UPC - GIA

Project 2:
Semantic Data Management for AI

Advanced Databases

Autors:
Elena Alegret & Sergi Tomàs & Júlia Orteu

June 2024

Contents

Abstract

This project outlines the implementation of best practices in data science pipelines, focusing on Data Ingestion, Data Engineering Pipeline, and Data Analysis Pipelines. Key elements include Data Sources, Data Collectors, and separate zones to ensure optimal data flow. Utilizing knowledge graphs created from tabular databases sourced from platforms such as *Airbnb*, *Mossos d'Esquadra*, and *TripAdvisor*, the project develops a tourist visualization tool for Barcelona using RDF. Additionally, a machine learning model is introduced to help property owners predict optimal rental prices based on property characteristics.

To access the interactive interface integrated into your web browser, visit the BCN Map4Tourism interface.

Key Words: Data engineering, data pipelines, organizational structure, data ingestion, data analysis, knowledge graphs, tourism visualization, Barcelona

1 Introduction

Barcelona, renowned for its culture, has long been a magnet for tourists from around the globe. However, beneath its well-known facade lies a growing concern: an increase in crime rates that has earned it the reputation of being one of the most dangerous cities in Spain. Despite this concern, tourism in Barcelona continues to grow, raising questions about the potential relation between these two phenomena.

In response to this belief, our project aims to explore the potential correlation between crime rates and tourism. We intent to construct robust pipelines for data collection, storage, and analysis, with the goal of developing a tool that enables users to visualize the impact and distribution of various crime rates on local businesses and tourist attractions.

Drawing from a diverse array of data sources, including official crime statistics, tourism metrics, and business records, we seek to understand how these two factors could affect the tourist experience and its decrease. 0ur hypothesis postulates that the tourism industry may inadvertently foster criminal activities, thereby deteriorating the crime rate and potentially jeopardizing the safety and well-being of both residents and visitors alike.

Furthermore, this project incorporates a machine learning model designed to assist property owners in predicting the optimal rental price based on the attributes of their properties. By leveraging knowledge graphs, we enhance data integration, semantic richness, and query capabilities, ensuring a seamless user experience.

This project makes use of three official data sources, detailed in the section 2, Data Ingestion, to explore the correlation between crime rates and tourism in Barcelona.

This project utilizes three official data sources, detailed in section 2, Data Ingestion, to explore the correlation between crime rates and tourism in Barcelona. The resulting tool offers multifaceted functionalities, providing insights into local crime rates and enabling users to explore nearby restaurants based on their preferences. Additionally, the ML model aids flat owners in determining the optimal rental price, thus has a positive effect in the decision-making process for both tourists and property owners.

2 Data Ingestion

In this section, we present the data sources used and describe the implementation of Data Collectors.

2.1 Data Sources

In this subsection, we dive into the datasets used in this project, providing details of our Crime rates, Airbnb listings, tourist attractions and reviews in Barcelona and the Google Maps API.

1. **Airbnb / OpenDataSoft**: OpenDataSoft extracts data from [1], a platform designed to provide information regarding Airbnb's listings in Barcelona. This dataset includes attributes such as location, price, and property characteristics.

    - **Usage**: This dataset is used to feed our ML model, enabling property owners to predict the optimal rental price based on their apartments' features. Additionally, this dataset is also used in the user-friendly interface to allow tourists to view apartments on the Barcelona's map along with the correlation to other datasets, such as criminal dataset.

2. **Mossos d'esquadra**: The Catalan government offers an open data platform where various crimes committed in Catalonia, including Barcelona, are published alongside their corresponding police departments. This dataset, accessible through [2], provides insights into the types and frequencies of crimes reported in Barcelona.

    - **Usage**: This dataset is analyzed to identify crime patterns in relation to tourist areas and Airbnb's listings. By integrating this data, we aim to provide a clear view of the crime's distribution per neighbourhood, aiding tourist on the decision-making process.

3. **TripAdvisor**: TripAdvisor [3] is platform for users to submit reviews of restaurants, hotels, and various attractions. Leveraging the TripAdvisor API, we extracted data on the 10 nearest points of interest surrounding a subset of 3000 Airbnb apartments in Barcelona. Additionally, we collected up to 10 reviews (if available) for each point of interest.

    - **Usage**: This dataset allows tourists to discover popular nearby sites and read reviews about them.

4. **GoogleMaps API**: Google Maps API is a platform for accessing geolocation data and related information. By integrating Google Maps API data into our project, we gained spatial context, allowing for a more comprehensive understanding of the geographic landscape under examination.

    - **Usage**: This dataset correlates the latitude and longitude of Airbnb listings with their respective neighborhoods. This helps locating the properties and facilitates the correlation of neighborhoods with other datasets, such as crime statistics and tourist attractions.

These datasets collectively enable us to achieve the two primary objectives of this project: to explore the relationship between crime rates and tourism in Barcelona, and to provide property owners a tool for predicting optimal rental prices. Detailed metadata for each dataset can be found in the *Data Collectors* directory of the deliverable.

2.2 Data Collectors

In this section, we detail the process of data collection and storage for our project.

All datasets were obtained through API (Application Programming Interface) calls, allowing us to retrieve the necessary information programmatically.

> **Reasoning Behind the Decision:**
> APIs provide structured access to data, ensuring seamless integration across different systems and platforms. They offer real-time or near-real-time data, ensuring the information obtained is up-to-date and accurate. Additionally, they facilitate controlled access to data, allowing organizations to enforce security measures and regulate data usage.

Following, the data was stored in a **.parquet format** in the Data Lake. Each dataset is stored as a separate *.parquet* file in the Data Lake. This approach ensures data integrity and facilitates seamless data access and processing.

> **Reasoning Behind the Decision:**
> Parquet is a columnar storage format that is highly optimized for query performance and efficient storage. It offers benefits such as compression, efficient encoding, and support for complex nested data structures.

Each dataset is stored separately as individual *.parquet* files. This practice is common in data engineering for several reasons:

> **Reasoning Behind the Decision:** Storing the datasets separately, we ensure that each dataset can be accessed, processed, and updated independently. This modular approach facilitates data management and enhances the overall performance and scalability of the data processing pipeline. This separation is important as it allows for the integration and transformation of data in subsequent stages, where datasets are combined and analyzed together.

3 Data Engineering Pipeline

The Data Engineering Pipeline encompasses three key components: Data Formatting, Data Quality and Data Preparation Pipeline.

3.1 Data Formatting Pipeline

In this section, we detail the process of the Data Formatting Pipeline, where data is homogenized according to a canonical data model.

Firstly, a connection with DuckDB, a file-based database, has been established to enable the writing of formatted data. This has been done to ensure that the data is available for further analysis.

Next, a Spark session has been initialized to process the data. It has been configured to use the DuckDB JDBC library, allowing connection to the DuckDB database and writing in the formatted data.

Subsequently, the *.parquet* files containing the raw data from Airbnb, the criminal dataset, and TripAdvisor locations and reviews have been loaded into Spark DataFrames. It is worth mentioning that, due to issues with the Airbnb dataset, preliminary preprocessing has been performed. This includes converting array-type columns to comma-separated strings and removing those that cannot be preprocessed (Spark does not work with array-type columns). This decision has been made to ensure that the data is in a suitable format for further analysis.

Finally, the Spark DataFrames have been written to tables in the same DuckDB database, *barcelona.db*. This allows to homogenized data to an appropriate format. It should be noted that there is one table per dataset.

> **Reasoning Behind the Decision:**
> DuckDB is a fast and efficient in-memory database, enabling quick processing of large datasets. By storing Spark tables in DuckDB, query speed is increase and latency is reduced compared to other storage options. Additionally, DuckDB provides a SQL interface, making data access and manipulation easier for future analyses. This offers flexibility and scalability, allowing quick access to stored data and facilitating integration with other systems.

3.2 Data Quality Pipeline

This section delves into the Data Quality Pipeline, highlighting the tasks and techniques employed to ensure data integrity and accuracy.

The three main tasks related to data quality are:

- Identification of Data Quality rules on the datasets: Identify potential data quality issues; errors, inconsistencies, outliers, etc.
- Assessment of the Quality of the Data: Evaluation of the data quality to determine its accuracy, completeness, consistency, and timeliness.
- Application of Data Cleaning processes: Address the issues identified during the data quality assessment; removing duplicate values, correcting formatting errors (column name errors), imputing missing values, standardizing data, etc.

Due to our lack of expert knowledge on the selected fields (criminal and tourism), the preprocessing will be mainly restricted to missing values imputation or the removal of incorrect variables.

It is worth highlighting that the decision to correct the formatting errors in the column names is to facilitate the search of relationships between tables in the *Data Explotation Pipeline*. For instance, in order to be able to implement the *Data Analysis Pipeline*, we have seen the need to introduce the *GoogleMaps API*, where we converted the adresses collected into Geographical coordinates.

GoogleMaps API was applied on the *Data Quality Pipeline* for reasons associated to the limit of tokens available for API requests.

The details of the quality filtering implementation can be found in the *DataQualityX.ipynb* files located in the *data_ quality_ pipeline* directory within the deliverable, where X represents Criminal, Airbnb, or Tripadvisor. We have chosen to separate the preprocessing into three distinct notebooks to ensure a more structured and organized workflow. Nonetheless, for a comprehensive view, the entire preprocessing procedure is consolidated in the *DataQualityPipeline.ipynb* file.

Please note that changes are applied directly to the database rather than being defined as constraints within each table.

> **Reasoning Behind the Decision:**
> As our data ingestion relies on diverse APIs, expanding the database with additional rows implies rerunning the entire system and pipelines. Consequently, imposing constraints on the database may seem less pertinent. Instead, we use the insights gained from dataset exploration to dynamically process our data.

To provide an example of one of the filters applied in the form of a Denial Constraint, we present the following expression:

barcelona_neighborhoods = [Eixample, Sants-Montjuïc, Les Corts, Horta-Guinardó, Sant Martí, Nou Barris, Sarrià-Sant Gervasi, Gràcia, Sant Andreu, Cuitat Vella]

$$\forall t \in \mathrm{R} \neg(\mathrm{t.area\_basica\_policial} \notin \mathrm{barcelona\_neighborhoods})$$

- Ensure only data registered in Barcelona is considered. Since police areas are separated by city district, this has to be assessed using police district rather than the city name.

All operations are implemented using Spark and all changes are stored in the Trusted Zone, the tables are the same as the Formatted Zone even though the reliability of the data has been assessed. The final relational database looks like the following:

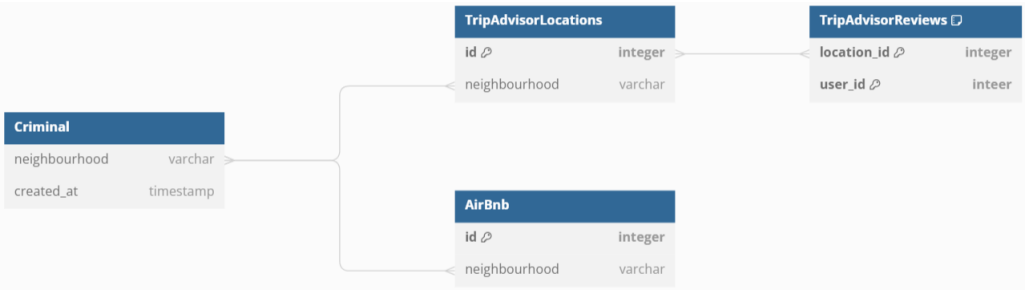

Figure 1: A representation of the variables' role in each dataset.
Note: Not all columns of each table are being represented, only the ones which state as Primary or Foreign Keys.

3.3 Data Preparation Pipeline

In this section, we integrate the datasets located in the Trusted Zone in order to ensure their preparation for the Data Analysis Pipelines. It aims to provide an in-depth analysis of the dataset. It identifies and selects columns that are most relevant for the analysis, discarding those that do not contribute to the visualization or modeling goals.

Firstly, the datasets - Airbnb, TripAdvisor, Criminality - are loaded from the Trusted Zone using a combination of JDBC with DuckDB to ensure that the loaded data is the most current.

> **Important to Remeber:**
> The Trusted Zone stores a DuckDB database, a columnar database management system used for efficient querying and data manipulation. Additionally, Apache Spark is utilized to handle large-scale data processing, providing the computational power needed to manage and analyze extensive datasets.

This pipeline is divided into three main sections, each corresponding to a table in the DuckDB file. Each column is evaluated for its relevance to our analysis and modeling tasks. This examination ensures that only the most significant variables are retained, thereby facilitating the transformation to the Knowledge Graph. Morevoer, is it wirth mentioning that the relationship between datasets is based on common keys or structures such as geographical locations.

- **Criminal Dataset**: This dataset includes a variety of columns that provide information about criminal activities. For the user interface, we only need the following columns:
  - num_mes, regio_policial (Neighborhood), ambit_fet (Crime type), nombre de victimes.
- **Airbnb Dataset**: This dataset includes a variety of columns that provide information about Airbnb listings, such as location, price, and property characteristics.
- **TripAdvisor Dataset**: This dataset includes a variety of columns that provide information about tourist attractions, restaurants, and hotels, along with user reviews.

Once this is completed, we store the data into the *data_explotation* directory. Subsequently, as we aim to work with a knowledge graph, we will need to follow these steps:

1. **Construct the RDF and Store the TTL File**
2. **Create Triples**

Reasoning Behind the Decision:
During the Data Quality phase, the primary focus is on ensuring the accuracy, completeness, and consistency of the data. This involves identifying and correcting errors, handling missing values, and ensuring that the data conforms to expected formats and ranges.

In contrast, the Dataset Exploration phase is more analytical. Here, the goal is to enhance the dataset's relevance and utility for specific analysis and modeling tasks. By selecting columns at this stage, we can focus on the most informative and pertinent variables, ensuring that our models and visualizations are built on the most relevant data.

Reasoning Behind the Decision:
Utilizing knowledge graphs enhances data integration and semantic richness, allowing us to unify datasets and represent complex relationships clearly. This approach supports advanced queries and analysis. It ensures flexibility, scalability, and interoperability, making our data analysis more efficient and effective.

3.3.1 Construct the RDF Schema and Instance Generation

In the RDF schema creation phase, we used DuckDB and SparkSQL to define the schema and create instances from our datasets. This process involves several steps to ensure that our data is accurately represented and integrated into the knowledge graph.

The schema definition involves specifying the ontology that defines the structure of our knowledge graph, including the definitions of classes, properties, and relationships between different entities.

The schema is divided into four main elements. The primary class is *Location*, which contains the common attributes - *district, longitude, and latitude* -. These attributes appear in all the duckdb tables and allow us to interlink the entities within the knowledge graph. Within this main class, we defined two subclasses which help categorize the different types of locations:

- *Entertainment*: Includes attributes specific to entertainment venues - restaurant and reviews of sightseeings.
- *Apartment*: Includes the attributes relevant to apartments, - price, room_type, etc -.

Additionally, we defined the *Incident* class to represent incidents. This class is linked to the *Location* class through the shared attribute *district* based on the *isinDistrict* relation. This linkage allows us to associate incidents with specific neighbourhood.

