Model 1 Experiment 1

This experiment uses the 1000 trajectories input in "./experiment_input/experiment1_1000traj.pkl".

In [1]:

```
# import the libraries
import numpy as np
import pandas as pd
import folium
from trajectory_clustering import TrajectoryClustering
```

In [2]:

```
# the input data
taxi_1000_df = pd.read_pickle("./experiment_input/experiment1_1000traj.pkl")
taxi_1000_df
```

Out[2]:

	TRIP_ID	TIMESTAMP	MISSING_DATA	TRAJECTORY	TRAJ_LEN
310308	1378760461620000008	1378760461	False	[[41.169564, -8.595108], [41.169735, -8.593992...	174
1302708	1397135305620000504	1397135305	False	[[41.150385, -8.607006], [41.14953, -8.607267]...	119
1066640	1392714116620000403	1392714116	False	[[41.166387, -8.577999], [41.166414, -8.577981...	113
1616481	1402560252620000041	1402560252	False	[[41.183451, -8.695215], [41.18346, -8.695206]...	108
1578304	1401954490620000098	1401954490	False	[[41.20614, -8.572644], [41.207184, -8.573454]...	189

In [3]:

```
tc = TrajectoryClustering()
```

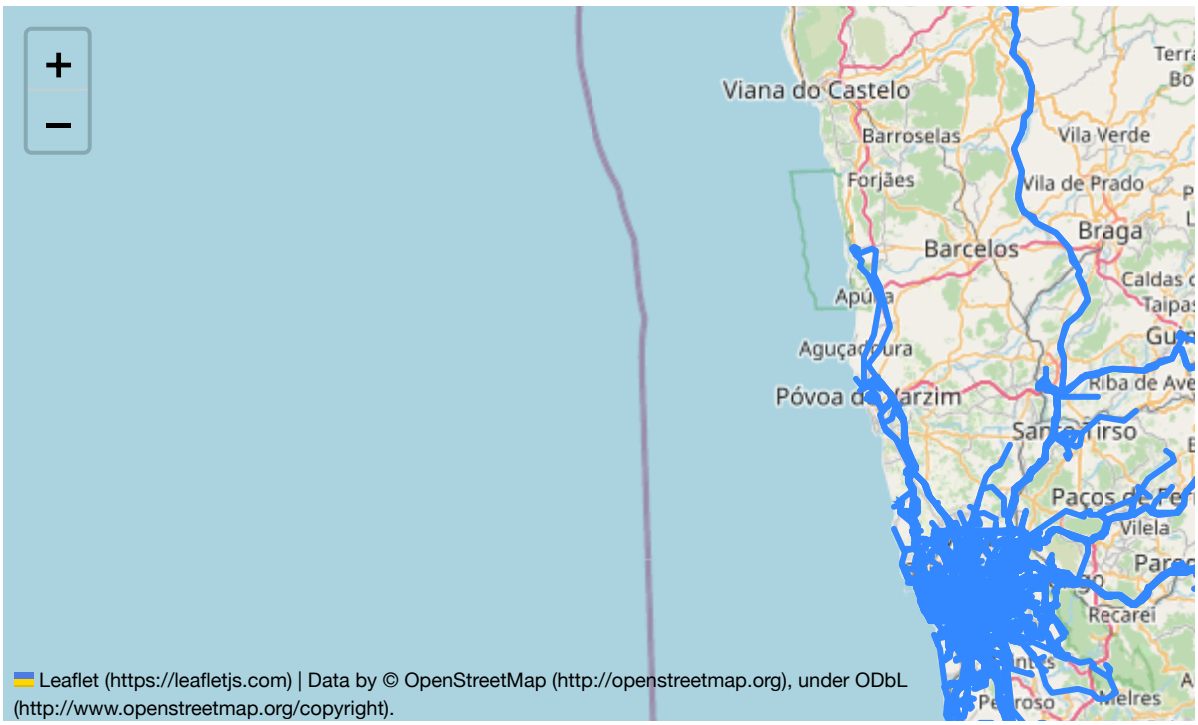
In [4]:

```
traj = tc.get_trajectories(taxi_1000_df)
```

In [5]:

```
tc.plot_all_trajectories_only(traj)
```

Out[5]:



Trajectory Clutering

1. Scale Trajectory to UTM coordinates

In [6]:

```
trajectories_xy = tc.latitude_longtitude_coord_conversion(traj)
```

2. Reduce each trajectory to predefined length (=15) using RDP algorithm

In [7]:

```
traj_to_keep, traj_xy_reduced = tc.rdp_reduce(traj, trajectories_xy, 15) #RDP reduce
```

Reshape the trajectory to 2-D arrays, so it could be input to HDBSCAN algorithm

In [8]:

```
traj_xy_reshaped = tc.reshape_trajectories(15, traj_xy_reduced)
```

```
(1000, 30)
```

