Identifying Urban Functional Regions

Cardiff University School of Computer Science and Informatics

BSc Computer Science

Author: Joshua Rimes c22002134

Supervisor: Dr. Padraig Corcoran

Moderator: Dr. Crispin Cooper

Abstract

This project explores the spatial correlation between urban functional regions, identified through clusters of Points of Interest, and socio-economic variables within urban areas. By integrating geospatial analysis, clustering algorithms, and socio-economic datasets, the aim was to investigate whether the spatial distribution of functional services such as retail, sports, and entertainment correlates with socio-economic trends across different urban areas. A custom Python-based system was developed to automate data preprocessing, clustering using DBSCAN, and correlation testing using Pearson's coefficient.

The results suggest that while meaningful correlations do exist, particularly in certain age groups and service types, they are often subtle and spatially dependent. Retail clusters showed modest correlation with age demographics, while sports & entertainment clusters showed stronger relationships with both younger and older populations. Several technical limitations were encountered, including memory constraints and time limitations that restricted the analysis to a single city and a limited set of POI classifications. Nevertheless, the project demonstrates the viability of using POI clustering as a proxy for functional urban layout and highlights its potential value in informing urban planning and policy decisions. Future work could expand this framework to include more cities, a broader array of POI categories, and higher-resolution spatial units, offering even deeper insights into the interplay between urban function and human demographics.

Acknowledgements

I would like to thank my supervisor Dr. Padraig Corcoran for his enthusiasm, knowledge, and help across the entire project which was of great value. I would also like to thank my girlfriend Isobel and housemates Jacob, Louie, and Ross for keeping me motivated, as well as my family for their support.

Table of Contents

1. Introduction

Urban environments are shaped not just by their buildings and roads, but by how their functional elements are distributed throughout the city. Functional elements are the building blocks of urban areas; every building has a functionality and contributes towards the success of urban growth. Whether it is a shop providing groceries, a factory providing mechanical parts, or a police station providing emergency services, these elements provide stability and opportunity to the surrounding area. Understanding how this layout affects the people who live and work there is vital for planning future development, improving quality of life, and ensuring sustainable urban growth **(Khalil 2012)**. This project explores whether the spatial distribution of Points of Interest (POIs) can be used to identify functional regions within cities, and whether those regions show any correlation with socio-economic variables such as income, employment, and age distribution.

The goal of this project is to develop a Python-based system that clusters POIs based on their location and type, then overlays the results with socio-economic heatmaps to assess potential correlations. The system will help determine whether the structure and layout of urban functions influence the economic and social dynamics of a city. The intended beneficiaries of this research include urban planners, data analysts, and researchers in urban geography and socioeconomics who wish to explore lightweight tools for mapping urban function. The system's simplicity and interactivity make it useful not only in academic settings, but also for local councils or government organisations that want to analyse cities without relying on large-scale GIS platforms.

My project is based on the assumption that POI data can act as a meaningful proxy for urban function, and that demographic and economic datasets can be accurately aligned with POI clusters using shared spatial references. The broader outcome of this project is a functional web-based application that offers an interactive visualisation of POI-based clusters and their potential relationship with socio-economic indicators. Whether the system finds strong, weak, or no correlations, the result will contribute to our understanding of how urban form and human activity are interlinked.

To provide further context, the following key terms are used throughout this report:

- Socio-economic: A term used to describe the interaction between social and economic factors, such as employment, age range, education, income levels, and health. In this project socio-economic variables are used to evaluate the conditions of different urban areas.
- Quality of life: This refers to the general well-being of individuals and societies, encompassing not only wealth and employment but also the built environment, physical and mental health, education, and recreation. Mercer's Quality of Living Survey includes criteria such as personal safety and security, health, transport

infrastructure, availability of consumer goods, and adequate housing, schooling, and recreation opportunities **(Faust 2021)**.

- Urban layout: The spatial structure and organisation of a city, including the positioning of roads, public spaces, commercial zones, and residential areas. It has a significant influence on transport efficiency, accessibility to services, and socio-economic equity.
- POI (Point of Interest): A POI is any specific location in a city that someone might find useful or interesting—this includes retail shops, schools, hospitals, parks, transport stations, and more. In this project, POIs are classified, spatially analysed, and clustered to infer the functional character of different urban regions.

1.1 Project Aims

The aim of this project is to determine whether urban layout and functional POI clusters, have a measurable relationship with socio-economic conditions in UK cities. This aim is supported by three core aims:

Aim 1

I will implement a Python system that preprocesses geospatial POI data, automates the identification of POI clusters using a clustering algorithm, and visualises the resulting clusters interactively on a map. The system will allow users to overlay these clusters with heatmaps of socio-economic variables such as income or age distribution. This enables direct visual and analytical comparisons between urban function and socio-economic structure. The system will be built using Python libraries such as Plotly for visualisation and Dash for interactivity. There are minimal risks associated with this objective, as the technical implementation depends on well-supported open-source tools.

Aim 2

I aim to collect and analyse POI data from at least two cities in the UK. Initially, Cardiff and Bristol were chosen due to their similar size and population, which provides a fair basis for comparison. Using this data, I will identify and record functional clusters, then compare those clusters to census-based and publicly available data covering variables such as employment rates, economic inactivity, and age structure. One of the primary risks here lies in data availability, specifically, whether high-quality, granular socio-economic data exists for both cities in a consistent and comparable format. While POI data is widely accessible through platforms like Digimap **(Appendix 1)**, socio-economic data may be fragmented or outdated. To mitigate this, the project focuses on datasets from reliable sources such as the Office for National Statistics (ONS). Another risk is geographic bias; both cities are located in the southwest of the UK, and their socio-economic similarities may be due to regional effects rather than urban layout. If needed, I may consider substituting one city for another with more distinct characteristics, although this introduces further complexity in terms of comparability and data availability.

Aim 3

The final and most important objective is to evaluate whether there is any measurable correlation between POI clusters and socio-economic patterns in the chosen cities. This is the crux of the investigation. A strong correlation could suggest that the spatial layout of urban functions influences quality of life or economic performance. Conversely, finding no correlation would still be a meaningful result, indicating that other factors, beyond layout, may dominate socio-economic outcomes. Since every conclusion (positive, negative, or neutral) contributes to a better understanding of urban dynamics, there is no risk of failure in achieving this objective from an academic perspective.

If correlation was discovered between POI clusters and socio-economic patterns, it could have implications for how urban environments are analysed and managed. City planners and policymakers could use insights from the correlations to inform strategic interventions aimed at improving quality of life, targeting economic investment, or addressing social inequality in specific regions. For example, identifying underserved areas lacking in key functional POIs, such as healthcare, education, or retail, could guide resource allocation to improve accessibility and equity. Understanding how different age groups or income brackets align with the spatial distribution of functional elements could support more inclusive planning decisions. This kind of data-driven spatial analysis could be a valuable tool in building smarter and more responsive urban development.

2. Background

This section provides the contextual foundation for the project, outlining the importance of urban planning, the impact of spatial layout on socio-economic conditions, and the methodologies used in previous studies to identify functional regions. By examining both theoretical and practical approaches to urban structure analysis, this background establishes the motivation behind using POI data and socio-economic overlays to investigate the functionality of urban areas. It also highlights the gap in existing research that this project aims to address.

2.1 Why is Urban Planning and Layout Important

The significance of this project lies in the importance of layout of urban areas. Cities rely on urban planning to remain functional, grow in population, and attract businesses. Every crucial aspect of an urban environment is under the effect of how its layout is planned **(Cesar 2023)**. On an economic level, a city must meet the needs of workers to retain them and build successful businesses. Companies will set up offices in areas that attract employees and can provide comfort to those employees living close by.

To grow, a city must have a core that can support the expansion of urban areas. In extreme cases such as London, multiple cores form to adjust to the sheer scale of its urban areas. These cores have a focus on corporate and entertainment functions; offices and headquarters are likely found, along with retail and attractions such as shopping centres and cinemas. Outside of the cores, the functionality of urban areas depends on multiple variables discussed in section 2.2.

The ideal urban layout is not a feasible system to achieve; the characteristics of an ideal urban layout system may exist in the absence of external disturbances such as political, social, and economic considerations **(Gushchin et al. 2017)**. However, if a structure or system can be found to improve the growth of a city, we can improve all aspects of life in an urban setting. The potential of growth in a city includes improved health and quality of life, less environmental impact, better economy and resource utilisation, national development, disaster prevention and greater credibility, more efficient problem solving and creativity, and more **(Cesar 2023)**.

2.2 Existing Views on the correlation between socio-economic factors & urban layout

The relationship between urban layout and socio-economic factors has been extensively discussed in planning and development literature. Highly developed cities contribute significantly to the economic health and productivity of society, while poorly planned urban areas often experience negative outcomes such as heavy traffic congestion, inadequate housing, and overstretched infrastructure **(Cesar 2023)**. A well-planned city layout provides access to essential services, amenities, and points of interest, which in turn improves the quality of life and promotes healthier lifestyles among residents **(Cesar 2023)**.

One factor highlighted in modern studies is street accessibility. Research from the Centre for Cities **(Magrini 2017)** indicates that the street layout of a city can heavily impact its economic success. In their study, a strong positive relationship was found between a city's street 'accessibility score' and its overall economic performance. Cities with more accessible, grid-like street patterns typically facilitate better movement of people and goods, which enhances economic activity. However, this research mainly focuses on street patterns and does not directly address the influence of broader functional regions or POI distributions.

Socio-economic indicators are often used to assess the success of an urban area, but quantifying these indicators can be challenging. Although the concept of quality of life has been in the development discourse for some time now, measuring it in a city is quite difficult as the aspects to be measured are still questionable **(Khalil 2012)**. For example, Mercer's Quality of Living Survey lists factors such as housing, economic environment, public services, and recreational facilities as key criteria, but achieving a universally accepted standard remains difficult. Urban initiatives often prioritise areas like shelter, access to services, and fostering local economic development to improve living conditions **(Khalil 2012)**.

Another perspective comes from examining how public spaces are used within urban environments. Critiques suggest that analysis of public spaces can sometimes be distorted to fit a particular thesis, and that definitive evidence supporting either positive or negative claims about public space design is limited **(Carmona 2010a; Carmona 2010b)**. This implies that while socio-economic outcomes and urban design are linked, drawing conclusions without acknowledging context-specific variables can be misleading.

The current body of research supports the view that socio-economic outcomes are closely tied to urban structure. However, while there is strong qualitative agreement on this relationship, the quantitative evidence remains complex and nuanced. This reinforces the importance of careful, data-driven methods in attempts to identify functional regions and assess their socio-economic impact.

2.3 Research on POI & Socio-Economic Datasets

The success of this project relies heavily on the use of reliable and well-structured data sources for both Points of Interest (POI) and socio-economic variables. Multiple data providers were investigated, but not all sources were suitable in terms of accessibility, relevance, or data structure.

Foursquare offers a sophisticated place engine that uses machine learning to synthesise and validate over 16 billion data points to ensure a high level of accuracy in its location-based services **(Foursquare 2024)**. However, despite its vast scope and technical robustness, Foursquare is primarily designed for commercial use cases and does not offer freely accessible geographic POI data for student projects, making it unsuitable for this system.

OpenStreetMap (OSM) is an open-source mapping platform that provides detailed geographical data, including points of interest and street-level detail **(OpenStreetMap [no date])**. While it is a valuable tool for viewing urban layouts, the data it uses is crowdsourced which is unreliable. Its reliability can significantly differ across different geographic areas which means there is a strong emphasis on assessing OSM data quality before applying it to specific use cases in many previous studies **(Biljecki et al. 2023)**.

Commercial providers like Precisely offer access to curated POI datasets, which may contain thousands of classified entities tailored for business use **(Precisely Inc. 2023)**. However, like Foursquare, the cost and licensing restrictions of these datasets make them inaccessible for educational research at the undergraduate level.

In contrast, Digimap emerged as the most viable solution for both POI and socio-economic data. Provided through a Cardiff University subscription, Digimap offers access to a wide range of datasets specifically intended for educational and research purposes **(Digimap 1996)**. The Ordnance Survey Map section contains extensive POI data, with a structured hierarchy consisting of 9 top-level groups, 52 mid-level categories, and over 600 detailed classifications. This granularity was particularly beneficial for categorising urban functions in this project. Additionally, the POI dataset offers four levels of positional accuracy and includes 25 attribute fields per record, enabling precise and flexible filtering of data to meet project requirements. Unlike OpenStreetMap, the Ordnance Survey is a national mapping agency with trained professionals who guarantee quality data.

Socio-economic datasets were also sourced through Digimap from the Office for National Statistics, which includes variables such as population density, age distribution, profession types, employment rates, and public health data. For further granularity and updated statistics, additional datasets were downloaded from the Office for National Statistics (ONS) website **(Appendix 2)**, including 2011 Census data that categorises residence type by age, as well as economic indicators like regional Gross Domestic Product (GDP) and employment statistics for specific urban regions such as Cardiff **(ONS 2024; ONS 2025)**.

Together, these datasets provided a robust foundation for identifying spatial correlations between POI clusters and socio-economic variables. The richness and structure of the Digimap and ONS data allowed for detailed filtering and overlay techniques, which are critical for building a system that aims to visualise functional regions within urban spaces.

2.4 Research on POI Data

The structure and representation of Point of Interest (POI) data play a critical role in identifying and analysing urban functional regions. I researched range of representations that are available for structuring POI datasets, each with advantages depending on the analysis being performed. Coordinate-based representation is the most common in spatial analysis, where each POI is geolocated using latitude and longitude, allowing for precise spatial mapping and distance-based clustering. This format is particularly useful for visualisations such as scatter maps and geospatial overlays **(Kelly 2024)**.

Address-based representation complements coordinate-based data by providing textual identifiers for each POI, improving human readability and usability in applications like search queries and routing algorithms. Categorical representations group POIs based on attributes such as industry type or urban function. This method allows data to be organised into high-level classifications, making it easier to filter and analyse specific types of urban activity.

Many datasets also implement hierarchical categorisation, which enables users to navigate between broader and more granular classifications of POIs. For instance, the Ordnance Survey POI classification system groups data into 9 main functional groups, including 'Accommodation, eating and drinking', 'Education and health', and 'Retail', with a total of 52 subcategories nested within these top-level groups **(Ordnance Survey Ltd 2020)**. This structure not only improves semantic clarity but also allows for advanced filtering and visual labelling, such as assigning colours or symbols to each group for improved map interpretation.

