Scooter Trajectories Clustering

University of Verona
Computer Engineering for Robotics and Smart Industry
Machine Learning and Deep Learning
2020/2021

Mirco De Marchi - VR445319

***Abstract*—The aim of this report is to present the main unsupervised learning techniques used for scooter trajectories clustering. The dataset that I used contains a big amount of positions taken in rentals that run through some cities in Italy. The objective is to find recurrent places crossed by the trajectories. This report presents some analysis on the positions and an implementation of heuristics that manage data in a systematic way: *timedelta heuristic*, *spreaddelta heuristic*, *edgedelta heuristic* and *coorddelta heuristic*. Then I performed the most traditional machine learning clustering techniques on the generated dataset: *K-Means*, *Mean Shift*, *Gaussian Mixture*, *Full Hierarchical Agglomerative*, *Ward Hierarchical Agglomerative*. All the features are extracted with *Principal Component Analysis (PCA)* with an improvement that select the components to focus on and have been integrated with the informations obtained from the heuristic procedures in order to optimize feature extraction.**

I. Motivation

Motion trajectories are really difficult to analyse and handle because of the amount of data. There could be errors in position tracking, due to localization issues, and usually data are not organized as you expect. Consequently trajectories are difficult to represent, filter and manage in relation with themselves or other informations. First of all, trajectories clustering is a challenge for its intrinsic difficulty in being treated and today is an ambitious topic in data science research. Moreover, trajectory clustering can help for several applications:

- Monitoring: understand main points of interest and common places visited by tourists;
- Forecasting: prediction of possible destinations starting from the current position and previous ones;
- Viability: traffic monitoring and kind of user's activity extracting semantic concepts from trajectories;
- Smart city: data support to city plan and smart transportation management;
- Security: trajectories which are significantly different from others in terms of some similarity metric may be viewed as outliers;
- Video analysis: movement pattern analysis from video (after extracting trajectories from video data);

II. State of art

Yuan et al. (2012) [1], in their trajectory clustering review, show some techniques widely used for moving object clustering. The most advanced deep learning techniques use an architecture called *Auto-Encoder*. Instead, in machine learning, there are some very recent algorithms, but all evolution and improvement of the traditional clustering techniques. On basis of full analysis on moving object clustering, the algorithms can be divided into 5 categories which are listed as follows:

- *Spatial based clustering*: find out trajectories which are similar in geometrical properties. *Palma et al. (2008)* [2] detected pauses and moves in trajectories by using an improved version of *DBSCAN* algorithm.
- *Time depended clustering*: time information can be very crucial for analysing moving object locations which are changing over time. *Nanni and Pedreschi (2006)* [3] considers time gaps between trajectory positions and proposes *T-OPTICS* as an adaption of *OPTICS*.
- *Partition and group based clustering*: trajectory data are often very long and complex and find trajectory partition or local patterns approaches can be solutions with low memory usage and time cost. *Lee et al. (2007)* [4] were the firsts to implement a *partition & group* framework for trajectory clustering based on a formal theory.
- *Uncertain trajectory clustering*: model and reduce the uncertainty of trajectories location between discrete time updates and noise in position tracking. Nock and Nielsen 2006 [5] implemented *Fuzzy C-Means (FCM)* that is an efficient algorithm for clustering data with noise.
- *Semantic trajectory clustering*: depending on the capabilities of the device, the instant speed or stillness, acceleration, elevation, direction and rotation, etc., can't be acquired directly. Several works start considering geographical informations as a background from which extracting higher semantic level informations. *Palma et al. (2008)* [2] introduced a new model for trajectory semantic: stops and moves, where stop is a part in trajectory where the object has stayed for a minimal amount of time (e.g. an airport, a touristic place).

In this report I will deal with techniques that involves the first 3 categories of current researches. I will present some partition and group heuristics, based on time gaps, and I will show how the traditional clustering algorithms can be used for trajectory clustering relying on positions spatial information.

III. Objectives

The objectives of this project is grouping trajectories in order to find common locations crossed by people that rent a scooter to visit a city in Italy. The model built has to be able to distinguish for example the positions related to a trajectory

that takes from the train station to the city centre, the ones that move in the city centre, the ones that run through the periphery and so on.

In particular the steps that involves the project are:

1) Dataset filtering and merging in order to build a dataset tidier and semantically correct;
2) Analyse features and represent the trajectories in line plots;
3) Group the positions for each rental and sort them through the timestamp;
4) Apply heuristic techniques on features;
5) Apply unsupervised learning techniques on features (as K-Means, Mean Shift...);
6) Study the results and evaluate the goodness of clustering;

IV. Methodology

The original dataset is composed by 4 tables in CSV fomat: *pos.csv*, *rental.csv*, *user.csv*, *device.csv*. This dataset is a subset of another dataset with some sensitive data dropped (like user name or email) and positions manumit in order to protect proprietary data. Although this dataset is only a subset of the proprietary data, the positions amount is really huge and it weighs about 2GB.

The *pos.csv* table contains all the positions, characterized by latitude, longitude, speed, timestamp and a device id used to join with *device.csv* table. The *device.csv* and *user.csv* tables contain the kilometres travelled respectively by a scooter and by a user. At last, the *rental.csv* table contains the start position and end position in couple of longitude and latitude of the rental trajectories with related start and stop timestamps, and all the ids used to join the other tables with this one (dataset diagram in figure 19). The figure 1 shows all positions in the two Italy cities.

