# YEAR 2020-21

| | |
|---|---|
| <u>STUDENT</u> ID: | **20164326** |
| MODULE CODE: | **GEOG0051** |
| MODULE NAME: | **Mining Social & Geographic Datasets** |
| COURSE PAPER TITLE: | **Coursework01** |
| WORD COUNT: | **2453** |

# Task 1: Mobility Patterns Analysis in Cambridge

## 1.0. INTRODUCTION

This report provides commentary and justification for the methods selected and discusses the findings and interpretations at reasonable depth.

The project utilises the suggested Python libraries taught across the module while also leveraging other modules where appropriate and justified (i.e. Seaborn for expanding Matplotlib's plotting capabilities)

```
 1 # built-in
 2 import time
 3 from datetime import datetime
 4 from IPython.display import IFrame    # for embedding an HTML iframe into the IPython notebook
 5
 6 # numeric
 7 import numpy as np                    # for working with datasets as arrays/matrices
 8 import pandas as pd                   # for working with datasets as dataFrames/Series) - based on NumPy.
 9 import geopandas as gpd               # for working with geospatial data as dataFrames/Series - based on pandas
10
11 # networks
12 import networkx as nx                 # for creating networrts
13 import osmnx as ox                    # for retrieving datasets (incl networks) from OSM
14
15 # visualising
16 import matplotlib
17 import matplotlib.pyplot as plt       # for plotting data onto a graph figure
18 import matplotlib.cm as cm            # for accessing built-in colormaps and utilities
19 import matplotlib.colors as colors    # for converting converting numbers to RGB
20 import seaborn as sns                 # for plotting statistical graphs. Built on Matplotlib and integrated with pandas data structures.
21 import folium                         # for visualising data on a Leaflet map
22 import folium.plugins
23 from folium import FeatureGroup, LayerControl, Map, Marker
24 from folium.plugins import HeatMap
25 from folium.plugins import MarkerCluster
26 import contextily as ctx              # for visualising data on a retrieved basemap of web tiles. Works with GeoPandas/Rasterio
```

## 2.0. VISUALISE USER CHECK-IN LOCATIONS

### 2.1. Data Preparation Method

The ambition was to plot all check-in locations onto a map whilst distinguishing the selected users of interest, users [75027] and [102829]. Several methods were explored to assign these points to a Matplotlib map grid: these being, partitioning the dataframe by users, retaining a single dataframe but distinguishing users via a 'helper' column, and compiling lists from a conditional for loop. All methods were successfully implemented. The 'helper' column method involved the least activity, but plotting could only be achieved using the *pandas.plot()* function. This restricted the symbology to predefined 'cmap' themes. Despite considerable experimentation, no suitable cmap achieved the desired outcome of illuminating user check-ins on top of the wider sample. It was recognised that the For Loop method does not reflect the best method for machine optimisation, as it requires greater number of processes. It was nevertheless found this offered the most flexibility for selecting effective symbology.

### 2.2. Contextily Vs Folium Visualisations

For the end-user, the points are only meaningful when related to geographical context – a basemap of the Cambridge area was critical for visualising the check-in distribution effectively. For GeoPandas DataFrames, Contextily and Folium are the most established libraries for plotting web tiles. As both methods enabled the data to be represented in different formats, both were implemented. The Contextily approach outputted a static visualisation with all standard map element including a legend, grid, title and scale bar. Whereas the Folium approach borrowed Leaflet functionality to output a HTML interface where end-users can