Other representations such as attribute-based and graph-based models were not applicable in this project. Attribute-based POI data contains metadata fields such as business names, opening hours, and service descriptions, enhancing the ability to perform qualitative analysis. Graph-based representations treat POIs as nodes in a network, with spatial or functional relationships, such as proximity or co-location, represented as edges. This structure supports advanced algorithms like centrality analysis or spatial clustering using graph theory **(Kelly 2024)**.

For the purposes of this project, the coordinate and hierarchical classification provided by Ordnance Survey was chosen due to its detailed granularity and compatibility with geospatial analysis tools. Its combination of coordinate precision and structured classification made it ideal for visualising urban function and identifying patterns when overlaid with socio-economic data.

2.5 Research on similar studies

The identification of functional regions has traditionally been approached through two main classificatory frameworks: formal and functional regions. Formal regions are defined by homogeneity, where each area within the region shares one or more measurable attributes. This might include language, population density, or land use **(Brown and Horton 1971)**. Functional regions, on the other hand, are based on the interaction between spatial units; areas that exhibit a greater degree of internal connectivity than with areas outside the region. In this case, functional distance becomes the key concept, referring to how likely two spatial entities are to interact based on their attributes within an n-dimensional space **(Brown and Horton 1971)**. Markov chain analysis is one method used to map these interaction patterns, with mean first passage time (MFPT) applied to flow matrices to measure functional distance.

While such methods are effective for analysing macro-scale regional systems, they are not well-suited for fine-grained urban layouts. The large-scale nature of interaction-based functional regions often results in generalised zones that obscure smaller patterns of urban activity. This project aims instead to apply a more localised approach, aligning with intra-city functional dynamics derived from POI clustering, an approach that remains underrepresented in current literature.

I discovered that more recent research has begun to explore region identification at smaller urban scales using advanced machine learning models. In one such study, a combination of morphological and socioeconomic features was used to classify urban blocks into functional regions **(Yang et al. 2022)**. The study relied on building-level data and residential structures, which are not available in the POI dataset used for this project. Morphological features were extracted using a Graph Convolutional Neural Network (GCNN) applied to building block structures, while socioeconomic features were derived using a Word2Vec (W2V) model to analyse the internal POI structure of each block. The W2V embeddings created vector representations of POI categories based on semantic similarity, which were then aggregated to reflect the socio-economic profile of each area. These features were processed through a stacking ensemble model consisting of two base classifiers and a meta-classifier to assign probabilities that each block belonged to a specific functional region.

This method, while comprehensive, presents limitations that make it unsuitable for this project. Firstly, it relies on residential building data, which is absent from the Ordnance Survey POI dataset available via Digimap. Secondly, the approach is computationally intensive and heavily reliant on block-level morphology, information that is not only unavailable in this context but also less relevant when the goal is to identify regions based on urban function rather than structural form.

Another study examining the socio-environmental value of urban green spaces used remote sensing and land classification techniques to analyse the health and well-being benefits of public parks, gardens, and street greenery **(Wilkerson et al. 2018)**. This study reinforces the connection between urban spatial features and socio-economic outcomes, particularly in the context of quality of life. However, it does not address functional region identification or attempt to relate spatial clustering of amenities to demographic or economic indicators.

Similarly, advanced remote sensing research has applied satellite imagery to extract urban features and segment land usage types **(Warth et al. 2020)**. These studies focus primarily on physical or environmental characteristics, and while they do assist in land classification, they do not establish socio-economic correlations based on POI functionality.

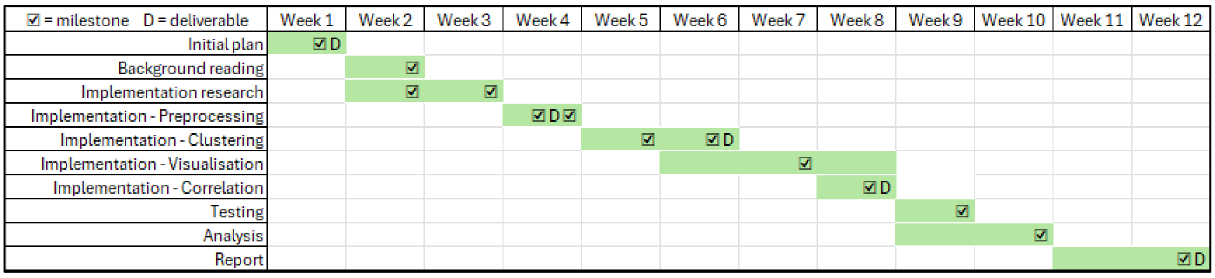
Despite a growing body of research into urban spatial analysis, there remains a gap in the literature concerning the direct correlation between POI-based functional clusters and socio-economic factors. Existing studies have either prioritised structural morphology, green space valuation, or macro-scale functional delineation, but none have proposed a lightweight, cluster-based system for identifying localised functional regions and directly comparing them with demographic data. This project addresses that gap by building a modular, interactive platform that visually overlays POI clusters onto socio-economic maps, allowing for accessible exploration of urban function in relation to social and economic conditions.

3. Design & Methodology

In this section, I will be discussing the methodology this project will use to ensure that milestones are met. It will also include the justification of the datasets that are analysed and used to create the data comparisons. Some of the Python libraries used to analyse and visualise the data will also be discussed so that their use is justified.

3.1 Methodology

I chose the waterfall methodology for this project as I had clear stages of development from the start. This methodology allows phases to cascade sequentially; only once a phase is completed can the next phase start **(Atlassian [no date])**. Although the waterfall method is rigid and does not incorporate the ability to revisit earlier phases like the agile method does, the obvious structure of this project means the development progress can be tracked more easily, especially with the help of a Gantt chart.



☑ = milestone D = deliverable	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Initial plan	🟩 ☑ D											
Background reading		🟩 ☑										
Implementation research		🟩 ☑	🟩 ☑									
Implementation - Preprocessing				🟩 ☑ D ☑								
Implementation - Clustering					🟩 ☑	🟩 ☑ D						
Implementation - Visualisation						🟩	🟩 ☑	🟩				
Implementation - Correlation								🟩 ☑ D				
Testing									🟩 ☑			
Analysis									🟩	🟩 ☑		
Report											🟩	🟩 ☑ D

Figure 1. Gantt chart visualising the progression of the project.

The Gantt chart shows a visual plan of the phases within this project. It clearly shows the week-by-week plan for the duration of the project, as well as the milestones and deliverables I set within the initial plan.

The milestones were written as a separate list to ensure that the project was progressing at a reasonable pace. For example, a milestone of making overlay heatmaps to find correlation between POI and socioeconomic data would not be marked complete until a visual map displaying the POI clusters was created. Although these plans and milestones are helpful for structuring a project, things often do not go as expected. Therefore, it is important to remain flexible and mindful of potential obstacles during development. Nevertheless, having a plan was very helpful for organising my work, as I needed enough time at the beginning to research new topics and develop knowledge of niche Python libraries to ensure my skills were adequate for the main development phase.

Since the very start of this project, I had planned to use the Python programming language. There were a variety of reasons for this choice; the main reason being that Python is the language I have the most experience in and the language I felt I could produce the highest quality code in. I also chose Python because of its reputation as a popular language for data processing and analysis. Python boasts a rich ecosystem of libraries and tools specifically designed for data analysis **(Karl 2024)**, making it perfect for building the system required in this project. The Dash Python library provided the core interactive framework for building the frontend web application for this project, this library uses the popular Flask web framework, which is another reason I decided on Python as the language to use for this project.

3.2 Choosing the datasets

For both the POI and socioeconomic data used to draw analysis, Digimap **(Digimap 1996)** was decided upon as the best source of data. Ordinance Survey data provides the most accurate and detailed POI data out of all the datasets I could have used. The 600+ official predetermined classifications met the level of detail necessary to make valid analysis of the POI clusters. Digimap also provided census data from 27th March 2011, from which a dataset on population age was downloaded. Data from the most recent census in 2021 was not accessible from Digimap and was significantly more difficult to find, hence the decision to continue with data from 2011. The twelve key socio-economic areas covered by the datasets were accommodation, car availability, no. of children, country of birth, economic status, household composition, long-term health, occupancy, population density, qualifications, social grade, and tenure.

The datasets from Digimap were downloaded in Comma-separated Value (CSV) file format due to my experience with the format and general ease of use. In the case of this project, using a program such as Excel made viewing CSV files very straightforward to work with. The system being built to process the data for this project only holds two CSV datasets at a time which relegated the need of a database system such as SQLite since no large datasets or complex relationships needed to be stored. The final justification of the decision to use CSV formatting was its common integration with Python; there are many popular libraries (e.g. pandas) that make CSV files simple to read from and write to.

The third dataset required for the system held the data for the areas used for the overlays of socioeconomic data. My plan was to use the Office for National Statistics' statistical geographies to divide the cities up into Super Output Areas **(Statistical geographies - Office for National Statistics 2021)**. Lower Super Output Areas (LSOAs) comprise of between 400 and 1,200 households, while Middle Super Output Areas (MSOAs) comprise of between 2,000 and 6,000 households. Originally, the plan was to use LSOAs because they provide more detailed information on the spread of socioeconomic data as well as aligning better with the size of the POI clusters, however, the sheer quantity of geographical coordinates used to create all the polygons for the LSOA shapes used too much memory on my PC. For the sake of time and specification constraints, MSOAs were chosen to be the geographical areas used for the socioeconomic overlays.

The MSOA dataset was downloaded in the GeoJSON format. GeoJSON is lightweight, human-readable, and well-supported by many mapping libraries, making it ideal for rendering polygonal boundaries like those defining MSOAs. Given the shift from LSOAs to MSOAs due to memory constraints, the GeoJSON format was a practical way to manage and visualise these larger, but fewer, geographical areas without sacrificing spatial accuracy or system performance.

3.3 Comparison of clustering algorithms

Clustering is a fundamental step in the analysis and visualisation of urban functional regions. The aim of this stage was to group POIs into spatial clusters based on geographic proximity, with the goal of identifying clusters that can be correlated to the socioeconomic data. To achieve this, I considered several clustering algorithms, including K-Means, Hierarchical Clustering, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

K-Means, while commonly used for spatial clustering, was quickly ruled out due to its inherent limitations. The algorithm requires the number of clusters to be defined in advance, which is impractical when dealing with real-world urban environments where the number of functional regions is unknown and potentially variable across different cities. K-Means is sensitive to outliers and assumes spherical clusters of roughly equal size which is not applicable in the case of irregularly distributed POI data. Hierarchical Clustering was also considered due to its flexibility and interpretability. However, the computational complexity of the algorithm becomes a significant drawback when applied to large spatial datasets. This lack of scalability made it unsuitable for the intended system, especially when integrated with a responsive web application.

I deemed DBSCAN as the most appropriate algorithm for clustering POIs in this project. It is a density-based clustering method, this means that it groups together points that are closely packed together based on a defined distance threshold and a minimum number of points within that distance. Unlike K-Means and Hierarchical Clustering, which are not density-based, DBSCAN does not require the number of clusters to be specified beforehand. It also handles clusters of arbitrary shapes and is resilient to noise, meaning it can effectively identify outlier POIs that do not belong to any functional cluster. These characteristics aligned well with the real-world distribution of POIs, where clusters are often irregular and interspersed with isolated points.

An important part of DBSCAN's effectiveness lies in the tuning of its parameters within the scikit-learn Python library: epsilon is the maximum distance between two samples for them to be considered as in the same neighbourhood, the default distance metric for this calculation was Euclidean distance, this metric is explored in section 4.2. The other important parameter, min_samples, is the number of samples in a neighbourhood for a point to be considered a core point. The use of DBSCAN was further justified by its seamless integration with the convex hull algorithm used to create polygonal representations of each cluster. It is a regularly used algorithm for clustering geographical data. This density-based approach complements the construction of spatial boundaries, which helped to create a visually logical way to represent functional regions on the interactive map interface.

The convex hull algorithm was chosen to generate simple polygonal boundaries around the points in each DBSCAN-identified cluster. While convex hulls provide a clear and computationally efficient method for enclosing a group of points, they can often overgeneralise the shape of a cluster by forming overly simplistic boundaries that ignore internal gaps or concavities. If I were to address this in the future, I could explore more advanced geometric techniques such as alpha shapes, which allow for the construction of concave boundaries and a more accurate depiction of a cluster's true shape. This aligns with findings highlighting the value of spatial generalisation methods like alpha shapes in capturing the real-world complexity of geographic phenomena **(Duckham et al. 2008)**. Their work demonstrates how alternative cluster formations can yield more accurate representations of irregular spatial distributions, making them particularly relevant to urban analysis where built environments rarely conform to convex geometries. Integrating such techniques could significantly improve the interpretability and accuracy of cluster boundaries in the context of POI-based functional region detection.

3.4 Visualisation and Interactivity

Once a clustering algorithm was chosen, the POI data could be represented as clusters of classified functions. To represent this visually, I opted to use Plotly, a Python library with the purpose of creating interactive data visualisations such as graphs and maps. Plotly's well documented functions provided a wide range of map types that meant multiple data formats could be visualised using a single map and traces.

The two map types chosen to display the data were scatter maps for the POI data, and choropleth maps for the socioeconomic heatmaps. Scatter maps were ideal for visualising the precise geospatial location of individual POIs, as they allow for each point to be plotted based on its latitude and longitude, with the option to colour-code by classification group. This allowed an immediate visual understanding of how different categories of urban functions are distributed across a cityscape, which paired with the use of lines drawn through the scatter map function, created clear visual representation of the clusters. In contrast, Choropleth maps are designed to represent aggregated data over predefined geographic boundaries, making them well suited for displaying socioeconomic indicators across MSOAs. The colour gradients applied in choropleth visualisations show the relative intensity of variables such as age distribution across the city. When I paired these two map types together, they provided complementary views of the urban environment: one showing granular, categorical distributions of functions, and the other highlighting broader demographic patterns across MSOAs.

To ensure that my plan was followed, I decided to create a web application so that further interactivity and visualisation could be achieved. For this, I used Dash, a Python framework specifically designed for building analytical web applications. Dash allowed me to flawlessly integrate the Plotly maps into an interactive interface, where users could toggle visibility, filter data by category, select different layers of socioeconomic information, and change some of the DBSCAN parameters. Behind the scenes, Dash is built on top of the Flask web framework, which provided the underlying server infrastructure and routing capabilities. This combination allowed me to develop a lightweight, responsive application that could render complex visualisations in a browser environment without the need for excessive front-end development.