The first thing that I noticed is that the dataset is not built very well and I would have the positions in relation with its rental and not with its device. Therefore I preferred to lose some positions saved by different devices, I joined the position and rental tables through their ids and I filtered it in order to obtain only the positions that belongs in the range of rental start-end time (figure 20). This operation was really tricky because the total amount of data is huge and I had to use optimized functions based on database join algorithms and chunks management to be able to handle these data in shot time. At last, the resulting table has been sorted for rental id, position id and timestamp, in order to have everything ready to perform plot representation and some feature analysis.

After that, I performed some analysis on features. I studied their distribution (figure 2) and I started to think how could be possible to group or divide positions. In particular I learnt that two trajectories can be similar in shape and can be divided in time. It means that a sequence of positions, a trajectory, can be similar to another one, evaluating his shape appearance and location in the geographical map, and therefore his sequence of their geographical coordinates. On the other hand, a trajectory can be different to another one if there is a temporal space between each other. The starting assumption of this analysis is that all positions belong to a rental, it means that starting from how data are constructed, I assume that a trajectory is the sequence of positions sorted in time that belong to the same rental (figure 3).

Dataset	Samples	Features
rental	14826	10
pos	817076	18
merge	817076	18
dataset	14826	13
partition city 1	608251	18
partition city 2	202795	18

Table I: Number of features and samples of generated dataset

As result I implemented 3 different clustering heuristics performed in a systematic and statistical way on the entire sequence of positions:

- **timedelta heuristic**: considers that a trajectory of a rental can be divided in a sequence of trajectories if the time gap between a position and previous one exceeds a *timedelta* value. First of all I calculated the time gaps for each set of positions grouping in rental:

$$TIMEGAPS = \{p.time - p[-1].time \mid \forall p \in POS\} \quad (1)$$

  where `p[-1]` is the previous position and the field `.time` is the position timestamp. The *timedelta* value can be assigned or can be automatically calculated. To calculate automatically the timedelta value I plotted the timegaps distribution and I exploited the statistical empirical rule to cover a percentage of timegaps distribution. In this case I take all the left tail of the distribution and the 43% of the right one, and that ones that remains outside are the positions candidates that divide a trajectory from another one. This operation has been performed for each rental positions in order to obtain a set of sub-trajectories of rental trajectory.
- **spreaddelta heuristic**: considers that a rental trajectory can be considered similar to another one if they spread a similar amount of area. I calculate the spread area for each rental trajectory in the following way:

$$SPREADS = \{max(t) - min(t) \mid \forall t \in TRAJ\} \quad (2)$$

  where `max(t)` and `min(t)` calculate respectively the maximum and the minimum latitude and longitude of a set of positions, and `TRAJ` is the set of trajectories that can be grouped for each rental, but even for the *timedelta heuristic* previously calculated in order to consider the time gaps division of trajectory and not only the rental division. Also in this case, I exploited the empirical rule to compute the trajectories similar for spread and I assigned $std(SPREADS)/4$ (20% of distribution) to the spreaddelta to automatically calculate it.
- **edgedelta heuristic**: acts as the *spreaddelta heuristic*, but it considers the edges of a trajectory, or rather the first position and the last position of a trajectory. The main problem here is that the distribution of edge positions

