SAFE PATH DETECTION VIA CLUSTERING

Statistical Learning - Final Project

Group 16

Authors:
Bassani Aurora, 1852791
Luciani Erica, 1868647
Mignella Laura, 1920520

1 Abstract

The goal of this project is to determine the safest walking route between two geographical points. To do this, we used several clustering methods and later compared the various results obtained to see which one best suited our purpose. Eventually we selected weighted k-means++ as the most appropriate.

2 Introduction

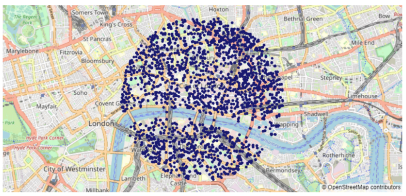
Nowadays there exists a lot of navigators that guide you through the cities, using any kind of transport means, including the walk. Most of them compute the paths basing on the time required for the arrival, the length, the cost, the presence of traffic lights and so on. Some of them also take into consideration the safety of the path, such as the Italian app W-her, and that's where our idea came from.
Our work is based on two clustering techniques, k-means++ and agglomerative clustering, to whose resulting clusters we have assigned a hazard index that we will explain later. Using this hazard index and the distance of each road from the clusters, we assigned a weight to all roads and then used the weights in the calculation of the safest path between two geographical data points.

3 Dataset

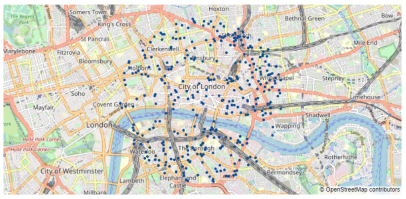
The data come from open datasets of the police of London containing data from July 2020 to June 2023 and due to the huge size of the city we decided to use in particular only those of City of London, which -of course- will be the geographical area we are going to focus on in this project. The dataset originally contained 12 columns, which carried the following information: Crime ID, Month, Reported by, Falls within, Longitude, Latitude, Location, LSOA code, LSOA name, Crime type, Last outcome category, Context. However, we decided to reduce the number of columns, retaining only those that we deemed useful for our purpose, and later, after realizing that some rows were duplicated or contained NA elements, we eliminated them, reducing also the number of rows as well. The resulting dataset thus has only 4 columns, namely Month, Longitude, Latitude, Crime type.
Since our purpose is to compute safe walking paths between two places, in order to take into account crime types we assigned a danger index from 1 to 10 to each of them, giving 1 to the least risky crimes for those who are walking and 10 to the most dangerous ones:

Crime type	**Value**
Violence and sexual offences	10
Possession of weapons	9
Robbery	8
Criminal damage and arson	7
Public order	6
Drugs	5
Burglary	4
Theft from the person	3
Vehicle crime	2
Bicycle theft	1
Shoplifting	1

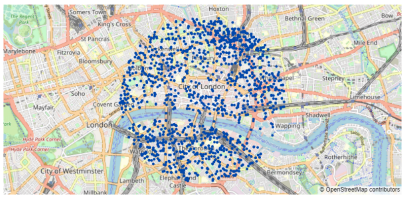
In the following plots (**Fig.1**) we can see how the different crimes are geographically distributed. What we can observe is that crimes of a certain type are roughly present all over the map, so that we cannot recognize evident clusters. Hence we cannot foresee how the clusters will be shaped.