interact with features. For instance, the visualisation is enhanced with pop-up boxes that details a point's attributes. Also, as Contextily tiles are projected into WGS84 Web Mercator, all points had to be reprojected to view alongside, whilst Folium conversely reprojects points on-the-fly by default. Moreover, Folium comes with plugins that enable the data to be interpreted in different ways. For instance, the heatmap plugin enabled the queried user check-ins to be seen amongst the wider sample, which provided scope to unpick trends. The data sample is large enough to make the resultant heatmap meaningful. However, Contextily provided ability to reduce the opacity of the tiles to give emphasis on the points, whilst Folium offered no flexibility to adjust. In conclusion, presenting both views as part of the project could be justified.
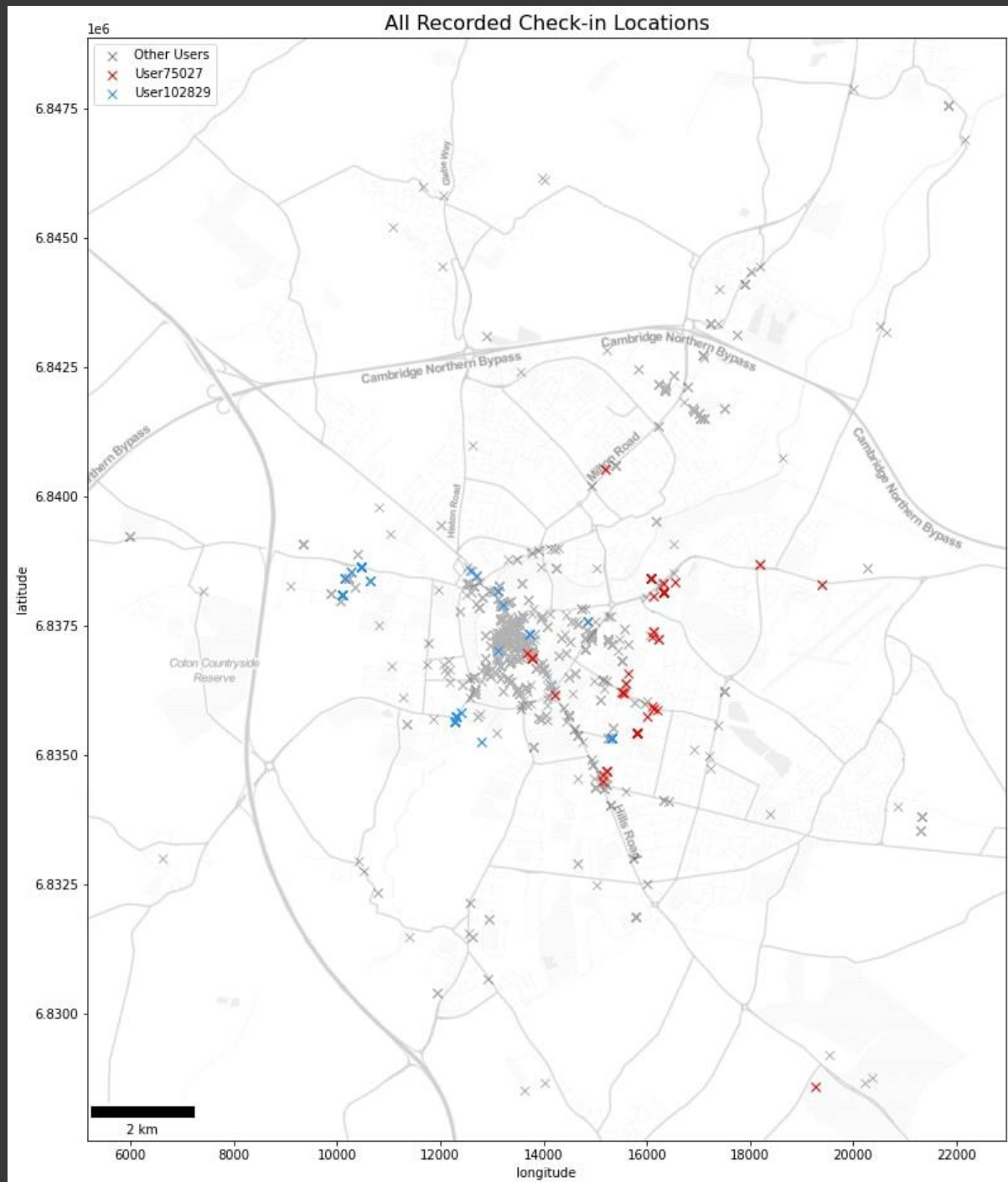


Figure 1.1 – Check-in distribution as static visualisation using Matplotlib with Contextily.
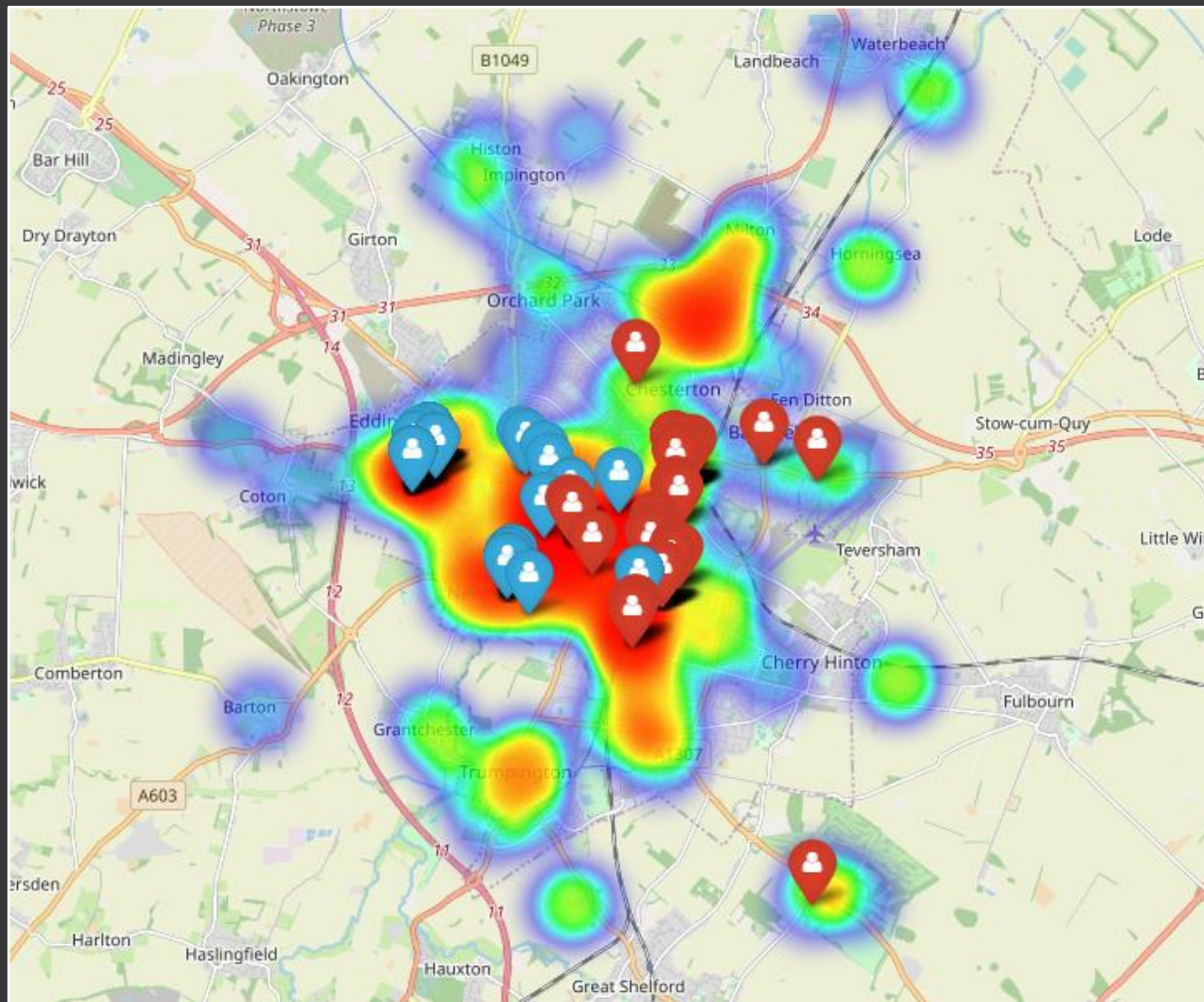
Figure 1.2 – Check-in distribution as interactive visualisation using Contextily.