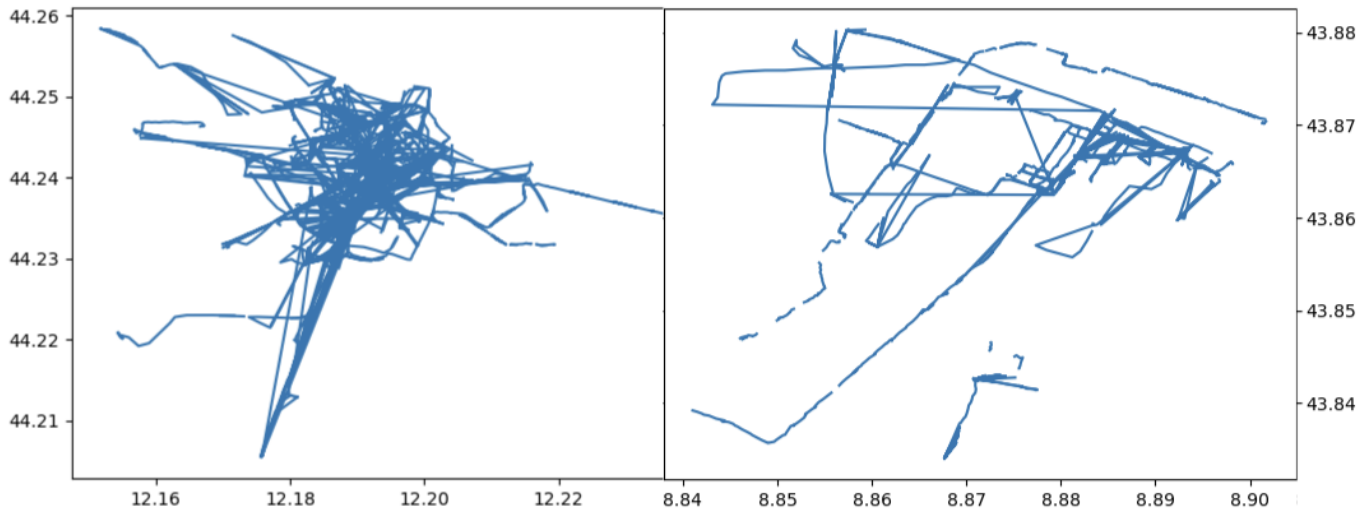

Figure 1. Trajectories without clustering in 2 Italy cities

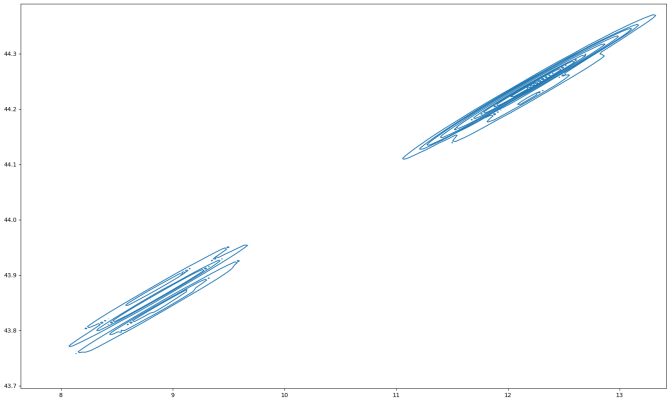

Figure 2. Distribution of positions: mainly developed in 2 Italy cities

has 2 centres or, in other words, it is bimodal. To resolve this issue I applied for simplicity only 1 iteration of *Mean Shift* starting from a random position in order to get closer to one of 2 centre. The value assigned to edgedelta is $std(meanshift(EDGES, iter = 1)) * 2$ and obtain the 95% of the subset of edges found by *Mean Shift*. The set of edges is calculated in the following way:

$$EDGES = \{concat(p[0], p[-1]) \mid \forall t \in TRAJ\} \quad (3)$$

- **coorddelta heuristic**: it is a combination of spread and edge heuristics in order to combine the main advantages of each other. To guarantee the convergence of this algorithm *edge heuristic* and *spread heuristic* have to manage the same subset of trajectory. Therefore first it is applied edge heuristic to find a subset of trajectories near a distribution centre and then both heuristics, edge and spread, are applied to the same subset.

After the implementation of these heuristic techniques, I started to prepare the features to be processed by clustering algorithms. In particular I decided to perform *Standardization*, *Normalization* and than *Principal Component Analysis (PCA)*. The component extracted by *PCA* can be decided in 3 different ways:

1) By a number of component decided a priori;
2) By the cumulative variance calculated by *PCA* over the number of features, considering to cover almost 80% of variance;
3) Constructing a list of features subset and performing *PCA* to produce 1 component for each features subset. The result will be a concatenation of columns produced by *PCA* for different subset of features.

As feature extraction, I considered that the work done for the heuristic algorithms could return useful for clustering algorithms. In particular $TIMEGAPS$, $SPREADS$ and $EDGES$ sets can be used as new features for my data. Therefore, I integrate my data features with the heuristic values and I run the pipeline of *Standardization*, *Normalization*, *PCA* and than clustering algorithms. In particular, I tried to perform this pipeline with space and time features together, with only space features, and then with space and heuristic features together. Moreover I tried to perform clustering algorithm on the whole trajectories and than also on positions partitioned in the two cities.

The clustering algorithms that I used are the following:

- **K-Means**: choose centroids that minimise the *inertia*, or *within cluster sum of squares (WCSS)* criterion. This is the simplest technique based on a distance metric, but it is also the faster and cheaper also in memory terms. Time complexity $O(n * k * l)$, where n is the number of data points, k is the number of clusters, and l is the number of iterations taken by the algorithm to converge.
- **Mean Shift**: discover blobs in a smooth density of samples with the purpose to find the mean of points within a given region. This algorithm, called density based, automatically sets the number of clusters, but it needs a bandwidth parameter, which dictates the region

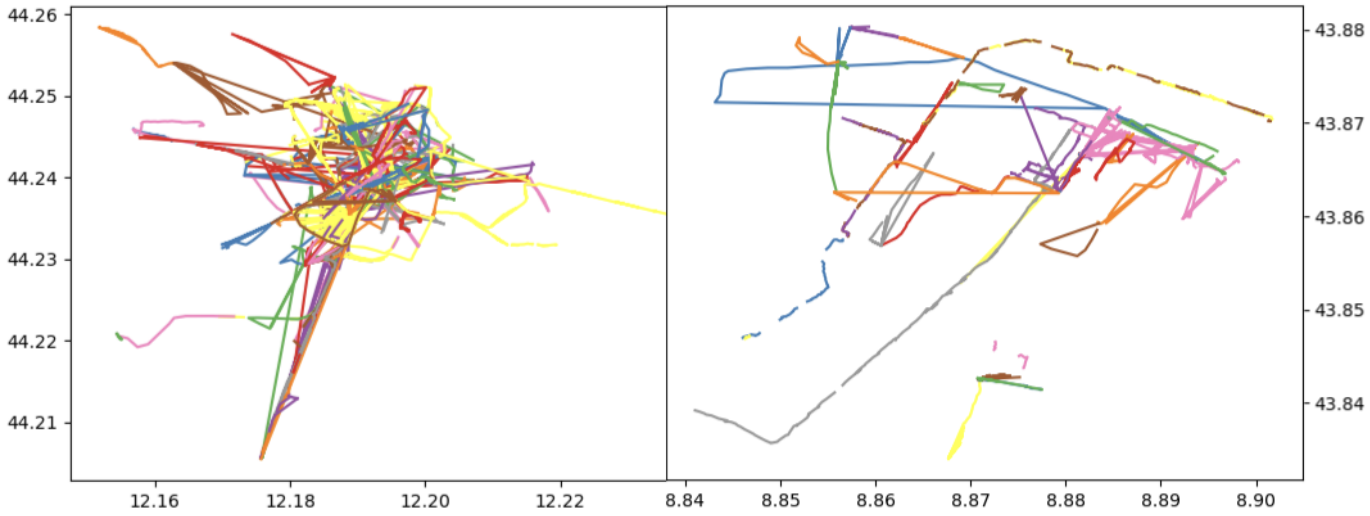

Figure 3. Trajectories belonging rentals

size to search through. Time complexity $O(n^2)$, where n is the number of data points.