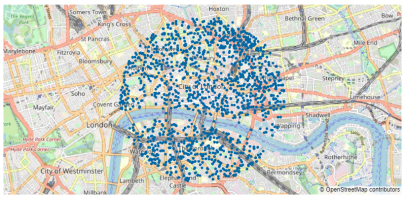
(a) Violence and sexual offences



(b) Possession of weapons

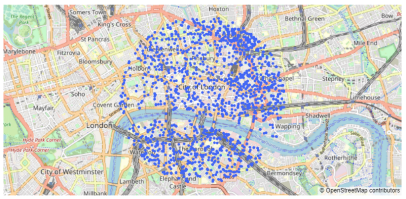


(c) Robbery



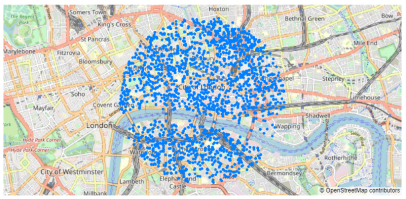
(d) Criminal damage and arson

(e) Public order



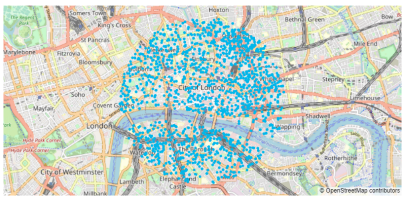
(f) Drugs

(g) Burglary

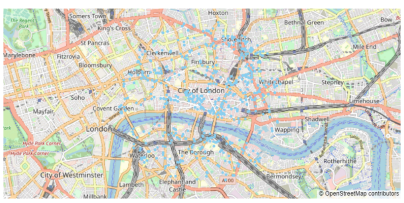


(h) Theft from the person

(i) Vehicle crime



(j) Bicycle theft



(k) Shoplifting

Figure 1

We then realized that, because the data are privatized -according to the London Police website- so that each crime is traced back to the nearest public place, multiple crimes turned out to have occurred at a single point and that indeed only about 4% of the rows in the dataset contained a unique latitude-longitude pair. To make sure that we did not have multiple overlapping points and that the final dataset had a row for each possible geographic location, we decided to assign to each latitude-longitude pair a score. While computing the scores we decided to take into consideration both the crime types scores and the year the crime was committed. When we are walking in a certain area, we usually try to get as far as possible from locations where we know a crime has recently happened, hence we wanted the crimes recently happened to have more weight than those happened in a distant past. Therefore for each unique geolocalization, we summed the danger indices separately for each year, we multiplied each sum for the weight of the year and then summed them up. We also tried to understand the frequency of each crime type in each year and took a look at the barplots in **Fig.2**. Of course, since the data from 2020 and 2023 are incomplete, the total number of crimes is much lower than in 2021 and 2022. Also, we can notice that the crime type that we labeled as the most severe -violence and sexual offences- is also one of the most frequent in all the years, along with theft from the person.