3. Trajectory Clustering using HDBSCAN

In [9]:

```
labels = tc.trajectories_hdbscan(traj_xy_reshaped, 4)
traj_dict = tc.map_trajectory_with_cluster_labels(traj_to_keep, labels)
```

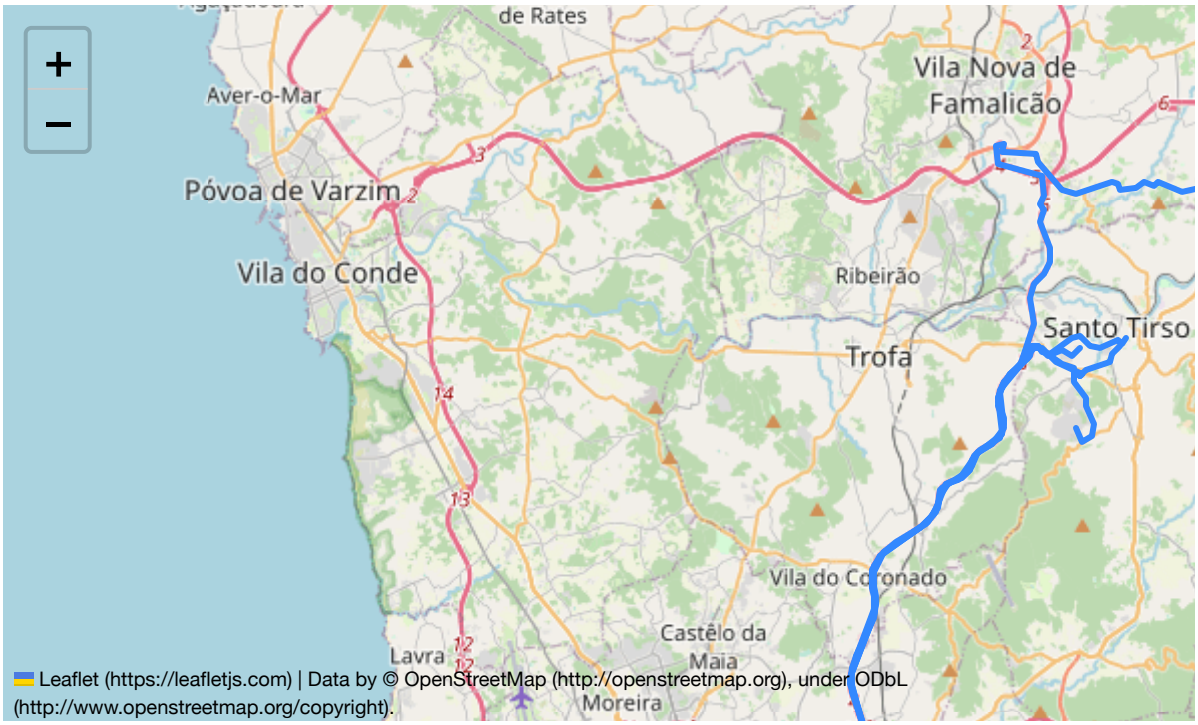
```
Estimated number of clusters: 9
Estimated number of noise points: 310
label: 1, #traj: 5
label: 5, #traj: 606
label: -1, #traj: 310
label: 7, #traj: 13
label: 0, #traj: 8
label: 6, #traj: 6
label: 8, #traj: 33
label: 4, #traj: 5
label: 3, #traj: 4
label: 2, #traj: 10
```