- **Gaussian Mixture**: assumes all the data points are generated from a linear combination of a finite number of Gaussian distributions with unknown parameters and implements the *expectation-maximization (EM)* algorithm for fitting mixture of Gaussian models. Time complexity $O(l * n^3)$, where l is the number of iterations and n is the number of parameters.
- **Full Hierarchy Agglomerative**: hierarchical clustering using a bottom up approach and minimizes the maximum distance between observations in pairs of clusters. Time complexity $O(n^3)$ where n is the number of data points. It also has a huge memory cost.
- **Ward Hierarchy Agglomerative**: hierarchical clustering using a bottom up approach and minimizes the sum of squared differences between all clusters. It is the same of the previous one, but it uses a variance minimizing approach similar to *K-Means* objective function.

V. Results

The best way to show the clustering results is plotting the trajectory with different colours in relation to the cluster which it belongs. In addiction, for clustering algorithms, I used a method of interpretation and validation of consistency within clusters of data, called *Silhouette*.

Silhouette clustering validation technique measures how similar an object is to its own cluster compared to other clusters. The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

The results of heuristic analysis are the following:

- *timedelta heuristic*: is the heuristic that perform the best result. In figure 4 you can see some rentals with their positions divided in subgroups of trajectories. That sub trajectories are built considering only the time gaps between the positions, no latitude or longitude has been used to produce this result. This result shows how the time sequences could be really useful to produce a good trajectory partition. Moreover, in figure 5, there is the *timedelta heuristic* representation in the entire city.