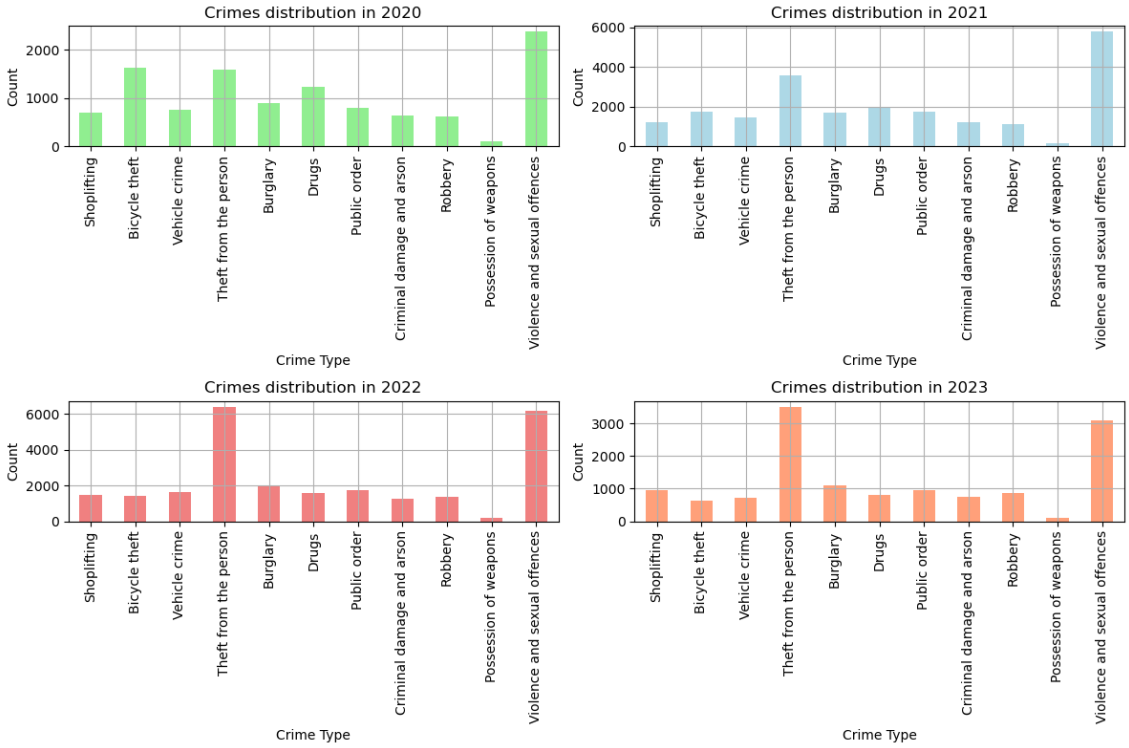


Figure 2

4 Models

As anticipated, the technique we experimented with to determine safer paths is clustering. The basic idea is to derive clusters based on the geographical location of the points and then find the path that is furthest from the clusters. We tried two different clustering methods, **k-means++** clustering and **agglomerative** clustering. In particular in the k-means++ we also exploited the scores computed in the previous section using them as weights, while in the agglomerative clustering we only build on the geographical coordinates. Starting from this, we can already expect that the centroids obtained with k-means++ are shifted more towards the points with higher danger scores. Both models require the desired number k of clusters, which we don't know a priori. Hence, for both models we varied k in the range $5 - 316$ with step 10 against 2575 points in our dataset and then picked the optimal one using *Silhouette* and *Gap Stastic* methods.
Silhouette coefficient is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1 and it's computed as average inter-cluster distance minus average intra-cluster distance all divided by the maximum of the two distances.
Gap Statistic method computes average distance between data points within a cluster for different clusters number. We would like the clustering to give us this mean distance that is lower than the one we have when data are randomly distributed without any meaningful cluster. In the end we analytically and graphically compare the safe paths obtained with the two methods.
Given a clustering, we find the safest path using the *NetworkX* function *shortest_path()* to which we pass edges weights computed as it follows:

1. we find all the clusters' centroids that are at most 200 meters apart from the midpoint of the edge;

2. we take the mean of those distances above and the mean of the weights of those clusters;

3. we assign as weight to the edge the ratio between the latter and the prior.

4.1 K-means++

4.1.1 Algorithm

K-means is a popular clustering algorithm used in unsupervised machine learning. Its primary goal is to partition a dataset into distinct groups, based on similarity among data points. Each cluster is represented by a centroid, which is the mean of the data points within that cluster. K-means works iteratively to assign data points to the nearest centroid and then updates the centroids based on the current assignments. The algorithm continues this process until the centroids no longer change significantly, at which point it converges, and the clustering is complete. K-means++ is an improvement over the standard k-means algorithm, specifically in the initialization phase. In traditional k-means, the initial placement of centroids is typically done randomly, while k-means++ chooses the first centroid randomly from the data points and then for each subsequent centroid, select the data point that is farthest from the existing centroids with a probability weighted by the squared distance to the nearest centroid. This probabilistic approach ensures that new centroids are positioned in areas of the dataset where data points are farther apart and leads to faster convergence and better final clustering results compared to the standard k-means algorithm.

4.1.2 Application and results

We implemented this method using the python function *KMeans()* from the *scikit-learn* library. The plots below represents the evolution of Silhouette score and Gap Statistic score as k varies. We can immediately notice that the two scores lead to very different conclusions. The prior suggests that the optimal number of clusters k is 315. The latter suggests k equal to 5 as the optimal; nevertheless in order to compare the results we take the second peak when k is equal to 75, also because we believe that 5 is too small to correctly describe micro areas of crimes.