4. Implementation

4.1 Handling the data

Handling the POI data throughout the system pipeline was the first critical stage of my implementation. This consisted of developing functions to clean the data that is inputted, cluster the data, create polygonal shapes for the clusters, and display the map. This section will focus on the preprocessing part of the data pipeline, the visualisation of the data will be covered in section 4.3 of this report.

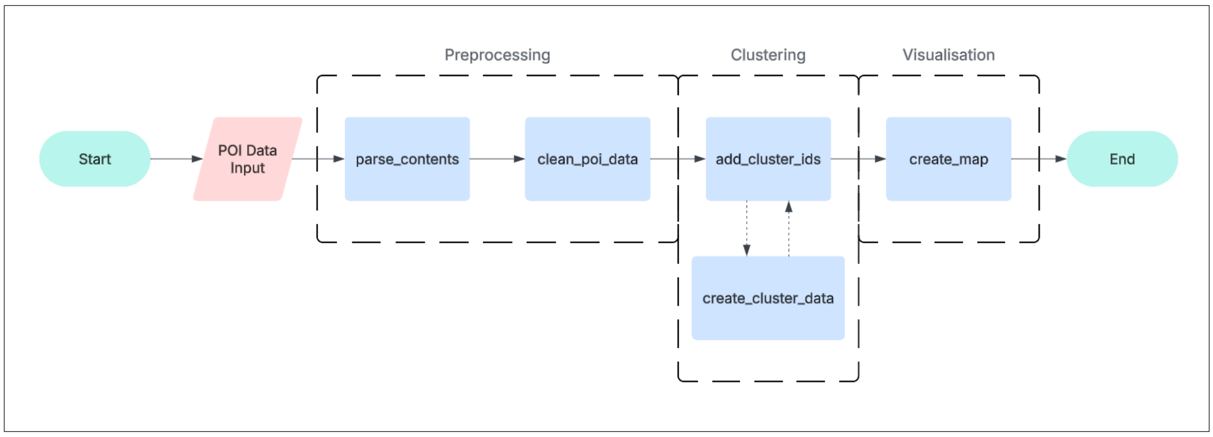
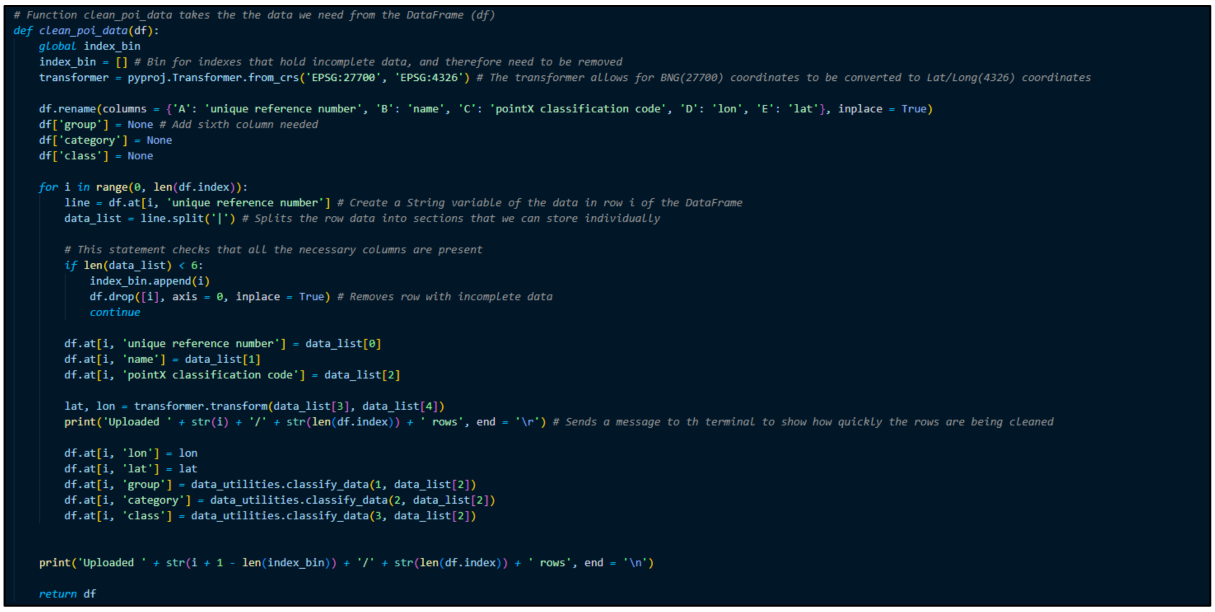


Figure 2. Simple flowchart showing the POI data pipeline

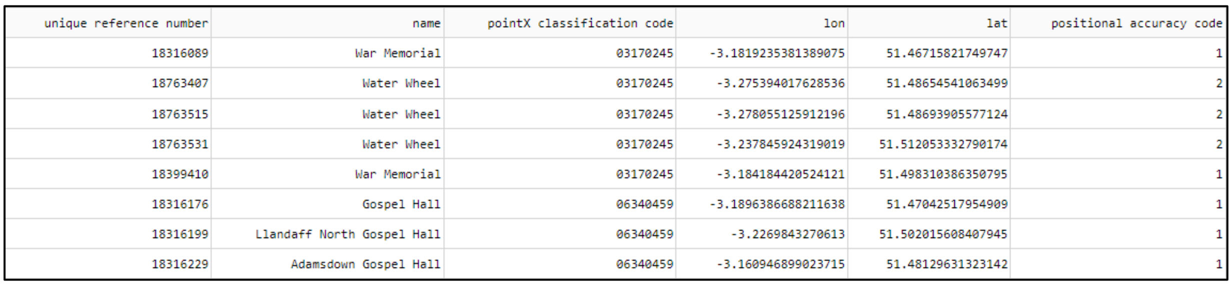
The initial POI datasets that were downloaded included numerous data columns that were not necessary for the system, so the first step was to remove these columns and structure the remaining data. The Python library, named pandas, being used to handle the datasets has a data structure called DataFrame. This structure stores data in a table what can be manipulated by a range of functions that made cleaning the dataset straightforward. I made a function ("clean_poi_data" in Figure 3) to handle all the initial DataFrame manipulation. After the dataset was first downloaded, it was apparent that the records of POIs were being held within one single column, this became an issue that, while easily solvable, meant incomplete records could not be identified straight away. Once the records where split into columns, the "len(data_list)" function could then check to see if the record was complete and drop the row if found otherwise.



```python
# Function clean_poi_data takes the the data we need from the DataFrame (df)
def clean_poi_data(df):
    global index_bin
    index_bin = [] # Bin for indexes that hold incomplete data, and therefore need to be removed
    transformer = pyproj.Transformer.from_crs('EPSG:27700', 'EPSG:4326') # The transformer allows for BNG(27700) coordinates to be converted to Lat/Long(4326) coordinates

    df.rename(columns = {'A': 'unique reference number', 'B': 'name', 'C': 'pointX classification code', 'D': 'lon', 'E': 'lat'}, inplace = True)
    df['group'] = None # Add sixth column needed
    df['category'] = None
    df['class'] = None

    for i in range(0, len(df.index)):
        line = df.at[i, 'unique reference number'] # Create a String variable of the data in row i of the DataFrame
        data_list = line.split('|') # Splits the row data into sections that we can store individually

        # This statement checks that all the necessary columns are present
        if len(data_list) < 6:
            index_bin.append(i)
            df.drop([i], axis = 0, inplace = True) # Removes row with incomplete data
            continue

        df.at[i, 'unique reference number'] = data_list[0]
        df.at[i, 'name'] = data_list[1]
        df.at[i, 'pointX classification code'] = data_list[2]

        lat, lon = transformer.transform(data_list[3], data_list[4])
        print('Uploaded ' + str(i) + '/' + str(len(df.index)) + ' rows', end = '\r') # Sends a message to th terminal to show how quickly the rows are being cleaned

        df.at[i, 'lon'] = lon
        df.at[i, 'lat'] = lat
        df.at[i, 'group'] = data_utilities.classify_data(1, data_list[2])
        df.at[i, 'category'] = data_utilities.classify_data(2, data_list[2])
        df.at[i, 'class'] = data_utilities.classify_data(3, data_list[2])


    print('Uploaded ' + str(i + 1 - len(index_bin)) + '/' + str(len(df.index)) + ' rows', end = '\n')

    return df
```

Figure 3. "clean_poi_data" used to clean the initial DataFrame of POI records

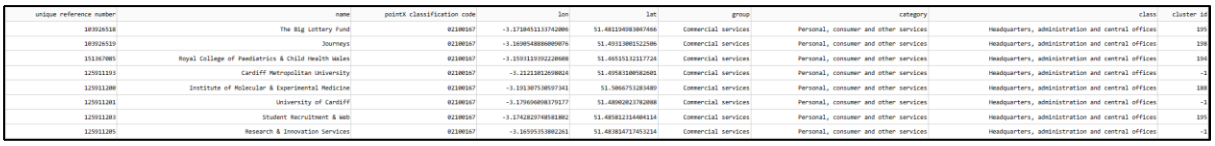
The six columns that remained after the “clean_poi_data” function were the attributes deemed necessary to keep. The reasoning behind retaining the unique reference number attribute was purely for testing purposes so that it was clear when observing specific POIs for the correct coordinate placement and other observational tests.



unique reference number	name	pointX classification code	lon	lat	positional accuracy code
18316089	War Memorial	03170245	-3.1819235381389075	51.46715821749747	1
18763407	Water Wheel	03170245	-3.275394017628536	51.48654541063499	2
18763515	Water Wheel	03170245	-3.278055125912196	51.48693905577124	2
18763531	Water Wheel	03170245	-3.237845924319019	51.512053332790174	2
18399410	War Memorial	03170245	-3.184184420524121	51.498310386350795	1
18316176	Gospel Hall	06340459	-3.1896386688211638	51.47042517954909	1
18316199	Llandaff North Gospel Hall	06340459	-3.2269843270613	51.502015608407945	1
18316229	Adamsdown Gospel Hall	06340459	-3.160946899023715	51.48129631323142	1

Table 1. Table showing a sample of the cleaned POI DataFrame before adjustments to the function

As seen in Figure 3, the function no longer keeps the positional accuracy code attribute. Given how accurate even a code 3 positional accuracy code is, the positional accuracy of the POIs is a minor issue when taken in the context of larger urban clusters. However, if it was seen as a crucial problem, any POIs with low positional accuracy, for example >= 2, could be filtered out and dropped in the same way the incomplete records are. The other three columns seen in Figure 3 were introduced later in the development of the system. These additions were solely for the purpose of displaying the classification information in the hover data of the POIs. The final POI DataFrame (Table 2) also has the additional cluster id attribute.



unique reference number	name	pointX classification code	lon	lat	group	category	class	cluster id
103926518	The Big Lottery Fund	02100167	-3.1710451133742006	51.481194983047466	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	195
103926519	Journeys	02100167	-3.1690548886009076	51.49313001522506	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	198
151367005	Royal College of Paediatrics & Child Health Wales	02100167	-3.1593119392220608	51.46515132117724	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	194
125911193	Cardiff Metropolitan University	02100167	-3.21211012698024	51.49583100582601	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	-1
125911200	Institute of Molecular & Experimental Medicine	02100167	-3.191307530597341	51.5066753283489	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	188
125911201	University of Cardiff	02100167	-3.179696098379177	51.48902023782088	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	-1
125911203	Student Recruitment & Web	02100167	-3.1742829748581802	51.485812314404114	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	195
125911205	Research & Innovation Services	02100167	-3.16595353802261	51.483814717453214	Commercial services	Personal, consumer and other services	Headquarters, administration and central offices	-1

Table 2. Sample from final POI DataFrame

To process the socioeconomic data, the “parse_contents” function was used again to load the dataset into a DataFrame. This dataset also had unused columns, so a similar function was used to minimise the data that is being stored. However, unlike the POI dataset, the downloaded socioeconomic dataset was already separated into columns meaning the only column that needed manipulation was the centroid coordinate column which was then split into latitude and longitude columns.

4.2 Clustering and spatial calculations

When approaching the second stage of the implementation, the distance measurements for the POIs had to be decided on. Within spatial data calculations on a two-dimensional plane Euclidean distance is universally accepted. Euclidean distance is simply the straight-line distance between two points. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), the Euclidean distance is given by:

$$d = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

(National Institute of Standards and Technology 2004)

However, since the POIs used the geographic coordinates system, the surface in which the points lie is curved. The haversine formula is a very accurate way of computing distances between two points on the surface of a sphere using the latitude and longitude of the two points. The haversine formula is a re-formulation of the spherical law of cosines, but the formulation in terms of haversines is more useful for small angles and distances (Kettle 2017).

The following equation where ϕ is latitude, λ is longitude, R is earth’s radius (mean radius = 6,371km) is how haversine(θ) = $\sin^2(\theta/2)$ can be translated to include latitude and longitude coordinates. Note that angles need to be in radians to pass to trig functions:

$$\begin{aligned}
a &= \sin^2(\phi B - \phi A/2) + \cos \phi A * \cos \phi B * \sin^2(\lambda B - \lambda A/2)\\
c &= 2 * \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})\\
d &= R \cdot c
\end{aligned}$$

(Kettle 2017)

Both distance metrics are widely used within spatial data analysis, meaning premade functions for measuring distance with both were readily available in Python libraries such as scikit-learn. However, the scikit-learn DBSCAN algorithm does not have an automatic distance calculator for haversine distance and requires for distances to be precalculated if using the haversine metric. When deciding on which metric to use, the distortion of using Euclidean distance on a spherical plane was considered. On a city scale, the curvature of the plane is ~ 0.09°, and over 100 meters, that curvature would be ~0.0008°. This miniscule angle would likely have insignificant effect on any distance measured with the Euclidean metric. Ultimately, I decided on using Euclidean distance for the sake of efficiency as I would not have to precompute any distance values before clustering.

Having chosen the DBSCAN clustering algorithm, further research was done on the mathematics and parameters of DBSCAN and the specifics of scikit-learn's function. The algorithm finds core samples of high density and expands clusters from them ("ExpandCluster" in Figure 5). It is good for data which contains clusters of similar density. It works by categorising data points into three types; core points - which have enough neighbours within a specified radius called epsilon, border points - which are near core points but lack enough neighbours to be core points themselves, and noise points - which do not belong to any cluster.