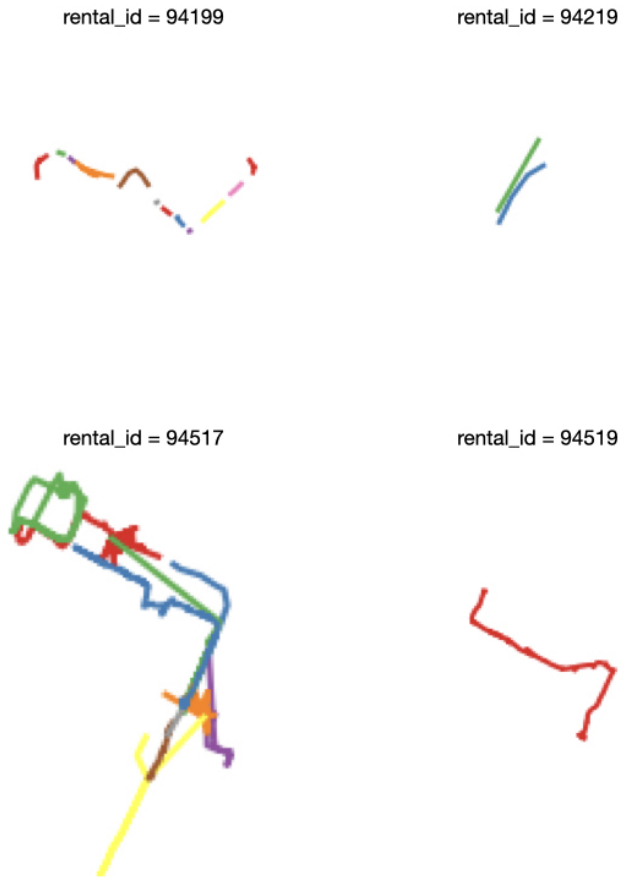

Figure 4. Timedelta heuristic line plot in details

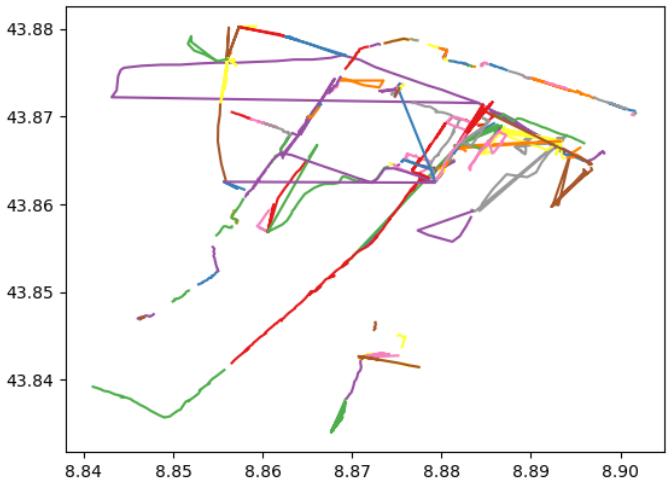

Figure 5. Timedelta heuristic line plot in the entire city

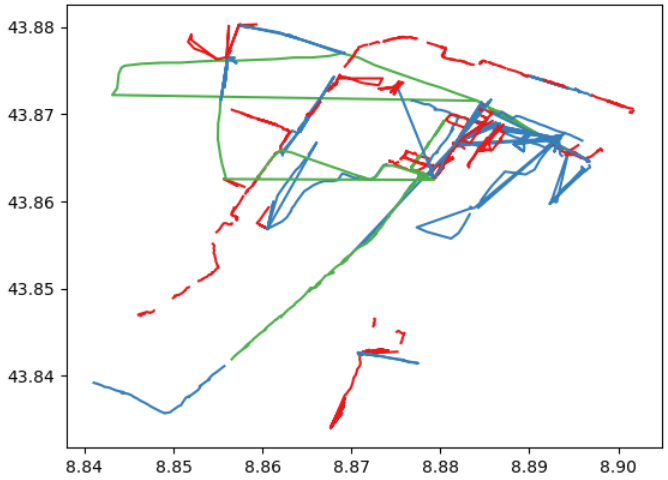

Figure 6. Spreaddelta heuristic line plot

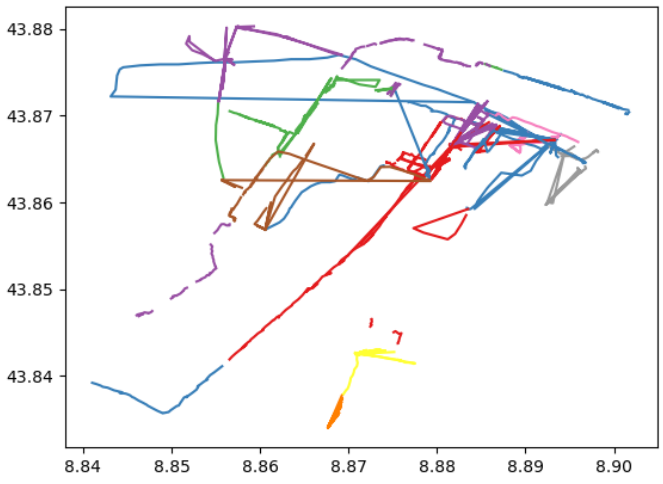

Figure 7. Edgedelta heuristic line plot

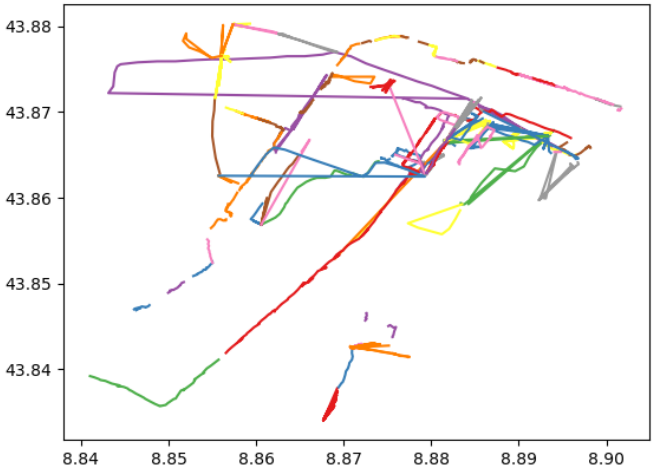

Figure 8. Coorddelta heuristic line plot

- *spreddelta heuristic*: in figure 6 you can see how the trajectories are clusterized in 3 main groups: wide area trajectories (green), medium area trajectories (blue) and small area trajectories (red). The trajectories showed in figure are grouped starting from the *timedelta* division performed before.
- *edgedelta heuristic*: in figure 7 you can see how the trajectories are clusterized in relation to the start and end positions of each trajectory generated by *timedelta heuristic*. This heuristic performs some good results, because it is able to recognize the neighbours trajectories, but there are also some result not expected due to the problem of bimodal distribution.
- *coorddelta heuristic*: in figure 8 you can see the trajectories clusterized with both *edgedelta* and *spreaddelta* techniques. It shows the same issues of *edgedelta heuristic*, but, with the *spreaddelta heuristic* addition, the trajectories are even more selective, to the point of returning nearly to the initial trajectories. This heuristic can be useful only if you want a model that slightly clusterize the trajectories.

For clustering algorithms, in addition to graph analysis, I also use the *Silhouette* validation. To evaluate the number of clusters for the algorithms that need it as parameter, I executed *K-Means* for a number of cluster in a range from 1 to 30 and I calculated for each cluster number the *WCSS error*. Then I plot the *WCSS* graph and I used *Elbow method* to choose the best value to use. As result of *WCSS* graph, I chose to set 5 clusters.

All clustering techniques were performed on each city individually, in order to facilitate the algorithms. The results of clustering techniques are the following:

- *Gaussian Mixture*: performs bad result 10 and the silhouette validation confirms it.
- *Mean Shift*: obtains the best result in terms of silhouette validation. The number of cluster generated is only 3 because *Mean Shift*, at the end of algorithm, performs a pruning operation of similar centres, taking only the