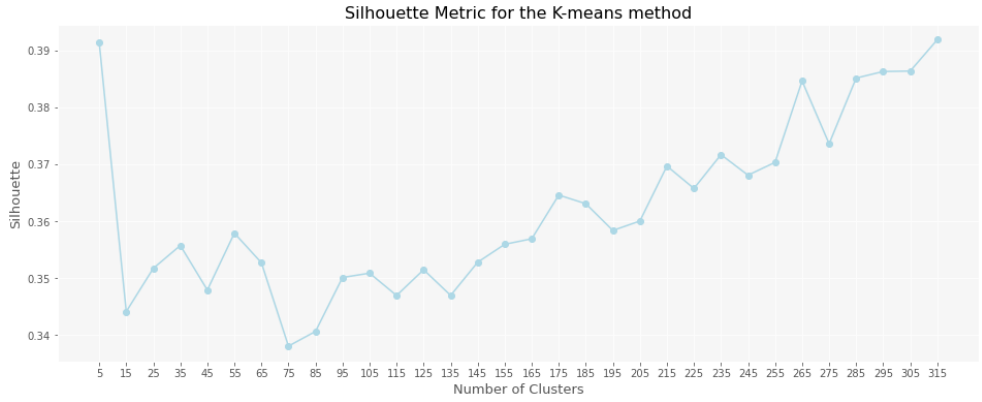


Figure 3

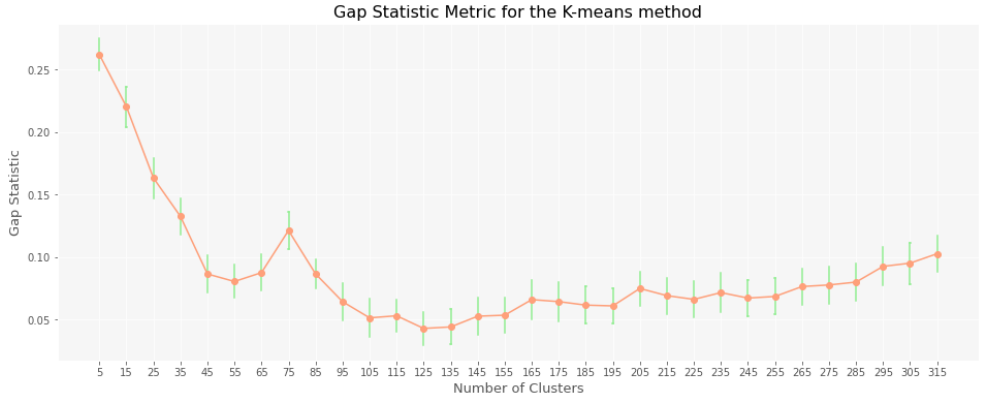
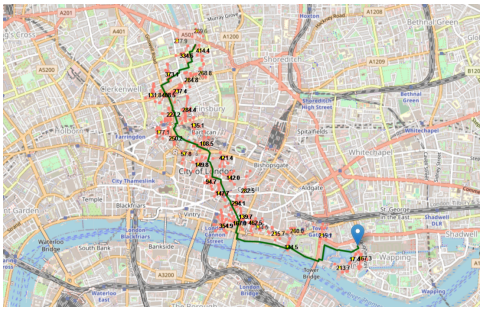
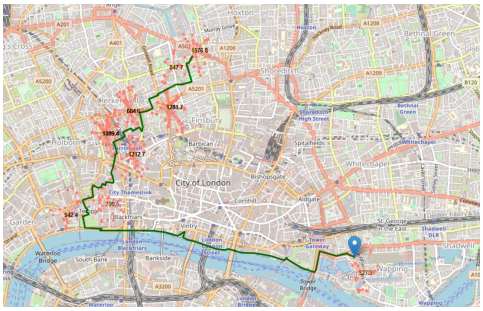


Figure 4

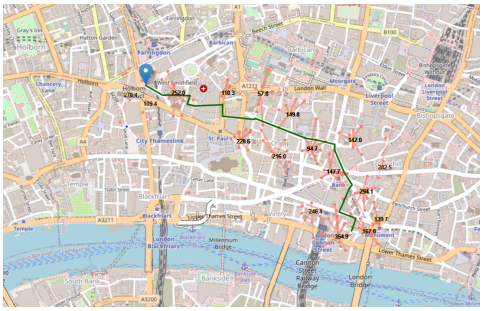
In support of the above, we now show a comparison of safest paths between two random locations, that are obtained with the two values of k.



