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*

Abstract

This project investigates the quality of life across Milan's neighborhoods, aiming to identify the most suitable areas for three distinct groups: students, families, and singles or couples without children. By integrating data from multiple sources, such as the Yelp API and official municipal datasets, we built a comprehensive database capturing various aspects of urban life. These include amenities such as restaurants, schools, universities, supermarkets, public transport, and parks. The data was carefully cleaned, processed, and organized into a MongoDB database with an embedded document structure. This report outlines the steps taken across the entire data management pipeline, with a focus on data acquisition, spatial integration, and quality assessment, to provide actionable insights into urban living in Milan.

Keywords: *Milan, Neighborhoods, Data Acquisition, Data Quality, MongoDB*

1. Introduction

Quality of life in a city is a multifaceted concept, varying significantly based on individual needs and preferences. Students may prioritize access to affordable dining options, coworking spaces, and public transport, while families often value proximity to schools, parks, and essential services. Singles or couples without children may be drawn to vibrant nightlife, cultural activities, and fitness facilities. Milan, as a dynamic and diverse city, offers a rich array of amenities, but these are unevenly distributed across its neighborhoods.

This study seeks to explore how Milan's neighborhoods align with the needs of these demographic groups by analyzing key urban point of interests and infrastructures, as well as taking into account the prices of the houses. By leveraging modern data acquisition techniques and spatial analysis, the project not only provides a snapshot of Milan's urban landscape but also establishes a framework for evaluating urban quality of life. The findings can guide individuals in choosing the best neighborhood to suit their lifestyle and help policymakers address areas needing improvement.

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 and official datasets, carefully processing the data to ensure consistency and spatial alignment.

2.1. Defining Neighborhood Boundaries

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.

2.2. Data Acquisition via APIs

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 OverpassTurbo, 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 (up to the API's maximum of 48) and pagination via an offset parameter. To ensure comprehensive coverage, up to 240 results per location were retrieved, depending on availability.

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, 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 uninterrupted 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 go to the Section boh. 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 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 Processing

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 OverpassTurbo, 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.

Here below is a more in detail description of how we dealt with the different datasets.

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.

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_22 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.
- 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.

3.2.4. Universities

In the university dataset, duplicates were 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 to ensure each faculty of a university appeared only once.

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.

4. Data Integration and 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 for students, families, and couples/singles.

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 *my_database*, 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

■ References

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