2.3.  Results & Findings

Figures 1.1 and 1.2 show that the majority of check-ins occurred around the city centre of Cambridge. While the check-in locations for user 75027 (aka 'User Red') and user 102829 ('User Blue') largely correlated with the wider sample. Most check-ins in the wider sample occurred in the inner-city centre, whilst the distribution of User Red skews further east of the city centre, whilst User Blue skews further west. More specifically, User Red tended to check-ins clustered around retail/shopping sites (i.e., Cambridge leisure Park, Cambridge Retail Park, Beehive Retail Park), as well as suburban highstreets. The anomaly tends to be near Wandlebury Country Park, outside of the city. User Blue tended to check-in in or around sites of education (i.e., Cambridge University's west site) and leisure facilities (i.e., Gonville & Caius Sports Grounds). In conclusion, both visualisations successfully depict the distribution of user check-ins in relation to the wider sample.

2.4. Comment on Privacy Implications of Mobility Analysis (see last lecture)

As micro-level spatiotemporal information is highly sensitive, the data was anonymised by the provider to protect the privacy of the persons who generated the data. It is nevertheless straightforward to deduced intricate inferences about the persons due to the granularity of the Gowalla dataset. This dataset was captured using GNSS receiver ubiquitous with modern-day smartphones, which can determine user position, velocity, and precise time (PVT). And by extension, expose everyday check-ins and journeys, with potentially a horizontal accuracy of less than 1m (given optimum conditions). It is not unreasonable to presume User Blue was a young student who enjoys sport, whilst User Red was young professional working in retail. Mining spatiotemporal data therefore not only raises moral and ethical dilemmas relating to

location privacy, but also how people are consciously or unconsciously represented by data scientists (Floridi and Taddeo 2016; Duckham and Kulik; Krumm 2019. It was recognised that this project must take a fair and considered approach to data usage.


## 3.0. <u>CHARACTERISATION FOR U75027 ON 30/01/2010 AND U102829 ON 11/05/2010</u>

Two maps were created to deduce displacement information of both users in geographical and network space.

### 3.1. DateTime Formatting

The brief gave a specific datetime query to be carried out on the dataset for each user. Although the dataset was provided with UK date formatting of %D%M%Y, amalgamating the datetime columns on import with Pandas' *pd.to_datetime()* introduced errors. Due diligence on outputs found a large proportion of dates, which included the 11th May (User Blue query) was automatically converted to the US formatting and required code intervention to rectify the problem. This was corrected by reading in date and time separately, converting to strings, amalgamating the strings, and converting those to datetime object.

### 3.2. Deriving Route Information

The dataset contained an array of spatially and temporally referenced points that represented the user's check-in history. However, the emphasis in this task was to define routes in-between the check-in locations. This was achieved by reworking the GeoDataFrame (gdf) by shifting one check-in location on one row, to its sequential check-in on the following row, *gdf[column].shift(-1)*. This resulted in rows from check-in A to B, from B to C and from C to D (etc.) – see table below. Although pending waypoints, it provided useful insight into geographical displacement and constitutes a starting point for network modelling.