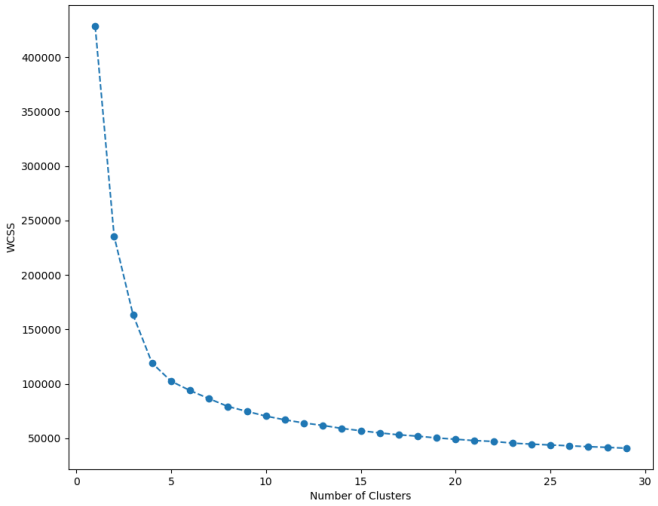

Figure 9. Within Cluster Sum of Squares (WCSS) graph for Elbow method in range 1 to 30 with K-Means

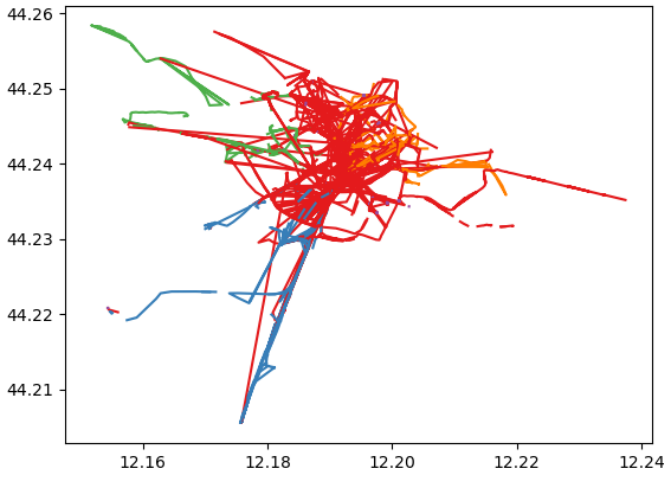

Figure 10. Gaussian Mixture line plot with silhouette -0.02

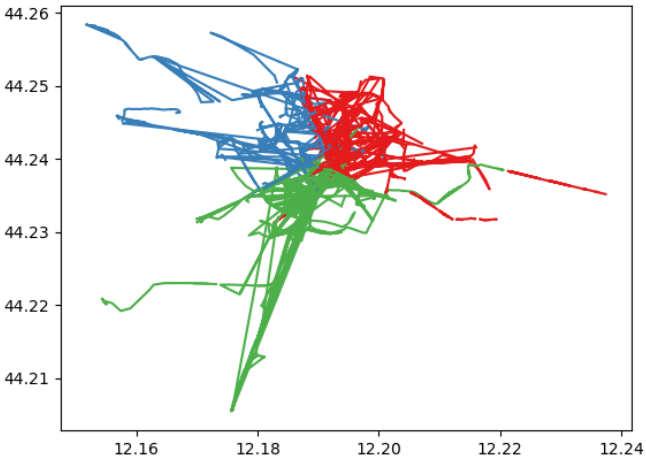

Figure 11. Mean Shift line plot with silhouette 0.40

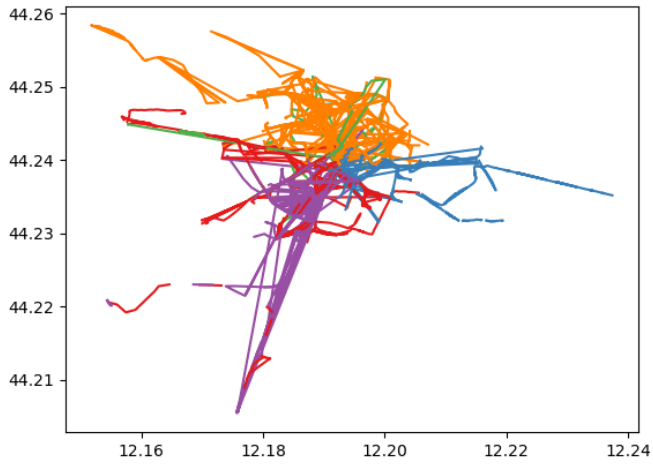

Figure 12. Full Hierarchical Agglomerative line plot with silhouette 0.16

more significant. In general, for trajectory clustering, I would like to obtain more clusters, but this is also an alert that the number of clusters can't be really higher.

- *Full Hierarchical Agglomerative*: worst both in silhouette validation term and representation term 12, but the main issue of this technique is a huge memory cost. The dedrogram of this hierarchical algorithm is shown in the this figure 13.
- *Ward Hierarchical Agglomerative*: performs well because it tries to group the centre of the city. It is really similar to *K-Means*, it is not as accurate as *K-Means*, but the main problem, like the *Full* version of this technique, is the huge memory cost. The dedrogram of this hierarchical algorithm is shown in the this figure 15.
- *K-Means*: it is the simplest algorithm but maybe the one that produces the best results. Even if it uses a distance metric to calculate the clusters, it performs very well not only in terms of plot representation, but also in silhouette value. In addition, it is really fast and cheap in memory. The main problem of this clustering result is the mixture of clusters in the city centre, that creates uncertainty.

VI. Conclusion

The best result are the one performed by *K-Means*: the feature are extracted with *Standardization*, *Normalization* and *PCA* with my custom feature subset implementation, the number of cluster given as input is estimated through the *Elbow method* from *WCSS* graph, and the results are given in terms of plot representation and *Silhouette score*.