Plot each group of trajectories

In [10]:

```
tc.plot_group_trajectories_only(1, traj_dict)
```

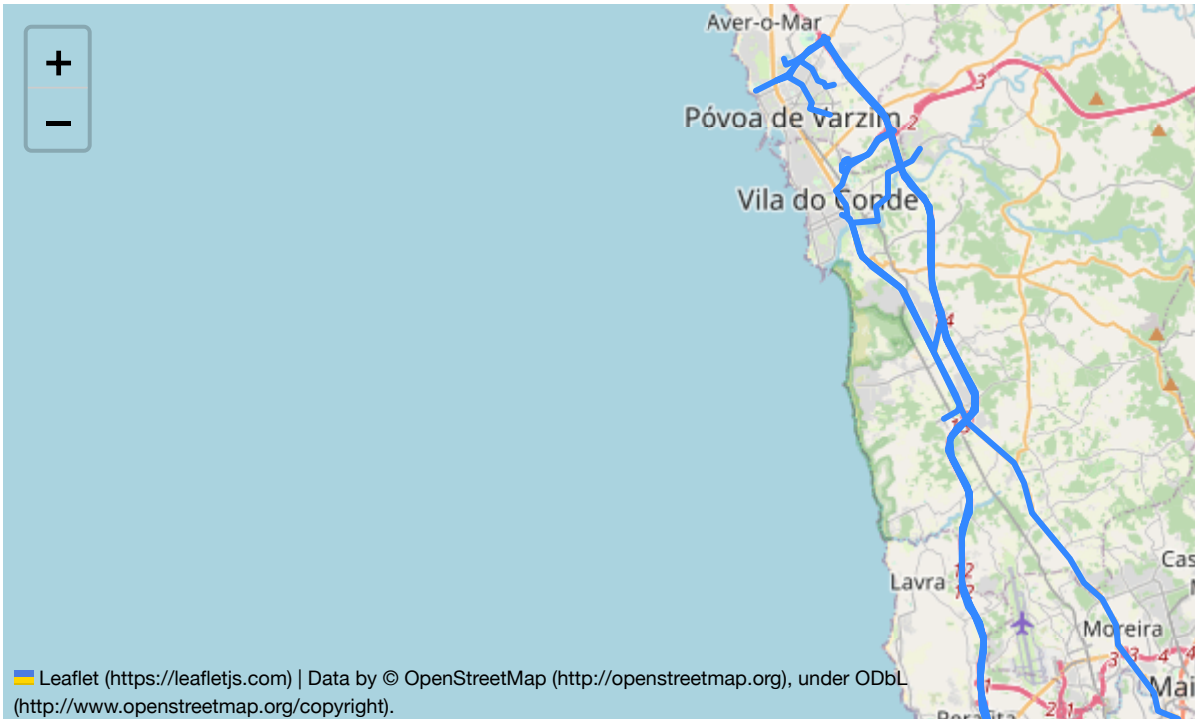
Out[10]:



In [11]:

```
tc.plot_group_trajectories_only(0, traj_dict)
```

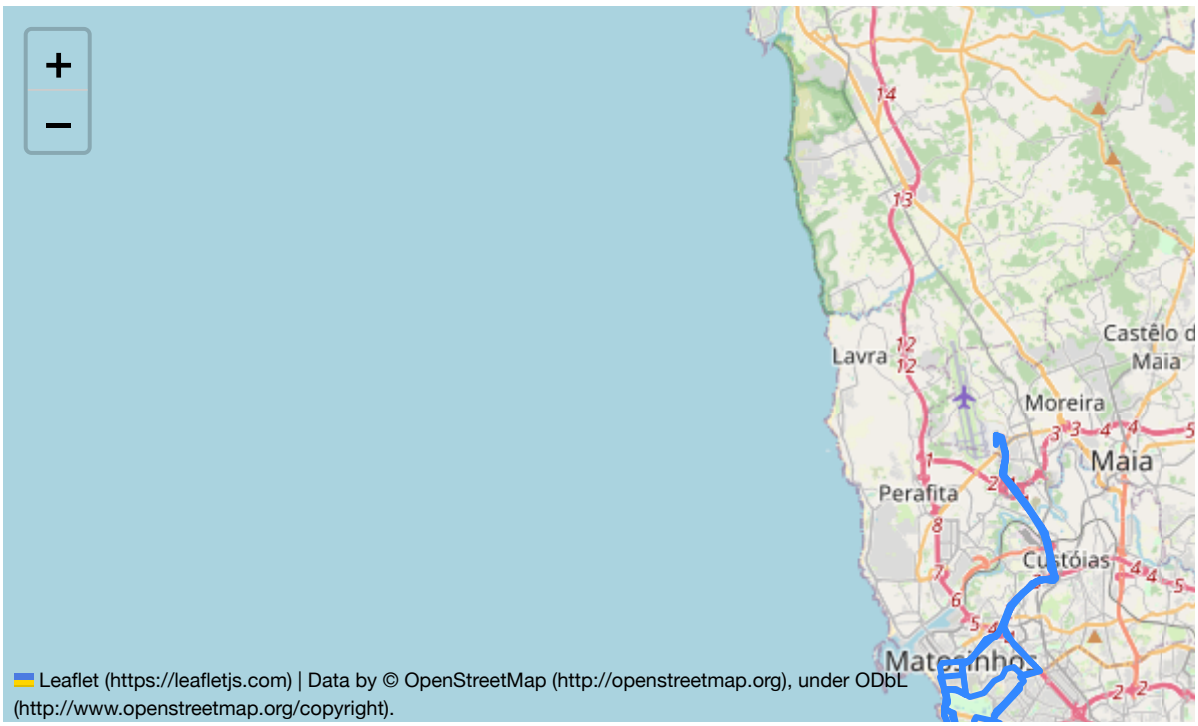
Out[11]:



In [12]:

```
tc.plot_group_trajectories_only(6, traj_dict)
```

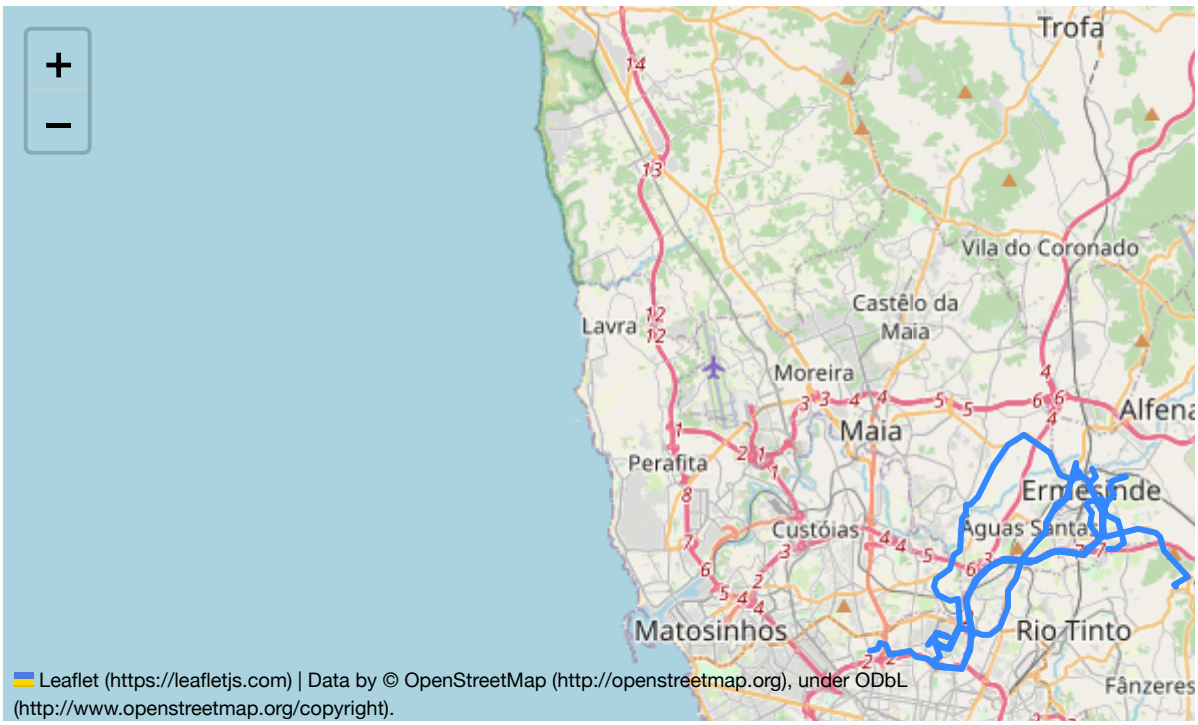
Out[12]:



In [13]:

```
tc.plot_group_trajectories_only(4, traj_dict)
```

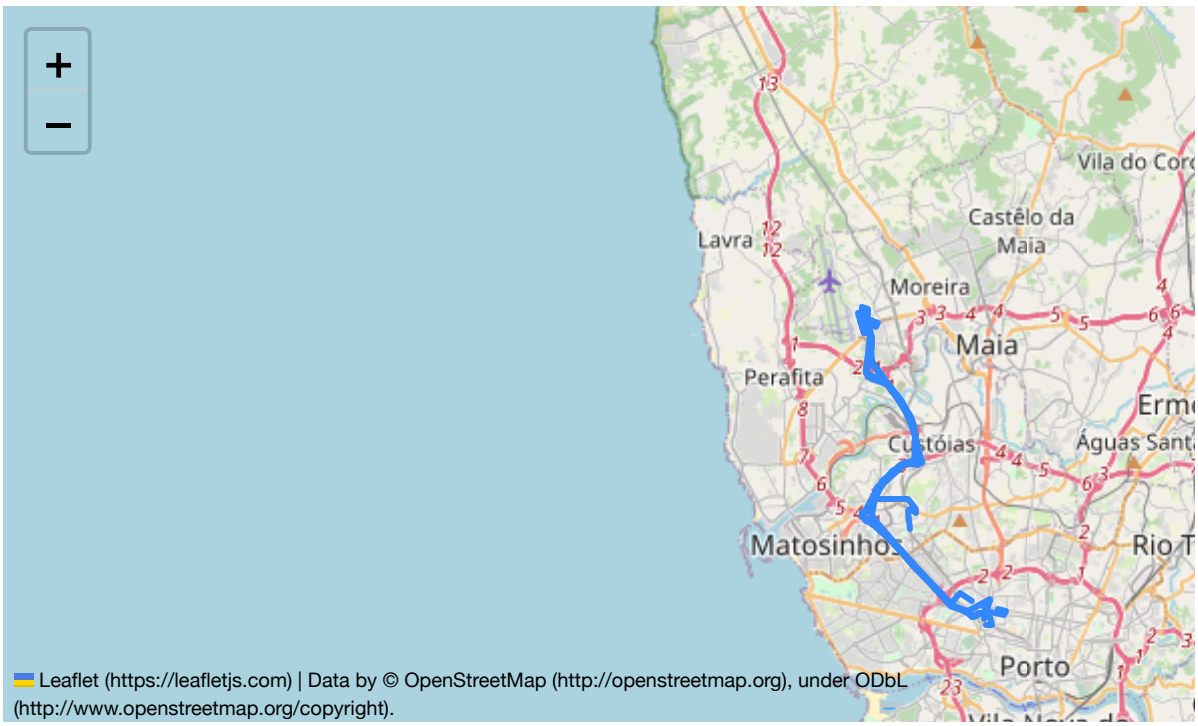
Out[13]:



In [14]:

```
tc.plot_group_trajectories_only(3, traj_dict)
```

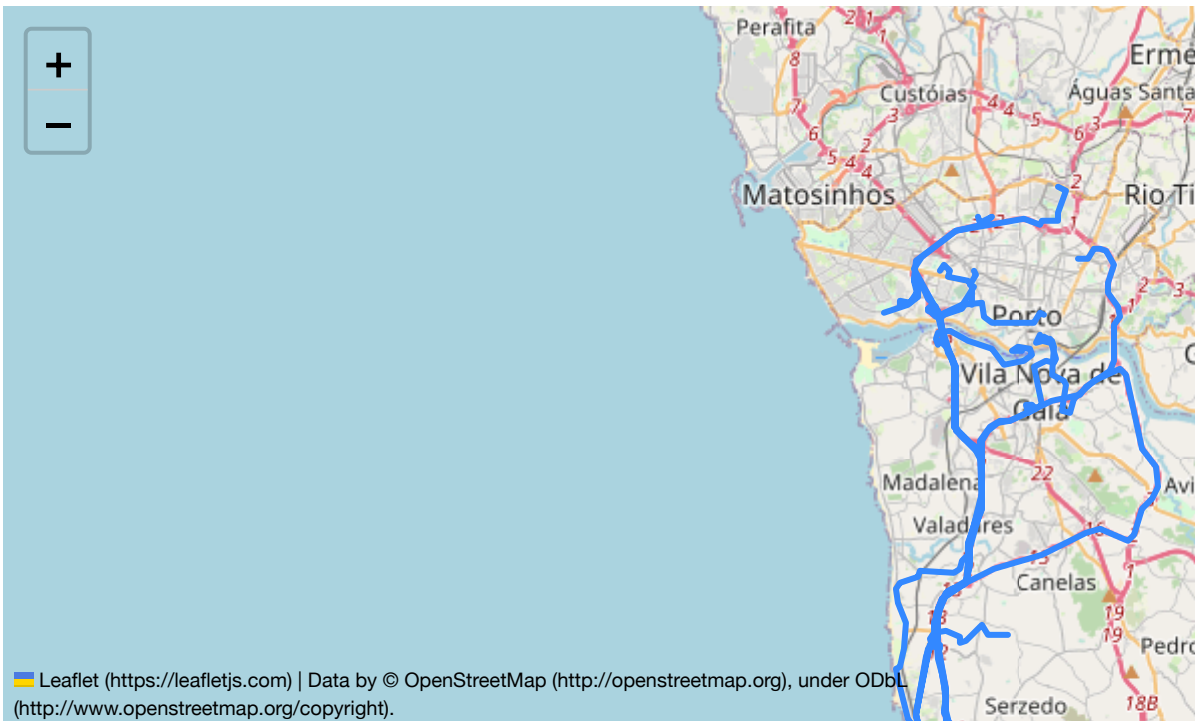
Out[14]:



In [15]:

```
tc.plot_group_trajectories_only(2, traj_dict)
```

Out[15]:



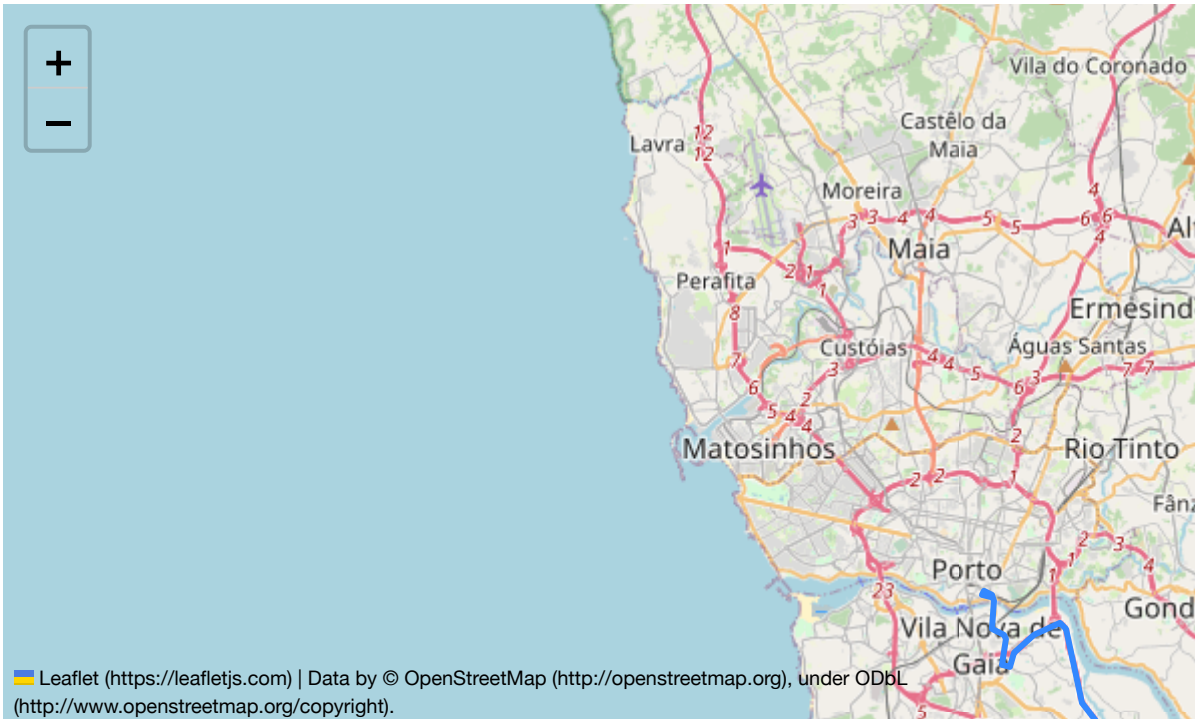
The well-clustered trajectories appears to be long and winding, spanning the entire distance. Many of them travel across cities.

Below is an example of such high-way trajectories. It travels between Porto and Espinho.

In [16]:

```
single_traj = traj_dict[2]['trajectories'][0]
m = folium.Map(location=single_traj[0], zoom_start=11)
folium.PolyLine(single_traj).add_to(m)
m
```

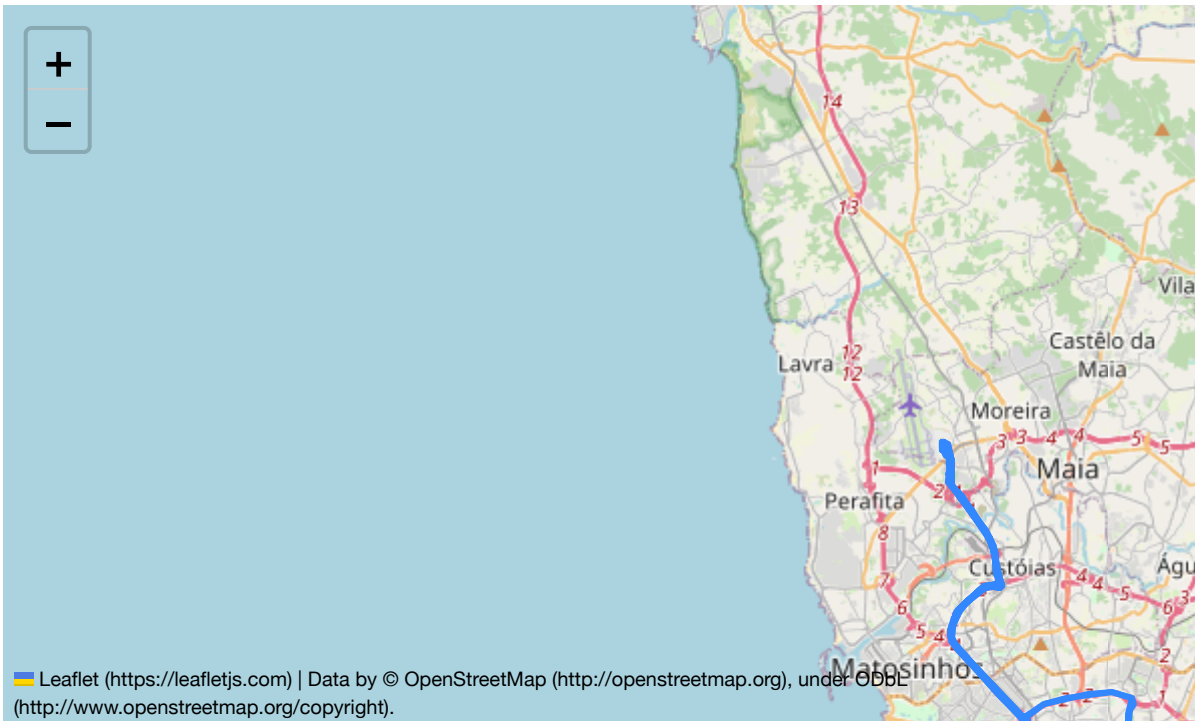
Out[16]:



In [17]:

```
tc.plot_group_trajectories_only(7, traj_dict)
```

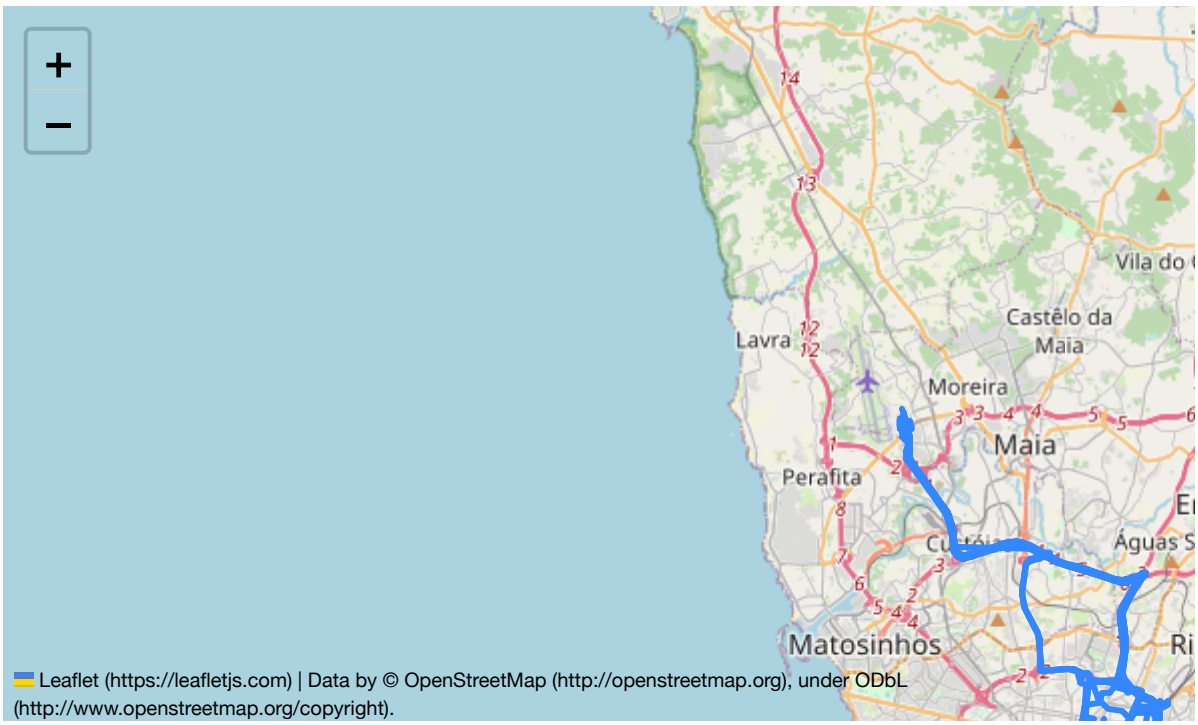
Out[17]:



In [18]:

```
tc.plot_group_trajectories_only(8, traj_dict)
```

Out[18]:

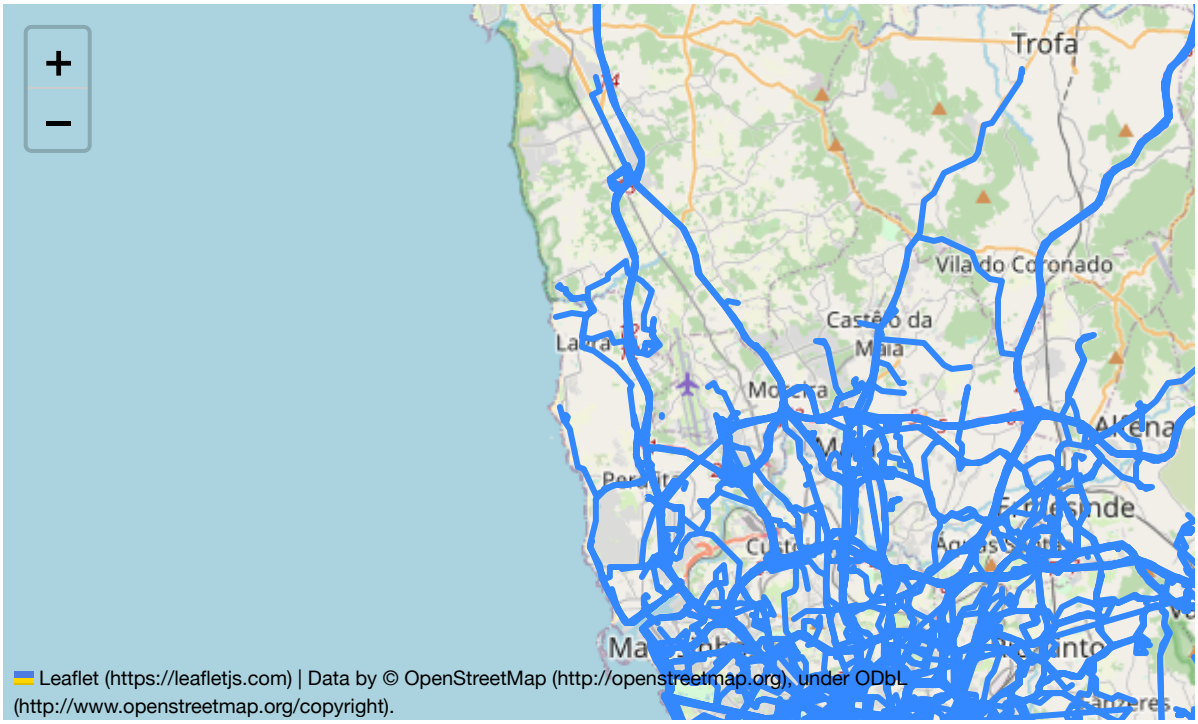


Noise group

In [19]:

```
tc.plot_group_trajectories_only(-1, traj_dict)
```

Out[19]:

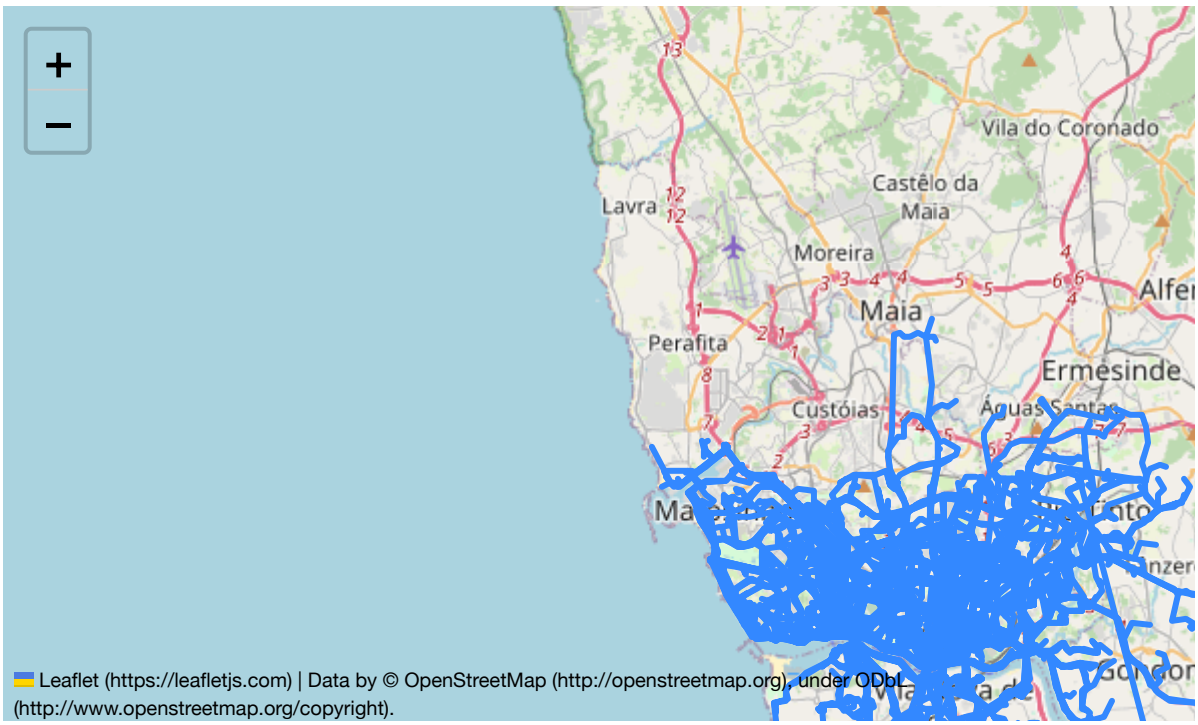


This group has not been clustered in finer granularity

In [20]:

```
tc.plot_group_trajectories_only(5, traj_dict)
```

Out[20]:



In [ ]: