Data Management Project

Analysis and Ranking of Milan's Neighborhoods

Edoardo Olivieri[1, †], **Federica Romano**[1, †] **and Francesca Verna**[1, †]

[1]*Università degli Studi di Milano-Bicocca, CdLM Data Science*
[†]*These authors contributed equally to the development of the project. In particular Federica Romano focused on writing the presentation slides, while Edoardo Olivieri and Francesca Verna prepared the report and the operational guide.*

Abstract

This project focuses on building a comprehensive database to help users identify the most suitable neighborhood in Milan based on their specific needs and preferences. By integrating data from multiple sources, such as the Yelp API, official municipal datasets and queries on Overpass Turbo, we captured various dimensions of urban life, including amenities like restaurants, schools, universities, supermarkets, public transport, and parks. The data was carefully cleaned, processed, and stored in a MongoDB database with an embedded document structure to enable efficient querying and analysis. This report outlines the steps taken across the data management pipeline, from data acquisition to spatial integration and quality assessment, to create a tool that supports individuals and families in making informed decisions about relocating to Milan.

Keywords: *Milan, Neighborhoods Ranking, Data Acquisition, Data Preparation, MongoDB*

1. Introduction

Quality of life in a city is a multifaceted concept that depends heavily on individual needs and preferences. Students may prioritize access to affordable dining options, coworking spaces, and efficient public transport, while families often value proximity to schools, parks, and essential services. Singles or couples without children, on the other hand, may seek neighborhoods offering vibrant nightlife, cultural activities, and fitness facilities. Milan, as a dynamic and diverse city, offers an abundance of amenities and services; however, these are distributed unevenly across its neighborhoods, posing challenges for individuals trying to identify the area that best suits their lifestyle.

This study aims to build a comprehensive database that can assist individuals in identifying the most suitable neighborhood in Milan based on their specific needs. By analyzing key urban points of interest and infrastructures, along with housing prices, this project provides a detailed view of Milan's urban landscape. Data from various sources, including the Yelp API, official municipal datasets and Overpass Turbo have been integrated, cleaned, and organized into a MongoDB database, allowing for efficient spatial analysis and personalized insights. This approach not only offers a practical tool for users but also establishes a framework for evaluating urban quality of life, with potential applications for both individual decision-making and urban policy improvements.

2. Data Acquisition

The data acquisition process was a critical step in building a comprehensive dataset of Milan's neighborhoods. The objective was to collect reliable and diverse information about key points of interest such as restaurants, nightlife venues, and cultural institutions, alongside other essential locations like schools, parks, supermarkets and transport facilities. To achieve this, we utilized APIs, queries on Overpass Turbo and official datasets, carefully processing the data to ensure consistency and spatial alignment.

2.1. Data from OpenStreetMap

A key challenge was associating each point of interest with its corresponding neighborhood. Official datasets did not provide predefined neighborhood boundaries, and many amenities, retrieved through the API, listed all available blocks as their location to bypass the Yelp algorithm and be recommended to users regardless of their actual location. To address this, custom polygons were created for each neighborhood using Overpass Turbo, a querying tool for OpenStreetMap. These polygons were saved in a GeoJSON file and served as the reference for spatial mapping throughout the project, ensuring precise assignment of data points to their respective areas. In addition, a specific query performed on Overpass Turbo was utilized in order to retrieve data about supermarkets in each neighborhood.

2.2. Data Acquisition via Yelp API

The Yelp API was used to retrieve information about restaurants, museums and nightlife locations. To perform the requests of the data, the Python library *requests* was used. From the GeoJson file obtained with the queries performed on Overpass Turbo, it was possible to get the complete list of neighborhoods in Milan. These names were used as input for the API queries to retrieve relevant business information.

Access to the Yelp API was authenticated using a private API key. For each neighborhood, businesses of a specific type were queried with parameters specifying the number of results per request and pagination via an offset parameter. While the API allows up to 50 results per request, there is a maximum limit of 240 results per location. To optimize request efficiency and avoid wasting API calls, 48 results were requested per query, for a total of 5 requests per location. This approach ensures the maximum results are retrieved without exceeding the API's limitations or resulting in uneven pagination.

The data acquisition involved iterating through the list of neighborhoods. For each query, key details such as business name, address, categories, average star rating, review count, price range (only for restaurants), and geographic coordinates (latitude and longitude) were extracted. This process was repeated until no further results were returned or the maximum limit was reached. To mitigate potential issues, error handling was implemented to log and skip neighborhoods where HTTP errors occurred. Furthermore, a brief delay was introduced between requests to adhere to API rate limits and ensure un-

interrupted data retrieval.

Once the data collection was completed, the results were stored in a Pandas DataFrame. Geographic coordinates were used to enrich the dataset by converting latitude and longitude into geometric points, facilitating spatial analysis. The DataFrame was subsequently converted into a GeoDataFrame with a standard Coordinate Reference System (EPSG:4326) to ensure compatibility with geographic analysis tools.

2.3. Downloaded Datasets

In addition to the API data, official datasets provided by the City of Milan Open Data Portal were downloaded. These datasets, in GeoJSON format, included information on schools, parks, pharmacies, transport stops, and other key locations. To see the detailed list of these datasets, please refer to the Datasets Section at the end of the report. Each dataset was then loaded into a GeoDataFrame using the *geopandas* library and converted to a common Coordinate Reference System.

In addition, to get all the information about the real estate prices in Milan, both tabular and geographic datasets were downloaded. The tabular dataset, in csv format, provides a summary of real estate valuations collected through the Osservatorio del Mercato Immobiliare (OMI), which analyzes and processes technical and economic data related to property values and rental markets. It contains information about market values and rental values, expressed in euros per square meter, and the property type of the real estate. The GeoJSON file contained geospatial data related to real estate zoning and boundaries.

A directory for storing all the downloaded GeoJSON files was created programmatically to ensure proper organization. Error handling was incorporated to manage potential issues during the download process, ensuring that the data retrieval was robust and reliable.

3. Data Preparation

After collecting the raw data from the primary sources outlined in Section 2, it was necessary to prepare the data to ensure it was ready for storage and subsequent analysis. This required performing integration and cleaning operations.

3.1. Spatial Join

To integrate the data collected from our primary sources, a spatial join was performed between the various datasets and the GeoJSON file obtained from Overpass Turbo, which contains the boundaries of Milan's neighborhoods. This operation was carried out using the *sjoin* function, which matches spatial geometries based on their spatial relationship. Specifically, the join was executed with the *how=left* and *predicate=within* parameters.

The *how* parameter determines the type of join to be performed, with *left* ensuring that all records from the first dataset are retained, even if they do not match any geometries in the second dataset. This approach is crucial to avoid losing any data points during the join process. The predicate parameter specifies the spatial relationship to evaluate. In this case, *predicate=within* ensures that each data point is matched to the neighborhood geometry it falls within.

This spatial join is essential for the analysis being conducted, as it allows each data point to be associated with a specific neighborhood. By linking individual business locations or venues to their respective neighborhoods, it becomes possible to analyze and compare the availability and distribution of amenities across different areas of Milan.

3.2. Data Cleaning

For each dataset, it is crucial to evaluate the quality of the data by checking for duplicates, errors, or inconsistencies.

One of the key operations performed on all datasets was removing rows where the geometry of a location was missing or invalid. Rows where the **Neighborhood** column contained *NaN* values were filtered out. These null values indicated that the spatial join did not result in a valid match between the dataset and the neighborhood boundaries, suggesting the location was not within any recognized neighborhood polygon. Removing these rows ensured that only valid, geospatially matched data was retained for further analysis, preserving the integrity of the dataset.

A more detailed description of how the different datasets were handled is provided below.

3.2.1. Restaurants, Museums, and Nightlife Venues

For datasets related to restaurants, museums, and nightlife venues, all duplicate records were removed to ensure that each venue was represented only once. Duplicates were identified based on attributes such as name, address, categories, rating, review count, and geometry. For restaurants, also the variable **Price**, which is not available for the other data, was taken into account.

3.2.2. Dog Parks and Playgrounds

For dog parks and playgrounds, duplicates identified based on **geometry** were retained. Some parks span across multiple neighborhoods, meaning they are shared between different areas. For example, the playground with identifier 5_227 on Via Manduria appears twice in the dataset, as it falls within two neighborhoods: *Parco delle Abbazie* and *Ronchetto delle Rane*. Both representations were retained to reflect the park's actual spatial distribution.

3.2.3. Sports Venues

The sports venues dataset presented duplicates with identical geometries but different **info** values, representing various activities offered at the same location. To address this:

- Rows with *CENTRO SPORTIVO*, a generic label, were removed if more specific activity descriptions existed for the same geometry. This was done after assessing that not all venues had at least one entry as *CENTRO SPORTIVO*.
- If a geometry had only a single row with *CENTRO SPORTIVO*, it was retained to avoid excluding entire venues.
- Similar activity labels were harmonized (e.g., *CALCIO A 5*, *CALCIO A 7*, and *CALCIO A 11* were merged into *CALCIO*).
- Rows labeled *PARCHEGGIO* were excluded, as they did not align with the dataset's focus on sports and recreation.
- Labels *NUOTO* and *FITNESS MACHINE* were replaced with *PISCINA* and *FITNESS* respectively.

3.2.4. Universities

In the university dataset, duplicates were firstly identified based on the **geometry** column. Since each record represented a degree program, multiple entries could exist for the same university. Duplicates based on **DENOMINAZ**, **FACOLTA**, and **geometry** were removed, ensuring that only unique entries remain for each university and faculty combination at a given location. This operation maintains the integrity of the data while preserving essential details for further analysis.

3.2.5. Schools

For schools, two datasets were combined, one for public schools and one for private schools. Duplicates based on **geometry** were not removed, as each institution could offer different services or programs. Before concatenation, relevant columns (e.g., name, address, educational level) were retained, and the **DESCRIZIONECARATTERISTICASCUOLA** column was added to the private schools dataset with the value *PARITARIA*. The column order was aligned to ensure consistency. The datasets were then merged using *pd.concat()*, resulting in a single comprehensive GeoDataFrame.

3.2.6. Transportation Systems:

Three datasets: train stations, bus stops, and metro stops were cleaned and concatenated:

- For the train stations dataset, the **Stazione** column was renamed to **Nome**, and a new column, **Mezzo**, was added with the value *Treno*.
- For the metro stops dataset, **nome** and **linee** were renamed to **Nome** and **Linee**, respectively, and **Mezzo** was populated with *Metro*.
- For bus stops, **ubicazione** was renamed to **Nome**, and the **Mezzo** column was set to *Bus*.

After aligning columns (**Nome**, **Linee**, **Mezzo**, **geometry**, and **Neighborhood**) across all datasets, they were merged into a single GeoDataFrame using *pd.concat()*.

3.2.7. Other Datasets

For supermarkets, coworking spaces, libraries, and pharmacies, no duplicated records were identified, so no additional cleaning was required.

3.3. Home Prices

For real estate price data, since they are available only in csv format, additional preprocessing steps were necessary before performing the spatial join. In particular, the dataset in csv format was filtered to retain only records for the year 2024. Additionally, rows were further narrowed down to include only specific property categories: *Abitazioni civili*, *Abitazioni di tipo economico*, *Abitazioni signorili* and *Ville e Villini*. For these filtered records, an additional column was computed, representing the average purchase price as the mean of the minimum (**Compr_min**) and maximum (**Compr_max**) values provided. Then a left outer join was performed on the variable **Zona** between this filtered dataset and the one containing geospatial data related to real estate zoning and boundaries.

Only after these steps, the spatial join was performed, with the difference that the *predicate* parameter was set to *intersects*, ensuring that a match occurs if the geometries from one dataset overlap with, or intersect the geometries in the other dataset.

After performing the spatial join between real estate zone data and neighborhood boundaries, the next step involved aggregating the combined data to summarize property prices at the neighborhood level. More precisely, the minimum purchase price (**Compr_min**) was determined as the lowest recorded property price within the neighborhood. Similarly, the maximum purchase price (**Compr_max**) represented the highest property price in that area. The average purchase price (**Compr_mean**) was also calculated, providing a central value for comparison across the neighborhood. Additionally, the geometry of each neighborhood was preserved by taking the representative geometry from the first record in each group, ensuring that the spatial boundaries were retained for further analysis.

3.4. Missing Values

Further quality checks were conducted to assess the completeness of the data, which represents the coverage with which a phenomenon is represented within a dataset. It was confirmed that most of the acquired datasets had no missing values, ensuring high data quality. However, there were notable exceptions. Specifically, the dataset containing information on restaurants, obtained via an API, revealed, out of 2,597 entries, 1,031 missing observations for the **Price** variable, 6 for **Categories** and 1 for **Business Address**.

Despite the high number of missing values, it was decided not to remove these observations to retain information on restaurants that still provided other valuable attributes, such as **Average Star Rating** and **Review Count**. Additionally, no imputation was performed using the mode of the price variable, expressed as euro symbols indicating price tiers (like €, €€, €€€), because this approach could have introduced bias. Imputation by mode would artificially over-represent the most common price tier, potentially distorting the diversity of the dataset and leading to inaccurate analyses.

The dataset that includes information about nightlife was observed to contain 2 missing observations for the variable **Categories**, while the museums dataset has only 1 missing value for the same attribute.

For the final dataset containing housing prices by neighborhood, it was observed that the neighborhoods *Chiaravalle*, *Quintosole*, and *Ronchetto delle Rane* had missing values for the variables related to housing prices, namely **Compr_min**, **Compr_max**, and **Compr_avg**. Since these variables are essential for subsequent analyses to enable meaningful comparisons across all neighborhoods, it was decided to impute the missing values using the global averages of each variable calculated from the available data.

For the dataset on coworking spaces, it was identified that four records had missing values for the variable **Orario di apertura**. It was decided not to remove these observations, as the information about the presence of coworking spaces in the respective neighborhoods is crucial and cannot be disregarded. Imputation was not performed because there is no reliable way to estimate the opening hours, and introducing potentially inaccurate data could compromise the integrity of the dataset.

For the dataset on supermarkets, there were four observations where the name of the supermarket was missing. These missing values were not removed because, within the context

of the project, the focus was on the presence of supermarkets in each neighborhood rather than their specific names. Removing these observations would have reduced the dataset's coverage of supermarket availability, potentially skewing the representation of neighborhood amenities.

4. Data Storage

To efficiently manage and analyze the large and complex geospatial datasets used in this research, MongoDB was selected as the database solution. This database was chosen due to its flexible and schema-less structure, which is particularly suited for handling complex, nested data such as the geographical and categorical attributes of neighborhoods. Its support for storing GeoJSON objects and performing geospatial queries enables efficient integration of spatial data, simplifying the management of diverse datasets, including points of interest (POIs), home prices, and transport lines. This capability makes it an ideal choice for evaluating the best neighborhoods in Milan based on the preferences of each individual.

The workflow began with connecting to a local MongoDB instance using the *pymongo* library. After setting up the connection, the various GeoJSON files obtained after the cleaning phase were loaded into Python using the *geopandas* library. To store the data, a database, called *ranking_milano*, and a collection, called *neighborhoods*, were created.

For each neighborhood in our study area, a base document was created. Each neighborhood document was initialized with key fields such as **geometry**, a nested **locations** dictionary for categorizing POIs, and a **home_prices** dictionary for property price statistics. A helper function named *append_to_neighborhoods* was used to iteratively map and append data from the POI GeoDataFrames to the corresponding fields of each neighborhood document. Field mappings ensured that all the relevant attributes from each dataset were aligned with the document schema. In addition to the POI data, the property price data, such as minimum, maximum, and average property prices, was integrated into the neighborhood documents. The resulting data structure captures a multidimensional view of each neighborhood, enabling efficient querying and analysis. Finally, the prepared neighborhood documents were inserted into the MongoDB collection using the *insert_many* method. This approach ensured that the data is well-organized, extensible, and ready for geospatial and categorical queries to evaluate the suitability of neighborhoods for different types of residents.

5. Queries

After having stored the data using MongoDB, it was possible to start our anlysis. To explore the code written for each of query, please visit our Operational Guide.

5.1. Most Diverse Neighborhoods

The first query performed aimed to identify the top five neighborhoods in Milan that offer the most diverse range of amenities. It works by calculating a diversity score for each neighborhood, which is based on the number of distinct amenities available. The diversity score is determined by analyzing the available categories of amenities within each neighborhood and counting the non-empty categories.

After the diversity score is computed, the neighborhoods are sorted in descending order, ensuring that those with the highest scores appear first. The query then limits the results to the top five neighborhoods with the highest diversity. This analysis is particularly useful for individuals seeking a neighborhood that caters to a wide range of needs and preferences, as it helps identify areas that stand out for their rich and varied amenities.

Rank	Neighborhood	Diversity score
1	Guastalla	13
2	Bovisa	13
3	Buenos Aires-Venezia	12
4	Città Studi	12
5	Bicocca	12

Table 1. Most Diverse Neighborhoods

5.2. Demographic-Based Neighborhood Scoring

The second analysis performed systematically evaluates Milan's neighborhoods based on their suitability for three demographic categories: students, families, and singles/couples.