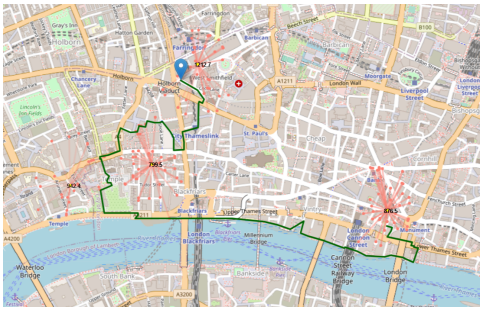
(a) $k = 315$



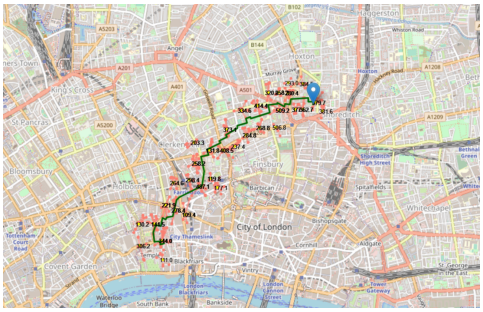
(b) $k = 75$



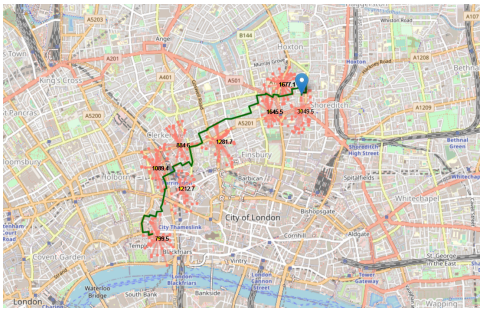
(c) $k = 315$



(d) $k = 75$



(e) $k = 315$



(f) $k = 75$

Figure 5

As we can observe from this itineraries obtained extracting random starting and arrival points, if the locations is not so near in the map, the one we get with $k = 75$ is way longer than the one with $k = 315$. That's given by the fact that if the two points are not in the same cluster, since the centroids are relatively few, it's highly probable that the path tends to pass over the edges of the clusters to go around them. We carried out a simulation as it follows, in particular:

- we extracted randomly 1000 pair of starting/arrival points;
- we computed for each one of them the safest path using the two different clustering obtained with the two different values of k;
- for each safest path we computed its length;
- we obtained the mean of the length of the safest path for each clustering.

As expected, the path with the smallest number of clusters is longer. For this reason, we thought to be more reasonable picking $k = 315$ when working with the k-means++.

k	**Path Length (m)**
75	3667.07
315	2625.55

4.2 Agglomerative

4.2.1 Algorithm

Agglomerative clustering is a hierarchical clustering algorithm used in unsupervised machine learning. Its main objective is to group data points into clusters by iteratively merging or "agglomerating" the most similar clusters until a single cluster contains all the data points. The process of agglomerative clustering typically starts with each data point as its own cluster, treating them as individual singleton clusters. Then, at each iteration, the algorithm identifies the two closest clusters based on a chosen similarity or distance measure and merges them into a single cluster. This process continues until there is only one cluster containing all the data points, or until a predefined stopping criterion is met. Agglomerative clustering creates a hierarchy or tree-like structure known as a dendrogram, which illustrates the sequence of cluster mergers. By cutting the dendrogram at a certain level, you can obtain clusters of varying sizes and shapes.

4.2.2 Application and results

We implemented this method using the python function *AgglomerativeClustering()* from the *scikit-learn* library. We can now see the results of the Silhouette score and the Gap Statistic for the selected values of k in the case of the agglomerative clustering. Let's remark that in this case we are not using the weights of each place and the clustering procedure is based only on the geographic features of the locations. As we can see from the two plots in **Fig.6** and **Fig.7** both the methods suggest an optimal value of k equal to 315.