**CHECK-IN DATA**

| ID | User_ID | loc_ID | lat | lon | geometry | date_time |
|---|---|---|---|---|---|---|
| 1228 | 75027 | 477840 | 52.157223 | 0.173005 | POINT (0.17301 52.15722) | 2010-01-30 15:13:43 |
| 1227 | 75027 | 31968 | 52.204923 | 0.145606 | POINT (0.14561 52.20492) | 2010-01-30 16:04:04 |
| 1226 | 75027 | 478133 | 52.205200 | 0.144522 | POINT (0.14452 52.20520) | 2010-01-30 16:06:57 |
| 1225 | 75027 | 478191 | 52.205650 | 0.144826 | POINT (0.14483 52.20565) | 2010-01-30 16:15:35 |
| 1224 | 75027 | 373382 | 52.209822 | 0.146594 | POINT (0.14659 52.20982) | 2010-01-30 16:59:45 |
| 1223 | 75027 | 159331 | 52.209487 | 0.144786 | POINT (0.14479 52.20949) | 2010-01-30 17:05:41 |
| 1222 | 75027 | 374196 | 52.211325 | 0.144424 | POINT (0.14442 52.21132) | 2010-01-30 17:20:55 |
| 1221 | 75027 | 31328 | 52.198955 | 0.127612 | POINT (0.12761 52.19895) | 2010-01-30 21:27:50 |
| 1220 | 75027 | 473651 | 52.202858 | 0.123643 | POINT (0.12364 52.20286) | 2010-01-30 22:14:11 |
| 1399 | 102829 | 1024097 | 52.196113 | 0.110121 | POINT (0.11012 52.19611) | 2010-05-11 06:30:18 |
| 1398 | 102829 | 373983 | 52.212543 | 0.094001 | POINT (0.09400 52.21254) | 2010-05-11 07:24:49 |
| 1397 | 102829 | 1031743 | 52.211119 | 0.095404 | POINT (0.09540 52.21112) | 2010-05-11 10:57:10 |
| 1396 | 102829 | 963000 | 52.196678 | 0.110528 | POINT (0.11053 52.19668) | 2010-05-11 18:03:09 |

**ROUTE DATA**

| route_id | user_id | orig_id | orig_lat | orig_lon | orig_geom | dest_id | dest_lat | dest_lon | dest_geom | departure_time | arrival_time | journey_duration |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 75027 | 477840 | 52.157223 | 0.173005 | POINT (0.17301 52.15722) | 31968 | 52.204923 | 0.145606 | POINT (0.14561 52.20492) | 2010-01-30 15:13:43 | 2010-01-30 16:04:04 | 3021.0 |
| 2 | 75027 | 31968 | 52.204923 | 0.145606 | POINT (0.14561 52.20492) | 478133 | 52.205200 | 0.144522 | POINT (0.14452 52.20520) | 2010-01-30 16:04:04 | 2010-01-30 16:06:57 | 173.0 |
| 3 | 75027 | 478133 | 52.205200 | 0.144522 | POINT (0.14452 52.20520) | 478191 | 52.205650 | 0.144826 | POINT (0.14483 52.20565) | 2010-01-30 16:06:57 | 2010-01-30 16:15:35 | 518.0 |
| 4 | 75027 | 478191 | 52.205650 | 0.144826 | POINT (0.14483 52.20565) | 373382 | 52.209822 | 0.146594 | POINT (0.14659 52.20982) | 2010-01-30 16:15:35 | 2010-01-30 16:59:45 | 2650.0 |
| 5 | 75027 | 373382 | 52.209822 | 0.146594 | POINT (0.14659 52.20982) | 159331 | 52.209487 | 0.144786 | POINT (0.14479 52.20949) | 2010-01-30 16:59:45 | 2010-01-30 17:05:41 | 356.0 |
| 6 | 75027 | 159331 | 52.209487 | 0.144786 | POINT (0.14479 52.20949) | 374196 | 52.211325 | 0.144424 | POINT (0.14442 52.21132) | 2010-01-30 17:05:41 | 2010-01-30 17:20:55 | 914.0 |
| 7 | 75027 | 374196 | 52.211325 | 0.144424 | POINT (0.14442 52.21132) | 31328 | 52.198955 | 0.127612 | POINT (0.12761 52.19895) | 2010-01-30 17:20:55 | 2010-01-30 21:27:50 | 14815.0 |
| 8 | 75027 | 31328 | 52.198955 | 0.127612 | POINT (0.12761 52.19895) | 473651 | 52.202858 | 0.123643 | POINT (0.12364 52.20286) | 2010-01-30 21:27:50 | 2010-01-30 22:14:11 | 2781.0 |
| 1 | 102829 | 1024097 | 52.196113 | 0.110121 | POINT (0.11012 52.19611) | 373983 | 52.212543 | 0.094001 | POINT (0.09400 52.21254) | 2010-05-11 06:30:18 | 2010-05-11 07:24:49 | 3271.0 |
| 2 | 102829 | 373983 | 52.212543 | 0.094001 | POINT (0.09400 52.21254) | 1031743 | 52.211119 | 0.095404 | POINT (0.09540 52.21112) | 2010-05-11 07:24:49 | 2010-05-11 10:57:10 | 12741.0 |
| 3 | 102829 | 1031743 | 52.211119 | 0.095404 | POINT (0.09540 52.21112) | 963000 | 52.196678 | 0.110528 | POINT (0.11053 52.19668) | 2010-05-11 10:57:10 | 2010-05-11 18:03:09 | 25559.0 |