The scoring process leverages a function that computes a neighborhood's score by evaluating its amenities and housing prices. Points of interest in the database are grouped by category, such as restaurants, libraries, universities, and normalized based on global minimum and maximum counts for each category. Categories that hold particular importance for each demographic are given higher weights in the scoring formula. For example, students benefit more from universities, coworking spaces, and public transport, while families prioritize schools, playgrounds, and parks. For distinct categories like universities and sports venues, duplicate entries are filtered to ensure accuracy in the count.

Housing prices are incorporated into the scoring system using normalization, where neighborhoods with higher-than-average prices receive a penalty proportional to their deviation from the global price range. This helps balance affordability with the availability of amenities.

The results for each demographic group are ranked in descending order of score, and the scores are scaled into percentages for intuitive interpretation. The top five neighborhoods for each category are then displayed, highlighting areas best suited to the preferences of students, families, and singles or couples.

The results for the three demographic groups considered are displayed in Table 2, Table 3 and Table 4.

Rank	Neighborhood	Score (%)
1	Buenos Aires - Venezia	100.00
2	Città Studi	69.46
3	Niguarda - Cà Granda	65.09
4	Duomo	64.24
5	Villapizzone	64.00

Table 2. Ranking for Students

Rank	Neighborhood	Score (%)
1	Buenos Aires - Venezia	100.00
2	Duomo	71.11
3	Città Studi	57.33
4	Niguarda - Cà Granda	54.22
5	Villapizzone	52.31

Table 3. Ranking for Singles and Couples

Rank	Neighborhood	Score (%)
1	Niguarda - Cà Granda	100.00
2	Buenos Aires - Venezia	98.89
3	Villapizzone	81.92
4	Stadera	81.13
5	Gallaratese	72.32

Table 4. Ranking for Families

5.2.1. Choropleth Map for Student Neighborhood Scores

Subsequently, a choropleth map (Figure 1) was generated using the Python *folium* library to visualize the suitability of different neighborhoods for students, based on the calculated score for each block.

The scores were normalized into percentages to facilitate comparison across neighborhoods. The YlOrRd color scale was applied, with the intensity of the color increasing as the suitability score rises, indicating that darker red shades represent neighborhoods more suitable for students.

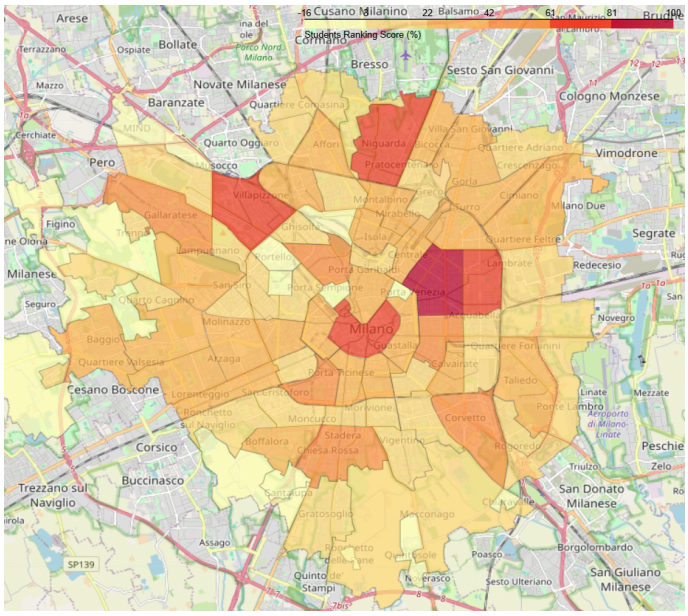

Figure 1. Choropleth Map for Students Scores

5.3. Neighborhood Matching Based on User Preferences

The third query is designed to gather user preferences interactively and then find the three neighborhoods that best match those preferences. The user provides details about various requirements, including faculty preferences, such as which university faculty they are interested in, the presence of specific amenities, like dog parks, libraries, coworking spaces, sport venues, supermarkets, museums, pharmacies, playgrounds, and transportation needs (metro, train, or bus services). Additionally, the user specifies the type of restaurant they prefer and the minimum number of restaurants they desire in the neighborhood. The user is also asked to provide a budget that represents the maximum price they are willing to pay per square meter for a home.

Based on these inputs, the query dynamically builds an aggregation pipeline that filters neighborhoods based on the specified preferences. The pipeline checks for the presence of each required feature, assigns a match score for each neighborhood, and sorts the neighborhoods first by match score, where higher scores represent a better fit, and then by average home price, favoring neighborhoods with lower prices.

The result is a list of neighborhoods that best meet the user's needs, with details about the match score, average price, and a breakdown of which requirements were met and which were not. If any of the user-specified conditions are not fulfilled by a neighborhood, they are flagged accordingly.

To illustrate how the query works, an example scenario was created. In this case, the user is a Physics student planning to move to Milan. Given this, the user specified a set of preferences to find the most suitable neighborhoods. The user is interested in finding a neighborhood close to universities with a Physics faculty. The presence of a library is important to facilitate study and research, and the user would also like to have access to coworking spaces for collaborative work. In terms of leisure and fitness, the user prefers a neighborhood with at least six restaurants with *seafood* as preferred type, while also requiring a sport venue with tennis facilities. The user needs a supermarket and a pharmacy nearby but does not prioritize dog parks, museums, or playgrounds. For transport, the user requests metro and bus services to ensure easy mobility around the city. Lastly, the user has set a budget of €3500 per square meter for housing, meaning they are looking for a neighborhood with average home prices that align with this budget.

The following table shows the result of the query with the above specified preferences.

Rank	Neighborhood	Not Fulfilled Categories	Score
1	Città Studi	Budget	10
2	Stadera	Faculty, Restaurant Cat.	9
3	Quarto Cagnino	Faculty, Library	9

Table 5. Top 3 Most Suitable Neighborhoods

6. Conclusions and Future Developments

This project aimed to explore and rank neighborhoods in Milan based on various factors, including the diversity of amenities, suitability for different demographic groups, and user-specific preferences. The analysis leveraged data on public amenities, housing prices, and neighborhood characteristics, processed through a series of geospatial and statistical techniques. The results of the analyses, including diversity scores and demographic-based rankings, offer a valuable starting point for individuals seeking to move to Milan or assess the city's various neighborhoods for specific needs.

One key suggestion for future development would be to incorporate data on crime rates for each neighborhood. Crime is a crucial factor influencing people's decisions about where

to live and it could help provide a more comprehensive picture of the safety and overall livability of each neighborhood.

Regarding the query that evaluates neighborhoods based on their suitability for students, families, and singles/couples, further refinement could be made by improving calculation of the score. This could be done by adding more factors that contribute negatively to the score, for example crime rates, pollution levels and overall cleanliness.

Another important aspect are the values of the weights. These could be better chosen by incorporating knowledge from an expert of the field or gathering data from public surveys, as a way to obtain more realistic weights to assign to each type of amenity.

For instance, if a ranking score provided by citizens for each neighborhood was made available, a case study could be conducted to explore which amenities, or combinations of amenities, contribute most or least to the overall neighborhood rating. This approach could help build a model that processes data about points of interest (POIs) and returns a score for each neighborhood as output.

Such a solution would greatly benefit people searching for a home by providing an impartial ranking of different neighborhoods. Furthermore, this model could also be scaled to work for other cities, however, it should be noted that cultural differences between cities could affect how a neighborhood is perceived.

For the query that provides personalized neighborhood suggestions based on user preferences, further development could focus on making the questionnaire more in-depth and user specific. For instance, the first question a user should be asked is in what category they fall in. These categories could be age groups, or as we used: students, single/couples and families. Then depending on the response, the following questions should follow a predetermined track, for example asking for specific preferences regarding restaurant reviews, such as whether high reviews are essential, or combinations of amenities, such as how much weight they place on having more restaurants compared to a nearby library. This personalized approach could increase the precision of the suggestions by allowing users to prioritize some attributes over others, thereby offering a more precise match for their ideal neighborhood. However, as for the previous case, more specific questions would require expert knowledge or extensive data gathered from public surveys.

Project Components

This section outlines the components of the project. These include:

- `OperationalGuide.pdf`, which is the operational guide for the reproducibility of the project;
- `MongoDB Integration and Queries.pdf`, which contains the specific queries executed for the project, providing further support to the operational guide.
- `geojson_files` folder, which contains all the data files obtained through APIs, queries on Overpass Turbo, and downloaded data from the Open Data portal of the Municipality of Milan;
- `PolyDatasets` folder, which contains the final datasets, where the spatial join and data preparation operations have already been performed;
- `Notebooks` folder, which contains all the notebooks required for the development of the project. Specifically, there is a separate notebook for each dataset obtained through APIs, while all operations performed on the downloaded data and the supermarket dataset obtained from Overpass Turbo are contained in the notebook called `DatasetMilano.ipynb`. The operations performed on the downloaded datasets containing information about home prices are described in the `PolyHomePrices.ipynb` notebook. The `DataManPolygons.ipynb` contains the query used to obtain the polygons of Milan. Lastly, the `IntegrationMongoDB.ipynb` notebook contains the code written to store the data in MongoDB and the code for the queries executed;

References

[1] Agenzia Entrate - OMI, "Quotazioni Immobiliari OMI: Compravendita e Locazione", [Online]. Available: https://dati.comune.milano.it/dataset/ds1996-quotazioni-immobiliari - omi - compravendita - e - locazione - riepilogo.

[2] Agenzia Entrate - OMI, "Zone e Perimetri OMI: 2024/1", [Online]. Available: https://dati.comune.milano.it/dataset/ds2831-zone-e-perimetri-omi-2024-1.

[3] Comune di Milano, "ATM - Fermate linee di superficie urbane", [Online]. Available: https://dati.comune.milano . it / dataset / ac494f5d - acd3 - 4fd3 - 8cfc - ed24f5c3d923.

[4] Comune di Milano, "ATM - Fermate linee metropolitane", [Online]. Available: https://dati.comune.milano.it/dataset/b7344a8f-0ef5-424b-a902-f7f06e32dd67.

[5] Comune di Milano, "Coworking nel Comune di Milano", [Online]. Available: https://dati.comune.milano.it/dataset/34b330d2-e34a-4d98-a327-447342001ee3.

[6] Comune di Milano, "Elenco e localizzazione degli edifici scolastici attivi - Scuole Statali -", [Online]. Available: https://dati.comune.milano.it/dataset/ds1863-elenco-e-localizzazione-degli-edifici-scolastici-attivi-scuole-statali-as-2020-2021.

[7] Comune di Milano, "Farmacie nel Comune di Milano", [Online]. Available: https://dati.comune.milano.it/dataset/ds501_farmacie-nel-comune-di-milano.

[8] Comune di Milano, "Istruzione: localizzazione delle sedi universitarie degli atenei milanesi", [Online]. Available: https://dati.comune.milano.it/dataset/ds94-infogeo-atenei-sedi-localizzazione.

[9] Comune di Milano, "Istruzione: Scuole Paritarie anagrafica", [Online]. Available: https://dati.comune.milano.it/dataset/ds1582-istruzione-scuole-paritarie-anagrafica-2020-2021.

[10] Comune di Milano, "Sistema Bibliotecario di Milano", [Online]. Available: https://dati.comune.milano.it/dataset/ee08abe0-aba1-44ab-b6ad-21fcd8a45100.

[11] Comune di Milano, "Sport: localizzazione degli impianti sportivi", [Online]. Available: https://dati.comune.milano.it/dataset/ds34_infogeo_impianti_sportivi_localizzazione_.

[12] Comune di Milano, "Territorio: localizzazione delle aree cani", [Online]. Available: https://dati.comune.milano.it/dataset/ds52_infogeo_aree_cani_localizzazione.

[13] Comune di Milano, "Territorio: localizzazione delle aree gioco", [Online]. Available: https://dati.comune.milano.it/dataset/ds724_infogeo_aree_gioco_localizzazione.

[14] Comune di Milano, "Trasporto pubblico: localizzazione delle stazioni ferroviarie", [Online]. Available: https://dati.comune.milano.it/dataset/802a1d1f-4203-44c9-b9f5-dd8aa1f6bc3c.

[15] Edoardo Olivieri, Federica Romano and Francesca Verna, "Analysis and Ranking of Milan's Neighborhoods-Operational Guide", [Online]. Available: https://github.com/edoardo-olivieri/DataManagement/blob/main/OperationalGuide.pdf.

[16] Edoardo Olivieri, Federica Romano and Francesca Verna, "MongoDB Integration and Queries.pdf", [Online]. Available: https://github.com/edoardo-olivieri/DataManagement/blob/main/MongoDB%20Integration%20and%20Queries.pdf.

[17] OpenStreetMap contributors, "Overpass Turbo, OpenStreetMap API", [Online]. Available: https://overpass-turbo.eu/.

[18] Yelp, "Yelp Fusion API", [Online]. Available: https://docs.developer.yelp.com/docs/getting-started.