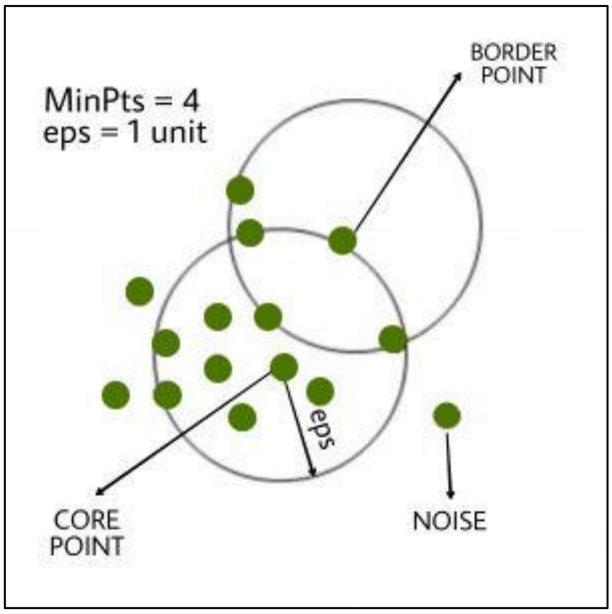
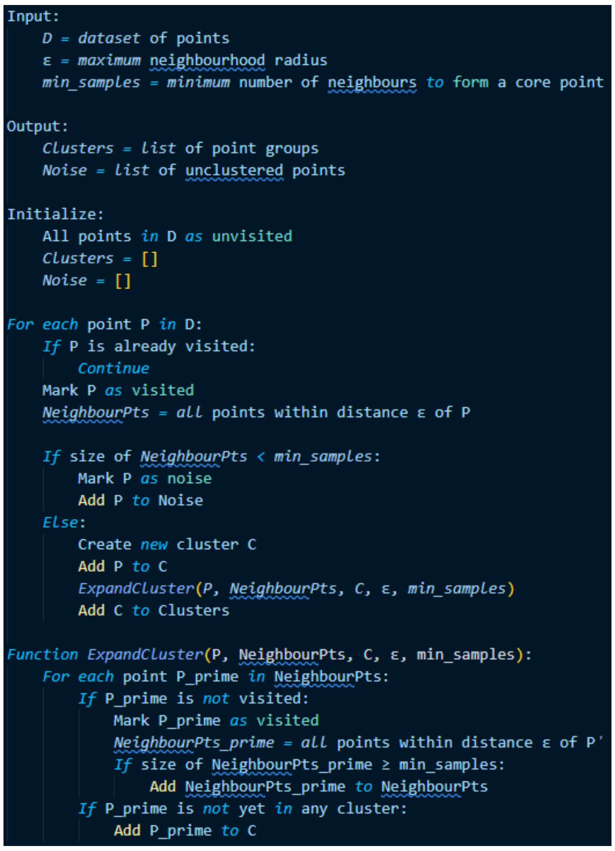


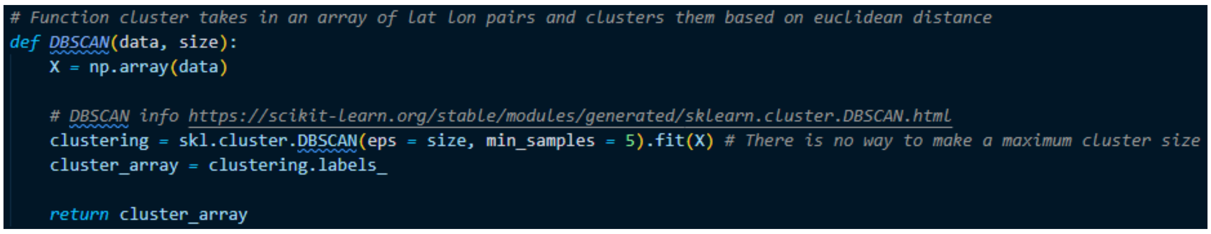
Figure 4. Diagram showing the three points in DBSCAN clusters



```
Input:
    D = dataset of points
    ε = maximum neighbourhood radius
    min_samples = minimum number of neighbours to form a core point

Output:
    Clusters = list of point groups
    Noise = list of unclustered points

Initialize:
    All points in D as unvisited
    Clusters = []
    Noise = []

For each point P in D:
    If P is already visited:
        Continue
    Mark P as visited
    NeighbourPts = all points within distance ε of P

    If size of NeighbourPts < min_samples:
        Mark P as noise
        Add P to Noise
    Else:
        Create new cluster C
        Add P to C
        ExpandCluster(P, NeighbourPts, C, ε, min_samples)
        Add C to Clusters

Function ExpandCluster(P, NeighbourPts, C, ε, min_samples):
    For each point P_prime in NeighbourPts:
        If P_prime is not visited:
            Mark P_prime as visited
            NeighbourPts_prime = all points within distance ε of P'
            If size of NeighbourPts_prime ≥ min_samples:
                Add NeighbourPts_prime to NeighbourPts
        If P_prime is not yet in any cluster:
            Add P_prime to C
```

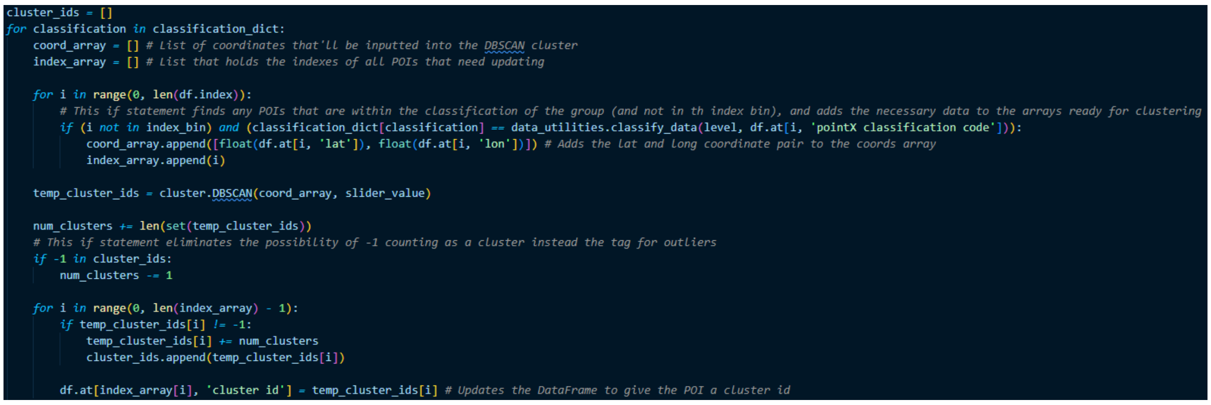
Figure 5. Pseudocode for DBSCAN algorithm

In the scikit-learn implementation **(Appendix 3)** there are multiple parameters that can impact the size and density of clusters, as well as the metric and algorithm used when computing the clusters. In the system I implemented, only the two parameters mentioned in 3.4 Visualisation and Interactivity were considered; the epsilon(eps) parameter was crucial in preventing the clusters from being too large. An eps value of 0.0005 limits a border point to being ~550 meters away from a core point, this meant that a variable eps value between 0.0005 – 0.005 gave accurate and usable cluster sizes on a city scale. The other parameter, min_samples, had to be adjusted according to what was assumed to be enough POIs to indicate a functional region. A min_samples value of 10 was chosen as it stops small areas, for example a row of shops, from being clustered. However, as discussed in sections 5 Analysis & 6 Conclusion & Future Work, small areas like these could also be clustered for comparison to the socio-economy on a lower level. This implementation has a worst-case memory complexity of O(n^2), where n is the number of data points. This can occur when the eps parameter is large and min_samples is low, while the original DBSCAN only uses O(n) memory **(Scikit-learn [no date])**.



```python
# Function cluster takes in an array of lat lon pairs and clusters them based on euclidean distance
def DBSCAN(data, size):
    X = np.array(data)

    # DBSCAN info https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
    clustering = skl.cluster.DBSCAN(eps = size, min_samples = 5).fit(X) # There is no way to make a maximum cluster size
    cluster_array = clustering.labels_

    return cluster_array
```

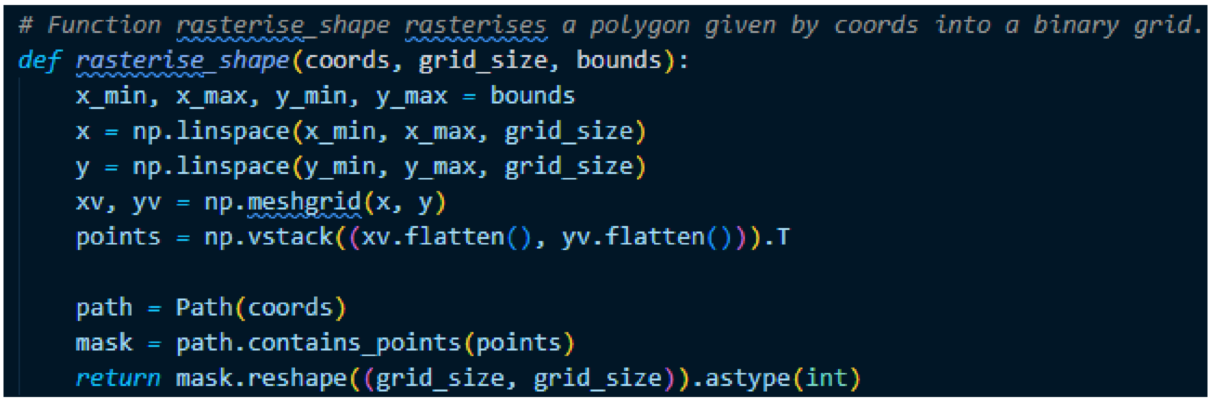
Figure 6. "DBSCAN" function with variable epsilon value



```python
cluster_ids = []
for classification in classification_dict:
    coord_array = [] # List of coordinates that'll be inputted into the DBSCAN cluster
    index_array = [] # List that holds the indexes of all POIs that need updating

    for i in range(0, len(df.index)):
        # This if statement finds any POIs that are within the classification of the group (and not in th index bin), and adds the necessary data to the arrays ready for clustering
        if (i not in index_bin) and (classification_dict[classification] == data_utilities.classify_data(level, df.at[i, 'pointX classification code'])):
            coord_array.append([float(df.at[i, 'lat']), float(df.at[i, 'lon'])]) # Adds the lat and long coordinate pair to the coords array
            index_array.append(i)

    temp_cluster_ids = cluster.DBSCAN(coord_array, slider_value)

    num_clusters += len(set(temp_cluster_ids))
    # This if statement eliminates the possibility of -1 counting as a cluster instead the tag for outliers
    if -1 in cluster_ids:
        num_clusters -= 1

    for i in range(0, len(index_array) - 1):
        if temp_cluster_ids[i] != -1:
            temp_cluster_ids[i] += num_clusters
            cluster_ids.append(temp_cluster_ids[i])

        df.at[index_array[i], 'cluster id'] = temp_cluster_ids[i] # Updates the DataFrame to give the POI a cluster id
```

Figure 7. POIs are clustered and assigned a cluster id

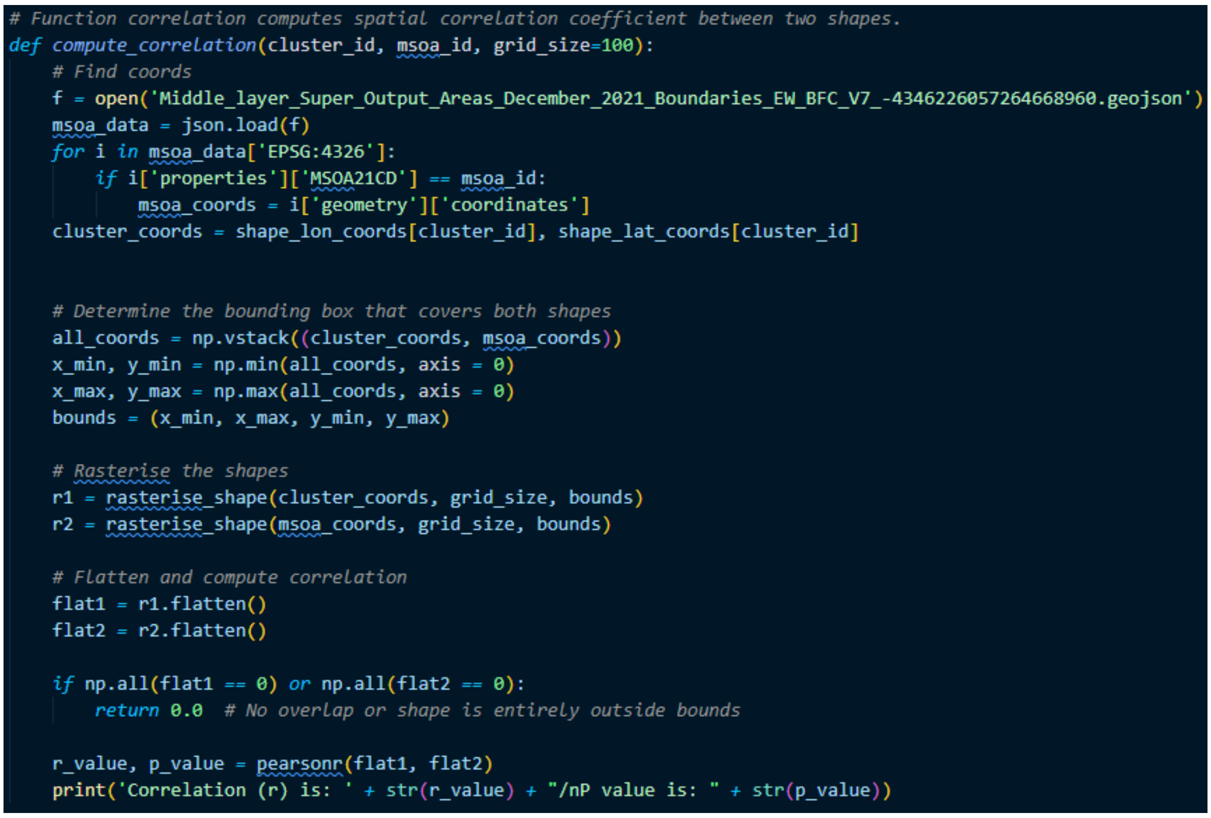
The final area of spatial calculation for the system was calculating the correlation between MSOAs and clusters. To achieve this, an algorithm was implemented to take in a cluster and a MSOA area and compute a spatial correlation coefficient to represent how much overlap there is and therefore whether there's a like between the two. To be able to compute a correlation, both areas had to first be rasterised. Rasterisation is the process of converting a vector image which is made up of polygons into a raster image which is made up of pixels or dots. A function (Figure 8) was made to rasterise the polygons using numpy and matplotlib functionality.



```python
# Function rasterise_shape rasterises a polygon given by coords into a binary grid.
def rasterise_shape(coords, grid_size, bounds):
    x_min, x_max, y_min, y_max = bounds
    x = np.linspace(x_min, x_max, grid_size)
    y = np.linspace(y_min, y_max, grid_size)
    xv, yv = np.meshgrid(x, y)
    points = np.vstack((xv.flatten(), yv.flatten())).T

    path = Path(coords)
    mask = path.contains_points(points)
    return mask.reshape((grid_size, grid_size)).astype(int)
```

Figure 8. Function "rasterise_shape" used to prepare shapes for computation

After rasterisation, the correlation is computed between the flattened 1D versions of the masks. The computation used Pearson correlation coefficient, which has a built-in scipy function **(Appendix 4)**. The Pearson correlation coefficient (r) is the most common way of measuring a linear correlation. It is a number between -1 & 1 that measures the strength and direction of the relationship between two variables **(Turney 2024)**. For this project, r values from 0 to -1 indicate that the two shapes do not overlap. The scope of this project does not cover the inverse correlation, only the extent of overlap between the two shapes. A p value accompanies every computation of r to show the statistical significance of that value. This is important as the project pivots on proving the use of POI clusters to reveal socio-economic correlation.