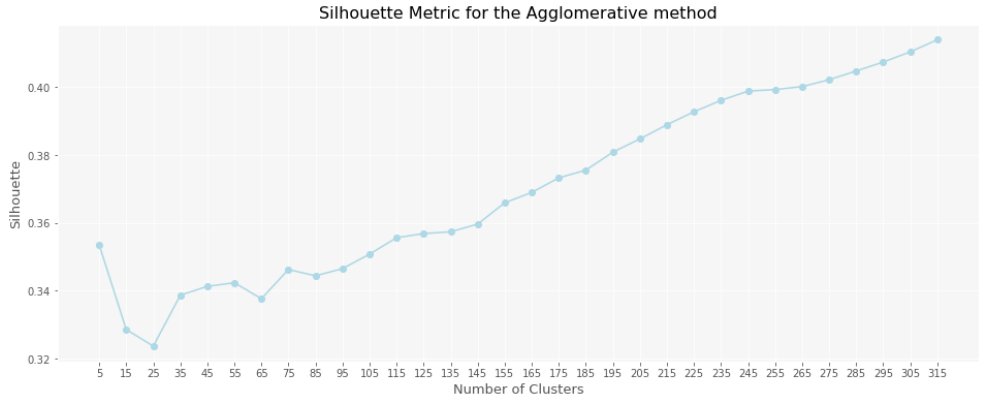


Figure 6

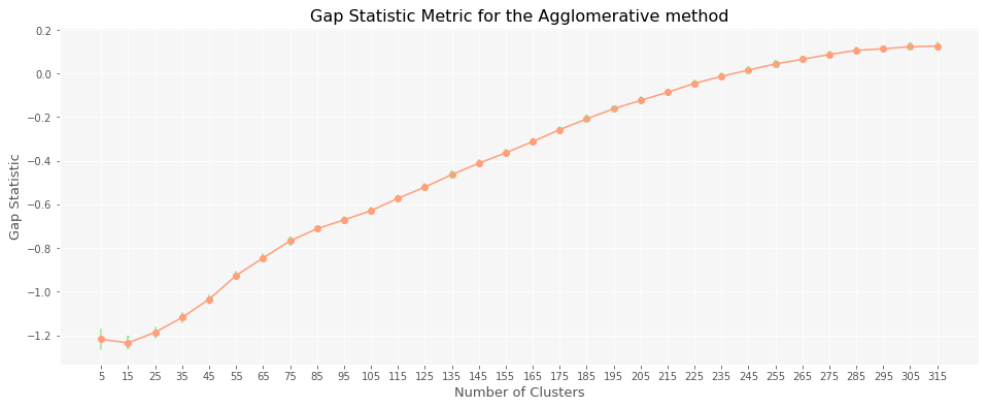
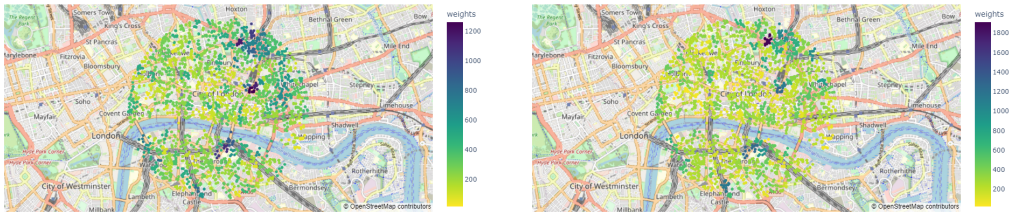


Figure 7

Let's now compare the clusters obtained with the agglomerative and the k-means techniques with the same number of $k = 315$. The colors od the clusters is given by the sum of the weights of the points within them. From the plots below we can notice that both k-means and agglomerative methods recognize three areas as more dangerous than others. This result seems reasonable since they are near train stations that we can expect to be more dangerous for a pedestrian than other areas.