In particular the feature extraction phase has been performed with different features configuration. The best result is produced with my custom implementation of PCA on the following subset of features:

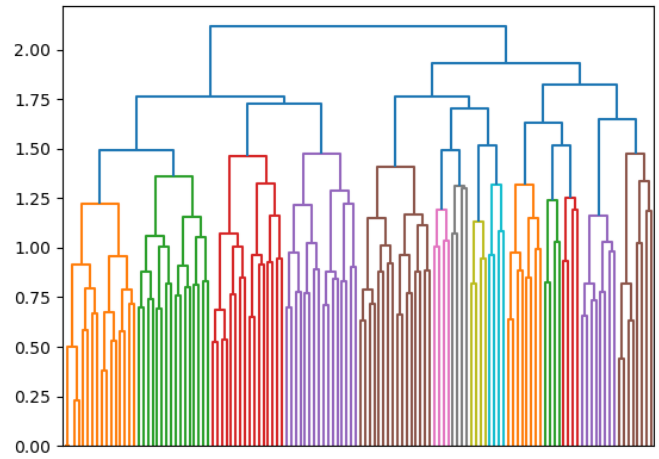

Figure 13. Full Hierarchical Agglomerative dendrogram up to level 5 of merge

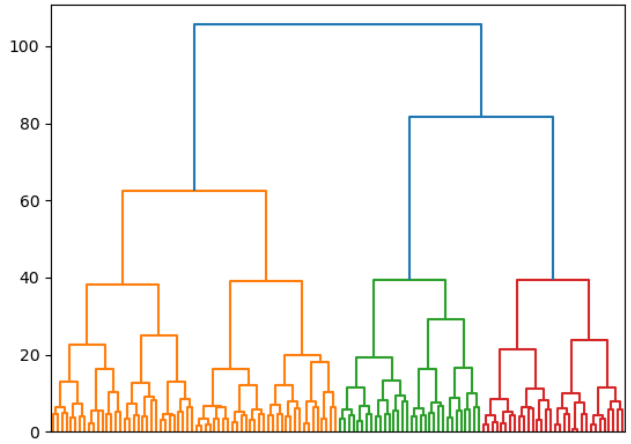

Figure 15. Ward Hierarchical Agglomerative dendrogram up to level 5 of merge

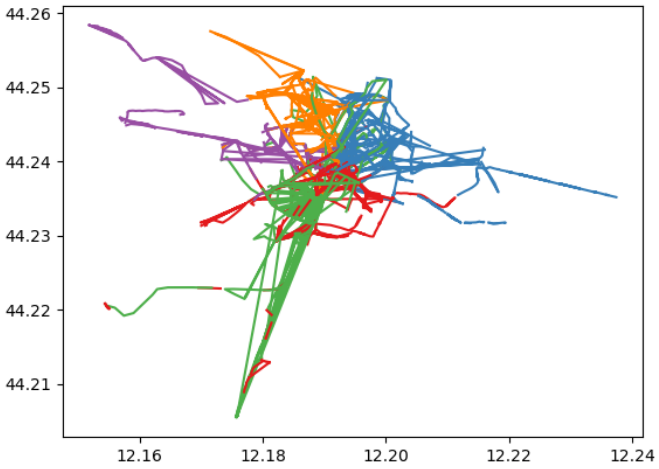

Figure 14. Ward Hierarchical Agglomerative line plot with silhouette 0.28

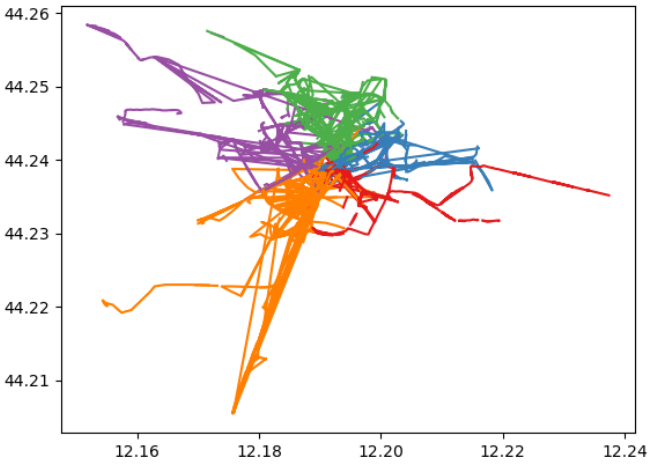

Figure 16. K-Means line plot with silhouette 0.352

$$\{\{latitude\}, \{longitude\},\\ \{spreadlatitude, spreadlongitude\},\\ \{edgelatitudestart, edgelatitudestop,\\ edgelongitudestart, edgelongitudestop\}\}$$

but it doesn't perform result widely better than the traditional PCA approach based on the 80% of cumulative variance.

Moreover, feature extraction without time features produces better clustering result. Time is a feature dimension that brings cluster algorithms to a worse prediction, because the same trajectory could be travelled in different time and instead I want to find the common trajectory independently by time. On the other hand, time is very useful to find subgroups of the same trajectory, as performed by *timedelta heuristic*.

Furthermore, I tried to compare *K-Means* clustering with and without PCA. The figure 17 shows *K-Means* clustering algorithm performed without PCA on latitude and longitude features only. This comparison shows how PCA improve the variance on clustering prediction, and the presence of heuristic features maintains the rental division, on which the heuristic algorithms has been executed.