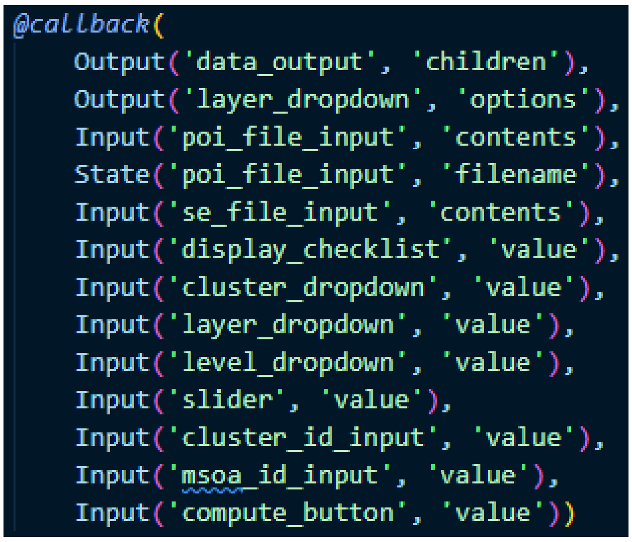


```python
# Function correlation computes spatial correlation coefficient between two shapes.
def compute_correlation(cluster_id, msoa_id, grid_size=100):
    # Find coords
    f = open('Middle_layer_Super_Output_Areas_December_2021_Boundaries_EW_BFC_V7_-4346226057264668960.geojson')
    msoa_data = json.load(f)
    for i in msoa_data['EPSG:4326']:
        if i['properties']['MSOA21CD'] == msoa_id:
            msoa_coords = i['geometry']['coordinates']
    cluster_coords = shape_lon_coords[cluster_id], shape_lat_coords[cluster_id]


    # Determine the bounding box that covers both shapes
    all_coords = np.vstack((cluster_coords, msoa_coords))
    x_min, y_min = np.min(all_coords, axis = 0)
    x_max, y_max = np.max(all_coords, axis = 0)
    bounds = (x_min, x_max, y_min, y_max)

    # Rasterise the shapes
    r1 = rasterise_shape(cluster_coords, grid_size, bounds)
    r2 = rasterise_shape(msoa_coords, grid_size, bounds)

    # Flatten and compute correlation
    flat1 = r1.flatten()
    flat2 = r2.flatten()

    if np.all(flat1 == 0) or np.all(flat2 == 0):
        return 0.0  # No overlap or shape is entirely outside bounds

    r_value, p_value = pearsonr(flat1, flat2)
    print('Correlation (r) is: ' + str(r_value) + "/nP value is: " + str(p_value))
```

Figure 9. Function "compute_correlation" prints the r value for the cluster & MSOA correlation

4.3 Creating an app for data visualisation

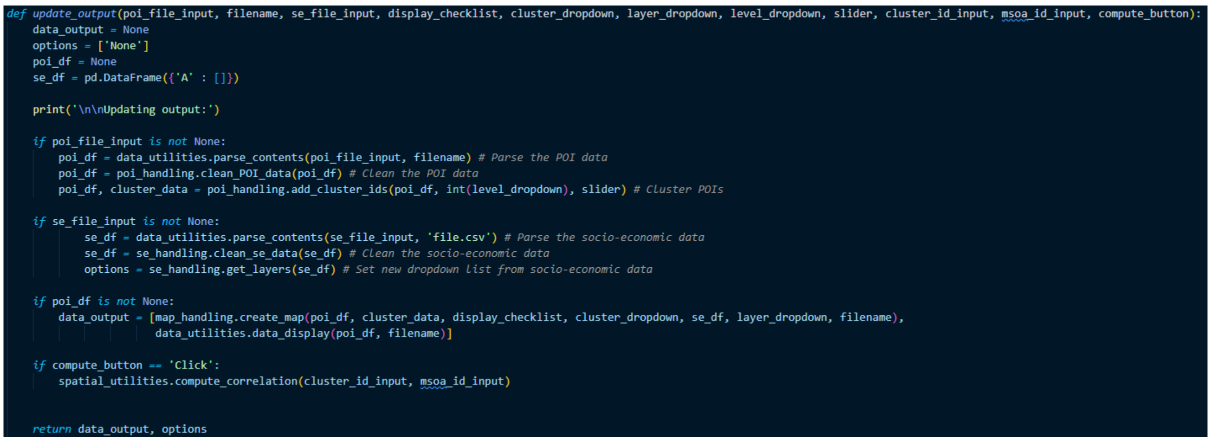
The final stage of implementation was to create a web application so that the clusters and socio-economic data could be visualised and interacted with. To do this, I started by building a Dash framework **(Appendix 5)** which uses embedded html to build the webpage. The Dash framework relies on callbacks to handle how the application reacts to buttons and other interactions. The input and outputs of the application are simply the properties of a component within the layout. When an input changes, the function that the callback decorator wraps will get called automatically. As the system continued to be built, I ran into an issue where the callback function would be re-run every time an interaction. This is usually fixed by sharing data between multiple callbacks, however, a lot of the inputs in the application require calculating and processing the clusters again which makes it difficult to make a multi-callback application.



```python
@callback(
    Output('data_output', 'children'),
    Output('layer_dropdown', 'options'),
    Input('poi_file_input', 'contents'),
    State('poi_file_input', 'filename'),
    Input('se_file_input', 'contents'),
    Input('display_checklist', 'value'),
    Input('cluster_dropdown', 'value'),
    Input('layer_dropdown', 'value'),
    Input('level_dropdown', 'value'),
    Input('slider', 'value'),
    Input('cluster_id_input', 'value'),
    Input('msoa_id_input', 'value'),
    Input('compute_button', 'value'))
```

Figure 10. Callback variables for the Dash application

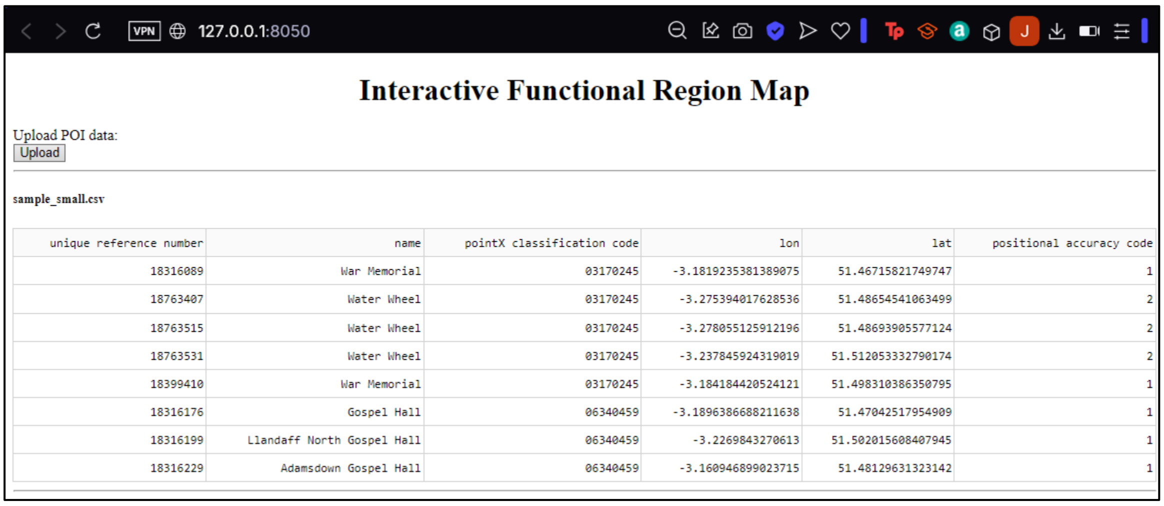
The application has two outputs, ten inputs, and a state for one of the inputs. One output controls the list of socio-economic layers that can be selected from the dropdown, these change depending on the layer columns within the uploaded socio-economic dataset. The other output is the data output which displays the map and data table. As seen in Figure 10, the callback inputs are all components of the application layout. The callback function "update_output" (Figure 11) takes the exact number of inputs as parameters plus optional states, iterates through the data pipeline if there is an uploaded dataset, and returns the exact number of outputs within the callback.



```python
def update_output(poi_file_input, filename, se_file_input, display_checklist, cluster_dropdown, layer_dropdown, level_dropdown, slider, cluster_id_input, msoa_id_input, compute_button):
    data_output = None
    options = ['None']
    poi_df = None
    se_df = pd.DataFrame({'A' : []})

    print('\n\nUpdating output:')

    if poi_file_input is not None:
        poi_df = data_utilities.parse_contents(poi_file_input, filename) # Parse the POI data
        poi_df = poi_handling.clean_POI_data(poi_df) # Clean the POI data
        poi_df, cluster_data = poi_handling.add_cluster_ids(poi_df, int(level_dropdown), slider) # Cluster POIs

    if se_file_input is not None:
            se_df = data_utilities.parse_contents(se_file_input, 'file.csv') # Parse the socio-economic data
            se_df = se_handling.clean_se_data(se_df) # Clean the socio-economic data
            options = se_handling.get_layers(se_df) # Set new dropdown list from socio-economic data

    if poi_df is not None:
        data_output = [map_handling.create_map(poi_df, cluster_data, display_checklist, cluster_dropdown, se_df, layer_dropdown, filename),
                       data_utilities.data_display(poi_df, filename)]

    if compute_button == 'Click':
        spatial_utilities.compute_correlation(cluster_id_input, msoa_id_input)


    return data_output, options
```

Figure 11. "update_output" controls output to the Dash layout

Once the callbacks had been established, I focused on the layout and components needed for the application. Uploading the POI dataset was necessary before any data processing or visualisation could be done so I started with implementing a Dash Upload component. To test whether the data had been uploaded, the application would display the uploaded and cleaned dataset as a table in the application (Figure 12).



Interactive Functional Region Map

Upload POI data:

sample_small.csv

unique reference number	name	pointX classification code	lon	lat	positional accuracy code
18316089	War Memorial	03170245	-3.1819235381389075	51.46715821749747	1
18763407	Water Wheel	03170245	-3.275394017628536	51.48654541063499	2
18763515	Water Wheel	03170245	-3.278055125912196	51.48693905577124	2
18763531	Water Wheel	03170245	-3.237845924319019	51.512053332790174	2
18399410	War Memorial	03170245	-3.184184420524121	51.498310386350795	1
18316176	Gospel Hall	06340459	-3.1896386688211638	51.47042517954909	1
18316199	Llandaff North Gospel Hall	06340459	-3.2269843270613	51.502015608407945	1
18316229	Adamsdown Gospel Hall	06340459	-3.160946899023715	51.48129631323142	1

Figure 12. Dash application showing upload button and data table

After confirming that a POI dataset could be uploaded, the focus shifted to creating a scatter map to display the POIs. The Plotly "scatter_map" (Figure 13) function takes in a dataset that is used to find the coordinates and other attributes of every record. "hover_data" is a list of information displayed in a small box about a POI when the mouse is hovering over it. The "color" variable is defined by the group in which the POI is classified, each colour was chosen from the standard Python predefined colour chart. The scatter map style was set to the OpenStreetMap, this was one of a few options for showing areas outside the United States of America and suited the level of detail this application required to make sense of the POIs when displayed on a map.

```python
fig = px.scatter_map(poi_data,
                     lat = 'lat',
                     lon = 'lon',
                     color = 'group',
                     hover_name = 'name',
                     hover_data = {'lat': False, 'lon': False, 'group': True, 'category': True, 'class': True},
                     color_discrete_map = color_map,
                     zoom = 13,
                     height = 600)
fig.update_layout(map_style = 'open-street-map') # The map figure uses the open street map base style
fig.update_layout(margin = {'r':0,'t':0,'l':0,'b':0})
```

Figure 13. Scatter map creation

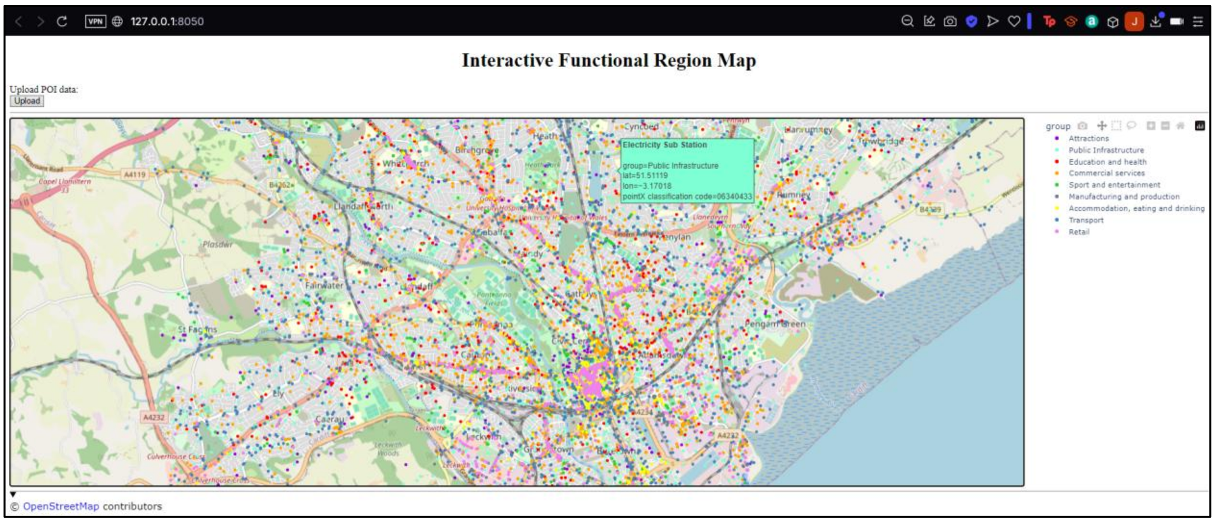
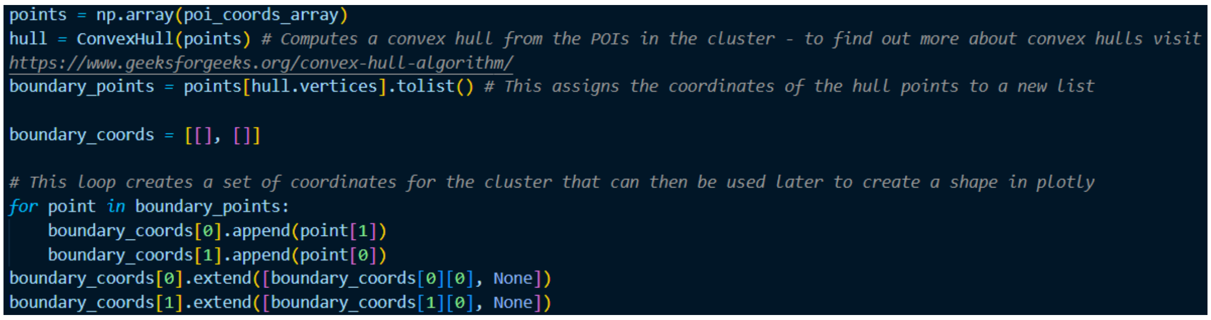


Figure 14. Dash application showing the map with POIs

Next, the clusters were added to the map along with some components for changing the cluster visibility and eps value. When the clusters were first displayed on the map, they would appear as rectangles based off the furthest border point east, west, north, and south in each cluster. After debugging the system, the issue was found to be with the convex hull algorithm being used to create the shape of the clusters; The boundary points were not being appended correctly to the lists of latitude and longitude coordinates. Once this issue was fixed the cluster shapes were being displayed properly (Figure 16).



```
points = np.array(poi_coords_array)
hull = ConvexHull(points) # Computes a convex hull from the POIs in the cluster - to find out more about convex hulls visit
https://www.geeksforgeeks.org/convex-hull-algorithm/
boundary_points = points[hull.vertices].tolist() # This assigns the coordinates of the hull points to a new list

boundary_coords = [[], []]

# This loop creates a set of coordinates for the cluster that can then be used later to create a shape in plotly
for point in boundary_points:
    boundary_coords[0].append(point[1])
    boundary_coords[1].append(point[0])
boundary_coords[0].extend([boundary_coords[0][0], None])
boundary_coords[1].extend([boundary_coords[1][0], None])
```

Figure 15. Convex hulls being used for cluster shape generation

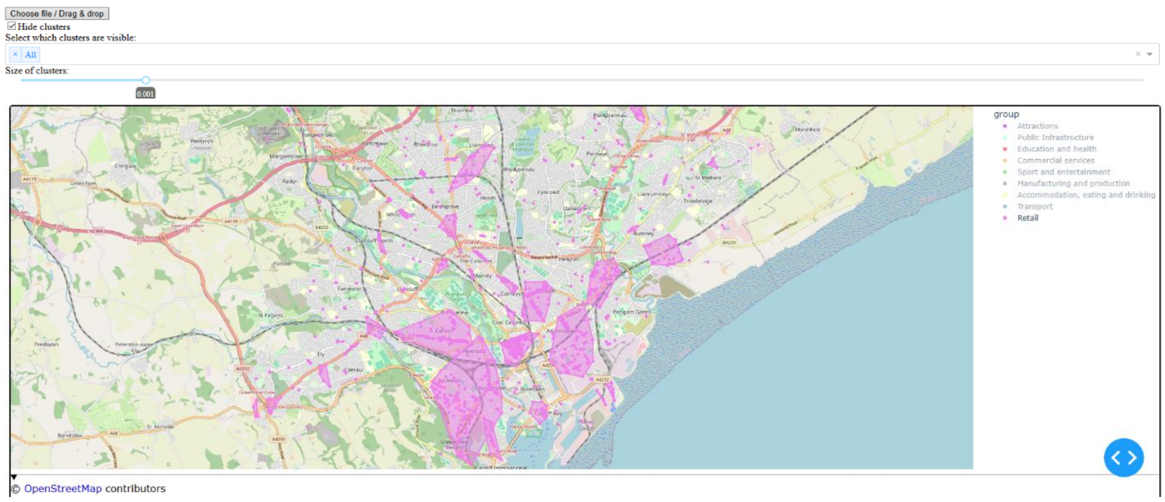


Figure 16. Dash application showing retail clusters on the map

As the number of components within the application grew, a day was spent applying CSS styling so that the application was more user friendly. The importance of styling was in the clarity of the button and other variable components, to make sure this was achieved, adequate spacing was added between the components and the font sizes was increased. Giving each component a border also helped with visual clarity as each interactive feature was contained in its own box.

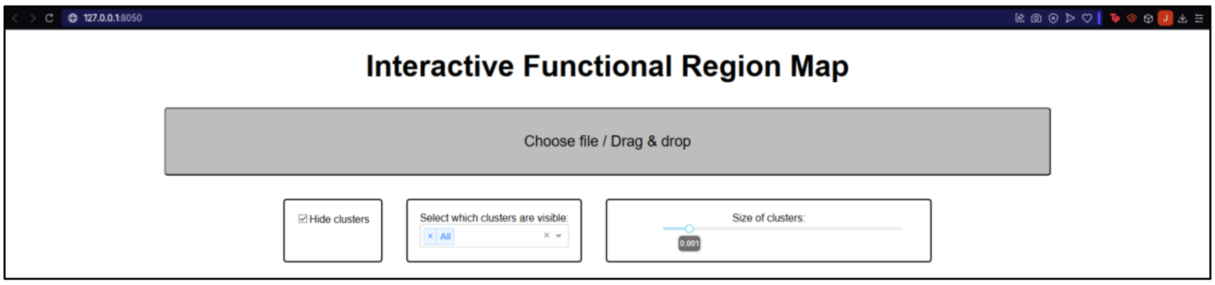

127.0.0.1:8050
Interactive Functional Region Map
Choose file / Drag & drop
Hide clusters
Select which clusters are visible:
All
Size of clusters:
0.001

Figure 17. Dash application with CSS styling

The penultimate part of implementing the Dash application was displaying the socio-economic data as an overlay. Before any socio-economic data could be processed or displayed, the geoJSON MSOA objects had to be downloaded. Using the example of Cardiff, 54 MSOA objects were selected to represent the area covered by the POIs. Some of these MSOAs were only partially within the urban area of Cardiff therefore are not covered by POIs. As the choropleth function takes a .json file without any need for preprocessing, there was no need to clean the MSOA data.

Once the data used to make the MSOA shapes, another Upload component was added so that socio-economic datasets could be uploaded to the system. Using the data handling functions made for the POI data, the socio-economic was cleaned and a DataFrame was created to store it. To test whether the socio-economic data was being uploaded, processed, and displayed correctly, the number fully employed individuals was chosen to be a predetermined layer. The choropleth map was then created as a trace of the POI scatter map (Figure 18).

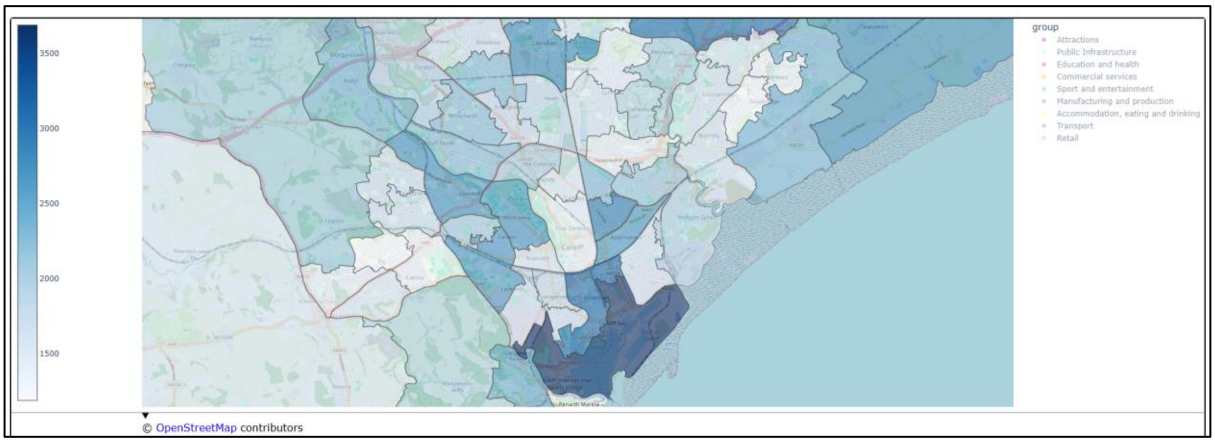


Figure 18. Dash application displaying the employee_fulltime overlay

After confirming that the socio-economic overlays work, I finished the application by creating a dropdown component so that a socio-economic layer can be selected from the layers within the dataset that’s uploaded to the system. I ran into an issue with callbacks where the input value for the dropdown could not be attained as there was not a placeholder value when the application was first initiated. This meant the “update_output” callback function was looking for an input value when it was not there. To handle this issue, a function was made to return the layer columns within the socio-economic DataFrame as a list of values to be used in the dropdown. The dropdown options were also reset every time the application is updated so that there will always be at least a placeholder value for the callback input to use.

The final part of the implementation was to allow the ids of a cluster and a MSOA to be inputted for correlation coefficient computation. To do this, I made two inputs and a button (Figure 19) so that the computation of the values only happens once the button is clicked.

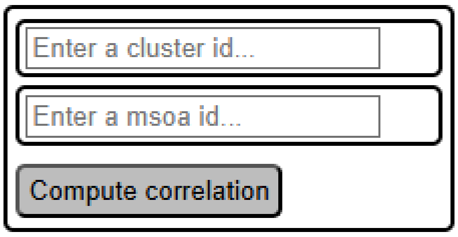
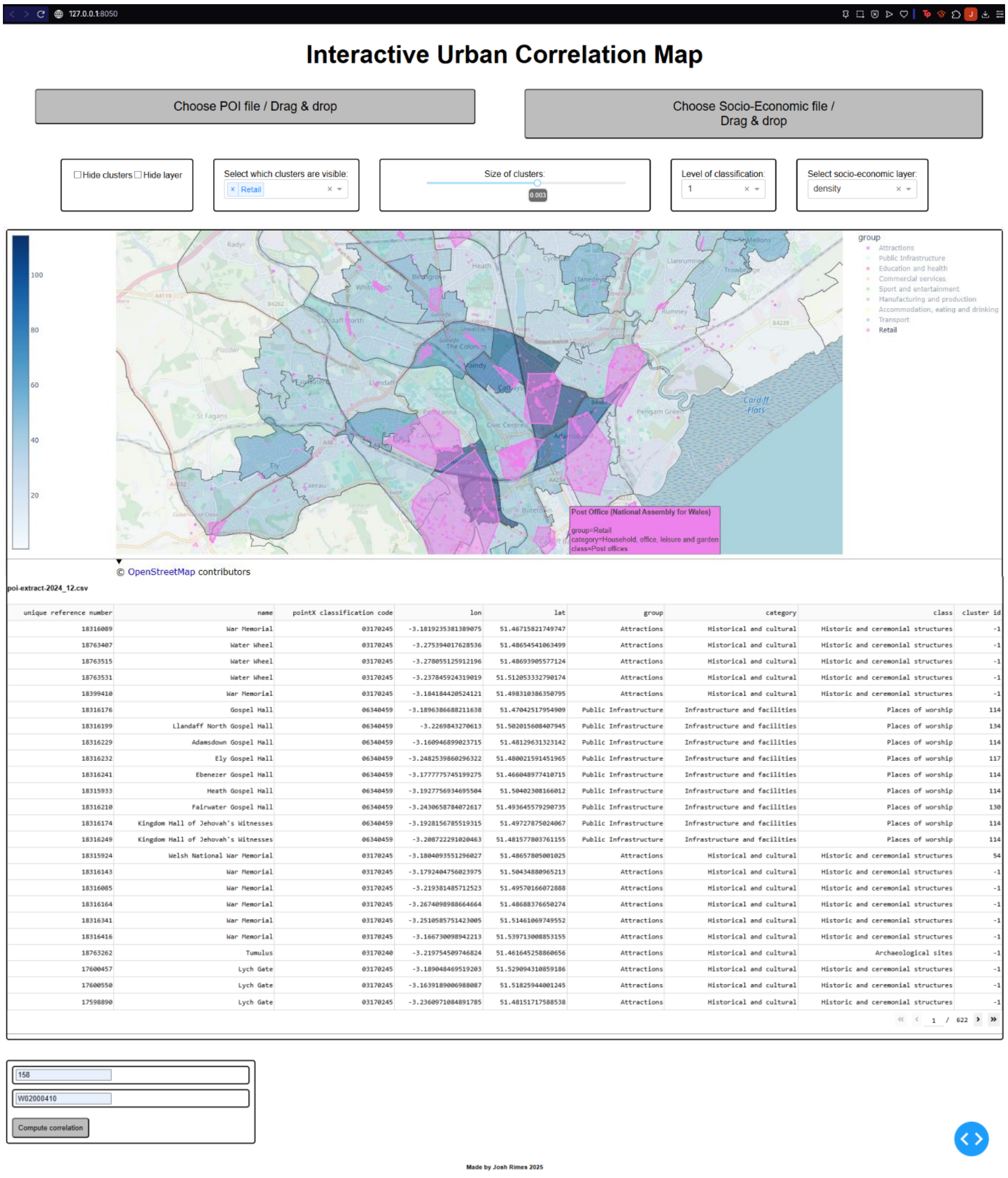


Figure 19. Div for inputting ids to be computed

Once these components had been added, the application was complete. I decided to change the title of the application to better fit the project. Figure 20 shows the entire web application set to show retail clusters over the population density socio-economic variable as an example.



Interactive Urban Correlation Map

Choose POI file / Drag & drop

Choose Socio-Economic file / Drag & drop

☐ Hide clusters ☐ Hide layer

Select which clusters are visible: × Retail

Size of clusters: 0.003

Level of classification: 1

Select socio-economic layer: density

Post Office (National Assembly for Wales)

group=Retail
category=Household, office, leisure and garden
class=Post offices

© OpenStreetMap contributors

poi-extract-2024_12.csv

unique reference number	name	pointX classification code	lon	lat	group	category	class	cluster id
18316089	War Memorial	03170245	-3.1819235381389075	51.46715821749747	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18763407	Water Wheel	03170245	-3.275394017628536	51.48654541063499	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18763515	Water Wheel	03170245	-3.278055125912196	51.48693905577124	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18763531	Water Wheel	03170245	-3.237845924319019	51.512053332790174	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18399410	War Memorial	03170245	-3.184184420524121	51.498310386350795	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18316176	Gospel Hall	06340459	-3.1896386688211638	51.47042517954909	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18316199	Llandaff North Gospel Hall	06340459	-3.2269843270613	51.502015608407945	Public Infrastructure	Infrastructure and facilities	Places of worship	134
18316229	Adamsdown Gospel Hall	06340459	-3.160946899023715	51.48129631323142	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18316232	Ely Gospel Hall	06340459	-3.2482539860296322	51.480021591451965	Public Infrastructure	Infrastructure and facilities	Places of worship	117
18316241	Ebenezer Gospel Hall	06340459	-3.1777775745199275	51.466048977410715	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18315933	Heath Gospel Hall	06340459	-3.1927756934695504	51.50402308166012	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18316210	Fairwater Gospel Hall	06340459	-3.2430658784072617	51.493645579290735	Public Infrastructure	Infrastructure and facilities	Places of worship	130
18316174	Kingdom Hall of Jehovah's Witnesses	06340459	-3.1928156785519315	51.49727875024067	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18316249	Kingdom Hall of Jehovah's Witnesses	06340459	-3.208722291020463	51.481577803761155	Public Infrastructure	Infrastructure and facilities	Places of worship	114
18315924	Welsh National War Memorial	03170245	-3.1804093551296027	51.48657805001025	Attractions	Historical and cultural	Historic and ceremonial structures	54
18316143	War Memorial	03170245	-3.1792404756023975	51.50434880965213	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18316085	War Memorial	03170245	-3.219381485712523	51.49570166072888	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18316164	War Memorial	03170245	-3.2674098988664664	51.48688376650274	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18316341	War Memorial	03170245	-3.2510585751423005	51.51461069749552	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18316416	War Memorial	03170245	-3.166730098942213	51.539713008853155	Attractions	Historical and cultural	Historic and ceremonial structures	-1
18763262	Tumulus	03170240	-3.219754509746824	51.461645258860656	Attractions	Historical and cultural	Archaeological sites	-1
17600457	Lych Gate	03170245	-3.189048469519203	51.529094310859186	Attractions	Historical and cultural	Historic and ceremonial structures	-1
17600550	Lych Gate	03170245	-3.1639189006988087	51.51825944001245	Attractions	Historical and cultural	Historic and ceremonial structures	-1
17598890	Lych Gate	03170245	-3.2360971084891785	51.48151717588538	Attractions	Historical and cultural	Historic and ceremonial structures	-1

« ‹ 1 / 622 › »

158

W02000410

Made by Josh Rimes 2025

Figure 20. The Dash application in its entirity

5. Analysis

To test whether clustering POIs can reveal correlation in socio-economic patterns, I analysed the correlation between the clusters of two different functional elements, retail and sports & entertainment, and the distribution of different age ranges across Cardiff. The retail categories include clothes stores, food retailers, and household stores, sport & entertainment categories include venues such as stadiums, gyms, cinemas, and theatres, all of which play a significant role in shaping the functional identity of urban regions. The intention was to demonstrate a proof of concept rather than produce a full demographic model, and due to time constraints, only age range was considered as a variable. Further variables could have been tested, such as employment rate or income levels, but for the purposes of this project it was sufficient to establish that a correlation could be found between age and the presence of certain POI types.

5.1 Retail

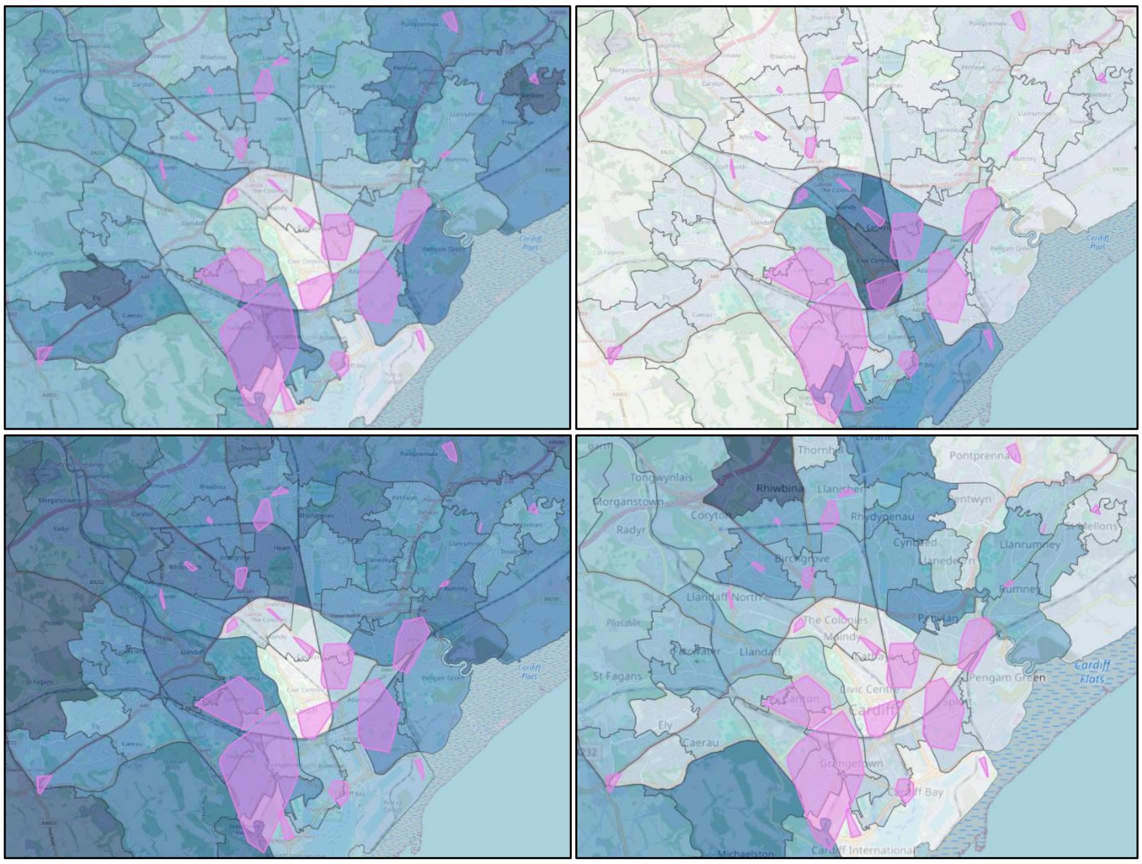


Figure 21. Top left: 0-15 Top right: 16-34 Bottom left: 35-64 Bottom right: 65+

The method used involved spatial correlation between POI clusters and MSOA areas. I implemented the system that displays the POI clusters and MSOA area so that overlaps can be observed and IDs can be used to compute the spatial correlation to show how strongly correlated the cluster's location is with different age group distributions. This was done using Pearson’s correlation coefficient (r value) and considering the statistical significance using the associated p values.

The first test involved a method I developed which I called the rP calculation. Each time a retail POI cluster overlapped an MSOA area - this occurred 37 times, an r value was calculated for that overlap and multiplied by the percentage of the population in that MSOA that fell within one of four age ranges: 0-15, 16-34, 35-64, and 65+. The resulting value, which I referred to as rP (r value * percentage), gave a weighted correlation that reflects both the strength of correlation and the size of the age group present. The average rP value was then calculated for each age range.

The final average rP values were as follows:

Age range	rP
0-15	5.68
16-34	13.7
35-64	11.2
65+	3.98

Table 3. Table showing age ranges and rP values for retail clusters

These results indicate that the strongest correlation between retail POI clusters and population was with the 16-34 group, followed by 35-64. This suggests that retail areas are more likely to be located in regions where the population is predominantly working-age adults and young adults. In contrast, areas with higher proportions of children and the elderly showed weaker correlations with the retail clusters.

It was important to verify that these findings were statistically significant. For each r value calculated, I also retrieved the corresponding p value to determine the validity of the correlation. A p value less than 0.05 indicates that the result is statistically significant. All the p values in this analysis that were above this threshold coincided with having extremely low r values which could be investigated further in future studies. Only p values below the threshold were used in calculating rP values, confirming the reliability of the correlations found.

In a second test, I examined the average percentage of each age group across all MSOA areas. For each age range, I filtered the data to include only MSOA areas where the percentage of that age group was equal to or greater than the average. I then calculated the average r value for cluster overlaps with these filtered areas.

The final average r values for areas with higher-than-average percentages for each age group were:

Age range	r
0-15	0.351
16-34	0.352
35-64	0.353
65+	0.348

Table 4. Table showing age ranges and average r values for retail clusters

These results show very close values across all four age groups, though again 35-64 was slightly higher. While this second test shows less variation between age groups, it still supports the idea that there is a measurable relationship between the presence of retail POIs and the population characteristics of the area.

5.2 Sports & Entertainment

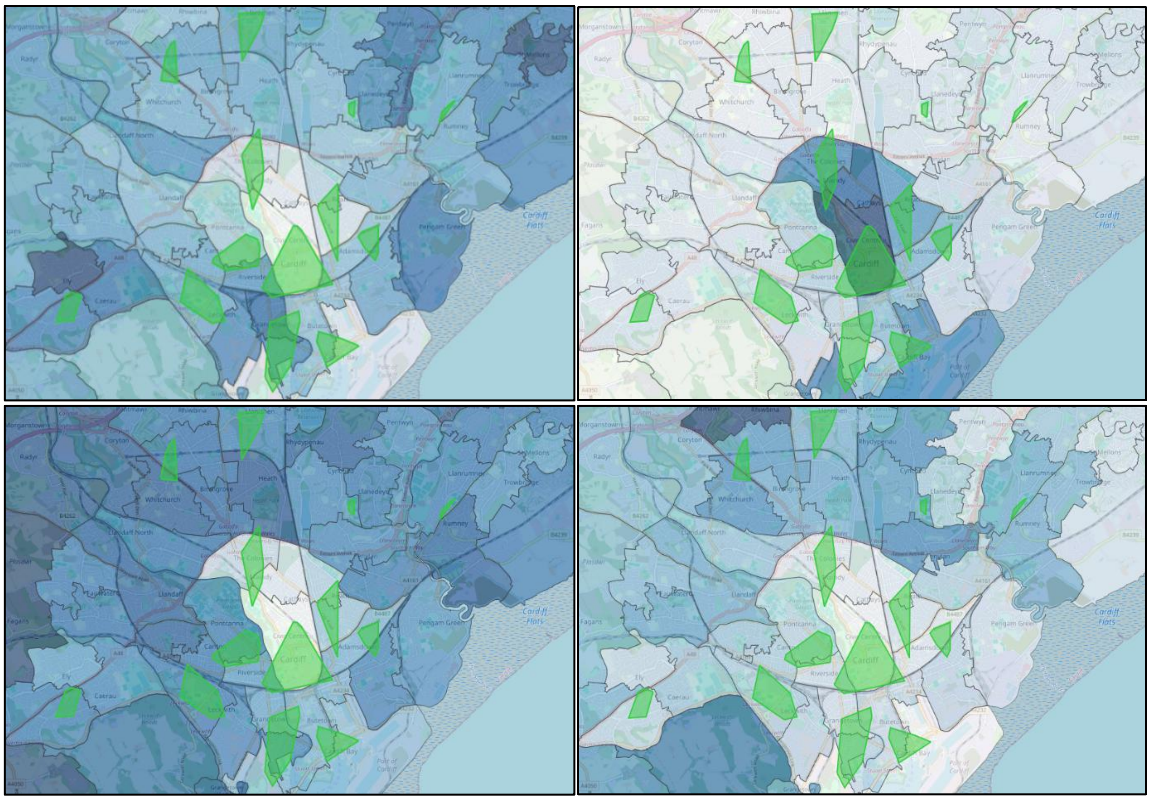


Figure 22. Top left: 0-15 Top right: 16-34 Bottom left: 35-64 Bottom right: 65+

The same two tests were conducted for sports & entertainment clusters. The first set of results were the average rP values for each age range. The values were as follows:

Age range	rP
0-15	5.39
16-34	14.46
35-64	10.49
65+	3.77

Table 5. Table showing age ranges and rP values for sports & entertainment clusters

These values suggest a similar pattern to what I observed in the retail analysis. Younger age groups, particularly those between 16-34, showed the strongest correlation with the locations of sports & entertainment clusters. This correlation likely reflects their higher engagement with leisure and recreational activities and their tendency to live or spend time in more socially active areas.

The second set of values, derived from the average r values in areas where each age range had an above-average presence, returned the following results:

Age range	r
0-15	0.401
16-34	0.281
35-64	0.414
65+	0.430

Table 6. Table showing age ranges and average r values for sports & entertainment clusters

Interestingly, while ages 16-34 had the highest rP score in the first set, it had the lowest r score in this second test. In contrast, the older age groups (35–64 and 65+) showed surprisingly high correlation values in this test. This may suggest that while sports and entertainment facilities are more concentrated in younger-populated areas, there are still notable overlaps in regions with higher proportions of older populations. This potentially reflects multi-use areas or broader community access to these facilities.

In comparison to the retail cluster analysis, the sports & entertainment clusters present a more nuanced picture. Retail clusters aligned more directly with younger populations in both sets of tests, whereas sports and entertainment clusters showed varying degrees of correlation across all age ranges. This could reflect the more diverse appeal of sports and entertainment venues and their broader spatial distribution across the city.

