4. Methodology

This chapter describes in detail the methodological approach adopted for analysing accessibility to public services in the study area. The entire process was divided into several phases, each of which played a fundamental role in building a solid data infrastructure and meaningful, interpretable indicators.

The first step was data collection, which involved integrating heterogeneous sources, such as geographic datasets, GTFS public transport data, and public services location, carefully selected to ensure spatial and temporal consistency. Next, the process of data preparation and transformation is described, which is necessary to obtain origin-destination matrices (OD matrices) consistent with the spatial structure adopted. In addition, the method used to retrieve data from other sources when not obtained from Google Maps will be shown, along with the conclusions reached.

A specific section is dedicated to the concrete creation of accessibility indicators, which required the aggregation and synthesis of information on distances and travel times between areas and points of interest. This section also explains the method used to impute missing distance data that could not be obtained in the previous phase. The indicators were then subjected to an interpretative analysis to assess their meaning, distribution and territorial implications.

Finally, the last part of the methodology explains the methods used for the data publication process in accordance with European open data standards. Figure 10 aims to clarify the stages of the methodology, which are detailed below.

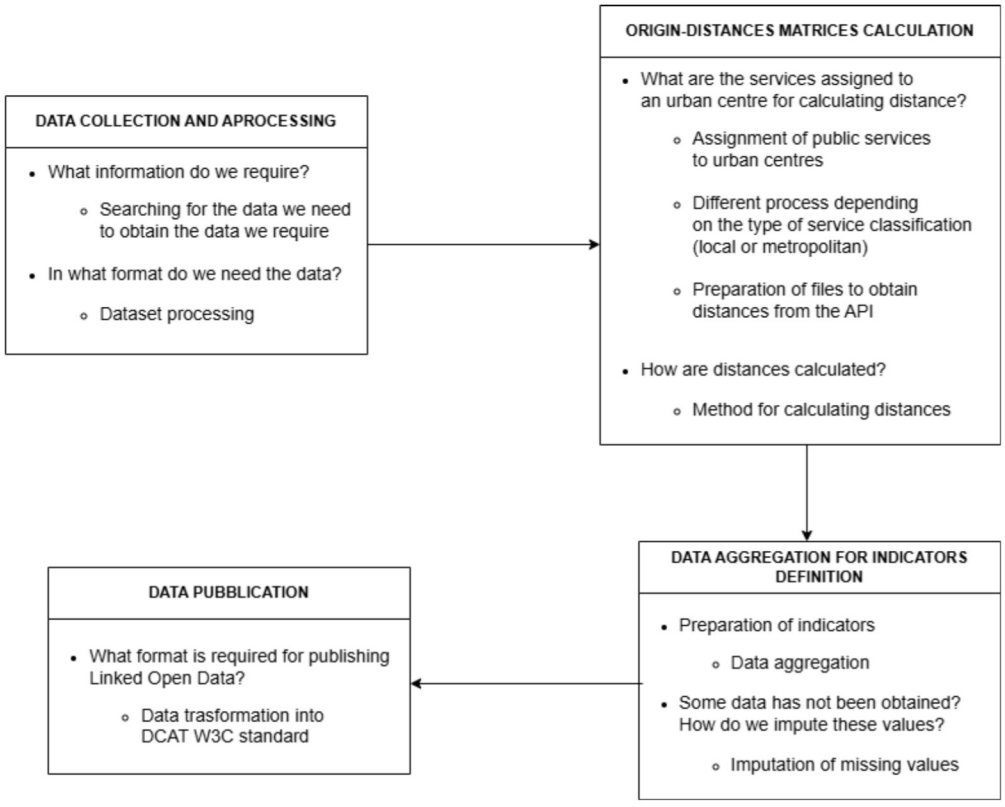


Figure 10: *Diagram explain the methodology phases*

The next 2 chapters concludes by describing the methods and tools used to visualise and publish the results, with the aim of making the data accessible and easily understandable even to public decision-makers and non-technical stakeholders, while ensuring the transparency and applicability of the entire process.

Before delving into the details of the methodology adopted, it is important to clearly reiterate the final objectives to be achieved through the application of the methods that are about to be described. The main objective of this work is to construct an accessibility indicator that quantifies the degree of accessibility of each urban centre in the Lazio region to public services of interest. In particular, the accessibility indicator is understood as the ease of reaching, in terms of both time and distance, the public services considered, which in this project are currently limited to the education and health sectors. The classification of services was carried out according to the criteria proposed by Kompil [13] and Verrachert [21], which distinguish between local and metropolitan services.

Category	**Sector**	**Type of service**
Local services	Education	Kindergarten Primary school Secondary school
	Health	General practitioners
Metropolitan Services	Health	Hospital facilities Emergency services

Table 7: Classification of public services considered for the calculation of accessibility indicators

Table 7 shows the categorisation adopted, highlighting the specific types of public services considered for each category and sector. For each type of service, the aim is to calculate an accessibility indicator expressed in terms of both time (travel time) and space (distance in kilometres), starting from urban centres and, in addition, from municipal boundaries. The accessibility indicator for each public service will be calculated considering two main modes of transport: by car and by public transport. This approach allows accessibility to be assessed on the basis of two of the main means of transport used to reach services, providing a more comprehensive and realistic view of the actual availability of services in the area.

4.1 Data collection

The first phase of the work involved researching and collecting the data needed to calculate accessibility indicators from population centres in Lazio to the main public services, with a particular focus on education and healthcare. The data were obtained from official sources, in accordance with European and Italian legislation on access to public data, and came from institutional websites or recognised national statistical bodies.

The raw data necessary and used to achieve the research objectives are, as follows:

1. Geometry of urban centres in the Lazio region: used to define the origin points for accessibility calculations and routing analyses.
2. Geometry of municipalities in the Lazio region: required for the territorial assignment of services and for spatial aggregations at the municipal level.
3. Road network of the Lazio region: necessary to support routing operations and to reposition centroids that fall outside urban geometries, ensuring realistic travel paths.
4. Population data for each urban centre and municipality: used for population-weighted aggregations and to meet project-specific criteria, such as evaluating accessibility in municipalities with fewer than 40,000 inhabitants.
5. Location data for schools in the Lazio region, including school category: serves as destination points in the OD matrix for assessing access to local educational services.
6. Location data for family doctors (general medicine) providing outpatient services: included as another category of local service for OD matrix computation.

7. Location data for hospitals in the Lazio region: considered as destination points for metropolitan-level services in the OD analysis.
8. Location data for emergency services in the Lazio region: similarly used to evaluate access to urgent care facilities at the metropolitan scale.
9. GTFS data of public transport providers in Rome: used to validate public transport availability and to complement missing data for transit-based accessibility calculations.

This initial section provides a basic explanation of the data collection process and includes a brief description of its structure and format.

In order to carry out the analysis, it was necessary to gather heterogeneous data from various official sources. The following main data sources were selected: the website of Italian National Institute of Statistics (ISTAT), OpenStreetMaps, the portal of the Italian Ministry of Education and Merit (MIUR), the portal of the regional health system of the Lazio Region (ASL LAZIO), the National Agency for Regional Health Services (PNE) and the official page of the administration of open data of Rome (OPEN DATA ROMA). The sources present datasets in different formats and not all data can be downloaded immediately or are structured in a uniform manner, making pre-processing and normalisation necessary.

Table 8: Sources, formats and update frequencies of datasets used for accessibility analysis

	Dataset	**Source**	**Format**	**Update_frequency**
1	Geometry of urban centres in the Lazio region	ISTAT [8]	Shapefile (.shp)	Every 10 years
2	Geometry of municipalities in the Lazio region	ISTAT [7]	Shapefile (.shp)	Yearly
3	Road network of the Lazio region	OpenStreetMaps [64]	Shapefile (.shp)	Continuously updated
4	Population of each urban centre and municipality	ISTAT [65]	CSV	Yearly
5	Location of schools with category	MIUR [66]	CSV	Yearly
6	Location of general practitioners	ASL LAZIO [67]	WEB	Varies by ASL
7	Location of hospitals	PNE [68]	CSV	Yearly
8	Location of emergency services	ASL LAZIO [69]	CSV	Varies by ASL
9	GTFS data of public transport companies in Roma	OPEN DATA ROMA [54]	ZIP	Monthly

Table 8 summarises the sources and formats for each dataset needed to achieve the project objectives, numbered according to the list of data types listed above. Some data are not available in structured or easily downloadable formats. In these cases, it was necessary to use web scraping techniques to extract the information directly from the official web pages. The purpose and reason for choosing this data will be explained in detail later on.

The objective of this first phase of the methodology is to build a solid and consistent data infrastructure containing all the information necessary for generating the accessibility indicators listed in table 7.A detailed description is provided below of the characteristics of the data from the various sources, the methods used for their acquisition, and the pre-processing operations carried out to adapt and prepare them for application in the subsequent methodological phases.

4.1.1 Geospatial data

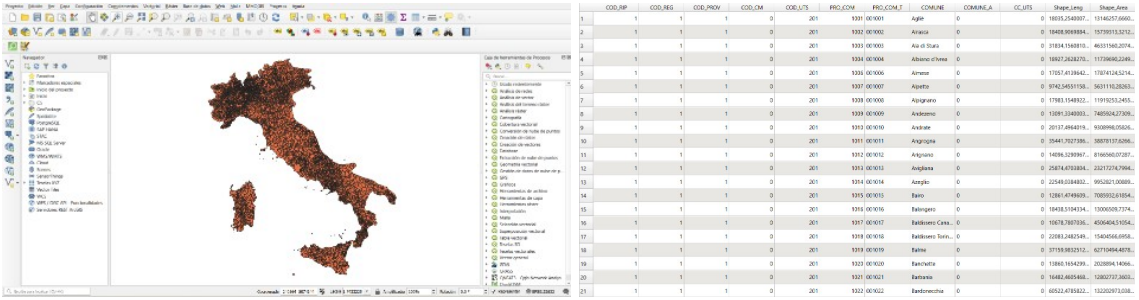
The geographical data was obtained from the official ISTAT website. The data was originally contained in a compressed archive (.zip) which, once extracted, contained vector files in shape file format (.shp). This data was imported into QGIS for initial visual inspection and for filtering, extraction and calculation of the necessary information. The datasets were then exported in GeoJSON format to facilitate their use in subsequent stages of the analysis. In particular, the goal of this step is extract the following key elements from the shape files of the municipalities and urban centres:

- The centroids of each municipality
- The list of neighbouring municipalities for each municipality
- The centroids of each urban centre

This information forms the spatial basis for constructing the origin-destination matrices and calculating the accessibility indicators that will be analysed in the next steps of the methodology.

Municipality data

After importing the vector layer (.shp) containing the raw data of Italian municipal boundaries into QGIS. The dataset covered the entire national territory. Figure 11a shows the initial cartographic representation of the layer, together with an example of the associated attribute table.



(a) *Municipality raw data visualization in QGIS* (b) *Variables of the municipality data*

Figure 11: *Visualization and variables of raw data about municipality in Lazio*

Since the analysis focuses exclusively on the Lazio region, it was necessary to extract a subset of data relating solely to municipalities belonging to that region. This operation was performed using the filtering function integrated in QGIS, accessible via the layer properties panel. In particular, the following query in SQL language (using OGR SQL syntax) was entered in the field dedicated to the filter expression:

```
"COD_REG" = 12
```

where `COD_REG` represents the variable containing the numerical code identifying the Lazio region within the ISTAT dataset. This condition made it possible to select only spatial elements with a region code equal to 12, generating a new vector layer containing only the municipalities of the Lazio region. This initial selection process optimised the dataset, bringing it into line with the geographical area of interest and improving the efficiency of subsequent spatial processing.

Once the vector layer containing the municipalities of the Lazio region had been isolated, two fundamental operations were carried out in QGIS:

1. **Extraction of centroids**: Using the integrated QGIS tool called Centroids, available in the process tools menu, in the vector geometry section, a new point layer was generated containing the geometric centroids of each municipal polygon. This step allows each municipality to be represented by a single point representing the centre of the town hall, which is essential for subsequent analysis. The output was then exported from QGIS in GeoJSON format.

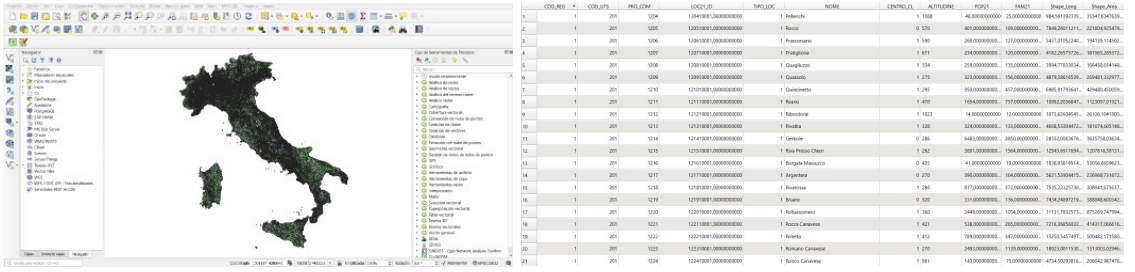
2. **Extraction of data relating to municipal boundaries**: As regards the extraction of neighbouring municipalities for each municipality, the process was divided into two distinct phases:

   - Attribute union operation by location in QGIS: Using the general vector analysis tool in QGIS called "Join attributes by location", a self-join operation was performed on the municipality layer, setting both the input layer and the comparison layer as the same municipal layer. 'Intersect' was chosen as the spatial predicate in order to identify all municipal polygons that intersect or share boundaries. The result of this operation was a new vector layer that was subsequently exported in GeoJSON format to facilitate further processing.
   - Cleaning and improving the GeoJSON dataset using Python scripts: Since the exported GeoJSON file was not completely clean or easily interpretable, a Python script based on the geopandas library was applied to optimise the information on neighbouring municipalities. Specifically, for each municipality contained in the GeoDataFrame, geometric (topological) adherence to other polygons was verified using the `.touches()` method, which returns adjacent polygons. The names of the neighbouring municipalities were collected in a list and assigned to the dedicated column 'neighbours' in the respective row of the GeoDataFrame. Finally, the dataset was simplified by keeping only the columns "MUNICIPALITY" and 'neighbours' and saved in a well-formatted JSON file, ready to be used in the subsequent stages of spatial analysis and territorial modelling.

The data relating to municipalities will be used to assign public services classified as local to each municipality in the region.

Urban centre data

As with municipalities, the layer relating to urban centres, downloaded from the ISTAT website, contains the geometric definition of all urban centres in Italy. As shown in the figure 12a, the dataset includes spatial information on each urban centre, but for the purposes of analysis it was necessary to isolate only the urban centres in the Lazio region.



(a) *Urban centre raw data visualization in QGIS* (b) *Variables of the urban centre data*

Figure 12: *Visualization and variables of raw data about urban centre in Lazio*

In addition to the spatial filtering operation to select only urban centres located within the boundaries of the Lazio region, further filtering criteria were applied. In particular, a selection was made based on the variable TYPE_LOC, to exclude urban centres classified as industrial areas or as "scattered houses" and "industrial areas", which are not relevant for the purposes of the analysis focused on residential areas. The SQL query (OGR SQL) used to obtain the relevant subset of data was as follows:

```
("TIPO_LOC" = 1 OR "TIPO_LOC" = 2) AND "POP21" > 0
```

Where the condition `'POP21' > 0` ensures that only urban centres with a population greater than zero are included. The variable POP21, which represents the population of each urban centre,

is particularly important, as it will be used in subsequent analyses to calculate accessibility indicators, weighting urban centres according to population density.

The last of the three specific objectives of this phase with spatial data is to generate a file containing the point centroids of urban centres. The centroids were initially calculated using the vector geometry tool integrated in QGIS, as described above for the municipalities. However, during a verification and validation phase, it emerged that many centroids were located outside the polygonal geometries of their respective urban centres. This phenomenon, highlighted, for example, in image 13, which shows the centroids calculated for the centres of "Centro" and "Forme di Suio-San Cataldo" di Castelforte (province of Latina), would have caused significant problems in the calculation of origin-destination matrices (OD matrices), particularly for the analysis of accessibility by public transport, as a centroid outside the geometry may not adequately represent the spatial position of the centre.

Figure 13: *Initial calculation of urban centre centroids – Municipality of Castelforte (LT)*

To resolve this issue, an analysis was conducted to realign the centroids using the road layer for the Lazio region as a reference. The road layer was imported into QGIS and filtered to retain only roads of territorial relevance within the region, using the following OGR SQL query:

```
"fclass" IN ('primary', 'secondary', 'residential', 'trunk', 'tertiary')
```

Next, using the 'Move points to nearest layer' tool in QGIS, the centroids outside the geometry were moved to the nearest road within the perimeter of the corresponding urban centre. This ensured that each centroid was correctly located within the geometry of the centre, thus improving the reliability of subsequent spatial analyses and accessibility calculations. Finally, the correct file containing the centroids of the urban centres was exported in GeoJSON format, thus ensuring optimal interoperability with the subsequent spatial analysis phases.

Next, using the 'Move points to nearest layer' tool in QGIS, the centroids outside the geometry were moved to the nearest road within the perimeter of the corresponding urban centre. This ensured that each centroid was correctly located within the geometry of the centre, thus improving the reliability of subsequent spatial analyses and accessibility calculations.

However, the validation process also revealed that several centroids, although technically located within the correct urban centre geometry, were placed in areas that were visibly disconnected from the actual built-up environment, such as isolated rural fields or agricultural zones. This became particularly evident during a visual inspection phase, in which all centroids were exported in KML format and overlaid on satellite imagery using Google Maps.

Although no further correction was applied in these cases, this observation highlights the inherent limitations in spatial data granularity and automated centroid generation. These anomalies in this work were accepted as part of the spatial heterogeneity of the dataset, and the analysis proceeded using the corrected centroids. Finally, the validated file containing the urban centre centroids was exported in GeoJSON format, thus ensuring optimal interoperability with the subsequent spatial analysis phases.

At the end of this first data collection phase, the following main files were generated: two GeoJSON files, containing the centroids of the municipalities and urban centres respectively, and a JSON file associating each municipality with a list of neighbouring municipalities. The next phase of data collection, which will be shown, is dedicated to the acquisition and preparation of data relating to education services, describing how the data was obtained and the adjustments necessary for analysis.

4.1.2 Locations data

The objective of this other sub-phase of data collection is to prepare data relating to the public services considered, in order to make them ready for the subsequent phases of the methodology. In particular, the objective is to obtain the following datasets:

- A dataset containing the locations of the schools of interest, with a variable indicating their category (e.g., nursery school, primary school, secondary school);
- A dataset containing the locations of general practitioners (family doctors);
- A dataset containing the hospitals in the area;
- A dataset containing emergency facilities;

These datasets will then be used to calculate accessibility indicators and to integrate spatial information on the distribution of public services in the Lazio region.

Educational centre data

In order to obtain a consistent and specific dataset of schools of interest in the Lazio region, after downloading it from the MIUR website in CSV format, the raw data contained in a CSV file was filtered using Python with Pandas.

The first step was to select only schools located in the Lazio region, based on the "REGION" variable, which allowed the analysis to be restricted to the territory of interest.

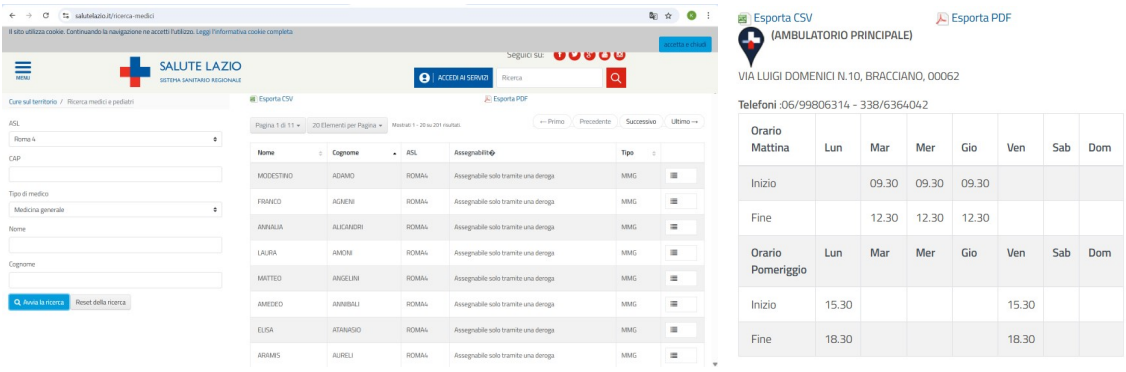
The dataset was then further refined to include only schools belonging to certain levels of education relevant to the analysis. In particular, schools with the following types of education were selected: nursery schools, primary schools, lower secondary schools and comprehensive schools. Although comprehensive schools do not appear in the classification of public services previously presented, they are essential institutions as they encompass all the mentioned school levels, and they play a crucial role in assigning school services to municipalities lacking full service coverage. This will be detailed later in the methodology. This was achieved by filtering the variable that describes the type of the school, which classifies the different levels of education. The resulting dataset includes only the relevant schools.

Finally, the data frame was exported to a new CSV file, ready for use in the next stages of analysis. This dataset is a fundamental part of the creation of accessibility indicators, as schools are one of the main public services of interest in the field of education.

Practitioners clinics data

Obtaining information on the locations of family doctors' surgeries was handled differently from other data, as the information was not available for direct download. In this case, it was necessary to apply a web scraping technique to extract the data from the family doctor search engine available on the ASL SALUTE LAZIO platform. The complete source code used to perform the web scraping to get the family doctors clinics is available at: https://github.com/keving-ing/Distance-project/tree/main/Raw_data_processing/web_scraping_salute.

A preliminary consideration that is fundamental for the collection of health data in the Lazio region is that the health system is organised into health zones (or metropolitan ASLs), given that the analysis was carried out only with municipalities with fewer than 40,000 inhabitants. The aim is to collect data related to ASL ROMA 4, ASL ROMA 5, ASL ROMA 6, ASL VITERBO, ASL LATINA, ASL RIETI and ASL FROSINONE. Those from other health zones refer only to data from clinics in the municipality of Rome.



Orario Mattina	Lun	Mar	Mer	Gio	Ven	Sab	Dom
Inizio		09.30	09.30	09.30			
Fine		12.30	12.30	12.30			
Orario Pomeriggio	**Lun**	**Mar**	**Mer**	**Gio**	**Ven**	**Sab**	**Dom**
Inizio	15.30				15.30		
Fine	18.30				18.30		

(a) *Search engine home page* (b) *Clinic address*

Figure 14: *Search engine of family doctors in the Lazio region - ASL SALUTE LAZIO platform*

The first step was to analyse the structure of the web page to understand how the data was organised. In particular, the HTML format of the page was examined.It can be observed in the image 14 that, after selecting the service type 'family doctors', a list of doctors appeared, and for each doctor there was a button that, when clicked, showed the details of the clinic, including the address and other relevant information.

The next step was to implement the web scraping process, which allowed us to automatically extract information from the HTML elements of the page, collecting the addresses of the family doctors' surgeries for each health unit. For this step, the browser automation tool `Selenium` was used, which allowed dynamic interaction with the SALUTE LAZIO platform, in particular with the family doctor search engine. The scraping methodology followed these steps:

1. Access to the platform: The web page containing the doctor search was opened via `Selenium`, using the Chrome browser driver. The URL for accessing the platform was the starting point for navigation;

2. Selection of the local health authority: The value corresponding to the relevant local health authority was selected iteratively from the drop-down menu dedicated to the selection of the local health authority. Interaction with the selection field was achieved using the Selenium command Select, which automated the selection of the correct value in the field with ID `_it_smc_laziocrea_sysmed_web_internal_portlet_SearchFormPortlet_asl`:

3. Starting the search: Once the ASL was selected, the "Start search" button was clicked, whose HTML element was identifiable by its ID `_it_smc_laziocrea_sysmed_web_internal_portlet_SearchFormPortlet_submit`. Clicking this button started the search for doctors, and the system loaded the results table;

4. Data extraction: Each doctor listed in the table was processed iteratively. The information was extracted from the rows of the table, identifiable by the CSS selector `.table-data tr`. For each row, the first name and surname data were retrieved from the table cells, located with the tags `<td>`. To access the clinic details, the 'Details' button (element with class `page-link`) was identified, which opened a window with the clinic address. The addresses were extracted from all text elements in paragraphs with class `p.text-uppercase`, filtering out the relevant ones and ignoring information about opening hours.

5. Navigating between results pages: The resulting page showed a list of doctors divided into several pages. Using the 'Next' navigation link, identified by the id text `Next`, the bot was able to move to the next page and continue collecting data. This process was repeated until all results pages were covered;

6. Saving data: Once the extraction was complete, the data relating to the doctors (name, surname and address of the surgery) was organised into a data frame using the pandas library, removing any duplicates. Finally, the dataset was exported in CSV format using pandas methods for later use in analysis.

This process made it possible to automatically collect the addresses of family doctors' surgeries in the area of interest, creating a dataset useful for analysing the accessibility of healthcare services in the area.

Hospitals and emergency ambulatory data

With regard to data relating to hospitals and emergency facilities, no complex pre-processing operations were necessary, as the data, already available in CSV format, contained detailed information on hospitals in the Lazio region, including addresses and information relevant to accessibility analysis. As indicated, the dataset was acquired directly from official sources and, as it was already correctly structured, it was used without significant changes. The information in the dataset was then used for the subsequent stages of the accessibility analysis, contributing to the creation of origin-destination matrices for the calculation of indicators.

At the end of the data collection phase, all the datasets needed to analyse accessibility to public services in the Lazio region were obtained and prepared. Data on municipalities, urban centres and public services (education, healthcare, hospitals and emergency services) were extracted, filtered and integrated into formats compatible with the subsequent analytical phases. With this data ready, the next step in the methodology involves calculating origin-destination matrices, a key step in determining accessibility to various public services based on distances and travel times between urban centres and points of interest. The next chapter is dedicated to this phase, which will describe the methods used to construct and analyse the matrices, integrating geospatial and socio-demographic information into a territorial accessibility model.

4.2 Calculation of origin-destination matrices

The calculation of origin-destination matrices (OD matrix) is a crucial step in assessing accessibility to public services. In this phase, the calculation of distances is separated into two sub-phases, based on the classification of services as either local or metropolitan. In particular, local services, such as schools and general practitioners, will be analysed separately from metropolitan services, such as hospitals and emergency facilities, which may require different modes of access or allocation due to their distribution across the territory. The OD matrices for local and metropolitan services will then be calculated taking into account the specific characteristics and administrative constraints when it comes to assigning which service to which urban centre.

The methodology is not limited to simply calculating distances, but also includes a series of data preparation operations. This preparatory process is essential to ensure that distances are calculated correctly. In particular, factors such as the correct definition of origins (urban centres) and destinations (services) and the correct assignment of distances and travel times based on the type of service must be taken into account.

The methodology considers online resources such as Google Maps to assess travel times from the centroids of the considered urban centres to the services. Google Maps is a very valuable source of information, as the spatial data of the road network are enriched with temporal data on travel times in different traffic conditions, providing important insights for accessibility. Google Maps employs graph data structures to calculate the shortest path from the origin (point A) to the destination (point B) and an algorithm to find the shortest path between a given origin and

destination. Google Maps determines the approximate arrival time taking into account factors such as the remaining distance, average speed due to real-time traffic conditions, historical data, and the officially recommended speed. Google Maps API requests are also made with the 'transit' mode to calculate the distance, in km, and time from an origin to a destination by public transport; It requests directions or distance using public transport routes (when available).

This work is based on realistic data, and Google Maps algorithms reflect more factors than a GIS transportation model based solely on infrastructure data. This section describes the details of data preparation and the methods used to calculate OD matrices, providing a comprehensive overview of the accessibility analysis process. The complete source code used to perform the OD matrix calculations is available at: https://github.com/keving-ing/Distance-project/tree/main/Script.

4.2.1 Distances to local public services: educational and general medicine

The process followed to construct the distance and time matrices from each urban centre to the most appropriate, educational and family doctors centre, in of their municipality, can be defined in four stages:

1. Identification of points of origin, municipalities and urban centres;
2. Identification of the service coordinates;
3. Assignment services to municipalities;
4. Call request to Google Distance Matrix API.

Identification of points of origin, municipalities and urban centres

- Municipalities with up to 40,000 inhabitants
  - First, the CSV file containing the population data for each municipality was uploaded, as shown in the data collection phase, obtained from the official ISTAT source. Using the pandas library in Python, the data was read and then filtered to select only municipalities with a population of less than 40,000 inhabitants. The result of this operation was a data frame containing a list of municipalities to be used for the next stage of analysis.
- Urban centres associated with each of these municipalities with population, except for those considered scattered and industrial zones
  - The GeoJSON file containing the spatial data of the urban centres' centroids, previously obtained during a pre-processing data collection phase, was uploaded. For each selected urban centre, the geographical coordinates were obtained in the UTM (Universal Transverse Mercator) reference system. Next, using a conversion function, the coordinates were transformed into the WGS84 geographic reference system (latitude and longitude), used for subsequent accessibility analyses. The final result is a dictionary that associates each urban centre with its ID and geographical coordinates (latitude and longitude).

Identification of the service coordinates

For some of the service locations, it was necessary to calculate the geographical coordinates (latitude and longitude), as the input data did not include this information in geographical format. Which addresses and why will be shown in the paragraph below. The method used to carry out this process is briefly described here to provide the reader with the necessary context. The process of converting addresses into geographical coordinates was carried out using the Google Maps Geocoding API. The method used to obtain the coordinates is as follows:

- Preparing the API request
  - For each address, contained in the dataset of the public services considered, a request was made to the Google Maps API using the address parameter (containing the address of the service) and the API key. The request was sent using the HTTP GET method to the geocoding endpoint, which returns a response in JSON format containing the geographical information related to the address.
- Cache management
  - To optimise the process and reduce the number of repetitive calls in case of errors, a cache system has been implemented. Before sending the request to the API, the script checks whether the address has already been geocoded, extracting the data from the local cache (stored in a JSON file). If there is a hit in the cache, the coordinates are returned immediately, avoiding a new API request.
- Response processing
  - Once the response is received from the API, the programme checks that the request status is 'OK'. If the response is positive, the geographical coordinates (latitude and longitude) are extracted from the JSON structure returned by the API and saved in the cache. Otherwise, an error is logged, and a log message is written to an error file to monitor geocoding issues.
- Saving the results
  - Saving the results The geocoded data is then returned for use in subsequent spatial analyses.

If an error occurs during the geocoding process (e.g., address not found or problems with the API), an error log containing the message returned by the API is written so that the problematic data can be dealt.

Once the geographical coordinates had been obtained through the geocoding process, it was necessary to perform a spatial validation of the geolocated points to verify that each point corresponding to a service was actually located within the correct municipal boundaries. The validation process was carried out using QGIS, a geographic information system software that allows spatial analysis to be performed. Each geolocated point (latitude and longitude) was associated with a specific municipality, as indicated in the database, which contains the name of the municipality to which it belongs as an attribute.

- Data preparation in QGIS
  - The geolocated data, in the form of points, was imported into QGIS in Geo-JSON format, with information about the points of interest (e.g., addresses of clinics or schools) and the municipality associated with each location. Similarly, the geometric layer of municipal boundaries, previously prepared, was loaded, containing the polygons that define the boundaries of each municipality in the Lazio region.
- Spatial check
  - For each geolocated point, a spatial intersection operation was performed in QGIS to check whether the point was within the polygon of the corresponding municipality. This was done using the "Check point within polygon" function, which checks whether the coordinates of the point are within the geometric boundary of the specified municipality.
- Validation results
  - Points that were within the boundaries of the municipality were considered valid, while those outside were flagged as anomalies. The results of this spatial validation were used to exclude or correct problematic data, ensuring that all locations were geographically correct and actually represented a point within the municipality to which they belonged.

- Manual corrections
  - For points that were not validated, manual corrections were made or, in some cases, the problematic points were excluded from the analysis to avoid errors in the subsequent stages of calculating the origin-destination matrices.

This validation process ensured that all geolocated points were correctly associated with municipalities, minimising the risk of errors in subsequent spatial analyses and providing a solid basis for calculating accessibility indicators.

Assignment services to municipalities

The first step involves identifying each of the infrastructures located within a municipality and automatically assigning it to the relevant municipality. After completing this step, the result is a JSON file containing the services available in each municipality. If a service is not available in a municipality, its corresponding category will remain empty. The output should follow the JSON structure outlined below:

```
{
    "MUNICIPALITY": {
        "SERVICE_TYPE": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ],
        "SERVICE_TYPE": []
    }
}
```

If a municipality lacks services of a required type, a search is conducted to find this missing public service type in neighbouring municipalities. The method applied is as follows:

(a) Identify the neighbouring municipalities of the municipality under consideration;

(b) Reference the JSON file generated in the first step to check if the required service type is available in any of the neighbouring municipalities;

(c) Consider all the centres found for the required service in the neighbouring municipalities;

(d) Calculate the network road distance using Google Maps between the centroid of the municipality and the centres identified:

- if more than 10 educational centres are found, only the 10 closest centres, based on Euclidean distance, will be considered for further distance calculations using Google Maps;
- in these calculations, the latitudes and longitudes of the service's centres are used as the coordinates for the Euclidean distance calculation.

(e) The Google Maps API call to calculate distances is performed using the following parameters, without considering the traffic:

```
params = {
    "origins": origin_str,
    "destinations": destination_str,
    "key": GOOGLE_MAPS_API_KEY,
    "mode": "driving"
}
```

(f) Assign the closest centre (in terms of distance in km) to the municipality.

(g) If no centres are found in the neighbouring municipalities, extend the search to the neighbours of the neighbouring municipalities for services.

The final JSON file will now have each types centre category filled in for each municipality

```
{
    "MUNICIPALITY_1": {
        "SERVICE_TYPE_1": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ],
        "SERVICE_TYPE_2": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ],
    },
    "MUNICIPALITY_2": {
        "SERVICE_TYPE_1": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ],
        "SERVICE_TYPE_2": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ],
    }
}
```

Call request to Google Distance Matrix API

Before making the API call, to simplify the process, the first step is add to the JSON file with the municipality and the services of the municipality, also the urban centres of the corresponding municipality.

```
{
    "MUNICIPALITY": {
        "SERVICE_TYPE": [
            {
                "SERVICE_DENOMINATION": "SERVICE_DENOMINATION",
                "SERVICE_LOCATION": "SERVICE_LOCATION"
            }
        ]
        "URBAN_CENTRE": [
            URBAN_CENTRE_1,
            URBAN_CENTRE_2
        ]
    }
}
```

Once the JSON is built, the number of elements required to complete the matrix will be determined. For each urban centre centroid, the distance to the educational centres within its municipality will be calculated. The following configurations are considered for the calculation of educational distance:

- Distance in km and time, at 8:00 am considering traffic (DRIVING)

```
params = {
    "origins": "|".join(origins),
    "destinations": "|".join(destinations),
    "key": GOOGLE_MAPS_API_KEY,
    "mode": "driving",
    "departure_time": time_unix_period  # Consider current traffic
}
```

- Distance in km and time with public transport at 8:00 am

```
params = {
    "origins": "|".join(origins),
    "destinations": "|".join(destinations),
    "key": GOOGLE_MAPS_API_KEY,
    "mode": "transit",
    "departure_time": time_unix_period
}
```

The API call is then made, with the origins being the centres of a municipality and the destinations being the different educational centres. Care is taken to avoid calculating the same distance twice, for example, if a school appears in two different categories. The calls are batched to comply with the following restrictions:

- In a single call, the maximum number of elements (in this case, the number of centres x the number of educational centres) cannot exceed 100;
- The number of origins (centres) and destinations (educational centres) must not exceed 25.

The objective of this phase was to obtain a final JSON file containing the distance between the centroid of each urban centre and the local service of interest for that urban centre. If the municipality of the urban centre already has the requested service, the distance is calculated for all the service facilities within the municipality. If the municipality does not have the service, the nearest service among the neighbouring municipalities is first assigned. Next, the distance is calculated only for the single infrastructure of the assigned service, considering the location of the latter in relation to the urban centre.

4.2.2 Distances to metropolitan public services: hospitals and emergency facilities

The method for calculating the distances to public services classified as metropolitan services, which include hospitals and emergency services, slightly differs from the method described earlier for local services. This process can be defined in four phases, with the only difference being in the method used for phase three. The methods used for the other phases are the same as those described for local data.

1. Identification of points of origin, municipalities and urban centres (refer 4.2.1 for detailed methodology).
2. Identification of the service coordinates (refer 4.2.1 for detailed methodology);
3. Assignment of services to urban centres;

4. Call request to Google Distance Matrix API (refer 4.2.1 for detailed methodology).

At this stage, the focus shifts to the methods used for assigning services to urban centres. In accordance with the constraints of the Italian healthcare system, this phase is treated separately, as the assignment methodology differs for hospitals and emergency services.

Assignment hospitals to urban centres

Before illustrating the methods used for the assignment of hospitals, two fundamental considerations must be taken into account:

- Service classification: As highlighted in the literature, it would be more accurate to classify the service as "healthcare provision" rather than simply "hospitals". For example, the classification could include medical, surgical, paediatric, and neonatal services, among others. However, due to the lack of open data from official sources, the analysis in this work is limited to calculating accessibility to "hospital services". Distance data to hospitals is stored in cached structures, allowing the possibility of future integration without the need to recalculate distances already computed.
- Regional health system division: The healthcare system in the Lazio region is divided into "health zones", represented by Local Health Authorities (Aziende Sanitarie Locali, ASLs). These are public entities within the Italian public administration, responsible for delivering healthcare services within a defined geographical area, usually at the provincial level. The ASLs are responsible for implementing the National Health Service and other legal obligations within their respective territories. Consequently, for administrative and economic reasons, each resident of a municipality will have access to the services available within the ASL of which their municipality is a part.

The method for assigning hospitals to urban centres follows the steps outlined below:

(a) The Euclidean distance from the centroid of the urban centre is calculated to all the hospitals located within the health zone to which the municipality belongs. This step allows the identification of the proximity of hospitals to the urban centre in terms of straight-line distance;

(b) The calculated distances are sorted in ascending order, from the closest to the farthest hospital;

(c) The calculated distances are sorted in ascending order, from the closest to the farthest hospital;

```
params = {
    "origins": "|".join(origins),
    "destinations": "|".join(destinations),
    "key": GOOGLE_MAPS_API_KEY,
    "mode": "driving",
    "departure_time": time_unix_period
}
```

(d) The driving distance is automatically saved in a cache data structure, making the previous calculation redundant. As a result, the driving distance between the urban centre and the hospital is already computed during this step of assignment, and therefore, step 4 becomes unnecessary for future calculations;

(e) The urban centre is assigned to the closest hospital based on the calculated driving distance, ensuring that the service is linked to the nearest hospital;

(f) The result of this phase is a JSON file structured as follows:

```
{
    "MUNICIPALITY": {
        "urban_centres": [
            URBAN_CENTRE_1,
            URBAN_CENTRE_2
        ],
        "DISTANCE": {
            "URBAN_CENTRE_1": {
                "HOSPITAL_ASSIGNED": {
                    "distanza_m": distance,
                    "tempo_s": time
                }
            },
            "URBAN_CENTRE_2": {
                "HOSPITAL_ASSIGNED": {
                    "distanza_m": distance,
                    "tempo_s": time
                }
            }
        }
    }
}
```

Although step 4 is effectively redundant due to the caching mechanism, it is still carried out for the public transport distance and to ensure that the distances are added to the JSON file in a consistent format, making the file compatible with other data aggregation processes in the subsequent stages.

Assignment emergency facility to urban centres

In the case of emergencies, an individual in a state of emergency is not required to go only to the emergency facility within the ASL of their municipality of residence. Therefore, the following methodological steps are used for assigning emergency facilities to urban centres. As previously mentioned, the datasets for hospitals and emergency facilities are slightly different, as not all hospitals within the hospital service provide emergency services.

(a) The Euclidean distance from the centroid of the urban centre is calculated to all emergency facilities in the Lazio region. This allows identification of the proximity of emergency services in terms of straight-line distance;

(b) The calculated distances are sorted in ascending order, from the closest to the farthest emergency facility;

(c) The driving distance from the centroid of the urban centre to the 3 closest emergency facilities, based on Euclidean distance, is calculated using the Google Maps API;

```
params = {
    "origins": "|".join(origins),
    "destinations": "|".join(destinations),
    "key": GOOGLE_MAPS_API_KEY,
    "mode": "driving",
    "departure_time": time_unix_period
}
```

(d) The driving distance is automatically saved in a cache data structure, making the previous calculation redundant. Therefore, the driving distance between the urban centre and the emergency facility is already computed during this step, and step 4 becomes unnecessary for future calculations;

(e) The urban centre is assigned to the 3 closest emergency facilities based on the calculated driving distance, ensuring that emergency services are assigned to the nearest facilities;

(f) The result of this phase is a JSON file structured as the previous for hospitals.

It is important to note that, in this phase, the public transport distance is not calculated, as it would not be appropriate to use public transport in an emergency situation.

4.2.3 Distances with public transport

As outlined in the methodology, the distance values are obtained using the Google Maps API, both for car travel and public transport. However, as previously mentioned and as will be further detailed in the chapter on results, some public transport distances could not be obtained. This issue is assumed to be caused by the fact that Google Maps does not contain updated data for all transport companies, the possibility of service unavailability during selected hours, or areas that are not covered by the public transport network.

Given these limitations, an alternative methodology was implemented to obtain the missing distances for public transport. This section outlines the alternative methods used, specifically:

- OpenTripPlanner (OTP) for routing: The first alternative method involved creating a routing network using OpenTripPlanner (OTP), which is capable of handling bus schedules and routes by integrating GTFS data (General Transit Feed Specification). OTP allows for accurate calculation of distances by considering transport schedules and route details. This solution helps overcome the limitations of Google Maps by providing more precise transit information for specific routes and times.
- Web Scraping with Moovit application: Another method used to obtain public transport distances involved scraping data from the Moovit application. Moovit provides real-time transit information, including bus schedules, routes, and travel times. By scraping the results from Moovit for the desired origin and destination points, it was possible to obtain public transport distances and travel times for cases where Google Maps failed to provide accurate data.

OpenTripPlanner (OTP) for routing

The first alternative methodology for calculating public transport distances is based on the use of OpenTripPlanner (OTP), a widely used open-source tool designed for multi-modal transport routing. OTP can process various types of data such as GTFS (General Transit Feed Specification) data and road networks to create a routing graph. This allows for the calculation of travel itineraries that combine walking and public transport, addressing the need for accessibility analysis across different transport modes. The process is divided into the following key steps:

(a) The first step in the process involves preparing the necessary data for OTP. This includes downloading and organizing the GTFS data, which provides schedules and stop locations for public transportation, as well as the road network data for walking routes. The GTFS data is typically made available by local transit authorities, while the road network is either provided directly or can be generated from available geographical data sources. Once all the data is gathered and placed in a single directory, the OTP graph is built using the following command:

```
java -Xmx4G -jar otp-2.4.0-shaded.jar --build data/
```

This command initializes the OTP application and processes the GTFS and road network data, creating the underlying graph used for routing. The `-Xmx4G`option ensures that OTP is allocated sufficient memory (4 GB in this case) to handle the large datasets.

(b) Once the graph is built, the OTP server is started to handle routing requests:

```
java -Xmx4G -jar otp-2.4.0-shaded.jar --serve --load data/
```

The `-serve` flag starts the OTP server, and the –load flag ensures that the data is loaded into memory. The server runs on localhost by default, providing an API endpoint to handle requests for routing between origin and destination points.

(c) A Python script is then used to interact with the OTP server. This script sends requests to the OTP API for calculating routes between an origin and a destination, both of which are specified by their geographical coordinates (latitude and longitude). The script uses the `requests` library to interact with the OTP server's API. The key parameters in the request are as follows:

- `fromPlace` and `toPlace`: These specify the origin and destination coordinates for the routing request.
- `time` and `date`: These parameters define the departure time for the journey
- `mode`: The mode is set to 'TRANSIT, WALK', indicating that OTP should find a route that combines both walking and public transit.
- `maxWalkDistance`: This parameter limits the maximum walking distance to 1 km for each journey leg.
- `arriveBy`: This is set to 'false', meaning the route is calculated assuming departure at the specified time, rather than arrival.

If the route is found, the script processes the itinerary, which includes walking legs and transit connections. The total duration and distance for the itinerary are calculated and saved to the log file.

(d) If OTP is unable to find a route (e.g., no connection exists between the origin and destination), the script logs this as an error with the message: error: No transit connection found between origin and destination. This ensures that any failed requests are captured for further analysis.

(e) The script processes multiple origin-destination pairs in a batch, reading them from a log file. The regular expression pattern in the script extracts latitude and longitude coordinates for both the origin and destination from the log file.

(f) The results of the routing request, including travel times and distances, are logged in a file. The output includes details such as the travel time, distance, and individual route steps (e.g., walking and transit segments) with information on the transport route, agency, and the specific stops or stations involved.

(g) If a successful route is found, the results are formatted and saved as follows:

```
OTP found: ## minutes, ## km
1. WALK from 'Location A' to 'Stop 1'
2. BUS from 'Stop 1' to 'Stop 2' with route 'Line 1'
3. WALK from 'Stop 2' to 'Destination B'
```

This method is used for all distances that need to be calculated, allowing for comparison between them and with the results obtained using the Google Maps API.

This alternative method was primarily implemented as a validation step, in order to assess whether different routing engines (in this case, OpenTripPlanner and Google Maps) would produce divergent results under the same input conditions. However, contrary to initial expectations, the results obtained using OTP did not reveal substantial differences when compared to those retrieved via the Google Maps API.

According to the official Google documentation, the platform already integrates all publicly available GTFS feeds when generating public transport itineraries. As a consequence, the accessibility calculations based on Google Maps can be considered reliable and inclusive of local transit data. Therefore, the OTP implementation served as a methodological cross-check rather than a core component of the final accessibility analysis.

Distance Calculation Using Moovit

To complement the OpenTripPlanner approach, an alternative method for obtaining public transport distance data involves using Moovit, a widely used mobile application for real-time public transport information. This method relies on web scraping to collect data directly from the Moovit interface, which provides the most accurate and up-to-date transit information available. The process for calculating public transport distances using Moovit is detailed below.

Moovit does not provide an official API for free, so the data was collected using web scraping techniques. The following steps describe how the scraping process works:

(a) Initially, the address of the origin and destination are provided as inputs to the Moovit application. These address can either be extracted from a previously processed dataset (such as centroids of urban centres) or entered manually. The system accepts not only latitude and longitude as parameters.

(b) he script navigates to the Moovit website or app interface using a headless browser (e.g., Selenium), where the coordinates are entered into the input fields for "Origin" and "Destination".

(c) The time of travel and the transport modes (e.g., bus, metro) are also specified. The user can set the desired time of arrival or departure. By default, the current time is used, but this can be adjusted for different scheduling scenarios.

(d) The script simulates a user clicking the “Find Route” button on the Moovit interface. This action triggers the back-end calculation for the public transport itinerary, which includes walk, bus, metro, or other modes of transport. Moovit uses real-time data to generate this itinerary.

(e) Once the request is processed, the results are displayed on the screen. The script waits for the page to load and then extracts the relevant data (i.e., travel time and distance). This is done using web scraping tools to parse the HTML and extract the required information from the page elements.

(f) After scraping the travel time and distance data, the information is stored in a structured format, such as a CSV or JSON file, for further analysis and integration with other datasets (e.g., for OD Matrix calculation).

In this phase, the distances between urban centres and public transport services have been calculated using alternative methods to Google Maps, specifically leveraging OpenTripPlanner (OTP) and Moovit for more accurate and comprehensive routing data. These methods have allowed the integration of real-time transport information, improving the reliability of distance and time calculations for public transport services.

The results of these calculations are stored in a structured format (JSON), including both distance and travel times, for each origin-destination pair. The data is now prepared for further processing, ensuring consistency and compatibility with other datasets.

With the distances and travel times calculated and stored, the next step in the methodology is the aggregation of the data. In this next phase, the individual data points will be aggregated at before at urban centre level, then at the municipality level, combining all relevant data to derive meaningful indicators of accessibility. This will allow for the calculation of weighted averages, standard deviations, and other necessary metrics for further analysis of accessibility across different public services.

4.3 Data Aggregation for indicator creation

Once the distances have been calculated as described in the previous step of the methodology, these data must be aggregated and combined to achieve the primary objective: to obtain an accessibility indicator for each type of public service, classified in Table 7, with a value representing the distance in kilometres and travel time in minutes, both for car access and public transport.

This section of the methodology outlines the methods used for data aggregation, distinguishing between two phases: aggregation of data related to distances calculated using the driving mode and those based on public transport. The need for this separation arises from the fact that certain results (which will be detailed in the results section) for public transport distances could not be obtained due to the lack of publicly available data, both from the Google Maps API and from the GTFS data of the transport companies operating in the Lazio region. The source code referring to the methods in this part of the methodology can be found at the link: https://github.com/keving-ing/Distance-project/tree/main/Raw_data_processing/Script.

4.3.1 Driving distance data aggregations

After obtaining all the origin-destination distance matrices, the first step in the aggregation process is to compute the values for each urban centre, followed by aggregation at the municipal level.

- **Aggregation at the urban centre level**: Aggregation at the urban centre level is necessary because multiple infrastructure locations of the same service may have distances calculated for a single urban centre. Aggregation is performed by calculating the mean distances and standard deviations for each service type (e.g., early kindergarten, primary education, secondary education, family doctors, hospitals, and emergency services). As a result, a data frames is obtained for each urban centre, with the following structure:

Table 9: Structure of the data frame with data aggregated by urban centre - method: DR

		Public services	
Urban centres		**Mean**	**St. Dv**
	Distance [Km]		
	Distance [min]		

The mean distance (mean_km) and mean time (mean_min) for each urban centre are calculated using the following unweighted formulas:

$$\text{mean\_km} = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\text{mean\_min} = \frac{\sum_{i=1}^{N} y_i}{N}$$

Where:

- x_i and y_i are the distances and durations, respectively;
- N is the total number of distances or durations for the given urban centre.

The standard deviation is calculated using the pandas library, following the formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Where:

- σ is the standard deviation.
- x_i represents each individual distance or duration value;
- μ is the mean of the distances or durations;
- N is the total number of distances or durations.

If, for a given service in a specific urban centre, there is only one infrastructure available, the standard deviation cannot be calculated. In this case, the standard deviation values for that service type will be omitted from the results.

- **Aggregation at municipality level**: After aggregating the data at the urban centre level, the next step is to aggregate the data by municipality. This involves calculating the average distance and time for each urban centre (the data aggregated in point 1) of the municipality, for each public service. Using these values, the weighted average and the combined weighted standard deviation are calculated for each municipality, with the total population of urban centres serving as the weight. The dataset of the urban centres contains for each urban centre its population, so this information was used. This results in a data frame with the following structure:

Table 10: Structure of the data frame with data aggregated by municipality - method: DR

Municipality		**Public services**	
		Mean	**St. Dv**
	Distance [Km]		
	Distance [min]		

The weighted average for each service type (e.g., mean distance to the service and mean travel time) was calculated for each municipality. The weight used in the calculation was the population of the urban centres within the municipality. For each service type, the weighted average is computed as follows:

$$\text{mean\_km} = \frac{\sum_{i=1}^{N}(x_i \cdot w_i)}{\sum_{i=1}^{N} w_i}$$

$$\text{mean\_min} = \frac{\sum_{i=1}^{N}(y_i \cdot w_i)}{\sum_{i=1}^{N} w_i}$$

Where:

- x_i and y_i represent the distances and durations for the urban centre i;
- w_i is the population of the corresponding urban centre;
- N is the total number of urban centres in the municipality.

The standard deviation for each service type is computed at the municipality level to capture the variation in distances and times across the urban centres. The formula for the standard deviation is:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Where:

- σ is the standard deviation,
- x_i represents each individual distance or duration value;
- μ is the weighted mean of the distances or durations;
- N is the total number of distances or durations.

4.3.2 Public transport data aggregations

Before moving on to see how the data on access distances by public transport were aggregated, it is important to note that some distances could not be calculated and are therefore missing, due to a lack of data. A process of imputation of values was necessary.

- **Aggregation at the urban centre level**: The aggregation for public transport data is practically the same as that described in the method for aggregation at the level of car distance data (see point 4.3.1), except that the number of services available for the distances is taken into account and specified. Therefore, in addition to the average and standard deviation information, the data frame will also contain the number of facilities for the service out of the total number in that urban centre that were considered. The structure of the data frame is as follows:

Table 11: Structure of the data frame with data aggregated by urban centre - method: TP

Urban centres		**Public services**		
		Mean	**St. Dv**	**Present / Total**
	Distance [Km]			
	Distance [min]			

So now, for the types of services where there is no value for that service in public transport, assigning this value is necessary, as missing data compromises the accuracy of aggregation by municipality. The following section illustrates how values were assigned for the urban cores that do not have public transport distance data for a given type of service.

Imputation of public transport distances missing values

Given the spatial nature of the data and the lack of strong predictive variables available at the urban centres level, regression-based imputation methods were deemed inappropriate. Instead, a conservative penalized approach was chosen, leveraging the maximum observed value at the municipality level as a bounding reference. The penalty was scaled based on the proportion of available data and the absence of services in the area, following principles commonly used in territorial accessibility studies. [70], [71] This method avoids underestimation and allows a flexible control of uncertainty via a tunable parameter λ. Missing public transport distances (km and minutes) values were imputed using a penalized maximum-based approach, parametrized by λ. This process was applied independently for each public service type $c \in \{\text{Kindergarten}, \text{Primary}, \text{Secondary}, \text{General\_medicine}, \text{Hospitals}\}$, and for each urban centres i.