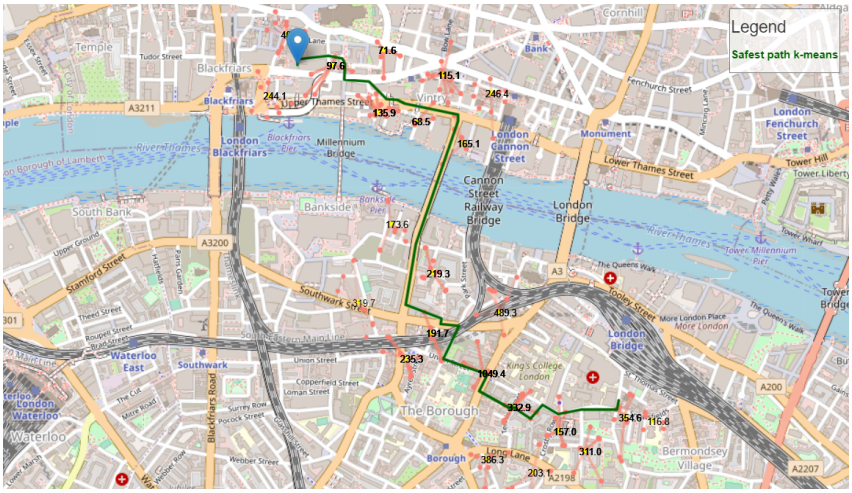


(a) k-means++ with $k = 315$ (b) agglomerative with $k = 315$

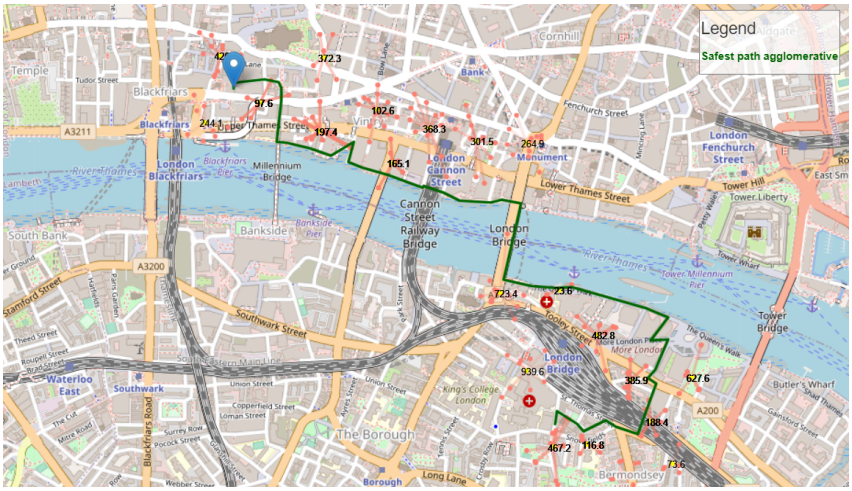
Figure 8

4.3 Models comparison

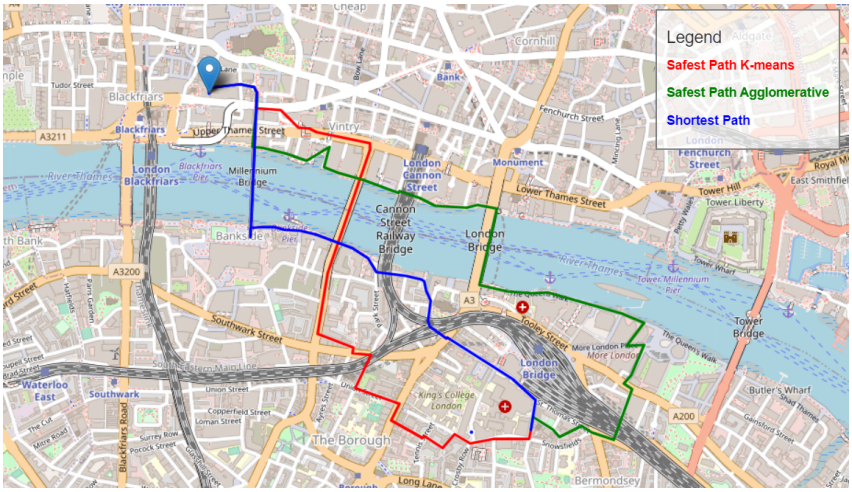
This section is dedicated to comparing the results of the k-means++ and agglomerative with the chosen k. Here, as an example, we show in separate pictures the two safe routes with the nearest clusters and then we show the two safe routes plus the shortest route in a single map.



(a)

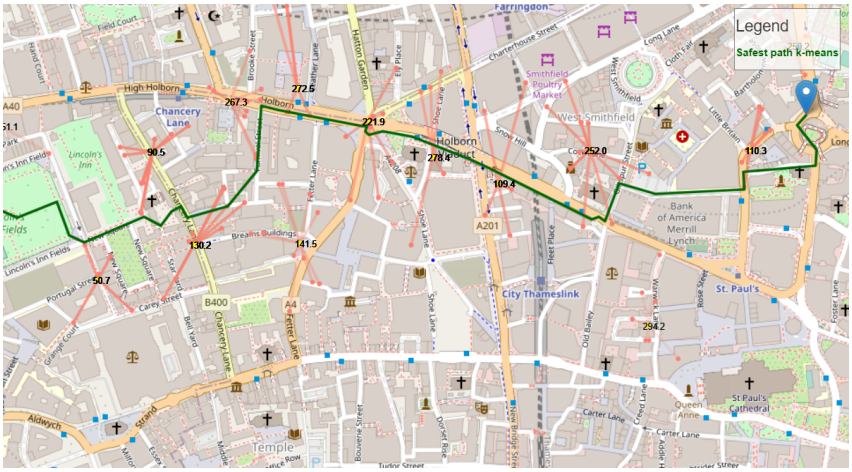


(b)

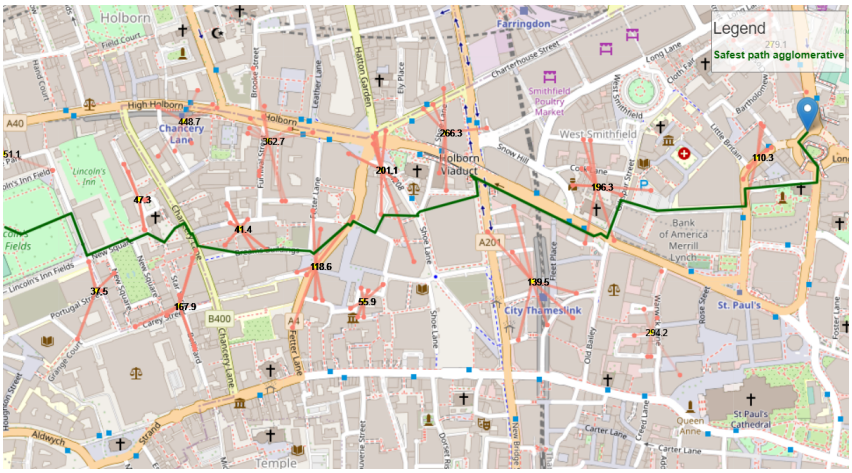


(c)

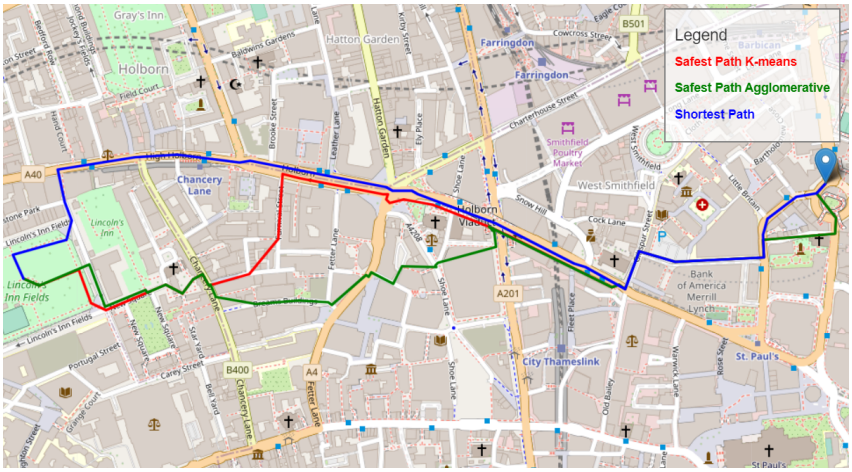
Figure 9: Paths from *55 Weston St, London SE1 3RA, Regno Unito* to *146 Queen Victoria St, London EC4V 4BQ, Regno Unito*



(a)



(b)



(c)

Figure 10: Paths from *Lincoln's Inn Fields, London, Regno Unito* to *206 Aldersgate St, London EC1A 4HD, Regno Unito*

Obviously we cannot get something general since they are only two examples, so what we did was to get 1000 couples of starting/arrival points, compute the safest and shortest paths for each pair and extract some statistics: average path length, average dangerousness of nearest clusters, average path distance from clusters, average path dangerousness.

	Shortest	K-means	Agglomerative
Path Lenght (m)	1999.24	2625.55	2649.76
Clusters Danger	//	259.98	263.59
Clusters Distance	//	130.11	130.79
Path Danger	//	99.23	102.36

The agglomerative clustering gives a *slightly* worse performance, but it didn't really come as a surprise, because both methods are based on the distance between points, but k-means++ also takes into account the weights.

5 References

1. data.police.uk/data/
2. hastie.su.domains/Papers/gap.pdf
3. towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c
4. acikbilim.yok.gov.tr/handle/20.500.12812/143435
5. cdn.techscience.cn/ueditor/files/cmc/TSP_CMC_69-2/TSP_CMC_18128/TSP_CMC_18128.pdf