### 3.3. Displacement Plot

A network graph was created using the NetworkX library to visualise the sequence of check-ins on their respective sample date. This was read into a network graph using the OSMnx library, where the check-ins became nodes while routes became edges. Figure 2.1 illustrates the sequence of check-ins and the 'as-the-crow-flies' displacements. Note that both user graphs have been plotted on the same map for the end-user's benefit, which involved specifying bounding box dimensions that fitted both sets of point coordinates.

It can be gleaned from Figure 2.1 that User Blue's journey started and ended from the same location in the suburb of Newnham and to the University of Cambridge site. Conversely, User Red's journey covered a greater distance from the southeast outskirts of Cambridge to the city centre via retail sites in the east.
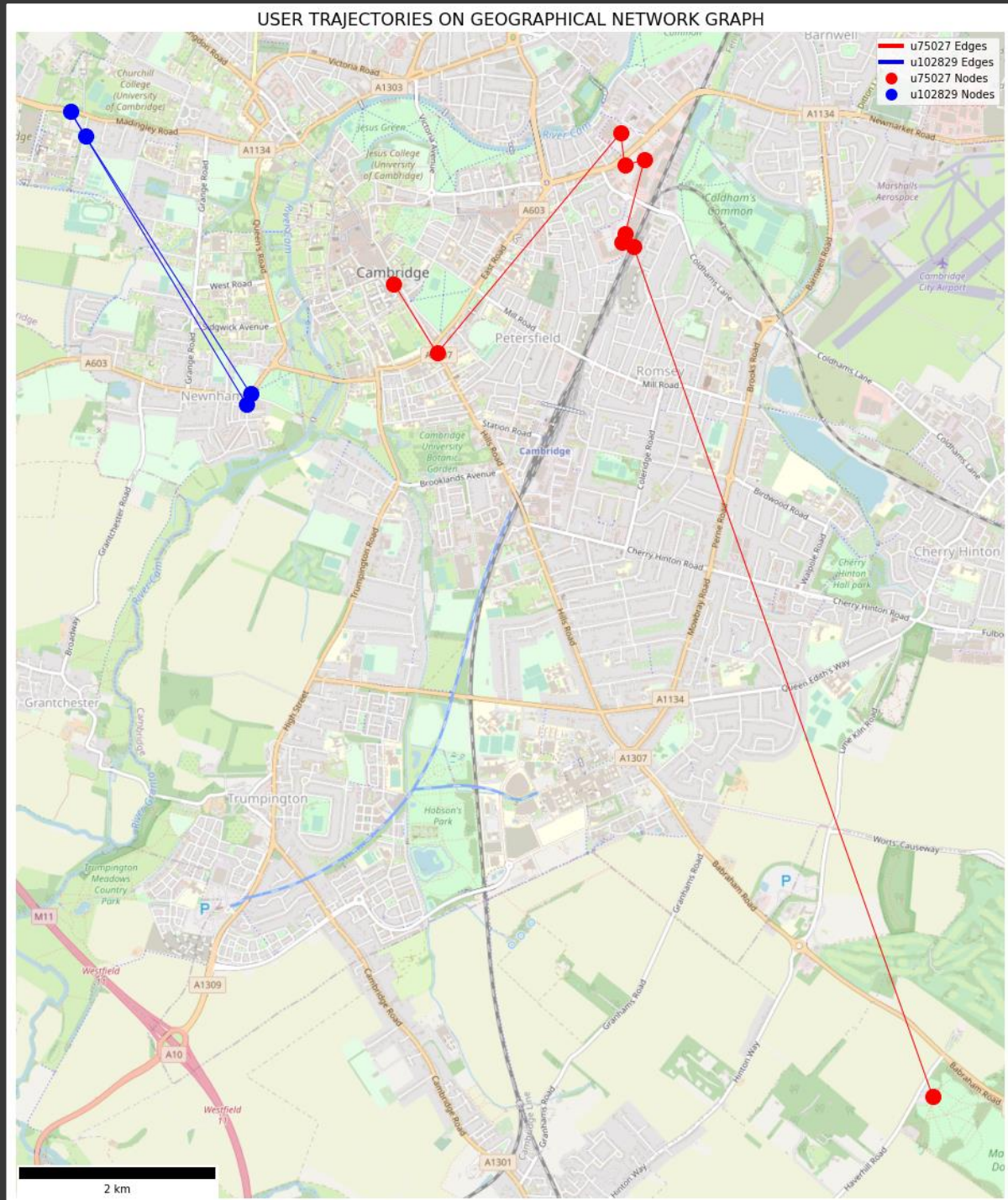


Figure 2.1 – Displacements on geographical plot

## 3.3. Street Network

Figure 2.1 provides displacements information that provides useful context. However, there is much missing between these milestones in the user's daily journey. The dataset does not specify connections between check-ins. As streets are nodes and connections (between streets) are edges/junctions, this information could be inferred by referencing an existing street

network and paths derived from using shortest path logic. The OSMnx library, an extension of NetworkX capability, provides the ability to access OpenStreetMap's warehouse of network datasets by specifying a location query (i.e., geocoded address, bounding box) and network type (i.e., walk, drive, all). For the former, a minimum bounding box (MBR) was calculated to request only parts of the network dataset that falls within the range of the points. As the brief does not specify the user's mode of travel, the requested network type was purposely requested was generic, *network_type='all_private'* – an unrestricted, undirected graph. It was nevertheless acknowledged that specifying different network types would request different iterations of the network dataset, which would greatly influence the outcomes of the route path. The network dataset provided waypoints to 'colour in' the gaps between the known check-in locations. A For Loop was used to iterate through a selection of nodes (path based on weighting); setting origin as source; finding neighbours of source; giving each neighbour; a distance from source and add to a queue; removing source node; setting the closest node in the queue as the new source and repeating until there the final destination was reached.

Figure 2.2 – Routes on Network Graph Map
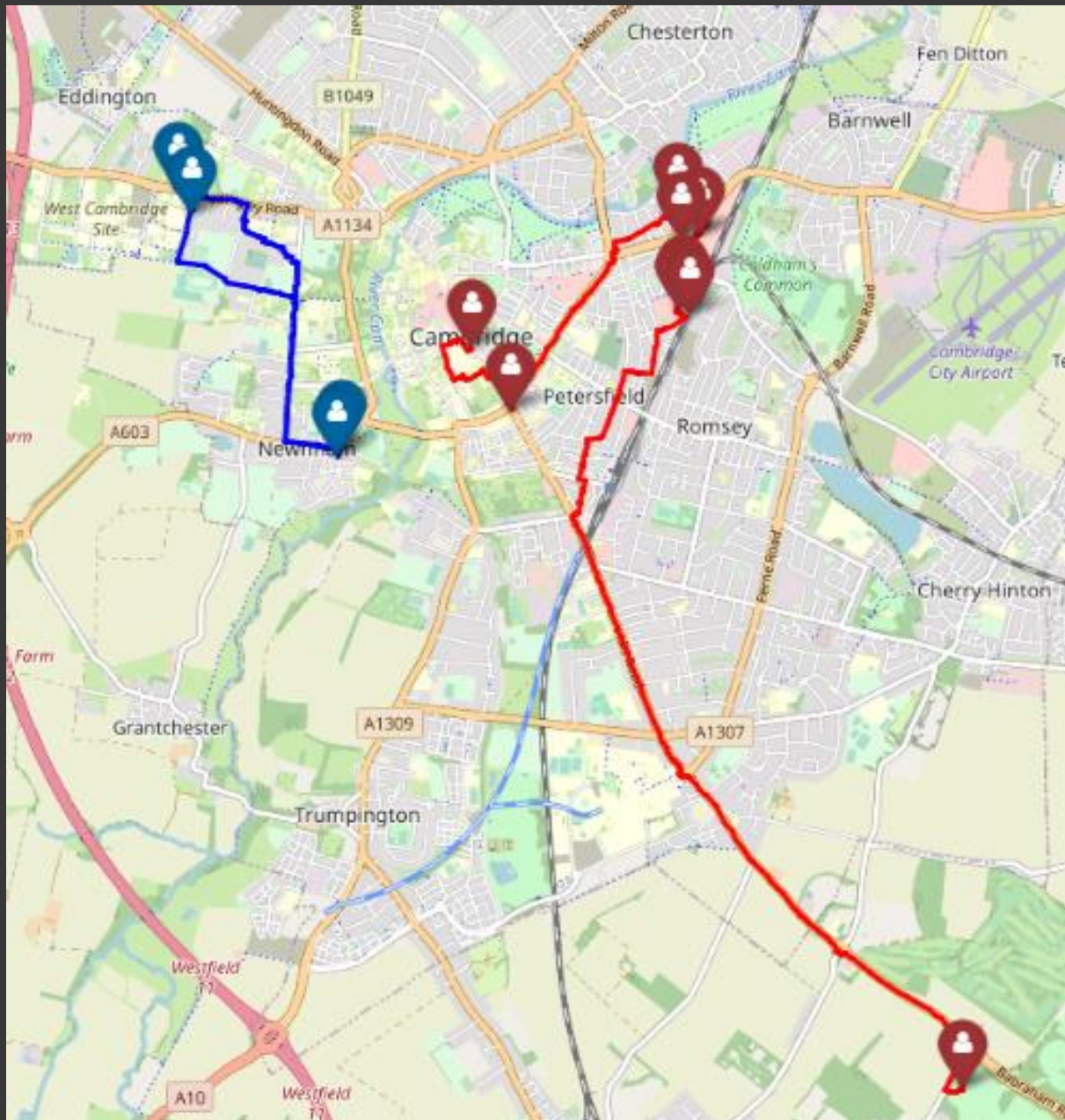
Figure 2.3 – Routes on Folium Map

3.3. Compute Displacement Statistics

|  | u75027 | u102829 |
| --- | --- | --- |
| Max Displacement | 7125.5670 | 2892.1630 |
| Average Displacement | 1463.4955 | 1923.9434 |
| Total Displacement | 11707.9640 | 5771.8300 |

The table details the maximum, average and total distance travelled on the day and summarise answers in a table. These were calculated by iteratively compiling a list of weights in lengths(m) between each route's origin and destination nodes and identifying from the list: the highest value (for maximum displacement), the mean value (for average displacement) and the sum of all values (for total displacement). It was unsurprising that these values are higher than values identified for geographical displacement, which uses 'as-the-crow-flight' logic to accumulate distances. More insight could have been gathered by investigating other individual mobility measurements such as radius of gyration (mean distance covered from the 'centroid' of traces), most frequented locations or travel time budgets.

3.4. Discrepancies

It must be reiterated that the network path is likely to deviate somewhat from the actual path taken. Unlike machines, people are innately unpredictable. They're not always willing to take the shortest route, or even a route known by the dataset. Similarly, the environment is not consistent and factors such as traffic and weather influence mobility patterns. In addition, nodes only represent a known point in continuous space and someone's actual location might lie somewhere distant from nodes, or in-between multiple nodes. One can anticipate from this ambiguity that, the deviation between modelled and actual routes would increase over larger displacements in areas with greater network connections. An additional process to reduce this discrepancy could be to create a function that compares the network duration (from calculating time across edges) to the actual time (from calculating difference between check-ins) and reroute accordingly until the actual duration requirement is met. Street networks nevertheless provide a reasonable framework for analysing mobility.

## 4.0. <u>COMPARATIVE ANALYSIS OF CHECK-IN FREQUENCIES & NETWORK CENTRALITY</u>

4.2. Detecting Closeness Centrality

Closeness centrality obtained by specifying 'Cambridge, UK' as the spatial query in *graph_from_address()* function and using NetworkX to inscribe a centrality measure to each node (line) as an edge attribute. This was plotted to illuminate the 'global centre' of the network where nodes are closest. This was found to be located towards the east of the main shopping district in the city centre where there are greater number of nodes and node connections. It must be reiterated that the spatial query and distance type used in the acquisition of the network boundary has a bearing on the network graph obtained. For instance, *dist_type='bbox'* clips the network to a distance, whilst *dist_type='network'* uses machine learning to only retain only nodes within some network distance from the center-most node. Moreover, obtaining Cambridge boundaries using *geocode_to_gdf('Cambridge,UK')* provides a jagged administrative boundary that encompassed the wider suburbs and countryside. In conclusion, a different collection of nodes could change, albeit subtly, the global centre.