Let:

- $T_i^{(c)}$ be the observed (possibly missing) travel time for category c in urban centre i,
- $T_{\max}^{(m,c)}$ be the maximum observed (non-missing) value for category c in municipality m,
- $p^{(m,c)}$ be the proportion of non-missing values for category c in municipality m,
- $\lambda \in \mathbb{R}^+$ be the penalization factor,
- $\delta_i^{(c)} \in \{0, 1\}$ be a binary indicator: $\delta_i^{(c)} = 1$ if the urban centre has a school of category c, 0 otherwise.

Imputation is performed only if:

- $T_i^{(c)}$ is missing,
- $T_{\max}^{(m,c)}$ is available (i.e., at least one valid value in the municipality),
- $\delta_i^{(c)} = 0$ (i.e., the urban centre does not have a school of that category).

The imputed value is then computed as:

$$\hat{T}_i^{(c,\lambda)} = T_{\max}^{(m,c)} \cdot \left[(1 - p^{(m,c)}) \cdot \lambda \cdot \beta + p^{(m,c)}\right]$$

where $\beta = 1.5$ is a fixed penalty multiplier applied to control the degree of imputation. No imputation is performed if the above conditions are not met.

- **Aggregation at municipality level**

  Given that the original dataset provides accessibility indicators at the *urban centre* level, values were aggregated to the *municipality* level for spatial consistency and interpretability. For each school category $c \in \{\text{Kindergarten}, \text{Primary}, \text{Secondary}, \text{General\_medicine}, \text{Hospitals}\}$, and for each imputation scenario defined by a parameter λ, the population-weighted average travel time for municipality m was computed as:

$$\bar{T}_m^{(c,\lambda)} = \frac{\sum\limits_{i \in \mathcal{N}_m} T_i^{(c,\lambda)} \cdot P_i}{\sum\limits_{i \in \mathcal{N}_m} P_i}$$

  where:

  - $\mathcal{N}_m$ is the set of urban centres belonging to municipality m,
  - $T_i^{(c,\lambda)}$ is the (possibly imputed) mean travel time for urban centre i, category c, and lambda value λ,
  - P_i is the population associated with urban centres i.

The methodology outlined in the chapter, from the identification of urban centres and the calculation of distances to the assignment of services, has culminated in the creation of a structured file containing these indicators in CSV format. This file is ready for use in the next stages of analysis and visualization.

Statistical analysis of the obtained indicators has been conducted to examine the distributions and to address the research questions outlined in the objectives of the study. These analyses provide valuable insights into the accessibility of different public services, enabling a deeper understanding of the spatial inequalities in service availability.

The results of this methodology will be integrated into a dataset and displayed through the visualization techniques described in the subsequent chapters. This will allow decision-makers and stakeholders to easily interpret and act upon the insights generated from the data.

4.4 Publication of the data

The final step in the methodology, after collecting, processing and finally aggregating the data to define the indicators, is to publish the data infrastructure that has been created.

First step, to ensure that the datasets are compliant with Open Data publication standards, particularly those required for integration into the European Data Portal, the data must be transformed into RDF (Resource Description Framework) format. RDF enables data to be linked, semantically described, and easily consumed across distributed systems, in line with the specifications defined in the DCAT (Data Catalogue Vocabulary) standard maintained by the W3C [72].

For this purpose, a dedicated RDF transformation pipeline developed by the Department of Computer Engineering at University Carlos III of Madrid (UC3M) was adopted. This pipeline is part of an ongoing initiative for publishing accessibility indicators for the Community of Madrid, and it has been adapted in this work for datasets concerning the Lazio Region.

The pipeline takes as input a series of structured CSV files and produces RDF-compliant metadata and indicator values, serialized for publication. However, the transformation process requires that the input files strictly conform to a predefined schema, aligned with the accessibility data model adopted by the Madrid Region. This includes using standardized field names, consistent indicator coding, and proper formatting of geographic and statistical values.

As a result, prior to execution, all CSV files produced in this project were restructured to meet the required specifications. This step ensures full interoperability with the RDF pipeline and guarantees that the output complies with the standards defined for Linked Open Data across Europe.

In parallel with DCAT-based RDF serialisation, the semantic structure of the dataset has been modelled according to the ontology defined in the LoDCOREMadrid project 15. This ontology describes the entities and relationships that are fundamental for representing territorial indicators in an interoperable manner.

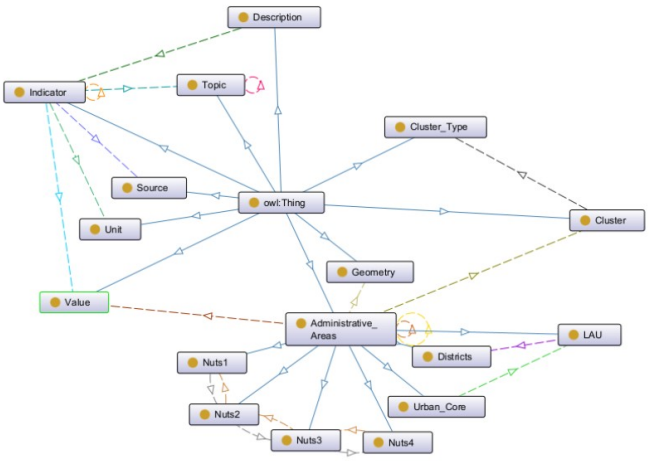


Figure 15: *Ontology as defined in the LoDCOREMadrid project [73]*

Specifically:

- Each indicator is an instance of the `Indicator` class and is described by a `Description`.
- It is associated with an `Administrative_Area` (e.g. LAU or Urban Centre).
- It has a `Unit`, a `Value`, and a `Source` from which it was obtained (`is_obtained`).
- It is organised into `Topics` through the `belongs_to` property.

This ontological approach allows for a formal representation of data that facilitates both integration with existing datasets and semantic querying via SPARQL, ensuring full compliance with Linked Open Data publication standards.

Furthermore, the RDF output is also published via a Virtuoso triple store, which enables SPARQL-based querying and integration into the European data infrastructure. Virtuoso is a high-performance, scalable graph database engine designed to store and manage RDF data. It supports the SPARQL query language and provides a public endpoint for Linked Open Data consumption. Publishing the accessibility indicators through Virtuoso ensures that the data is not only compliant with semantic web standards but also queryable via SPARQL endpoints, discoverable by third-party platforms, and aligned with the best practices for Linked Data publication.

The entire pubblic publication process has been adapted to the LoDCOREMadrid framework, an initiative of the Department of Computer Engineering at the Carlos III University of Madrid for the publication of Linked Open Data relating to territorial indicators on attractiveness and accessibility. The resulting files have been described using RDF triples, using standardised prefixes and properties consistent with the LoDCORE project specifications.

Below is a simplified example of the description of one of the datasets produced, relating to the average time taken to access educational services by public transport in Roma:

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
<http://lodcoremadrid.uc3m.es/dataset/educacion\_tiempo\_tp\_normalizado\_lazio>
a dcat:Dataset;
    dct:title "Educación - Tiempo transporte público normalizado por población
    (Lazio)"@es;
    dct:identifier "3.98";
    dct:publisher <https://lodcoremadrid.org> ;
    dcat:theme <http://lodcoremadrid.uc3m.es/theme/educacion>;
    dcat:distribution <https://example.org/data/educacion\_tiempo\_tp
    \_normalizado\_lazio.csv>
```

This approach not only ensures publication in accordance with Linked Open Data standards, but also allows for future semantic interoperability with similar datasets, facilitating integration into visualisation or automated analysis tools.