Reasoning Behind the Decision:
By dividing the schema into main elements and subclasses, we ensure that our data is organized in a logical and semantically meaningful way. This method not only enhances the semantic richness of our data but also ensures that our knowledge graph is built on a solid foundation, ready for complex analysis and querying.

To create instances, we utilized functions defined in the *utils.py* script. The *create_instance* function converts rows from our tables into RDF instances by mapping data from each row to the corresponding RDF class. For example, each row in the *criminal_dataset* table is transformed into an instance of the *Incident* class. Each table has its own instance generation function.

Once the RDF is constructed, the data is exported into a Turtle (.ttl) file located within the *data_explotation* directory.

> **Reasoning Behind the Decision:**
> By automating the process of instance creation using functions, we ensure consistency in our RDF data. This boosts the semantic richness and makes it more interoperable.
>
> The generated RDF data is then exported into a Turtle (.ttl) file, which is a widely used, compact, and readable serialization format for RDF. It is designed to be easily understood by humans while still being parsed for machines. Turtle's syntax is less verbose compared to other RDF formats - RDF/XML -, which simplifies the process of writing and debugging RDF data. Moreover, it supports a variety of data types and structures, helping to represent complex datasets.

3.3.2 Create Triples:

Generates subject-predicate-object triples to represent data relationships.

3.4 Model Oriented Explotation

We built two different graphs in order to complete both analytic procedures. The one previously mentioned, which comprised all of the accessible information, and another to work on the modeling procedure that will come next. This second graph lacks information such as restaurant and attraction names, as well as the publication title of the AirBNB listings. This is done to help the model distinguish between useful and useless information. Furthermore, we discretized the variable that will be our aim, namely apartment pricing.

> **Reasoning Behind the Decision:**
> Initially, this was done in order to make direct link prediction on the graph, it was the solution to predict the price without having to make use of a regression model, which is not possible on the graph using the desired methods. Although our final model is not making this predictions and a regressor could have been implemented, we keeped this transformation to facilitate the model task.

The price was discretized in the following bins:

- **(0) Low Price**: 0 to 50 €/night
- **(1) Medium-Low Price**: 50 to 150 €/night
- **(2) Medium-High Price**: 150 to 250 €/night
- **(3) High Price**: 250+ €/night

The ranges were decided in exploration and analysis to maximize the significance on each group of apartments.

4 Data Analysis pipelines

4.1 The Data Analysis pipeline using SPARQL

In this part, we present our first data analysis; where we explain the implementation and the platform designed to offer users a more easy exploration of Barcelona's diverse neighborhoods.

We integrate the datasets located in the Trusted Zone to ensure their preparation for the Data Analysis Pipeline 1 and create our application.

Migration to SPARQL: This application was originally developed using PySpark for data processing and filtering. In the new version, we have migrated the filtering and querying operations to SPARQL to leverage a graph-based RDF approach.

The pipeline is structured into functions that apply tasks to consolidate the datasets. It is important to note that the exploration zone only prepares data for analysis. The preprocessing of the dataset is done in the Data Quality Pipeline, see Section 3.2.

> **Reasoning Behind the Decision:**
> Migrating to SPARQL allows us to take advantage of RDF's flexible data modeling capabilities, enabling more efficient and scalable data integration and querying. This enhances the application's ability to handle complex relationships and provides a more robust framework for future expansion.

The process begins with loading the RDF graph containing the data into memory. We define the necessary namespaces to structure the data correctly. Next, we utilize SPARQL queries to filter and retrieve the relevant data needed for the application.

Query Functions

- **Filter Apartments**: This function filters apartments based on user-defined criteria such as price, room type, number of bathrooms, and minimum nights. It constructs a SPARQL query dynamically based on the input parameters to retrieve the appropriate data.
- **Filter Locations**: This function filters entertainment locations (like restaurants and attractions) based on criteria such as average rating and selected neighborhoods. Similar to the apartment filter, it constructs a SPARQL query to fetch the required data.
- **Popup Content for Reviews**: This function generates content for map popups, displaying reviews and ratings for selected locations. It executes a SPARQL query to get a review for each location and formats the content accordingly.

4.1.1 Visualization and User Interaction

The BCN Map4Tourism interface serves as a tool for tourists and renters in Barcelona, offering a space for neighborhood exploration, restaurant discovery, and crime rate analysis.

Users are provided by a visual interface where they can select specific neighborhoods to explore, aided by a map visualization with markers representing Airbnb listings and nearby restaurants or attractions.

Through interactive sliders and checkboxes, users can define their preferences, filtering Airbnb listings by review scores, price range, and property features like the number of bathrooms and beds. Additionally, they have the option to display TripAdvisor-rated restaurants and attractions on the map.

The BCN Map4Tourism interface helps users to focus their exploration on their preferences by neighborhood, nearby amenities and attractions.

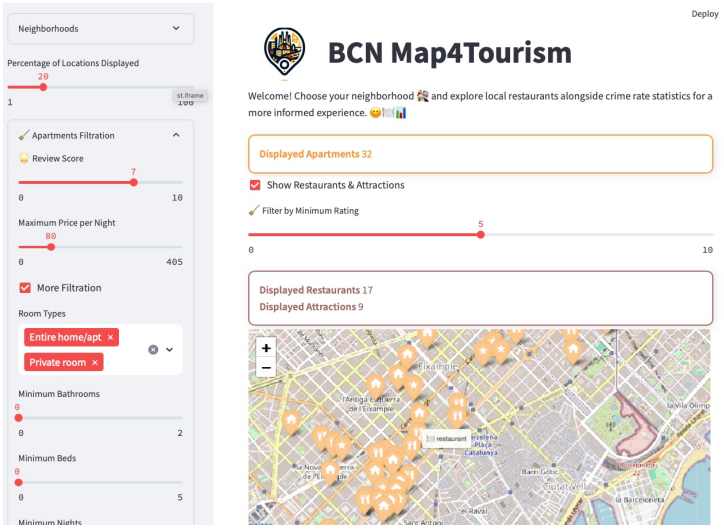

Figure 2: Interface.

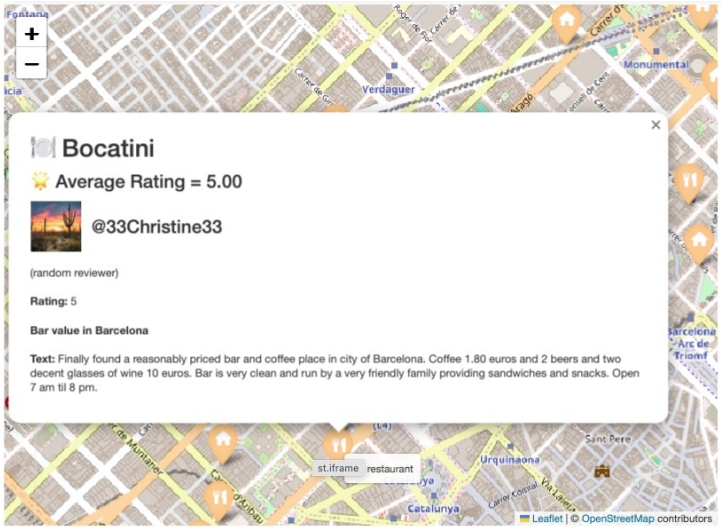

Figure 3: Review selected randomly from the dataset.

All details about the interface implementation can be found in the file *app.py* in the *App-Data_ Analysis_ pipeline1* directory of the deliverable.

4.2 AirBNB price modeling

Taking advantage of the graf properties, we are going to use a embedding based model that can predict the renting price of an apartment in Barcelona. The complete pipelines can be found in the repository in the **Model-Data_Analysis_Pipeline**. This is a summary of the model.

It is decomposed into 2 main models:

- TransE embedder. We trained a TransE projection model to reduce the graph's relations to a 100-dimensional embedding space. The embeddings generated by this model for apartments will be employed in the second face of the modeling.

> **Reasoning Behind the Decision:**
> TransE is a fast model, which we valued, since the computational capacities we have are limited and we need to do a bunch of iterations to try to make the model work as good as possible, this was the option selected. It is likely that some other projection model had worked better.

- **MLP for Classification**: As stated in the Explotation Zone, our target (the Airbnb price) has been binned into 4 diferent targets regarding diferent price ranges. This model will be trained using the supervised pairs embedding(apartment), price_range(apartment). This will be done by using an MLP Classificator.

The data will be splitted just into train and test datasets due to our lack of more data. The only triplets taken from the graph will be the ones regarding the apartment relation with the price.

> **Reasoning Behind the Decision:**
> We also want the embedding model to learn the appropiate embeddings for the test apartments and the information contained in the it's characteristics and location. To predict a real new apartmemnt, it should be included in the graph, to fine-tune the projection model later.

The results of the complete pipeline are the following:

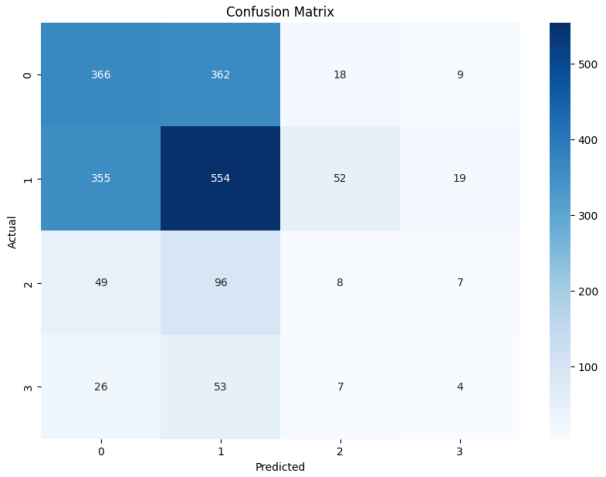

Figure 4: Enter Caption

As it stands, the model is not an excellent estimator. While it performs better than the random baseline (which would yield 25% accuracy), it rarely predicts the two most expensive classes accurately, and when it does, the predictions are generally poor.

Through experimentation, we have determined that these are the best achievable results with the current data before the model begins to overfit. This suboptimal performance could be attributed to several factors, including data unpredictability (though this is unlikely), an insufficient number of data instances, or an inadequate embedding method (TransE), potentially due to a too small embedding dimension.

5 Conclusions

This project has been a great way to understand how more complex data processes work. Designing a full set of pipelines, including the use of knowledge graphs, the implementation of a interface, the use of embedding models and machine learning models, has been both challenging and insightful. Addressing the collection and processing of data is crucial and defiantly not a trivial task. We also have had the opportunity to work with `pyspark` `sparksql` and `sparkql` to virtually (since all is executed in a single machine) parallelize our code and processes.

We leveraged knowledge graphs and TransE models to effectively represent and analyze relationships within our data. Additionally, we integrated machine learning models to predict the target variable. The use of `sparql` enabled us to query our knowledge graphs.

We are proud of the final result, of our BCN Map4Tourism, a very useful tool for minorities to look for they ideal apartment and attractions in Barcelona while being able to develop their awareness around the dangers of the city of Barcelona.
.

References

[1] Airbnb. *Inside Airbnb*. URL: https://insideairbnb.com/ (visited on 04/22/2024).

[2] Mossos d'esquadra. *Criminal Dataset of Barcelona*. URL: https://analisi.transparenciacatalunya.cat/Seguretat/Fets-delictius-i-infraccions-administratives-de-l-/y48r-ae59/about_data (visited on 04/22/2024).

[3] Tripadvisor. *Tripadvisor Dataset*. URL: https://www.tripadvisor.com/ (visited on 04/22/2024).