5.3 Analysis Conclusion

These two sets of tests confirm that there is a demonstrable correlation between POI clusters and demographic variables, in this case, age range. This proves the concept that POI data can be used to identify functional and demographic regions in a city. Although limited in scope, the system successfully demonstrates that meaningful insights can be derived from POI clustering when combined with spatial and demographic data. The results are consistent with real-world expectations, and they form a solid basis for further development and testing.

One area that could be explored in future analysis is the effect of adjusting the min_samples parameter used in the clustering algorithm. In this project, a higher min_samples value was chosen to identify mostly significant retail clusters across the city, which helped keep the analysis focused on larger urban centres. However, lowering the min_samples value would allow for the detection of smaller clusters that may exist in residential neighbourhoods or suburban areas like some of the smaller clusters already found at a higher min_samples value. These smaller clusters could reveal additional patterns of correlation, particularly with age groups like 0-15 or 65+, which may not be strongly represented in large central clusters. By investigating correlations at this finer level of detail, the system could potentially uncover more nuanced relationships between the built environment and demographic distribution.

In terms of how well the project goals were achieved, the system was able to demonstrate the central idea: that POI clustering can reveal real-world patterns. While not every part of the original proposal was implemented due to time limitations, the core functionality works and produces interpretable results. Given more time, I would test the system using a wider range of demographic and economic data to explore additional relationships.

I believe the methodology and technologies used were appropriate for the project. The combination of Python, Pandas, and DBSCAN was suitable for working with geospatial data at scale, and the testing approach using correlation statistics provided a clear way to evaluate the results. There is still room for improvement, and this will be discussed in the 6.1 Future Work section.

6. Conclusion & Future Work

6.1 Conclusion

The third and most important aim of this project was to investigate whether clustering POIs could be used as a basis for analysing urban functionality and its correlation with socio-economic factors. To achieve this, I implemented a Python system capable of preprocessing POI data, identifying spatial clusters using the DBSCAN algorithm, and visualising both clusters and socio-economic overlays. A key motivation behind this work was to determine whether POI data could offer insight into how urban spaces operate and how this correlates with demographic and economic conditions.

Although the original plan included analysing both Cardiff and Bristol to make comparisons between similarly sized cities, only Cardiff was analysed in full due to time constraints. However, the system was successfully implemented and used to identify and analyse functional regions of Cardiff, and from there, I was able to carry out an in depth analysis of spatial correlation between retail POI clusters and age related socio-economic data.

The analysis produced promising results. By applying a spatial correlation approach, two different test methods were conducted to measure the relationship between cluster locations and the percentage distribution of four age ranges. The first method introduced a metric called rP, a weighted correlation coefficient that multiplied the spatial r value of a region with the population proportion of a specific age group. The second test reinforced the findings by filtering for regions where each age group's percentage was above average and calculating the mean r value. The results again showed a positive correlation, with values ranging from 0.348 to 0.353 across age groups.

These findings support the initial hypothesis that POI data, when clustered and visualised, can be used to derive meaningful correlations with socio-economic variables. The methodology proved effective in highlighting where specific population groups are more likely to be located in proximity to retail functional regions. While this does not provide a full explanation of urban layout or causation, it demonstrates that POI clustering has the potential to contribute to urban studies and planning by offering an efficient, data-driven starting point for further investigation.

Despite these achievements, the project also faced a number of limitations. One big constraint was the computational capacity of my PC, which restricted analysis to only MSOA regions and a reduced set of POI categories. Using finer geographic divisions such as LSOA areas or expanding the POI classification to include the original 600+ types was not feasible due to memory and performance limitations. Additionally, the decision to focus solely on age demographics, rather than a broader set of socio-economic indicators like income, education or employment, was driven by time limitations. Nonetheless, for a proof-of-concept project, the decision to focus on a single demographic variable helped maintain clarity and feasibility within the timeframe.

The project achieved its main goal of developing and testing a system capable of identifying functional regions using POIs and analysing their relationship with socio-economic factors. It provides a strong foundation for future work and opens the door to more comprehensive urban analysis using similar methods.

6.2 Future Work

The outcomes of this project pave the way for several promising avenues for future work, both in terms of deepening the analysis and making improvements the underlying system.

One of the most immediate next steps would be to extend the analysis to additional cities. Comparing the results from cities with similar or different layouts, economies, and population distributions would help determine whether the observed correlations are unique to Cardiff or are indicative of broader patterns. Cities from different regions of the UK, or even internationally, could be included to account for location-based influences and validate the method’s generalisability.

The range of socio-economic variables used in this analysis was limited to age distribution. Future work could expand this to include a broader set of indicators such as average household income, education levels, employment rates, property prices, and health data. Many of these variables are available via government or open datasets, and their inclusion would provide a more comprehensive understanding of how functional regions impact, or are shaped by, different aspects of urban life.

There is also significant potential in the use of the full POI classification set. The POI dataset originally included over 600 unique types, offering an opportunity to move beyond broad categories like Retail or Health and instead explore more specific functional roles such as gyms, post offices, bakeries, or entertainment venues. This would not only make the clustering more detailed but might also uncover functional micro-regions and niche urban patterns that are currently obscured by generalisation.

Another key improvement would be in spatial resolution. While MSOA regions provided a manageable way to integrate socio-economic data, they are not optimal for fine-grained spatial analysis. Using LSOA areas, or even smaller geographic units if available, could improve the accuracy of spatial correlation tests. However, this would require more efficient memory management or more powerful hardware, as the current development setup encountered performance issues when attempting more granular analyses.

Experimenting with DBSCAN parameters, particularly lowering the min_samples value, could allow the system to identify smaller, denser clusters that may correspond to localised neighbourhood hubs or specialised commercial zones. This could be particularly useful in more suburban or sparsely populated areas where large clusters are less likely to form.

Further future work could also explore using temporal data. Functional regions may shift or behave differently at different times of the day or year. If POI data was combined with time-stamped mobility data, event calendars, or business hours could add another layer of analysis to this model.

Improvements to the user interface and interactivity of the visualisation component could make the tool more accessible to planners, researchers, or local governments who may benefit from quickly identifying socio-economically significant urban zones. Creating a lightweight, user-friendly dashboard or web-based version of the system would help bridge the gap between technical implementation and practical application.

In summary, this project has demonstrated the potential of POI clustering as a tool for urban analysis and socio-economic comparison. It has also revealed a wide range of opportunities for expansion and refinement, offering a valuable platform for continued development by future researchers, urban analysts, or students interested in smart city solutions and data-driven planning.

7. Reflection

Over the course of this project, I have experienced significant personal and academic growth, both in terms of the technical skills I developed and the deeper understanding I gained of the subject matter. Working alongside my supervisor, Dr. Padraig Corcoran, was a genuinely rewarding experience. His guidance helped shape my direction, especially during the early conceptual phases, and encouraged me to approach the project with both rigour and curiosity. I thoroughly enjoyed the learning and development process that unfolded throughout the last four months.

One of the most valuable aspects of this project was immersing myself in a topic that was entirely new to me, urban planning and the spatial layout of urban functions. At the same time, I began to explore socio-economic data, gaining insight into how demographics and services interact in the context of urban geography. This blend of spatial computing, data analysis, and social theory was both fascinating and intellectually inspiring.

A substantial portion of my time was devoted to learning and applying a broad range of Python libraries, many of which I had not used before. From data handling and geospatial analysis to clustering algorithms and web integration, I spent considerable time experimenting, testing, and debugging in order to create an effective system. This effort dramatically increased my proficiency in Python and my ability to independently learn complex technical tools.

Another challenge I faced was academic writing at this scale. Having not previously produced a report of this length or complexity, I had to develop my academic writing style from scratch. I quickly realised how critical clarity, structure, and justification are in communicating technical ideas effectively. Over time, I became more confident in my writing and found real enjoyment in improving how I expressed my analysis and findings. It is a skill I am proud to have developed and one that will benefit me far beyond this project.

I am incredibly proud of the work I have produced. The project has not only given me tangible technical experience, but it has also played a major role in shaping my aspirations for the future. Before beginning this work, I was unsure about which career direction I wanted to pursue after university. However, the enjoyment I've had engaging with data-driven research has sparked a strong interest in data science as a potential path. The problem-solving process, the creativity required in analysis, and the satisfaction of discovering meaningful insights have all been hugely motivating. This experience has been transformative; the technical knowledge, practical experience, and critical thinking I have developed will undoubtedly inform any systems I design in the future. More importantly, the lessons I've learned, about curiosity, persistence, and the value of thoughtful reflection, will stay with me throughout my career. I'm grateful for the journey this project has taken me on, and I look forward to building on it in whatever comes next.

Appendices

Dataset websites

1. Digimap: https://digimap.edina.ac.uk/
2. Office for National Statistics: https://www.ons.gov.uk

Python library websites

3. Scikit-Learn DBSCAN function: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
4. SciPy Pearson Correlation function: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html
5. Dash framework: https://dash.plotly.com

References

Atlassian, B. [no date]. *Waterfall Methodology for Project Management | Atlassian*. Available at: https://www.atlassian.com/agile/project-management/waterfall-methodology

Biljecki, F., Chow, Y.S. and Lee, K. 2023. Quality of crowdsourced geospatial building information: A global assessment of OpenStreetMap attributes. *Building and Environment* 237, p. 110295. Available at: https://www.sciencedirect.com/science/article/pii/S0360132323003220

Brown, L.A. and Holmes, J. 1971. The delimitation of functional regions, nodal regions, and hierarchies by functional distance approaches. *Ekistics* 32(192), pp. 387–391. Available at: https://www.jstor.org/stable/43617893

Cardiff's employment, unemployment and economic inactivity - Office for National Statistics. 2024. Available at: https://www.ons.gov.uk/visualisations/labourmarketlocal/W06000015/#inactivity

Carmona, M. 2010a. Contemporary Public Space: Critique and Classification, Part One: Critique. *Journal of Urban Design* 15(1), pp. 123–148. Available at: https://www.tandfonline.com/doi/full/10.1080/13574800903435651

Carmona, M. 2010b. Contemporary Public Space, Part Two: Classification. *Journal of Urban Design* 15(2), pp. 157–173. Available at: https://www.tandfonline.com/doi/full/10.1080/13574801003638111

Cesar. 2023. *The Importance of Urban Planning – The Seven Key Reasons*. Available at: https://www.archistar.ai/blog/the-importance-of-urban-planning-the-seven-key-reasons/

Duckham, M., Kulik, L., Worboys, M. and Galton, A. 2008. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition* 41(10), pp. 3224–3236. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0031320308001180

Edina. 1996. *Digimap*. Available at: https://digimap.edina.ac.uk/

Faust, F. 2021. How Liveable is my City? Part 2 | Urban Liveability According to the Mercer Quality of Living Report | QLab Think Tank GmbH. *Medium*. 14 October. Available at: https://medium.com/qlab-think-tank-gmbh/how-liveable-is-my-city-2452bb129292

Foursquare. 2024. *Who We Are | Foursquare*. Available at: https://location.foursquare.com/company/who-we-are/

Gushchin, A.N., Sanok, S., I., Pereverzeva, N., V. and Schneidmiller, N.F. 2017. *Does an ideal urban layout system exist?* Available at: https://www.revistaespacios.com/a17v38n24/17382416.html

Karl, T. 2024. *Benefits of Python for Data Analytics Explained*. Available at: https://www.newhorizons.com/resources/blog/benefits-of-python-for-data-analytics

Kelly, L. 2024. *What is POI Data? Everything You Need to Know*. Available at: https://datarade.ai/company/blog/what-is-poi-data

Kettle. 2017. *Distance on a sphere: The Haversine Formula*. Available at: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128

Khalil, H.A.E.E. 2012. Enhancing quality of life through strategic urban planning. *Sustainable Cities and Society* 5, pp. 77–86. Available at: https://www.sciencedirect.com/science/article/pii/S2210670712000455

Magrini, E. 2017. *How the layout of a city affects its economic success*. Available at: https://www.centreforcities.org/blog/layout-city-affects-economic-success/

National Institute of Standards and Technology. 2004. *Euclidean distance*. Available at: https://xlinux.nist.gov/dads/HTML/euclidndstnc.html

OpenStreetMap. [no date]. Available at: https://www.openstreetmap.org/about.

Ordnance Survey Ltd. 2020. *POINTS OF INTEREST CLASSIFICATION SCHEME*. pp. 1–17. Available at: https://www.ordnancesurvey.co.uk/documents/product-support/support/points-of-interest-classification-scheme.pdf

Precisely, Inc. 2023. *Precisely Points of Interest: Detailed POI data for location intelligence*. Available at: https://www.precisely.com/product/precisely-points-of-interest/precisely-points-of-interest

Regional gross domestic product: all ITL regions - Office for National Statistics. 2025. Available at: https://www.ons.gov.uk/economy/grossdomesticproductgdp/datasets/regionalgrossdomesticproductallnutslevelregions

Scikit-learn. [no date]. *DBSCAN*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Statistical geographies - Office for National Statistics. 2021. Available at: https://www.ons.gov.uk/methodology/geography/ukgeographies/statisticalgeographies

Turney, S. 2024. *Pearson Correlation Coefficient (r) | Guide & Examples*. Available at: https://www.scribbr.com/statistics/pearson-correlation-coefficient/

Warth, G., Braun, A., Assmann, O., Fleckenstein, K. and Hochschild, V. 2020. Prediction of Socio-Economic Indicators for Urban Planning Using VHR Satellite Imagery and Spatial Analysis. *Remote Sensing* 12(11), p. 1730. Available at: https://www.mdpi.com/2072-4292/12/11/1730

Wilkerson, M.L., Mitchell, M.G.E., Shanahan, D., Wilson, K.A., Ives, C.D., Lovelock, C.E. and Rhodes, J.R. 2018. The role of socio-economic factors in planning and managing urban ecosystem services. *Ecosystem Services* 31, pp. 102–110. Available at: https://www.sciencedirect.com/science/article/abs/pii/S2212041616302480

Yang, M., Kong, B., Dang, R. and Yan, X. 2022. Classifying urban functional regions by integrating buildings and points-of-interest using a stacking ensemble method. *International Journal of Applied Earth Observation and Geoinformation* 108, p. 102753. Available at: https://www.sciencedirect.com/science/article/pii/S0303243422000794