The tests performed shows also how difficult is the clustering operation on trajectories very distant from each other. In fact, the positions of this dataset involves mainly two Italy cities and a clustering algorithm on both will try to create 2 clusters. Since I would like to obtain more cluster and to group trajectory inside the cities, I tried to increment the number of clusters. A *K-Means* clustering execution on the whole positions with 5 clusters performs really bad results (figure 18). Therefore, clustering has always to be performed on a specific region of interest in order to optimize the results. This is the reason why clustering techniques has been performed on each city individually.

Lastly, I noticed that *silhouette score* is never higher than 0.5, while its maximum value could arrive to 1.0. This score could indicate that the result is not very good, but actually *silhouette score* is not an validation methodology so reliable, because it depends a lot on the data you are dealing with.

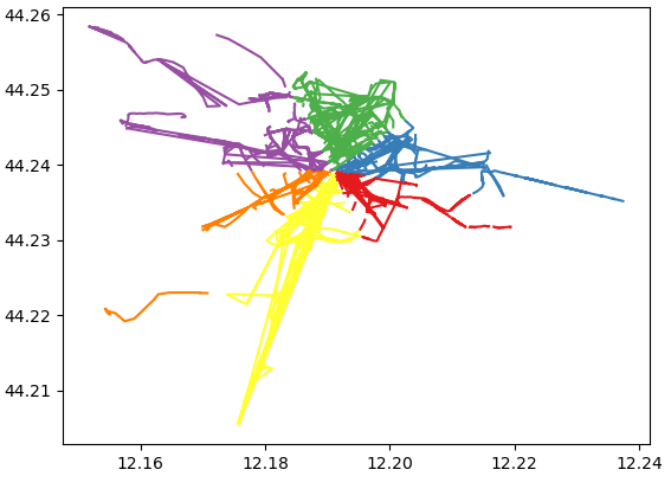

Figure 17. K-Means without PCA with only latitude and longitude features

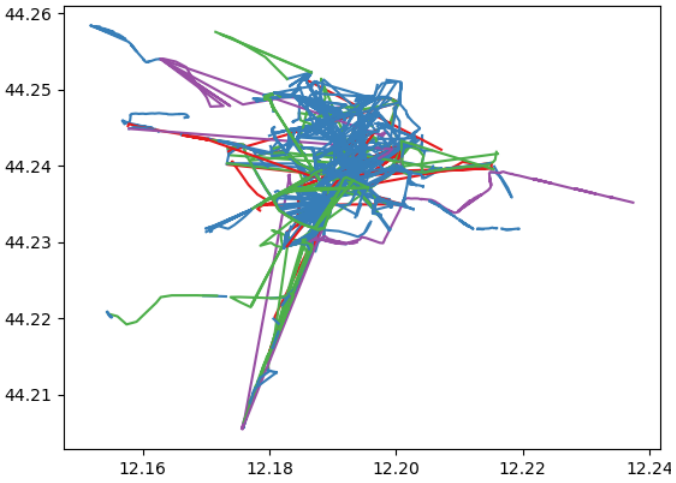

Figure 18. K-Means with 5 clusters, PCA, Standardization and Normalization performed on all positions showed on one city

In particular, *silhouette validation* uses a distance metric to produce this value, therefore it will be very reliable when used for circular shape clustering (in 2D) but for other type of distribution, you shouldn't give it too much importance, but rather consider a visual representation of results. In our case, trajectories will hardly have a circular shape, consequently an average positive silhouette score of 0.3 is pretty good for this kind of clustering.

For sure, *edgedelta* and *coorddelta* heuristic can be improved, and more advanced clustering techniques can be implemented in order to obtain better results.

References

[1] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, 01 2017.

[2] A. Palma, V. Bogorny, B. Kuijpers, and L. Alvares, "A clustering-based approach for discovering interesting places in trajectories," 03 2008, pp. 863–868.

[3] N. M and P. D, "Time-focused clustering of trajectories of moving objects," *Journal of Intelligent Information Systems*, vol. 27, no. 3, pp. 267–289, Nov 2006. [Online]. Available: https://doi.org/10.1007/s10844-006-9953-7

[4] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 593–604. [Online]. Available: https://doi.org/10.1145/1247480.1247546

[5] R. Nock and F. Nielsen, "On weighting clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1223–1235, 2006.

Figure 19. Original dataset ER model

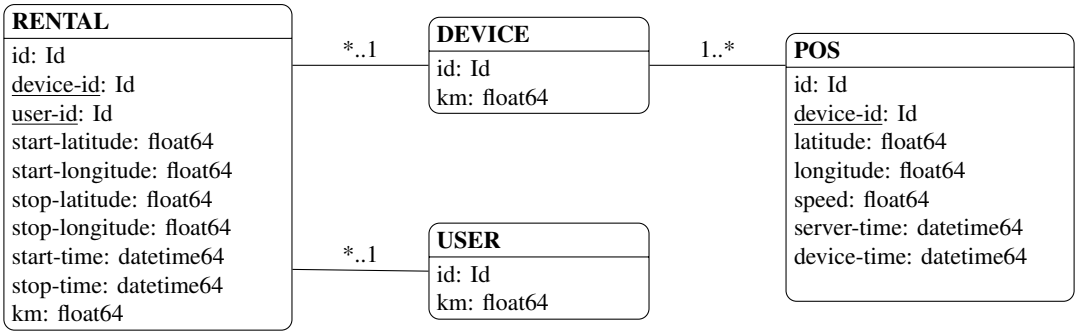

Figure 20. Generated dataset ER model