4.1. Describe pattern of all user check-ins in relation to closeness centrality

Although both the global centre and distribution of check-in locations both occur in the centre of the city, check-in locations tend be distributed further west of the global centre. To investigate the discrepancy, a selection of geometries were imported to reflect the urban make-up and space-based topologies of the city; buildings, Land-use: Retail & Commercial, Amenities and Bus-Stops. All such factors tended to correlate more with check-in locations than the global centre of the city. For instance, the majority of check-ins occurred in or around the commercial and retail space of 'The Grand Arcade'. This affirms that the most connected part of Cambridge is perhaps not necessarily the most popular for visitors. Urban planners can capitalise on centrality measures to better construct spaces that work for communities.
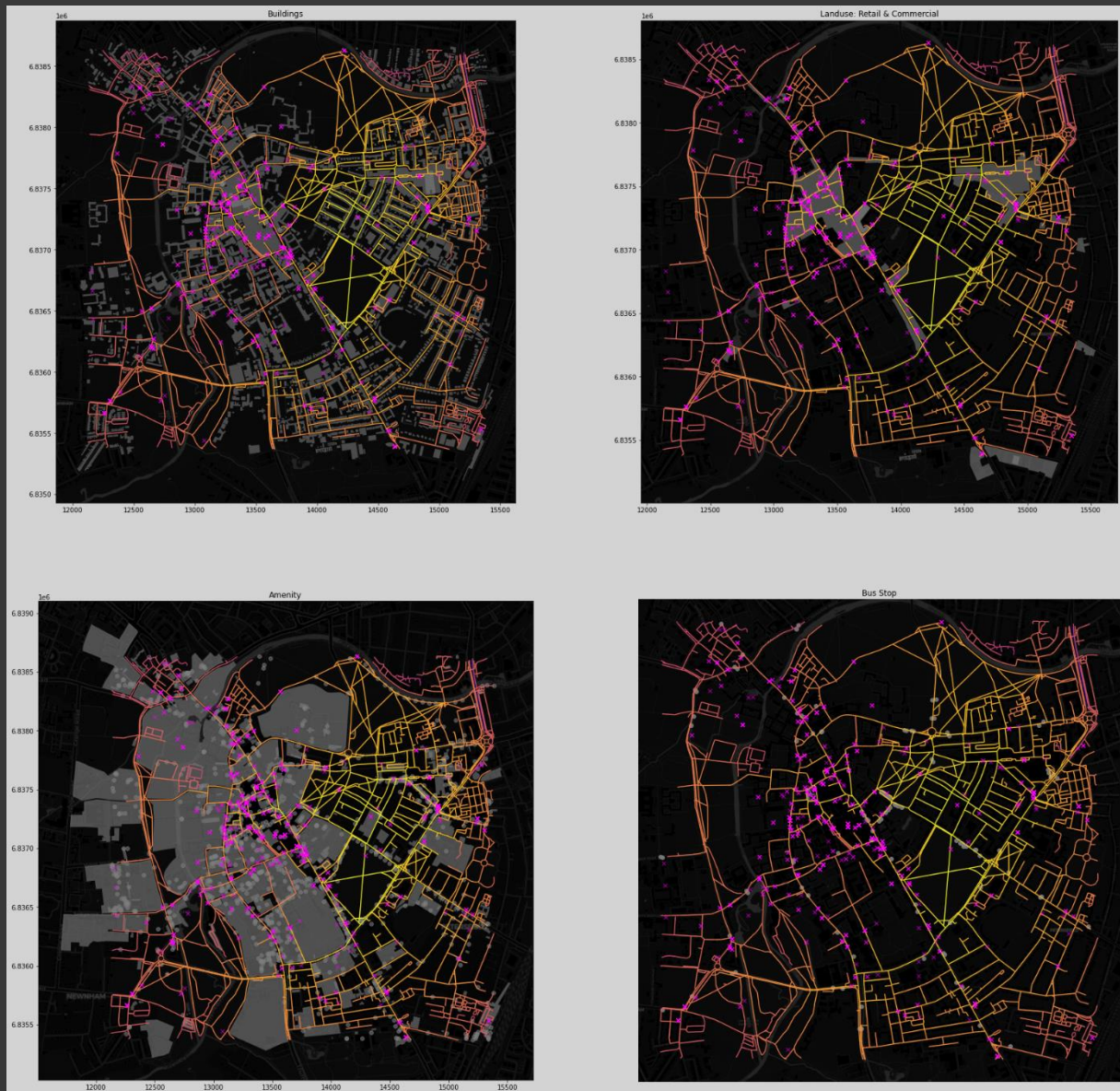
Figure 3 – Closeness Centrality, Check-in Locations & Space Topologies

## 5.0. URBAN PLANNING APPLICATION

The project proposes that a new fire station in the northeast region of the city centre near the roundabout that intercepts East Road, Elizabeth way and Newmarket Road. This area is close to the global centre of the city of Cambridge whilst backing onto major road arterials and numerous minor passthroughs, going into and out of the city centre. This area therefore constitutes the most connected part of the city that facilitates optimal response routes/times and serves optimal populations. This area is also separated from the main commercial and shopping district in the city in the west and therefore, would be subjected to lesser traffic congestion. To develop the idea for a new fire station in this zone, it is suggested that further network analysis is conducted to measure travel time radius and accessibility. This should form part of a multi-criteria analysis that assesses a range of social and physical factors adheres to urban design principles.

## Task 2: Machine Learning Analysis with Venue Review Data



Please be aware that I suffered continuous technical difficulties during Task2 and my progress was severely hampered by repeated crashes whilst handling the 'df_Calgary_pre.csv' I continued with best endeavours, but please refer to the Python notebook for direction and elaboration of methods.

## 6.0. LOADING AND CLEANING TEXTUAL DATASET

6.1. Load and understand the dataset



```
# number of unique users and items
print ("There are " + str(len(df['user_id'].unique())) + " unique users_ids in the dataset.")
print ("There are " + str(len(df['business_id'].unique())) + " unique business_ids in the dataset.")
[OUT] There are 25138 unique users_ids in the dataset.
There are 5234 unique business_ids in the dataset.
```

```
# top five users in terms of rating counts
```

```
ascending_list = df.groupby('user_id')['stars_y'].count().reset_index().sort_values('stars_y',ascendi
ng=False)
display(ascending_list.head())
[OUT]
```

| | user_id | stars_y |
|---|---|---|
| 13369 | Wx7cbLDqYEL3_aVZwh82Ww | 397 |
| 20669 | oUK6Xs5dPPnP4whFeZExGg | 368 |
| 13960 | YRcaNlwQ6XXPFDXWtuMGdA | 365 |
| 6150 | EsOu51dW3UTDJTApBxwz2g | 330 |
| 120 | -InhDRRVG7wrwsgAUvN4Qw | 314 |

6.2. Pre-process the text review data

## 7.0. BUILD A SUPERVISED LEARNING MODEL FOR TEXT ANALYSIS

7.1. vectorise the preprocessed review text data to give text features

7.2. Split dataset into a train and test-set

7.3. Train K >2 machine learning models to predict the star rating

## 8.0. RUN A LEXICON-BASED SENTIMENT ANALYSIS ON TEXTUAL DATA

8.1. Run NLTK Vader Sentiment Analyser on the textual data

Review Vs Sentiment Scores per Business