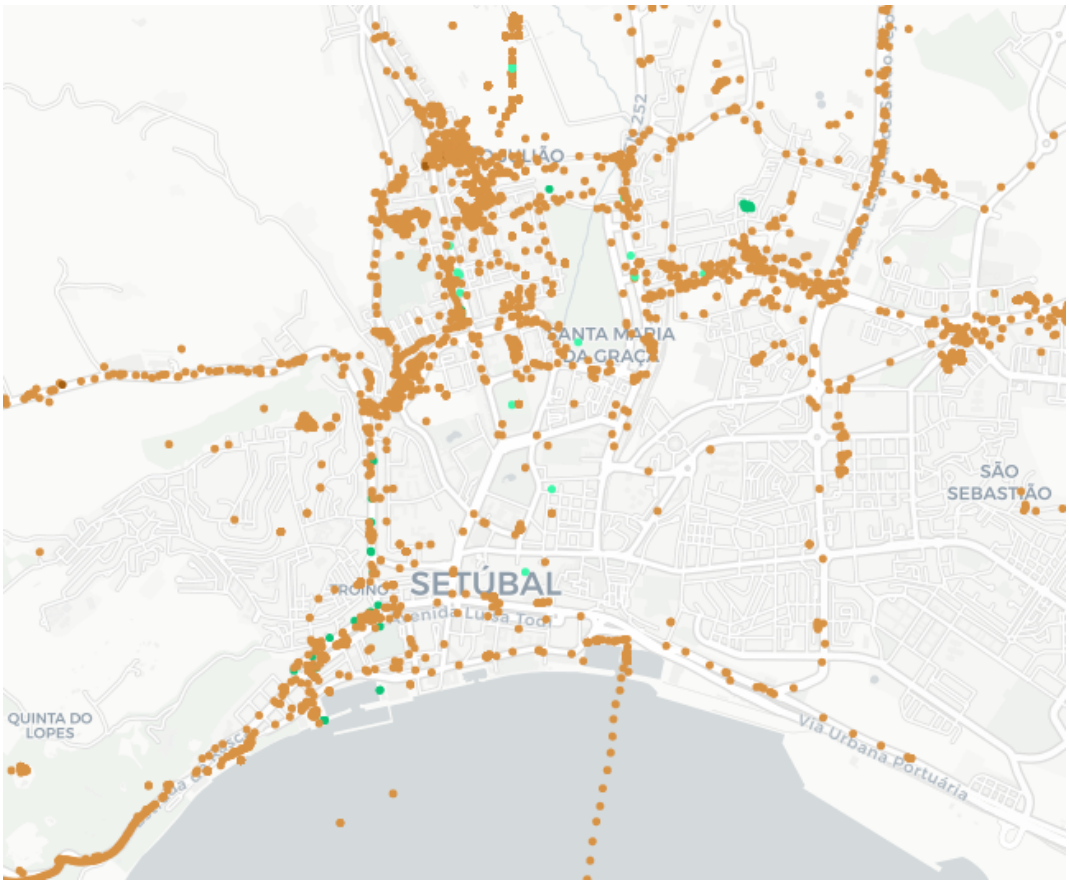

Figure 24- Resulting CLIQUE map using Raw Data

4.2.5 Raw Data Results

In the last 4 sub chapters, the algorithms DBSCAN, HDBSCAN, KMeans and CLICK were tested on the author's data without any rules or data filters.
The data used for the tests were only the latitude and longitude previously scaled according to the most commonly used standard (chapter 3.3).
The first conclusions can be drawn by a visual analysis of the results in a map: in HDBSCAN , KMEANS and CLIQUE results (Figure 20, Figure 23 and Figure 24 respectively), it was observed that the trajectories (possibly by car) appeared on the POI maps. This presents a problem because in reality they are data that will only be important for the study of the trajectories, and very rarely could be considered POI. Analyzing now in more detail each of the maps, we can verify that **DBSCAN** was the one that obtained more reasonable visual results, although it generated too many clusters (189). DBSCAN also marked a lot of points as outliers thus obtaining a map with less noise. In the case of **HDBSCAN**, as it is an algorithm developed to deal with various densities, it generated 300 clusters, again, too many if we consider how close they appear together in some areas of the maps. Observing the map from Figure 25 in more detail we can see that several of these clusters are actually the same area, where it should only be considered a POI. This problem can be solved using the "cluster_selection_epsilon" parameter. This method, described by Claudia Malzer and Marcus Baum, mixes DBSCAN with HDBSCAN, more specifically, returns the DBSCAN results only in the area affected by the points defined by the parameter. In this case the parameter needs an epsilon measure (like DBSCAN) which is used to join all clusters that are closer than that defined value. In this case, the clusters whose distance from the neighbors was less than 5 meters were merged (Malzer and Baum 2019).
Several clusters of this algorithm were also part of trajectories (Figure 20).

Figure 25- Section of Resulting HDBSCAN map showing a high density problem

Regarding KMEANS, as it is an algorithm that does not deal with outliers, we can always expect to see a map with all the entry points in the algorithm, which is what happened. This algorithm identified the 49 clusters previously defined by the Elbow Method, which we considered a good number and perfectly real. However, the clusters found were quite large, which makes the final result imprecise and, in this case, unusable (Figure 23).
Analyzing the map resulting from the CLIQUE algorithm it is possible to conclude that the results were not at all reasonable, having generated only 10 clusters and most of them so large that it is impossible to draw any kind of conclusions regarding POI. It should be noted that the trajectories are also present (Figure 24 and Figure 26).

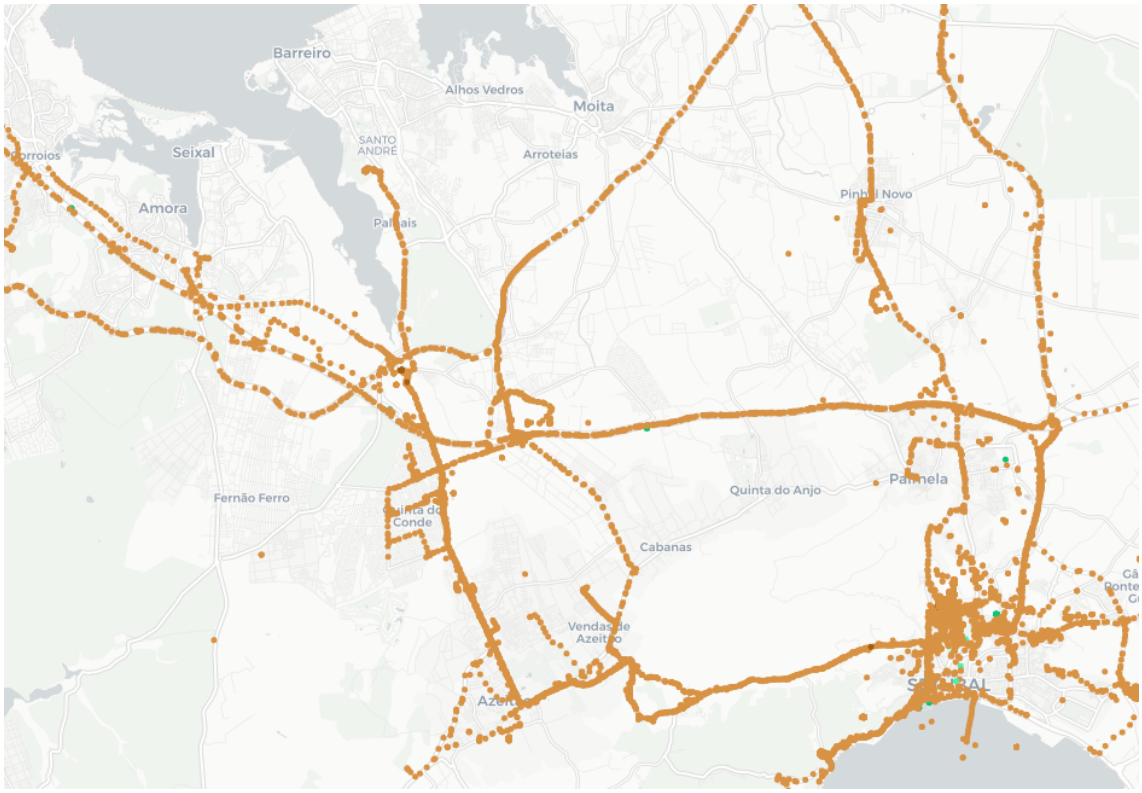

Figure 26- Resulting CLIQUE map using Raw Data and showing trajectories

Table 5-Cluster Info from Raw Data

	Silhouette	Calinski Harabasz	Davies Bouldin	Nº Clusters	Nº Noise Points	Process Time (s)
DBSCAN	-0.10	256.79	1.96	189	13061	3.86
HDBSCAN	0.20	2435.84	1.45	300	7129	1.45
KMEANS	0.53	3243119.64	0.51	55	0	1.01
CLIQUE	-0.48	14170.79	3.02	10	54	0.70

Now analyzing the metrics and information on clusters from Table 5
-Silhouette Score: according to the results, the CLIQUE is the one that produces the worst results, which is in accordance with what can be concluded by visually analyzing the resulting map. This score also indicates that KMEANS produces the best results, which as previously concluded, is no longer in accordance with the results of the maps in

this case (Figure 23). DBSCAN and HDBSCAN had bad scores especially DBSCAN possibly because it is the algorithm that detected the most noise, so it is possible to confirm that these metrics are not perfect for density clustering algorithms.
-**Calinski Harabasz**: In the case of the DBSCAN and HDBSCAN algorithms, the values seem consistent, with the HDBSCAN having a higher value due to having detected a larger number of clusters and presenting more varied densities. In relation to KMEANS and CLIQUE, these values are in accordance with the maps because the clusters are very large.
-**Davies Bouldin**: According to this metric, the lower the value, the better the performance of the algorithm. In this case the values make some sense because in KMEANS case, although it didn't have the best visual results, it was the only algorithm that had a heuristic to determine the best parameter to use (Elbow method), thus obtaining the lowest value. HDBSCAN and DBSCAN were the second and third best respectively and CLIQUE the worst with the highest value.
It should be noted that in all cases where KMEANS is used in this project, 30 repetitions of the experiments were made and an average was obtained since there is a random component in the choice of starting points.
We can see that DBSCAN was the algorithm that identified the most points as Noise followed by HDBSCAN.
As expected, CLIQUE was the fastest algorithm as it is grid based.

4.3 Speed split Data Testing

As a second approach, and to try to solve some of the problems identified above, the same 4 algorithms were tested on the data after it was filtered using some rules.
Previously to get the velocity columns, the calculation was done with the data from the next dataframe line, and in cases where the next line does not refer to data from the same trip, or even from the same day, unreal velocities appear. Considering this, the first filter used on the data was to convert speeds above 250 km/h into 0.
This conversion was done in order to be able to 'divide' the data more correctly, in this case it was considered that data whose speed corresponds to less than 10 km/h are stopping points and grouped in a separate dataset. The opposite are trajectories points.
What was expected from this separation was to obtain a map with clusters representing isolated locations and another map with paths between those same points.
Then the same 4 algorithms were tested on the "trajectory" data and on the "stop" data.

4.3.1 DBSCAN

By analyzing the "stop" map resulting from this test and comparing it with the previous one, it is possible to verify that although fewer clusters (137) have been identified, they were still quite a few. The trajectories that were on the map from Raw Data were no longer part of the clusters in this map (Figure 27).

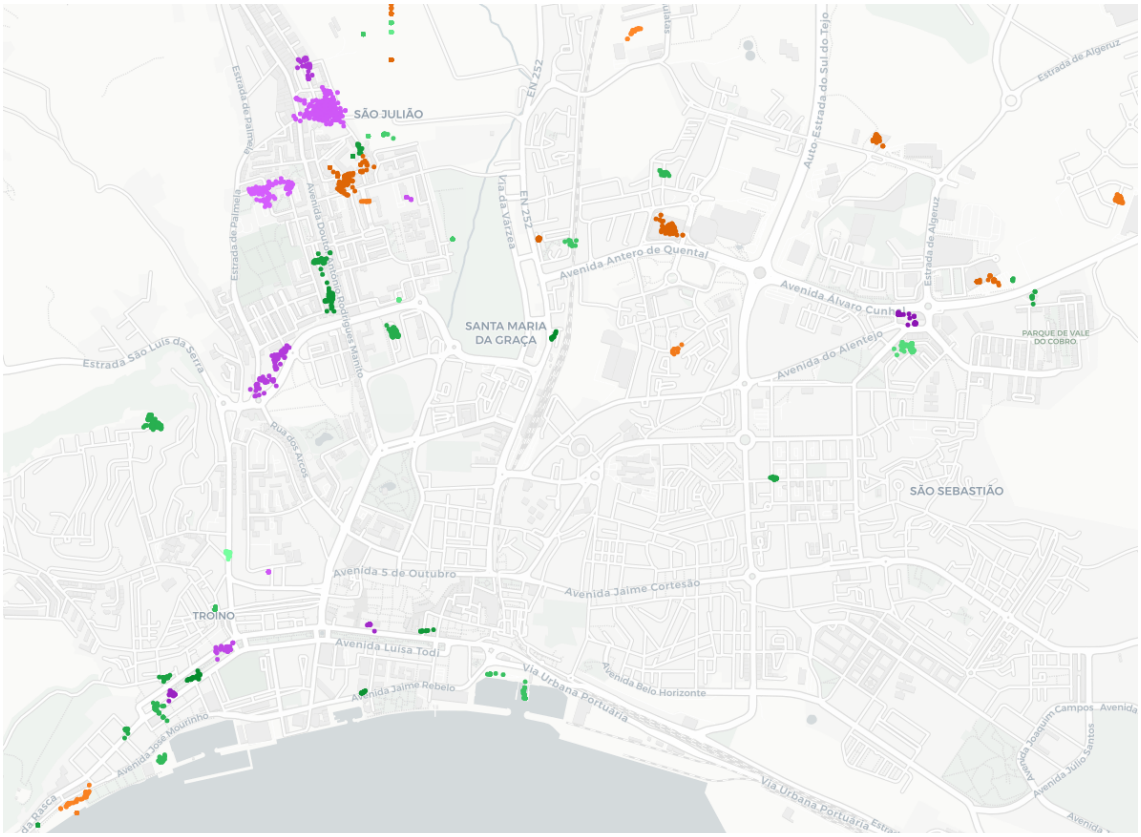

Figure 27-Resulting DBSCAN map using the "Stop" data

By analyzing the resulting "trajectory" map, it is possible to verify that it is a map that contains all the routes linking the points.
Although in this map only trajectories are supposed to exist, it also has many points that visually do not look like trajectories (Figure 28).

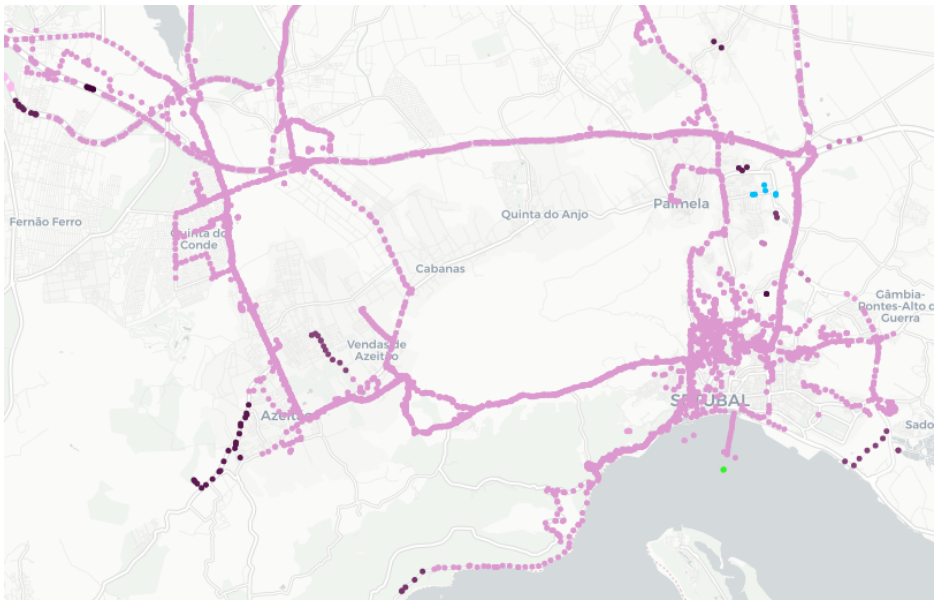

Figure 28- Resulting DBSCAN map using "trajectory" data

4.3.2 HDBSCAN

In relation to the HDBSCAN "stop" results, 26 clusters have been identified, which is a major improvement over previous results for this algorithm. As in DBSCAN it can be observed that the trajectories are no longer visible (Figure 29).

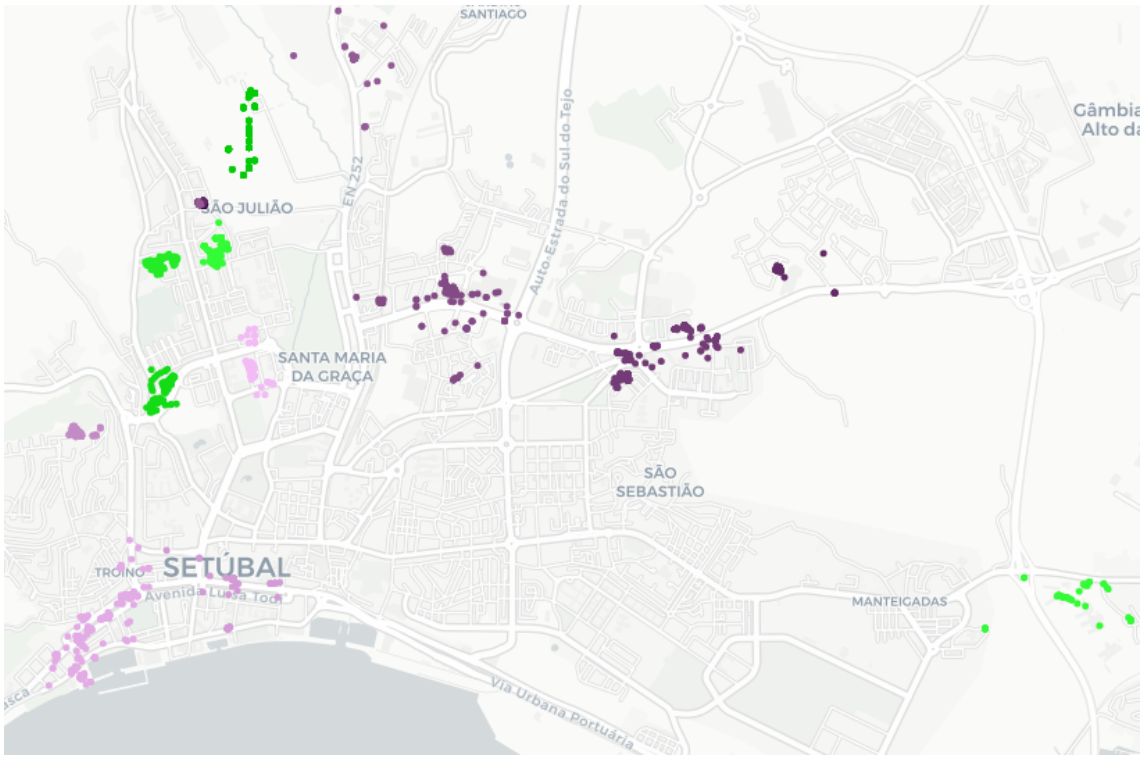

Figure 29- Resulting HDBSCAN map using "Stop" data

In some cases very large clusters are created as can be seen in the Figure 30.

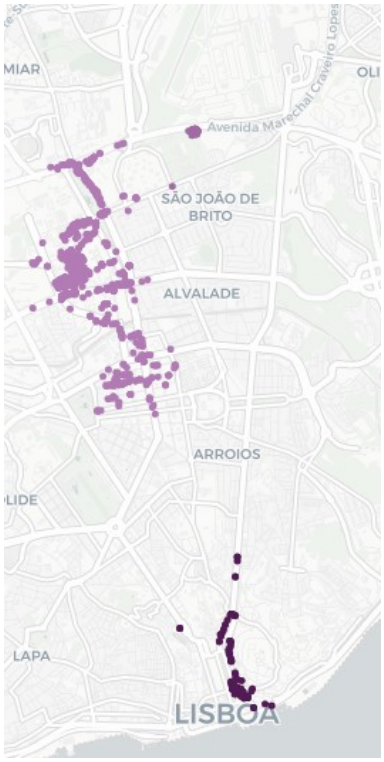

Figure 30- Example of HDBSCAN desnsity problem

Regarding the map resulting from the "trajectory" data test, we can see that this algorithm can obtain better results than DBSCAN, separating and better identifying the trajectories in relation to other isolated points (Figure 31).

Figure 31- Resulting HDBSCAN map using "trajectory" data

4.3.3 KMEANS

Analyzing the results of the KMEANS algorithm, it is possible to observe that now a larger number of clusters have been identified (49) in the case of "stops" and that although most trajectories are no longer present, there are still some (Figure 32).

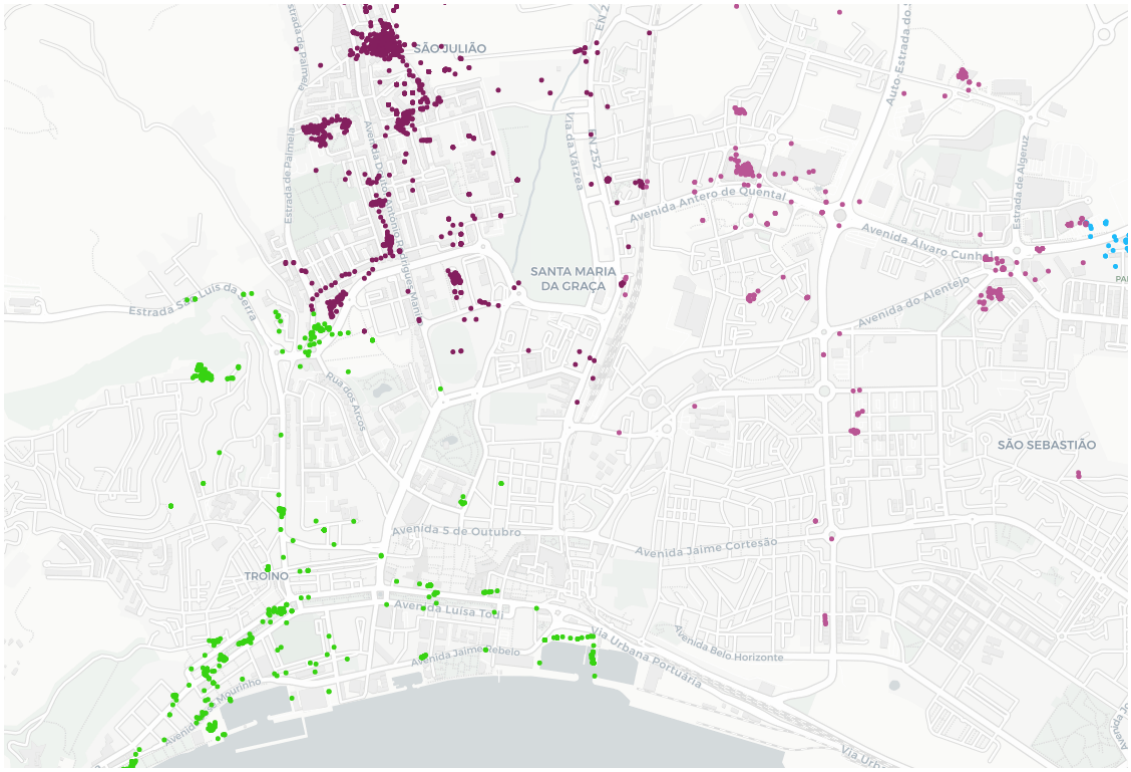

Figure 32- Resulting KMEANS map using "Stop" data

Observing the results from the "trajectory" data, 36 clusters were identified, and analyzing the map it was possible to conclude that although the trajectories are present, many outliers are too, also typical of this algorithm (Figure 33).

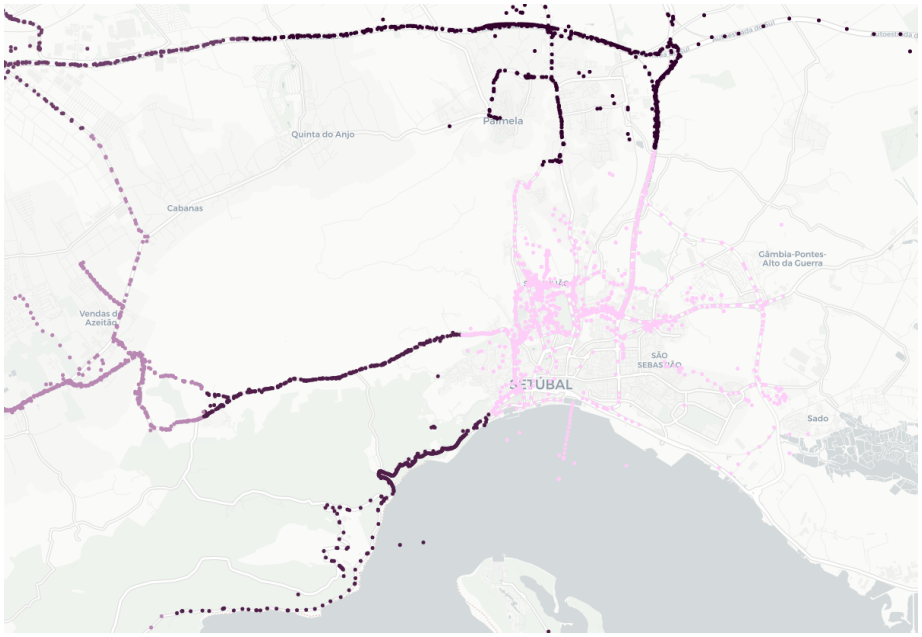

Figure 33- Resulting KMEANS map using "trajectory" data

4.3.4 CLIQUE

For the CLIQUE algorithm "stop" data, the results were not very different from the previous ones, in other words, although the trajectories are not present in most cases, the algorithm identified only 10 clusters, these being quite large or scattered, a bad result (Figure 34).

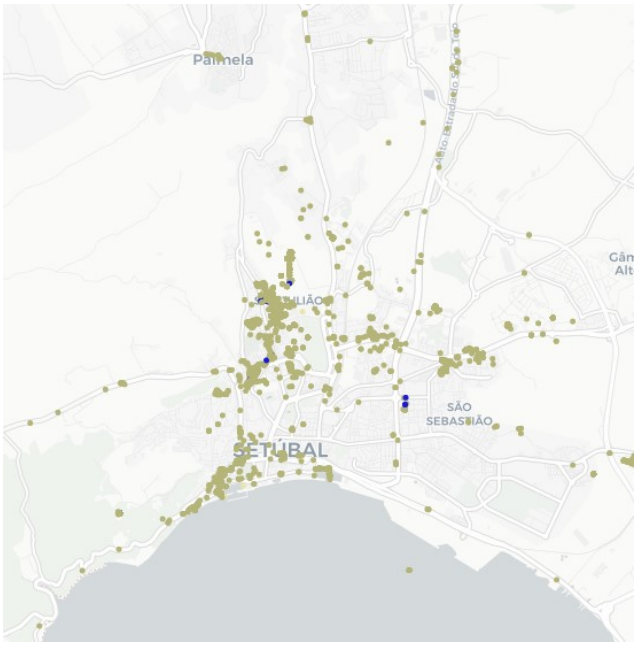

Figure 34- Resulting CLIQUE map using "Stop" data

In the case of the "trajectory" data, 8 clusters were identified but, as in previous cases, they are too large and scattered, thus making the results unusable for our purposes (Figure 35).

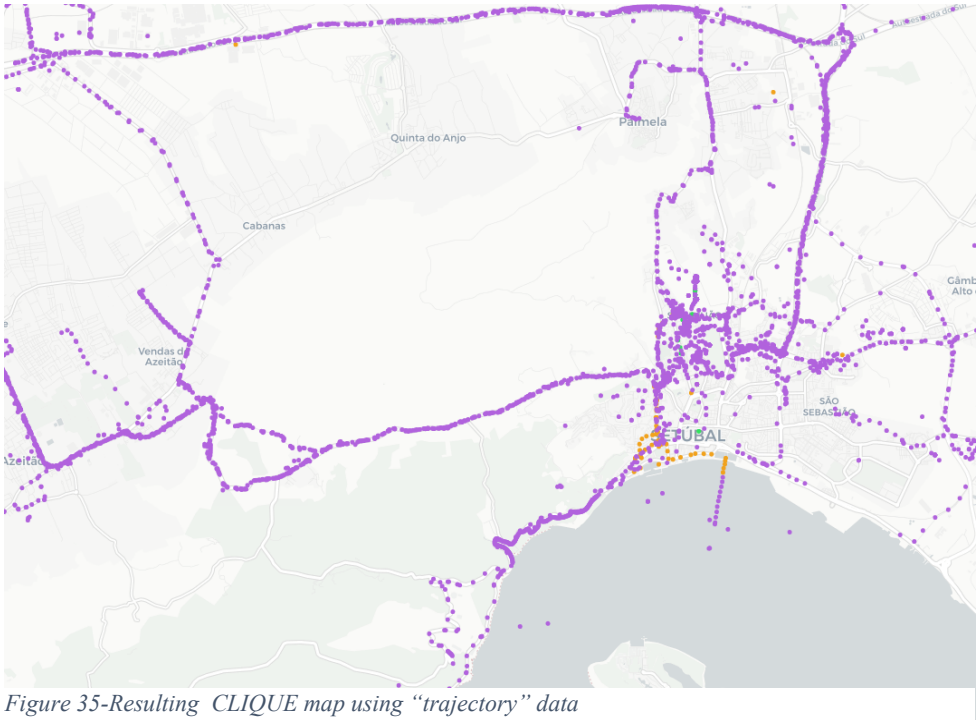

Figure 35-Resulting CLIQUE map using "trajectory" data

4.3.5 Results

As could be seen from the previous data, the separation of the "stop" and "trajectory" data resulted to some extent. More specifically, the trajectories hardly appeared in the "stop" data, which was positive. When they do appear, it means that they are points where there was a lot of traffic and therefore low speed on the road or in fact trajectories that were carried out on foot. In the "trajectory" cases, the results were not so positive because in most cases lots of other points show on the map, points that are actually outliers and this makes it very difficult visually to distinguish the trajectories.
There are also still too many clusters, and many of them cannot be considered POI because their constituent points all belong to the same day.

Table 6- Cluster Info from Stop & Trajectory (Traj) Data

Algorithms	Silhouette	Calinski Harabasz	Davies Bouldin	Nº Clusters	Nº Noise Points	Process Time (s)
DBSCAN stop	0.48	337.09	1.89	137	2760	3.11
DBSCAN traj	-0.56	135.38	1.98	228	386	0.98
HDBSCAN stop	0.53	41186.44	1.31	26	2373	0.42
HDBSCAN traj	0.02	1765.05	1.24	151	4395	0.21
KMEANS stop	0.68	11294664.55	0.40	51	0	0.32
KMEANS traj	0.47	650976.57	0.52	48	0	0.67
CLIQUE stop	-0.56	41663.97	2.50	10	46	0.44
CLIQUE traj	-0.34	321.79	2.39	8	56	0.43

In this approach, metric values have had some significant changes in Table 6
-**Silhouette Score**: as in the previous approach, the values for the CLIQUE show that the algorithm had negative results, it is also possible to verify that KMEANS presents the highest values again. In the case of "DBSCAN stop" it is already possible to get a positive value showing good results, in the case of DBSCAN trail as confirmed on the map, the results are not adequate.
In the case of HDBSCAN, the results are better than DBSCAN for both stop and trajectory.
-**Calinski Harabasz**: once again for KMEANS and CLIQUE the values are large derived from the size of their clusters.
This value is also higher in HDBSCAN compared to DBSCAN because although the number of clusters is much smaller, the size of the clusters is bigger.
-**Davies Bouldin**: In this metric the values follow exactly the same logic as in the previous approach, i.e. KMEAN remains the best parameterized algorithm, followed by HDBSCAN and DBSCAN. The CLIQUE shows high values, so it is considered the worst option.
In this approach DBSCAN has identified more noise in stop data and HDBSCAN in trajectory data.
A rather surprising result was the HDBSCAN runtime that exceeded the CLIQUE for both datasets. DBSCAN is still the one showing the lowest execution performance.

4.4 Stay Points Testing

For this test, a previously mentioned approach used by (Fu et al. 2016) and (Montoliu, Blom, and Gatica-Perez 2013), POI using Stay Points was used. This process consisted in creating clusters of the location data of each day separately, thus originating the so-called Stay Points. These Stay Points were then also used as input for the clustering algorithms, thus originating the POI. All possible combinations of clustering between the 4 algorithms were tested, so the Stay Points were found using each one of the 4 algorithms, and then on each of these results the 4 algorithms were again tested to find the POI.
The main advantage of this method compared to the previous one is to be able to identify the places where the user has been on several different days, which corresponds more to the correct POI concept. For example, a user attends the same place every day of the week but only for 5 minutes, will generate a number of points, if on another day the same user attends a different place but for 5 hours, that place will generate more points than the other place on all days of the week. In this example, the previous method would more easily consider the place where the user only went once as POI for the amount of points. With this method that will be tested, that location would not even be considered because it was only one day.

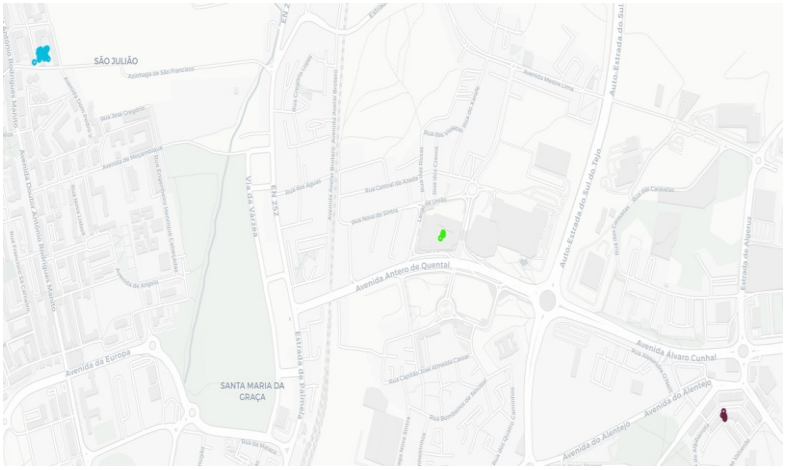

Figure 37- Resulting DBSCAN clustering from DBSCAN Stay Points

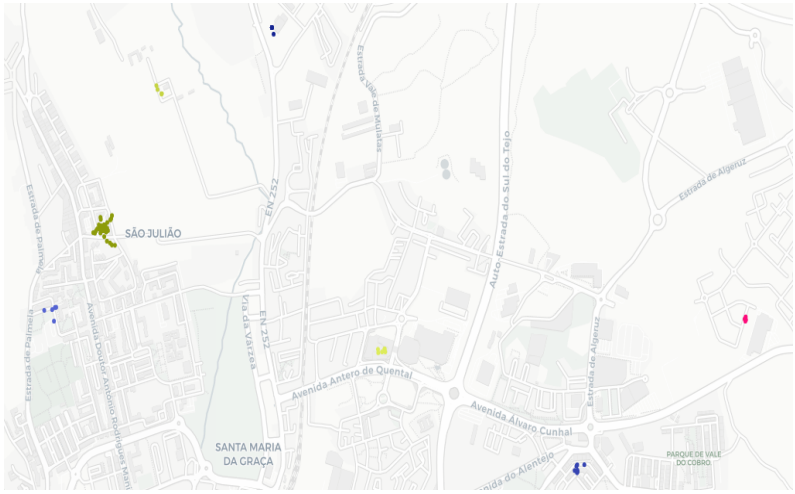

Figure 36- Resulting DBSCAN clustering from HDBSCAN Stay Points

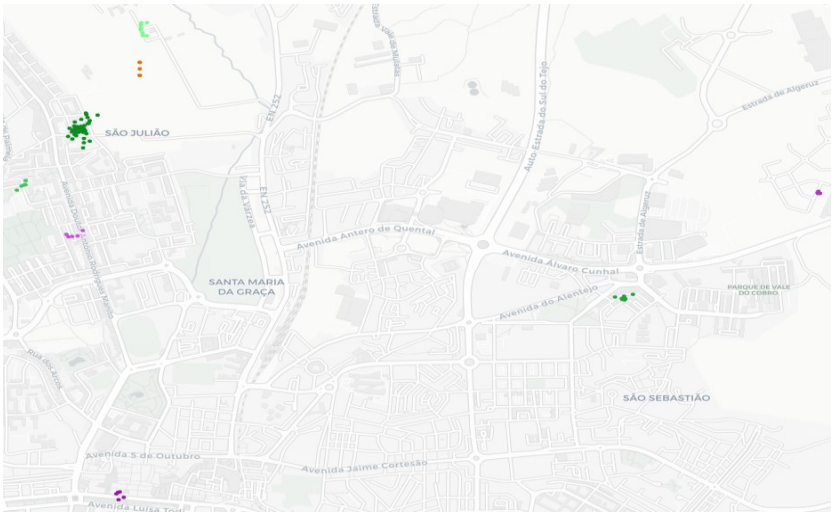

Figure 39- Resulting DBSCAN clustering from KMEANS Stay Points

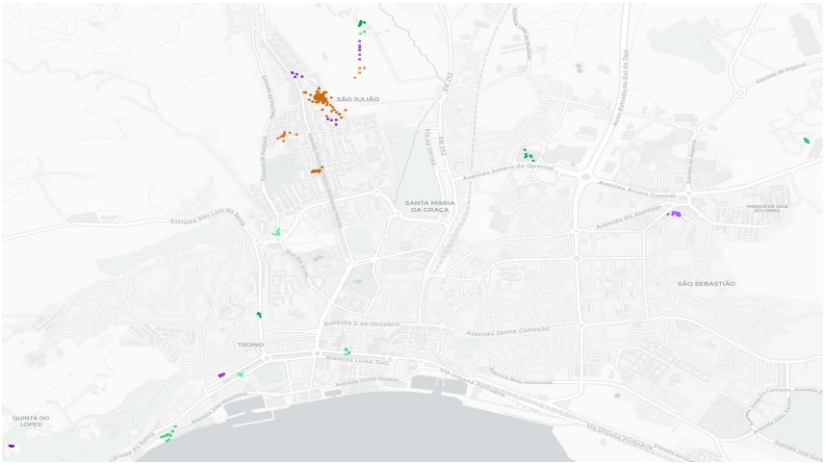

Figure 38- Resulting DBSCAN clustering from CLIQUE Stay Points

4.4.1 Results

After testing all algorithm combinations, the only ones that returned good results were DBSCAN_DBSCAN (Figure 37), HDBSCAN_DBSCAN (Figure 36), KMEANS_DBSCAN (Figure 39) and CLIQUE_DBSCAN (Figure 38) as they were the only ones that identified reasonably sized and problem-free clusters with different point densities, so these were the solutions proposed and tested on the volunteers data in the results chapter. All other combinations had density problems and originated clusters too large for the intended purposes, some of them the size of entire cities, making it impossible to draw any kind of conclusions about POI. The resulting maps of those combinations can be seen in Appendix B.
We can also see that with this approach there are no longer any visible trajectories on the maps.

Table 7-Cluster Info from Stay Points method

Stay Points	POI	Silhouette	Calinski Harabasz	Davies Bouldin	Nº Clust	Nº Noise Points	Process Time (s)
DBSCAN	**DBSCAN**	-0.19	0.46	1.99	10	268	0.03
HDBSCAN		-0.20	0.44	2.00	13	321	0.03
KMEANS		-0.29	0.42	2.10	9	344	0.04
CLIQUE		-0.06	0.62	2.36	27	638	0.14
DBSCAN	**HDBSCAN**	0.21	1039.92	1.55	22	124	0.0089
HDBSCAN		0.28	3789.79	1.73	23	157	0.0099
KMEANS		0.10	541.53	1.74	17	205	0.0099
CLIQUE		0.08	188.37	1.56	51	431	0.02
DBSCAN	**KMEANS**	0.82	71727.57	0.12	7	0	0.02
HDBSCAN		0.80	180280.50	0.34	7	0	0.02
KMEANS		0.81	42290.67	0.21	8	0	0.03
CLIQUE		0.79	34538.99	0.30	8	0	0.03
DBSCAN	**CLIQUE**	0.82	13381.92	0.11	7	0	0.19
HDBSCAN		0.68	19751.41	0.20	7	0	0.23
KMEANS		0.70	6728.85	0.14	15	0	0.20
CLIQUE		0.74	5371.20	0.17	16	0	0.24

In this test the values were not very different from previous cases as we can see in Table 7.

-**Silhouette Score**: In this metric the values were higher in the KMEANS algorithm as in the previous approaches, although in the CLIQUE they were also quite high, the results were not satisfactory. The lowest was DBSCAN followed by HDBSCAN as before.

-**Calinski Harabasz**: The values of this metric behaved in exactly the same way as in the previous test, i.e. KMEANS and CLIQUE with high values derived from cluster size, and lower values for DBSCAN and HDBSCAN, this being higher due to cluster size and lower noise identification.

-**Davies Bouldin**: As in the previous metric, the results were similar, except that this time CLIQUE had the lowest values on the list.

Although these metrics are not the most suitable to validate density clustering algorithms, it is possible to verify that the best algorithms for this approach are those with the lowest Calinski Harabasz and Davies Bouldin values simultaneously. Regarding the processing time it is also interesting how the CLIQUE was the slowest of all.

4.5 Stay Points with Possible Location

In order to create more value to the previous solutions, an attempt was made to identify each POI previously found. It was decided that as a last approach, an existing POI database would be used to which a query with the coordinates of the centroid of each POI identified by the algorithms would be made. After doing a search we came to the conclusion that the best existing service was from Google, but it was paid, so a second option was used. The Open Street Maps database through a python (Overpy) Wrapper that uses the Overpass API to get access to it. The coordinates of each POI identified and a radius was used as input, the result returned consists of all locations recorded in that area (Figure 40). This database still has many places and unmapped locations, so most of

the query results for each POI were empty. In the results chapter, a validation of each volunteer's data results was made by them. In the validation of the POI, it was also requested that in the case of the POI that had the results of the query, they were also validated. Taking into consideration the lack of data in the Open Street Maps database, it was considered that any percentage of right results would always be a plus.

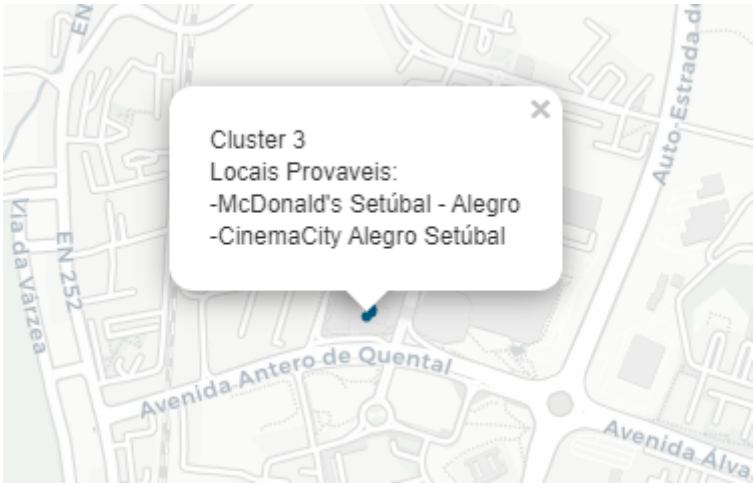

Figure 40- Example of Possible Locations in the resulting map

4.6 Regular Paths

As mentioned above, two different approaches will be used to detect regular paths. The first approach will consist of identifying individual trajectory points based on a list of conditions, followed by processing them by a snap-to-road algorithm to obtain the trajectories. The second approach will be the result of clustering all the points of each set start/destination.

4.6.1 Snap-To-Road

The first step in this approach was to process the data for each day in order to find all the starting and destination points of each trajectory. Then these data were all grouped in a

Figure 41- Example of GPS error, where the upper point is repeated more than 3 times

list and clustered using DBSCAN with eps= 30 / 6371000 (the same used for POI) and min_samples = 10. That is, the starting and destination points that occur at least 10 times were found. These data were then converted into a list that contains each route separately with the respective points and an index that defines the departure and destination. The trajectories found with the points connected by a line were then generated. But it is possible to find several flaws in this method. In the specific case of this Figure 41, the point above is a GPS error that was considered as one of the destination points.
In Figure 42 we can see another flaw. Although we can roughly perceive the trajectory, sometimes the paths used can be different which will undermine the final result making it impossible to obtain a complete trajectory from it.

Figure 42- Example of polyline different trajectories from the same group of start/destination

After some tests and analysis, it was possible to realize that most GPS errors take place when users are inside buildings and therefore the accuracy is lower. We also noticed that in these cases, the wrong points that are stored are often at exactly the same coordinates. To try to mitigate this problem, a condition has been implemented that eliminates all points that repeat more than 3 times.
After testing this method, the trajectory of Figure 41 is finally no longer considered and some points of the others have been also eliminated.
To try to get the exact trajectory, the last method of this approach is to export to individual maps all the trajectories of each set departure/destination Snapped-to-road. But how is this Snap-to-road done? One of the most important parameters of this Snap-to-road API algorithm is the ***network_type***, in which the way the route is traveled, car, train, walking, and more is selected. For this solution we decided to use the parameter **drive** because it is the most important and recurrent in most of the data. The **OSMNX API** is used and 2 points of the trajectory are processed at a time so that the snapped-to-road trajectory is generated between them until the destination point is reached. At the end of this processing, for each departure/destination route we will have several exported maps, some with similar routes and others possibly different, this is because from a departure to a destination we can go through different paths as previously mentioned in Figure 42. A list is also created with the distance of each of the courses of each trajectory and clustered

in order to obtain the most consistent distance. Then the probability of this being the real regular course is higher. The trajectories that belong to these clusters are the ones returned and sent to the volunteers for validation.
Although sometimes the trajectory is correct, the API used to generate the Snapped to road route sometimes does not work with the intended accuracy. We have detected that in many cases the algorithm does not detect any path between 2 points. Whenever this happens, we decided that on that mini-path where this error occurs, the *network_type* parameter would be changed to "walking" to check if a path is already found, if it still doesn't exist, then that trajectory is ignored.
In Figure 43 the red section represents a path generated with the walking parameter because with the drive parameter, no path was found between those two points. Looking at the map and we can conclude that in reality the path exists and therefore was a failure of the API algorithm.

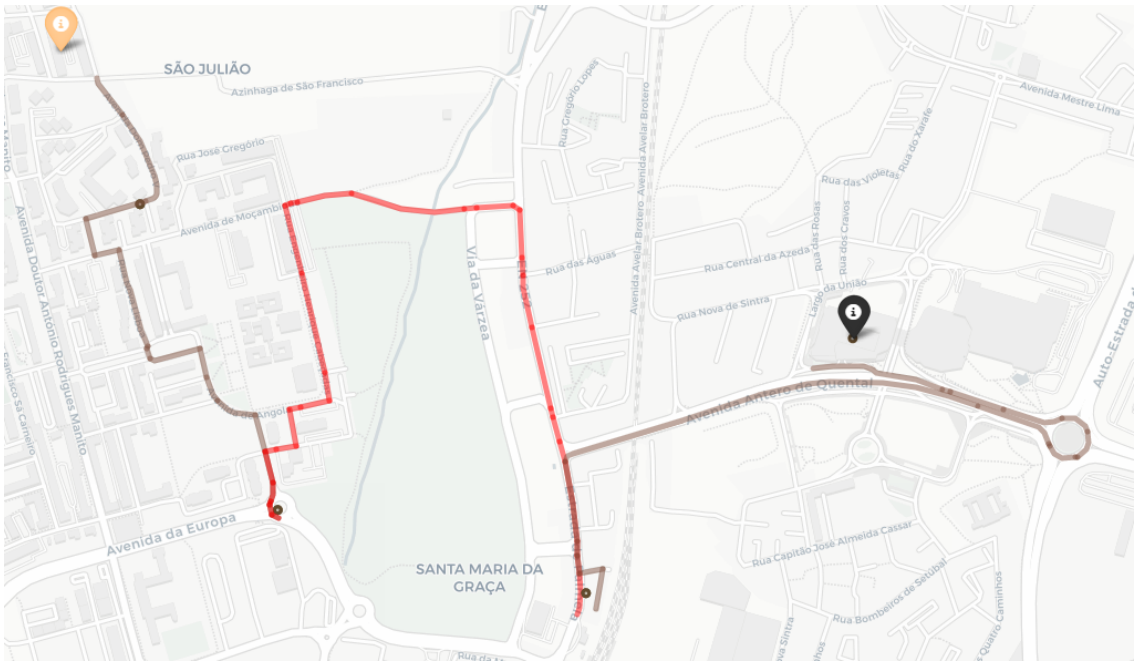

Figure 43- Example of API's "path not found" error and the "walking" solution presented by the red section

The two main problems that cause bad results for this snap-to-road algorithm are the low frequency of points during the trajectory, the existence of wrong GPS points and some bugs of the API as seen in Figure 43.
An example where the lack of points can be verified is in the case of Figure 44 where there is only one point between the starting location and the destination location. This point is actually part of an alternative route to that destination, but either the algorithm didn't find the way between the intermediate point and the destination, or it simply decided that going back and following the usual route would be faster. If there were one or more points on the alternative route, the probability of this "middle reversal" would have been much lower.

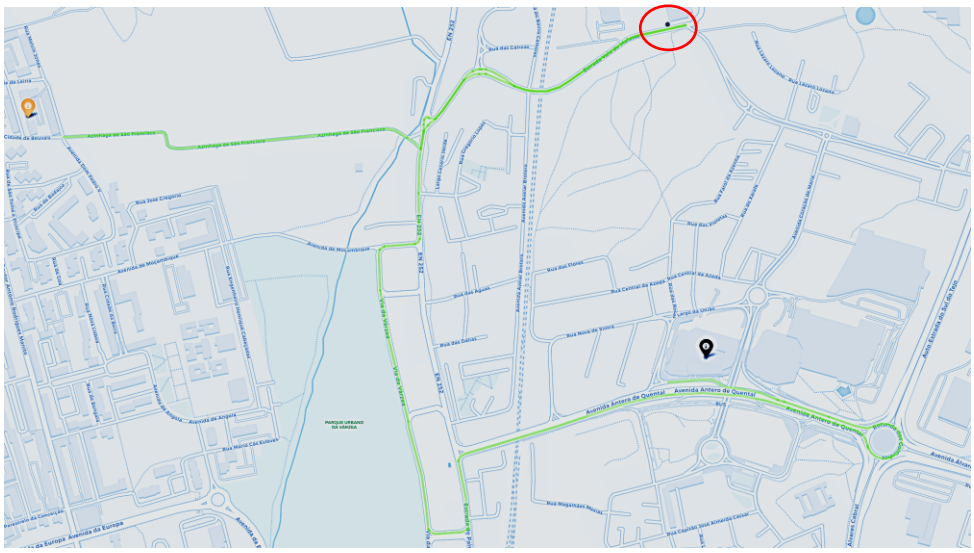

Figure 44- Lack of points limitation example

An example where GPS errors can cause trajectory problems can be seen in Figure 45 where the point being 3 or 4 meters to the side caused a small trajectory turn.

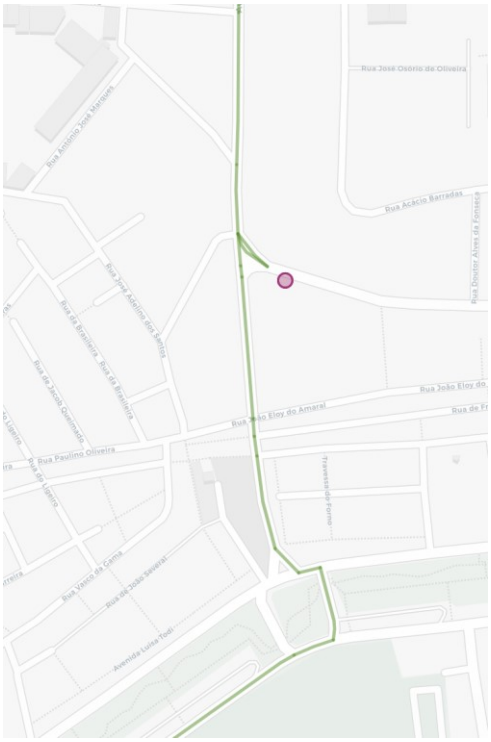

Figure 45- GPS error points limitation

4.6.2 Clustering

This second approach consists of quite simple steps. First the wrong GPS points are removed just like in the previous Snap-to-road approach when the points the repeat more than 3 times are removed, then all the points of each departure/destination set are grouped in a list regardless of the day. At the end, each set of points is clustered using DBSCAN and HDBSCAN. For DBSCAN an *eps* of 100 / 6371000 is used as in the rest of the approaches and a *min_samples* of 5 for low density and 10 for high density datasets. For HDBSCAN a *min_cluster_size* of 10 is used for low density and 15 for high density datasets.
As can be seen in Figure 46 (left-DBSCAN, right-HDBSCAN), in most cases and at low densities the algorithm only detects the starting and destination points. Also, in the same Figure, the right section shows a rare example where most of the route is in the cluster as well. Note that the grey points on the maps are outliers.

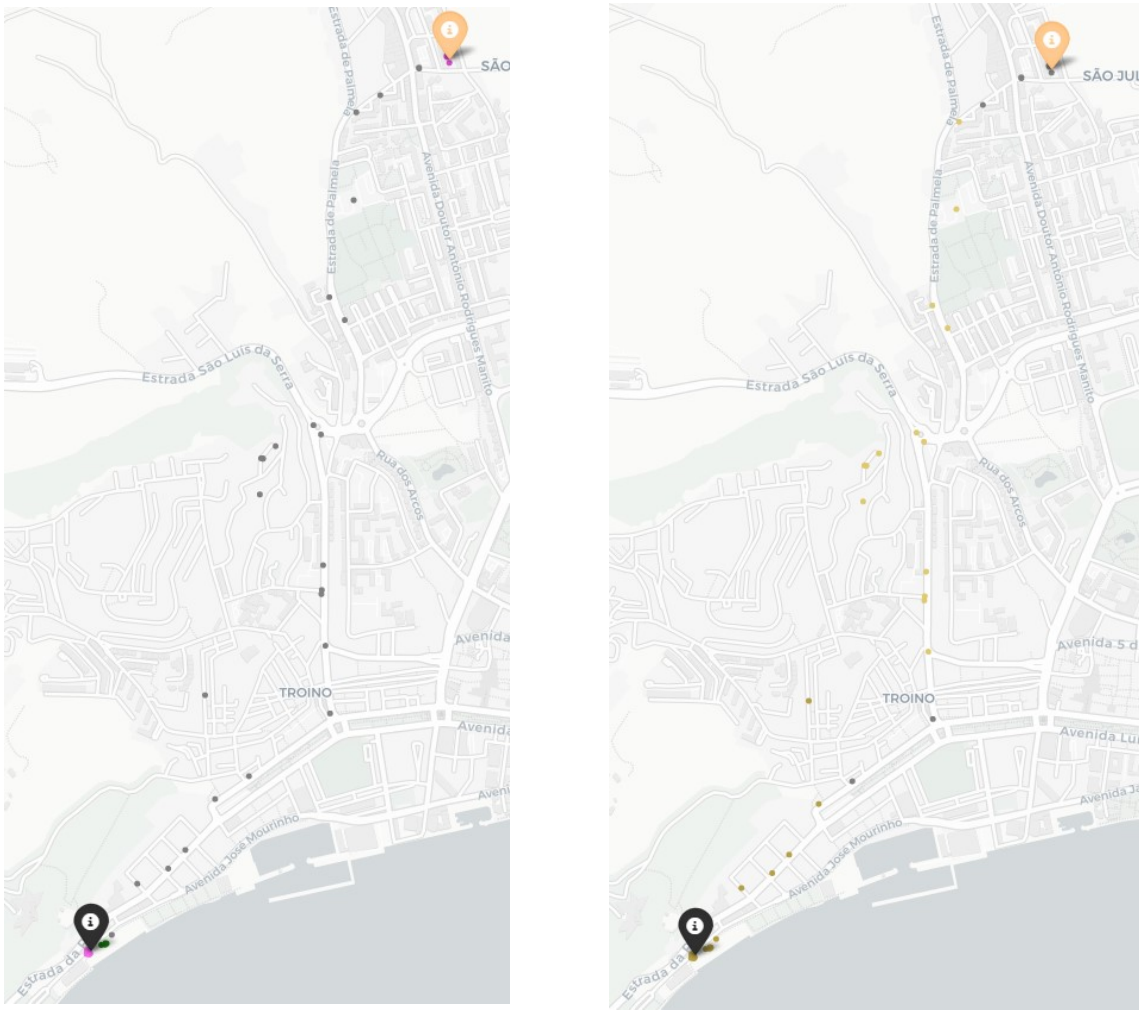

Figure 46- Example of low point density limitation when clustering trajectories

In datasets with a higher density it is already possible to identify a pseudo trajectory because not only one cluster is detected but several very close together (Figure 47).

Figure 47- Example of high density trajectory clustering

5 Results and Evaluation

In this chapter the solutions proposed above have been tested so that it is possible to conclude which are the best and whether they really solve the initial problem of this project. This test was done using several volunteers who agreed to share their location data.

5.1 POI

For POI detection 4 algorithms using Stay Points, DBSCAN_DBSCAN, HDBSCAN_DBSCAN, KMEANS_DBSCAN and CLIQUE_DBSCAN were proposed in the previous chapter. Taking this into consideration, the data of the volunteers were processed in order to be generated using the algorithms, 4 folium maps with the results obtained. The 4 corresponding Excel files containing the index, the average coordinates, the number of points and the probable locations of each cluster were also generated. There is also an empty column in these files with the label "check" where each volunteer confirmed if the cluster was indeed a POI and if the estimated possible location was true. The results can be seen in Table 8.

Table 8-POI evaluation results

Algorithms	***Total Points***	***Confirmed***	***Not Confirmed***	***Success %***
DBSCAN_DBSCAN	*92*	*80*	*12*	*86.95*
HDBSCAN_DBSCAN	*110*	*94*	*16*	*85.45*
KMEANS_DBSCAN	*51*	*38*	*13*	*74.50*
CLIQUE_DBSCAN	*100*	*70*	*30*	*70*

Table 9-Possible Locations evaluation results

Algorithms	**Total Nº of Possible Locations**	**Confirmed**	**Not Confirmed**	**Success %**
DBSCAN_DBSCAN	15	10	5	66.666
HDBSCAN_DBSCAN	14	13	1	64.285
KMEANS_DBSCAN	6	5	1	83.333
CLIQUE_DBSCAN	12	6	6	50.000

5.2 Regular Paths

For the detection of regular paths 2 different approaches were proposed in the previous chapter. As the clustering approach only resulted in experiments with extremely specific data sets, particularly with high density data and very short paths, we decided that it made no sense to test it and therefore to obtain the results we would only use the Snap-to-Road approach, because although this is somewhat fragile in relation to some specific data, it is also the only one that returns complete paths and with some conditions it was possible to extract the right ones.

As mentioned above, to generate these trajectories all sets of points that had the same starting point and destination regardless of the path taken were added to a list. This was done for all departure/destination sets. Then the distance of all these trajectories from each set was calculated and a clustering made on them. In the end, the largest cluster was selected, which also contained trajectories with very similar distances. After these conditions, the probability of the trajectory of this cluster with less path distance being the right one and with less errors is much higher.
To confirm this method, the resulting trajectory of each set was sent to the volunteers to confirm its validity by evaluating from 0 to 5.
The results are shown in Table 10.

Table 10-Paths evaluation results

Nº Paths	0	1	2	3	4	5
103	27	6	10	10	18	32

6 Discussion

This chapter represents the findings of this entire research process and consists of a brief discussion of the initial topic proposed and methods found in the literature. It also mentions the type of dataset chosen and what solutions are proposed to solve the problem. Finally, a discussion is made about the results obtained through the proposed algorithms, its limitations and future research.

According to the literature, people's GPS data is increasingly being used by applications to study their habits. This information is then used in many different types of applications or even for advertising purposes.

As defined at the beginning of this dissertation, the final objective is to identify user's behavior based on location data. But identifying user behavior is a difficult task composed of many layers, so as a first layer of processing on the way to this objective, this work aims to identify individual points of interest for each user and the most frequent journeys made.

The objective was also to extract and evaluate the information that can be extracted from mobile phone data. As most smartphones today have Google services, it was decided to use the location data in the format stored by these services.

In the reviewed literature, although there are several methods used to solve the problem of POI identification, the best results reported were, surprisingly, all obtained by clustering algorithms, so four clustering algorithms, DBSCAN, HSBSCAN, CLICK and KMEANs were chosen for the experiments. Although these algorithms were tested on the complete GPS data and also in datasets divided by speed, the results were never close to the ones reported because in many cases the speed is not enough to separate POI from trajectories, and, these methods do not guarantee that each POI identified has points of different days. The solution was clustering the data daily, identifying the stay points, and then clustering the stay points obtaining the POI, as proposed by some authors in the literature. All the algorithm combinations were tested in this two-step clustering and the best ones were identified: identifying stay points by daily clustering using each one of the previously mentioned four algorithms followed by clustering the results (stay points) using DBSCAN.

An API was also implemented which, through a set of coordinates, returned all the POI of the OpenStreetMaps database. For each cluster, a query was made to this database to try to "predict" each POI.

The data of each volunteer was processed by each of the 4 combinations resulting in 4 maps with the corresponding clusters and 4 excel files. These files were then sent to the volunteers and they confirmed if each cluster was indeed true and if the "predicted" location was correct.

Data from 9 volunteers was used, and as can be seen from Table 8 in the previous chapter, the combination with the highest success rate was **DBSCAN_DBSCAN** with a correct POI value of **86.95%**. Looking at Table 9 it was possible to see that 83.33% of all possible locations discovered by the KMEANS_DBSCAN combination were correct, it should be noted that most of the possible locations of this combination came from the data of only one volunteer, we believe that by obtaining a larger number of volunteers the results would be different making DBSCAN_DBSCAN the most successful. Cross-checking the data in the two tables we can conclude that for the best combination (DBSCAN_DBSCAN), 12.50% of the correct POI were also correctly predicted using the OpenStreetMaps database. We consider that this was a very good result and that it

fulfilled the objectives in a positive way, having thus identified most of the POI of all the datasets obtained.
To solve the problem of identifying recurring paths the solution was developed from the ground up as there is not much information in the literature. As mentioned in chapter four, two methods were tested, one of them was simply to create a list with the start and end points of each trajectory, to add all the points of the trajectories with corresponding start/end and to cluster each of these sets. In the end the results of the experiment were not acceptable so this method was not considered as a solution.
The other method, made from scratch, consisted of developing an algorithm that was able to identify trajectories using only the points and their temporal information. A list with all starting and ending points was created and individual maps were generated with all the points of the trajectories that occurred more than ten times. An API was used to Snap-to-Road these points to obtain a trajectory. As it is possible that from point A to point B there are several possible routes, the distance of each of these paths of each group of trajectories with the same place of origin and destination was calculated. Having the distance from each of these routes with the same starting point and destination, the predominant distance value in the list is identified and the corresponding path is returned.
Each of these trajectories was sent to the volunteers for evaluation with a score from zero to five. According to Table 10, of the six volunteers, 103 regular trajectories were found where 31.06% were evaluated with five points, 17.47% with four points, 9.70% with three points, 9.70% with two points, 5.82% with one point and 26.21% with zero points. Some of the trajectories received bad scores because the algorithm is made to consider trajectories in vehicles on the road, as some of the trajectories of the volunteers are made on foot, the algorithm sometimes gives incorrect results. We consider that for road trajectories, and taking into consideration all the limitations that were presented, this algorithm produced positive results that respond to the initial problem presented.

6.1 Research Limitations

In the course of this project and more specifically in the experiment phase, it was possible to identify several limitations that did not allow us to achieve better results.
The first limitation found was some lack of literature concerning the identification of regular paths.
Another limitation is the lack of GPS data density. When the location is on, smartphones only save GPS data on average every 5 minutes, which makes it difficult to identify some points of interest and makes the trajectories very inaccurate, because within 5 minutes the user may have used several different paths that were not saved.
It should be noted that in these circumstances, the accuracy of the GPS is low and therefore there were several wrong GPS points, particularly while the person was inside buildings. Although there was an attempt to identify and ignore these points, some remained and made it more difficult to identify trajectories,
For the identification of possible locations, the OSMNX API for OpenStreetMaps was used, but as this does not have as much information as for example Google Maps, not many probable locations were presented.
This API was also used to perform the Snap-to-road between several points and this is where we found other limitations, namely errors that appeared due to not having found a path between two points, even though they were on the same road. There were also many points that although they were far away from the others, they had a distance of 0 in relation to the following points, which made them have to be discarded.

Finally, there was also a limitation regarding the evaluation of clusters because although four metrics were used that show some facts about them, there is none that reasonably evaluates clustering algorithms of varying densities.

6.2 Future Research

For future research, if it is not imperative to use data from Google, it is recommended to use others that have been recorded with a higher frequency than those from google recorded every 5 minutes.
It is also very important to understand how incorrectly recorded GPS data can be better identified in order to improve trajectory identification results.
Something that can also improve the identification of POI is to produce or find a method to evaluate each of the clusters found, so that several "eps" and "min_samples" values can be tested end the perfect ones found in the DBSCAN algorithm.
For the identification of possible locations, it was concluded that OpenStreetMaps still contains a reduced database and its API still contains some bugs that compromise the result of Snap-to-road trajectories. Therefore, it is advisable to use a more complete API, such as Google's if there are enough funding for it.
Regarding the identification of regular trajectories, we think that there is still a lot of ground to explore. Joining some aspects of the approach presented, with clustering and time information would probably be a way to improve the results obtained.

Bibliography

Agrawal, Rakesh, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1998. “Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications.” In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’98, New York, NY, USA: ACM, 94–105. http://doi.acm.org/10.1145/276304.276314.

Angkhawey, U, and V Muangsin. 2018. “Detecting Points of Interest in a City from Taxi GPS with Adaptive DBSCAN.” In *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, , 1–6.

Ankerst, Mihael, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. “OPTICS: Ordering Points to Identify the Clustering Structure.” In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’99, New York, NY, USA: ACM, 49–60. http://doi.acm.org/10.1145/304182.304187.

Ashbrook, Daniel, and Thad Starner. 2003a. “Starner, T.: Using GPS to Learn Significant Locations and Predict Movement across Multiple Users. Personal and Ubiquitous Computing 7(5), 275-286.” *Personal and Ubiquitous Computing* 7: 275–86.

———. 2003b. “Using GPS to Learn Significant Locations and Predict Movement across Multiple Users.” *Personal and Ubiquitous Computing* 7(5): 275–86.

Birant, Derya, and Alp Kut. 2007. “ST-DBSCAN: An Algorithm for Clustering Spatial–Temporal Data.” *Data & Knowledge Engineering* 60(1): 208–21. http://www.sciencedirect.com/science/article/pii/S0169023X06000218.

Borah, B, and D K Bhattacharyya. 2004. “An Improved Sampling-Based DBSCAN for Large Spatial Databases.” In *International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings Of*, , 92–96.

———. 2007. “A Clustering Technique Using Density Difference.” In *2007 International Conference on Signal Processing, Communications and Networking*, , 585–88.

Borah, Bhogeswar, and Dhruba K Bhattacharyya. 2008. “DDSC : A Density Differentiated Spatial Clustering Technique.” *Journal of Computers* 3.

Brock, Jürgen Kai-Uwe, and Florian von Wangenheim. 2019. “Demystifying AI: What Digital Transformation Leaders Can Teach You about Realistic Artificial Intelligence.” *California Management Review* 61(4): 110–34. https://doi.org/10.1177/1536504219865226.

Caliński, T., and J Harabasz. 1974. “Communications in Statistics - Theory and Methods.” *Communications in Statistics* 3(1): 1–27.

Campello, Ricardo J.G.B., Davoud Moulavi, and Joerg Sander. 2013. “Density-Based Clustering Based on Hierarchical Density Estimates.” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7819 LNAI(PART 2): 160–72.

Cao, Xin, Gao Cong, and Christian S Jensen. 2010. “Mining Significant Semantic Locations from GPS Data.” *Proc. VLDB Endow.* 3(1–2): 1009–20. http://dx.doi.org/10.14778/1920841.1920968.

Chowdhury, Nirmalya, and Preetha Bhattacharjee. 2014. “Using an MST Based Value for ε in DBSCAN Algorithm for Obtaining Better Result.” *International Journal of Information Technology and Computer Science* 6: 55–60.

Davies, David L., and Donald W. Bouldin. 1979. “A Cluster Separation Measure.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1(2): 224–27.

Duan, Lian et al. 2007. “A Local-Density Based Spatial Clustering Algorithm with Noise.” *Information Systems* 32(7): 978–86. http://www.sciencedirect.com/science/article/pii/S0306437906000871.

Elbatta, Mohammed, and Wesam Ashour. 2013. “A Dynamic Method for Discovering Density Varied Clusters.” *International Journal of Signal Processing, Image Processing and Pattern Recognition* 6: 123–34.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. “A Density-Based Algorithm for Discovering Clusters a Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*,

KDD'96, AAAI Press, 226–31.
http://dl.acm.org/citation.cfm?id=3001460.3001507.

Fu, Zhongliang, Zongshun Tian, Yanqing Xu, and Changjian Qiao. 2016. "A Two-Step Clustering Approach to Extract Locations from Individual GPS Trajectory Data." *ISPRS International Journal of Geo-Information* 5: 166.

Gaonkar, Manisha Naik, and Kedar Sawant. 2013. "AutoEpsDBSCAN : DBSCAN with Eps Automatic for Large Dataset."

Ghahramani, Zoubin. 2019. "Https://Eng.Uber.Com/Applying-Artificial-Intelligence-at-Uber/." https://eng.uber.com/applying-artificial-intelligence-at-uber/.

Gonzalez, Marta C, Cesar Hidalgo, and Albert-Laszlo Barabasi. 2008. "Understanding Individual Human Mobility Patterns." *Nature* 453: 779–82.

Haenlein, Michael, and Andreas Kaplan. 2019. "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence." *California Management Review* 61(4): 5–14. http://10.0.4.153/0008125619864925.

He, Yaobin et al. 2014. "MR-DBSCAN: A Scalable MapReduce-Based DBSCAN Algorithm for Heavily Skewed Data." *Frontiers of Computer Science* 8.

Herrera, Juan C et al. 2010. "Evaluation of Traffic Data Obtained via GPS-Enabled Mobile Phones: The Mobile Century Field Experiment." *Transportation Research Part C: Emerging Technologies* 18(4): 568–83.
http://www.sciencedirect.com/science/article/pii/S0968090X09001430.

Hinneburg, Alexander, and Daniel A Keim. 1998. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise." In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, AAAI Press, 58–65. http://dl.acm.org/citation.cfm?id=3000292.3000302.

Karypis, G, Eui-Hong Han, and V Kumar. 1999. "Chameleon: Hierarchical Clustering Using Dynamic Modeling." *Computer* 32(8): 68–75.

Lian, Defu, and Xing Xie. 2011. "Learning Location Naming from User Check-in Histories." In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, New York, NY, USA: ACM, 112–21. http://doi.acm.org/10.1145/2093973.2093990.

Lin, Chih-Yang, Chin-Chen Chang, and Chia-Chen Lin. 2005. "A New Density-Based Scheme for Clustering Based on Genetic Algorithm." *Fundamenta Informaticae* 68(4): 315–31. http://widgets.ebscohost.com/prod/customerspecific/ns000290/authentication/index.php?url=https%3A%2F%2Fsearch.ebscohost.com%2Flogin.aspx%3Fdirect%3Dtrue%26AuthType%3Dip%2Ccookie%2Cshib%2Cuid%26db%3Da9h%26AN%3D18316605%26lang%3Dpt-pt%26site%3Deds-live%26sc.

Liu, Yanchi et al. 2010. "Understanding of Internal Clustering Validation Measures." *Proceedings - IEEE International Conference on Data Mining, ICDM*: 911–16.

Liu, Yaqiong, and Hock Soon Seah. 2015. "Points of Interest Recommendation from GPS Trajectories." *International Journal of Geographical Information Science* 29(6): 953–79. https://doi.org/10.1080/13658816.2015.1005094.

Malzer, Claudia, and Marcus Baum. 2019. "A Hybrid Approach To Hierarchical Density-Based Cluster Selection." : 1–17. http://arxiv.org/abs/1911.02282.

Milenova, Boriana, and Marcos Campos. 2002. Proceedings - IEEE International Conference on Data Mining, ICDM *O-Cluster: Scalable Clustering of Large High Dimensional Data Sets*.

Mitra, Sushmita, and Jay Nandy. 2011. "KDDClus : A Simple Method for Multi-Density Clustering."

Montoliu, Raul, Jan Blom, and Daniel Gatica-Perez. 2013. "Discovering Places of Interest in Everyday Life from Smartphone Data." *Multimedia Tools and Applications* 62(1): 179–207. http://infoscience.epfl.ch/record/192536/files/Montoliu_MTA_2012.pdf.

Moreira, Adriano, and Maribel Yasmina Santos. 2005. "1 Density-Based Clustering Algorithms DBSCAN and SNN."

Nishida, Kyosuke, Hiroyuki Toda, Takeshi Kurashima, and Yoshihiko Suhara. 2014. "Probabilistic Identification of Visited Point-of-Interest for Personalized Automatic Check-In." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, New York, NY, USA: ACM, 631–42. http://doi.acm.org/10.1145/2632048.2632092.

Paola, A de, A Giammanco, G lo Re, and G Anastasi. 2019. "Detection of Points of

Interest in a Smart Campus." In *2019 IEEE 5th International Forum on Research and Technology for Society and Industry (RTSI)*, , 155–60.

Parsons, Simon. 2011. "Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman, MIT Press, 1231 Pp., $95.00, ISBN 0-262-01319-3." *The Knowledge Engineering Review* 26(2): 237–38. https://www.cambridge.org/core/article/probabilistic-graphical-models-principles-and-techniques-by-daphne-koller-and-nir-friedman-mit-press-1231-pp-9500-isbn-0262013193/EB33DB2063BB50BBF6FD6C0087A7537C.

Peng, Liu, Zhou Dong, and Wu Naijun. 2007. "VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise." In *Proceedings - ICSSSM'07: 2007 International Conference on Service Systems and Service Management*, , 1–4.

Ram, A et al. 2009. "An Enhanced Density Based Spatial Clustering of Applications with Noise." In *2009 IEEE International Advance Computing Conference*, , 1475–78.

Ram, Anant, Jalal Sunita, Anand Jalal, and Kumar Manoj. 2010. "A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases." *International Journal of Computer Applications* 3.

Rousseeuw, Peter J. 1987. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Journal of Computational and Applied Mathematics* 20(C): 53–65.

Sander, Jörg, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1998. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications." *Data Mining and Knowledge Discovery* 2(2): 169–94. https://doi.org/10.1023/A:1009745219419.

Schubert, Erich et al. 2017. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN." *ACM Transactions on Database Systems* 42: 1–21.

Shaw, Blake, Jon Shea, Siddhartha Sinha, and Andrew Hogue. 2013. WSDM 2013 - Proceedings of the 6th ACM International Conference on Web Search and Data Mining *Learning to Rank for Spatiotemporal Search*.

Sheikholeslami, Gholamhosein, Surojit Chatterjee, and Aidong Zhang. 1998. "WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial

Databases." In *VLDB*,.

SlashData, and Developer Economics. 2020. *State of the Developer Nation - 18th Edition*. https://s3-eu-west-1.amazonaws.com/vm-blog/uploads/2019/04/SoN_16_report.pdf.

Wang, Wei, Jiong Yang, and Richard R Muntz. 1997. "STING: A Statistical Information Grid Approach to Spatial Data Mining." In *VLDB*,.

Wang, Xin, and Howard Hamilton. 2003. 2637 Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science) *DBRS: A Density-Based Spatial Clustering Method with Random Sampling*.

Xiong, Z, R Chen, Y Zhang, and X Zhang. 2012. "Multi-Density DBSCAN Algorithm Based on Density Levels Partitioning." *Journal of Information and Computational Science* 9: 2739–49.

Xu, Xiaowei, Jochen Jäger, and Hans-Peter Kriegel. 1999. "A Fast Parallel Clustering Algorithm for Large Spatial Databases." *Data Mining and Knowledge Discovery* 3(3): 263–90. https://doi.org/10.1023/A:1009884809343.

Zaïane, Osmar, Andrew Foss, Chi-Hoon Lee, and Weinan Wang. 2002. Proc. of PAKDD *On Data Clustering Analysis: Scalability, Constraints, and Validation*.

Zhang, L, C Chen, Y Wang, and X Guan. 2016. "Exploiting Taxi Demand Hotspots Based on Vehicular Big Data Analytics." In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, , 1–5.

Zhou, Changqing et al. 2004. "Discovering Personal Gazetteers: An Interactive Clustering Approach." In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*, GIS '04, New York, NY, USA: ACM, 266–73. http://doi.acm.org/10.1145/1032222.1032261.

Zhou, Changqing, Nupur Bhatnagar, Shashi Shekhar, and Loren Terveen. 2007. *Mining Personally Important Places from GPS Tracks*.

Appendices

Appendix A

Algorithms	Experiments	Results	References
VDBSCAN	Pre select different Eps values for different densities according to a k-dist plot	VDBSCAN is able to find clusters with varying densities, and has the same time complexity as DBSCAN	(Peng, Dong, and Naijun 2007)
DMDBSCAN	Pre select different Eps values for different densities according to a k-dist plot	DMDBSCAN is able to find clusters with varying densities, better result than DBSCAN and DVBSCAN	(Elbatta and Ashour 2013)
AutoEpsDBSCAN	Pre select different Eps values for different densities according to a k-dist plot	AutoEpsDBSCAN is able to find clusters with varying densities, better result than DBSCAN , VDBSCAN and only requires one input parameter	(Gaonkar and Sawant 2013)
GDBSCAN	Cluster points according to both, their spatial and their nonspatial attributes	Showed the effectiveness and efficiency on large spatial databases	(Sander et al. 1998)
"Haversine" DBSCAN	Automatically determine the parameters of DBSCAN according to pick-up and drop-off locations of taxis in Bangkok	Better results than DMDBSCAN in all 3 types of areas (low, medium and high density population)	(Angkhawey and Muangsin 2018)
DJ-Cluster	Explores the use of novel semi-automatic techniques to discover gazetteers from users' travel patterns	Although data from only the author was used, the results were superior compared to KMEANS	(Zhou et al. 2004)
ST-DBSCAN	Clusters spatial–temporal data according to its non-spatial, spatial and temporal attributes. Solves noise points not detected when different densities exist and	Promising algorithm when spatial-temporal data is used to be clustered	(Birant and Kut 2007)

	improves adjacent cluster detection.		
OPTICS	Algorithm that creates an augmented ordering of the database representing its density-based clustering structure.	Suitable for interactively exploring the clustering structure, offering addi- tional insights into the distribution and correlation of the data	(Ankerst et al. 1999)
GADAC	GADAC applies a genetic algorithm to regulate the radii of all circles, and an interactive system to determine the proper parameters to provide satisfactory shape covering.	Results demonstrate that the noise and all clusters in any data shapes can be identified precisely in the proposed scheme.	(Lin, Chang, and Lin 2005)
EDBSCAN	Enhanced DBSCAN algorithm which keeps track of local density variation within the cluster.	The proposed clustering algorithm could find clusters that represent relatively uniform regions without being separated by sparse regions.	(A Ram et al. 2009)
DD_DBSCAN	Extension of DBSCAN so that it can also detect clusters that differ in densities.	The clustering algorithm could find clusters that represented relatively uniform regions without being separated by sparse regions.	(B Borah and Bhattacharyya 2007)
DDSC	Extension of DBSCAN so that it can also detect clusters that differ in densities.	Clusters extracted from a dataset are non-overlapped spatial regions having different densities. Adjacent clusters may be very close, without being separated by any sparse regions.	(Bhogeswar Borah and Bhattacharyya 2008)
DVBSCAN	Proposal of a density varied DBSCAN algorithm which is capable to handle local density variation within the cluster.	The proposed clustering algorithm can find clusters that represent relatively uniform regions without being	(Anant Ram et al. 2010)

		separated by sparse regions.	
CHAMELEON	By basing its selections on both interconnectivity and closeness, the Chameleon algorithm yields accurate results for these highly variable clusters.	Chameleon's performance was compared against CURE and DBScan on four different data sets. It was able to detect genuine clusters better than the others	(Karypis, Han, and Kumar 1999)
SNN	Implementation of two density-based clustering algorithms: DBSCAN and SNN within the context of the LOCAL project as part of a task that aims to create models of the geographic space (Space Models) to be used in context-aware mobile systems.	The results show that SNN performs better than DBSCAN since it can detect clusters with different densities while DBSCAN cannot.	(Moreira and Santos 2005)
KDDClus	Serves as an enhancement to DBSCAN averaging the distances of a pattern to all k of its nearest neighbors smoothing out of noise while automatically detecting the "knees" from the k-distance plot.	The algorithm KDDClus is an enhancement to DBSCAN, in terms of automatically estimating the various density-based parameters for optimal clustering. Unlike DBSCAN, where only the kth nearest neighbor is considered during the distance computation	(Mitra and Nandy 2011)
LDBSCAN	A new clustering algorithm LDBSCAN relying on a local-density-based notion of clusters is proposed.	The experiments show that the proposed algorithm provides better results than OPTICS and overcomes the short-coming of DBSCAN. In addition, it takes the advantage of the LOF (local outlier factor) to detect the noises compared with other	(Duan et al. 2007)

		density-based clustering algorithms.	
DBSCAN-DLP	DBSCAN-DLP partitions a dataset into different density level sets by analyzing the statistical characteristics of its density variation, and then estimates Eps for each density level set, finally adopts DBSCAN clustering on each density level set with corresponding Eps to get clustering results.	Excellent performance on both synthetic and real-word datasets. It's time and I/O consuming when input dataset is enormously large.	(Xiong et al. 2012)
MDBSCAN	Testing MST (minimal spanning tree) based Value for eps in DBSCAN Algorithm to Obtaining Better Results	The modified version of the DBSCAN algorithm has successfully provided the natural group present in all the synthetic data set considered for experimentation	(Chowdhury and Bhattacharjee 2014)
MR-DBSCAN	MR-DBSCAN, a scalable DBSCAN algorithm using MapReduce, all the critical sub-procedures are fully parallelized so, there is no performance bottleneck caused by sequential processing.	Experimental results demonstrate that the algorithm can solve large-scale dataset with balanced load and have efficient speed-up and scale-up for skewed big data.	(He et al. 2014)
DBRS	The algorithm can identify clusters of widely varying shapes, clusters of varying densities, clusters which depend on non-spatial attributes, and approximate clusters in very large databases.	Experiments suggest that the algorithm scales well on high-density clusters. DBRS can deal with a property related to non-spatial attribute(s), by means of a purity threshold, when finding the matching neighborhood.	(X. Wang and Hamilton 2003)
HDBSCAN	Obtain a flat partition consisting of only the most significant clusters formulating the problem of maximizing the	Experimental results on a wide variety of real world data sets show better performance than other	(Campello, Moulavi, and Sander 2013)

	overall stability of selected clusters.	recent solutions to this problem.	

Appendix B

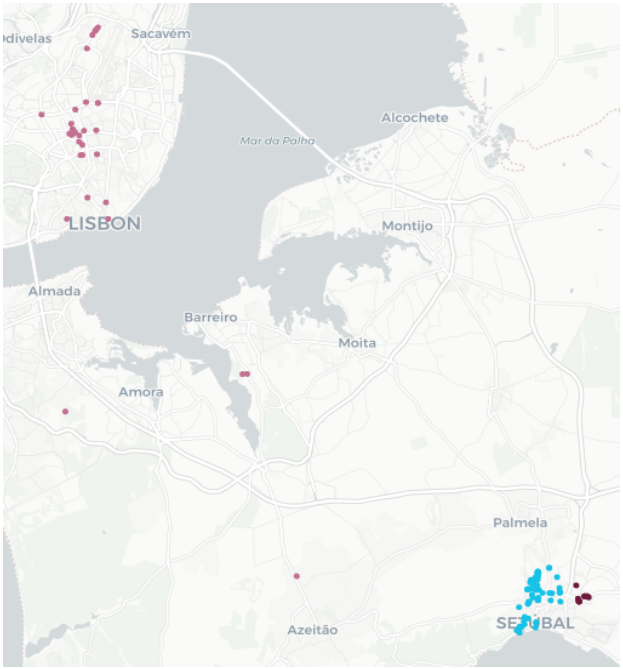

Figure 50- Resulting HDBSCAN clustering from DBSCAN Stay Points

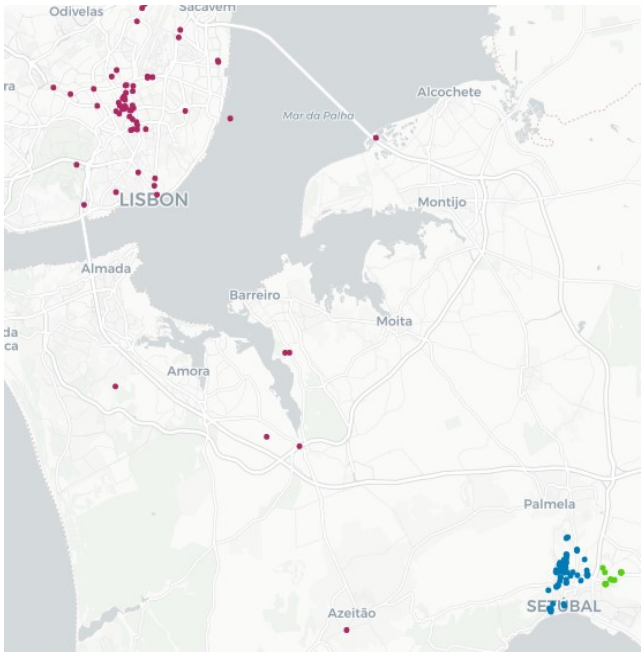

Figure 51- Resulting HDBSCAN clustering from HDBSCAN Stay Points

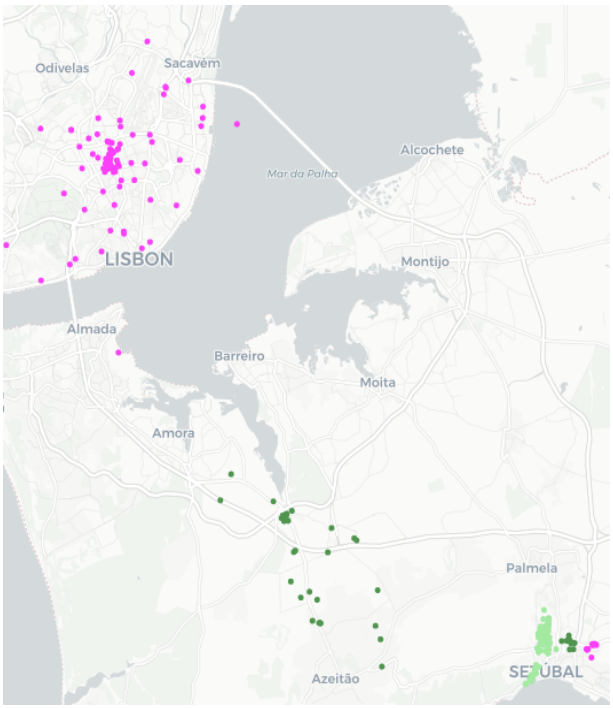

Figure 49- Resulting HDBSCAN clustering from KMEANS Stay Points

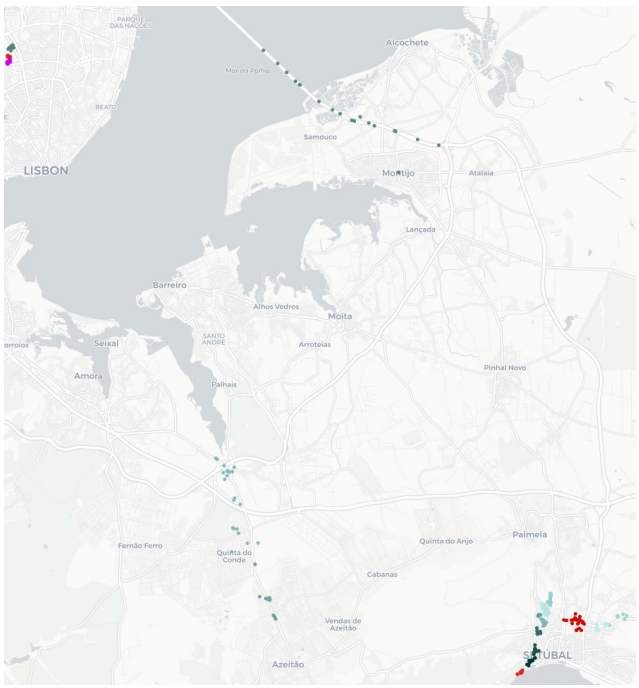

Figure 48- Resulting HDBSCAN clustering from CLIQUE Stay Points

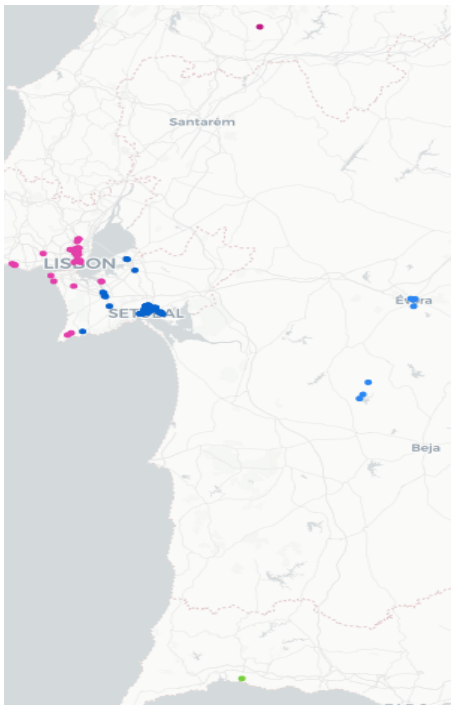

Figure 53- Resulting KMEANS clustering from DBSCAN Stay Points

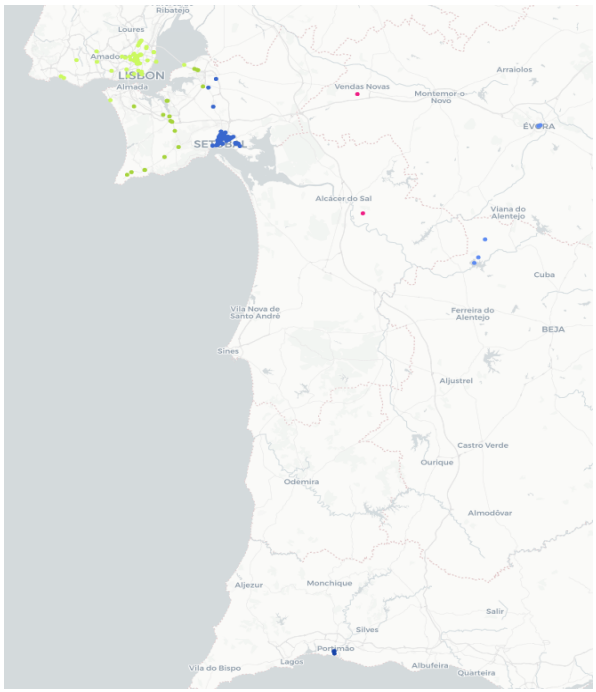

Figure 54- Resulting KMEANS clustering from HDBSCAN Stay Points

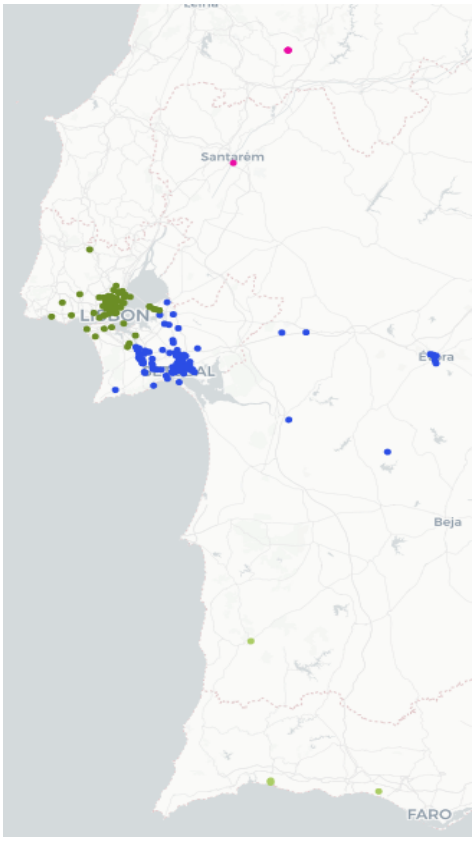

Figure 52- Resulting KMEANS clustering from KMEANS Stay Points

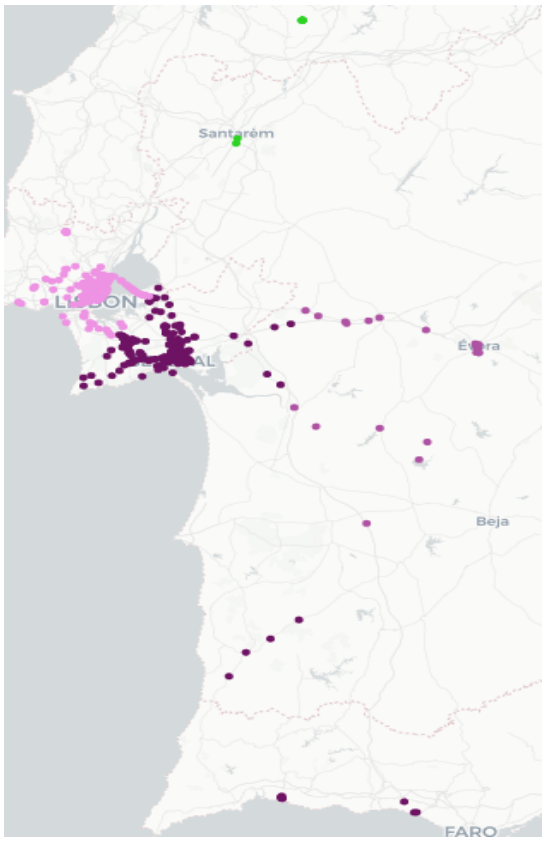

Figure 55- Resulting KMEANS clustering from *CLIQUE Stay Points*

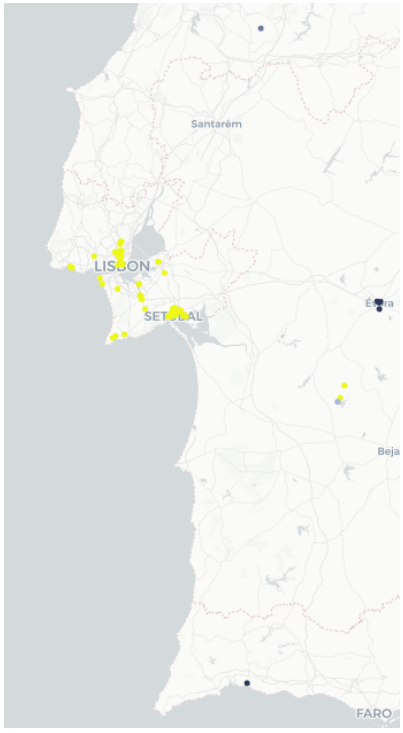

Figure 59- Resulting CLIQUE clustering from *DBSCAN Stay Points*

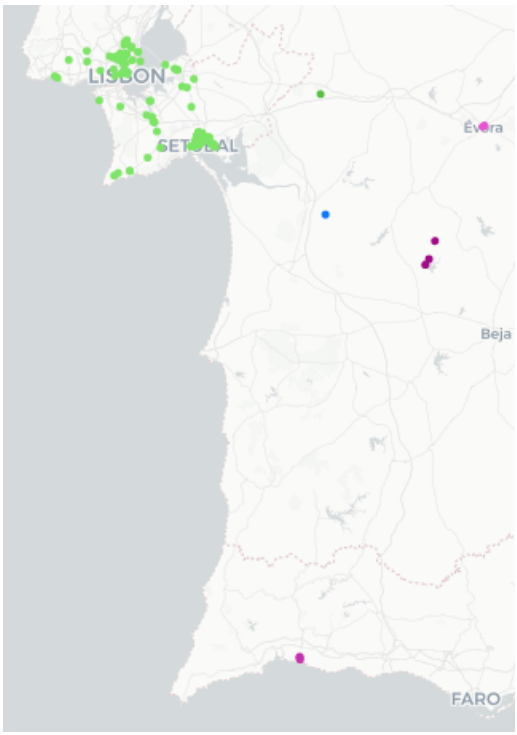

Figure 58- Resulting CLIQUE clustering from *HDBSCAN Stay Points*

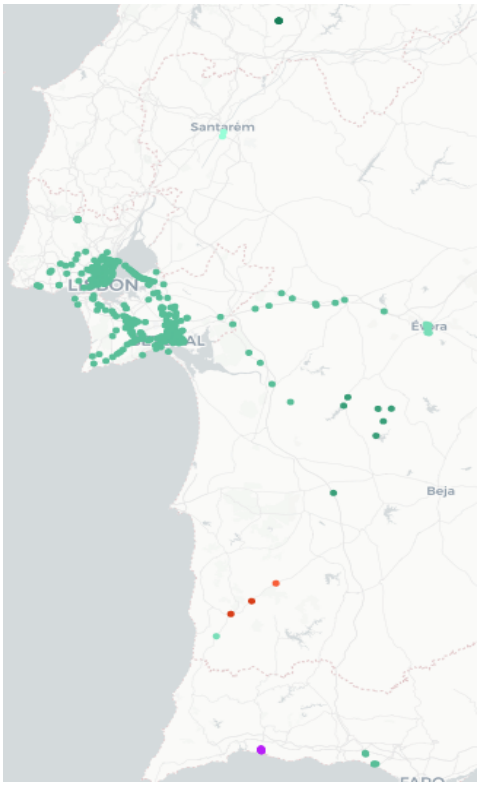

Figure 56- Resulting CLIQUE clustering from *KMEANS Stay Points*

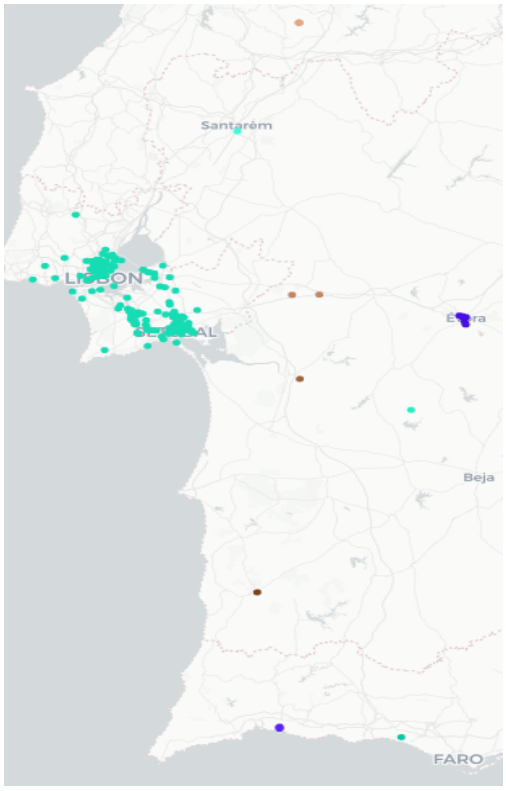

Figure 57- Resulting CLIQUE clustering from *CLIQUE Stay Points*