Big Data Management Delivery 2

Ximena Moure
Ange Xu
ximena.moure@estudiantant.upc.edu
ange.xu@estudiantant.upc.edu
Master of Data Science -BDM

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

June 16, 2023

Contents

1 Introduction

In the second phase of the project, the goal is to perform descriptive and predictive analysis of data related to Barcelona's housing and its relationship with its economy. In other words, we were asked to create the Formatted and Exploitation zones by using Apache Spark.

As for the datasets being worked with, we decided to use the given solution by the professors which involved utilizing .parquet files for Idealista listings and .json files for all the others. For the third dataset, we opted for one from Open Data BCN which contains information about cultural and leisure spaces and facilities found in Barcelona (from now on referred to as cultural dataset). It includes details about libraries, cinemas museums, parks, etc.

Regarding the Landing Zone where we read the formatted zone input data, we decided to upload the lookup-tables, the income dataset, and the cultural dataset to a Mongo database called landing with different collections for the different datasets. As the Idealista files are in .parquet format, we just kept them locally.

2 Formatted Zone

In order to implement the Formatted zone we read all the collections(except those from Idealista) from Mongo (the landing zone) using a Spark-Mongo connector.

During the implementation of this zone, we made a deliberate decision to minimize data transformations. The rationale behind this approach is to avoid limiting the potential analyses that can be performed in the Exploitation zone. By refraining from extensive aggregations, calculations, or column removal, we ensure that we retain all the relevant information that could be valuable for future analysis in the Exploitation zone. We did perform some basic data cleaning but nothing too complex.

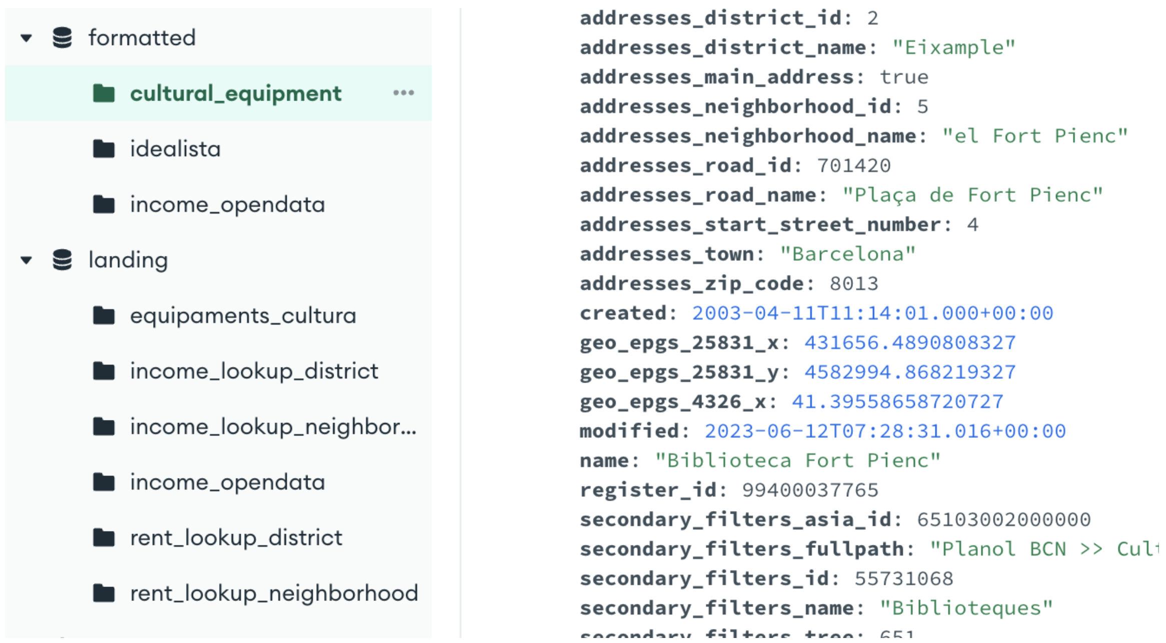
2.1 Data structure

We decided to create one database in Mongo with three different collections, one for each dataset. The reasons for choosing Mongo as the Formatted zone are the following:

- MongoDB's document-oriented storage allows for the storage of semi-unstructured data. This is particularly advantageous when dealing with diverse data sources, where the data may not adhere to a strict schema.
- As data sources evolve over time, the formatted zone may need to accommodate changes in data schemas. MongoDB's flexible schema design allows for easy schema evolution.
- It is designed to be highly scalable, allowing for horizontal scaling across multiple nodes.
- It has robust integration capabilities with various data processing and analysis tools.

As mentioned before we did not remove columns or did any major data cleaning. The reason is that we know that the main goal to have a formatted zone is to homogenize the data sources and given them the same format. Thus, for each data source, we store the same columns as in the Landing zone and we add 4 new ones that are the result of re-conciliating each dataset with the lookup tables. These new columns are: `"neighborhood_id_reconciled"`, `"neighborhood_name_reconciled"`, `"district_id_reconciled"`, `"district_name_reconciled"`.

Having these new variables will make the work on the exploitation zone much easier since the joining of the tables can be done in a simple way. Figure 1 depicts the structure.



```
addresses_district_id: 2
addresses_district_name: "Eixample"
addresses_main_address: true
addresses_neighborhood_id: 5
addresses_neighborhood_name: "el Fort Pienc"
addresses_road_id: 701420
addresses_road_name: "Plaça de Fort Pienc"
addresses_start_street_number: 4
addresses_town: "Barcelona"
addresses_zip_code: 8013
created: 2003-04-11T11:14:01.000+00:00
geo_epgs_25831_x: 431656.4890808327
geo_epgs_25831_y: 4582994.868219327
geo_epgs_4326_x: 41.39558658720727
modified: 2023-06-12T07:28:31.016+00:00
name: "Biblioteca Fort Pienc"
register_id: 99400037765
secondary_filters_asia_id: 65103002000000
secondary_filters_fullpath: "Planol BCN >> Cul
secondary_filters_id: 55731068
secondary_filters_name: "Biblioteques"
secondary_filters_tree: 651
```

Figure 1: Formatted zone database structure

3 Exploitation Zone

3.1 Descriptive Analysis

Four KPI's are calculated using Spark RDD operations. We decided to store the KPI in a Mongo database called exploitation, which contains four different collections(each collection corresponds to a view that contains one KPI). We also analyzed the possibility of using a Relational Database for this zone since the data that we are storing has a well-defined schema and structure. However, in the end, we decided to use Mongo, for many of the reasons described before but we do think that a relational database would be a good option too. Collections in Mongo:

- `first_kpi`: In this collection, each document has the following structure: "neighborhood", "district", "culturalType" and "#culturalPlaces".
- `second_kpi`: In this collection, each document has the following structure: "neighborhood", "year" and "avgPrice".
- `third_kpi`: In this collection, each document has the following structure: "neighborhood", "district", "propertyType" and "#properties".
- `fourth_kpi`: In this collection, each document has the following structure: "neighborhood", "AvgIncome-2007-2017", "AvgPrice-2020-2021'".

We would like to mention that for the KPIs, it was kind of complicated to take the year into account when joining the different datasets according to the year since they do not coincide in any year (one has years ranging from 2007 to 2017, and the other from 2020 to 2021).

3.1.1 KPI1:Cultural facilities by district and neighborhood

As a first KPI we decided to calculate the different types of cultural facilities that a neighborhood has. We also added the possibility to filter by the district. The presence of cultural facilities in a district or neighborhood can enhance the desirability and attractiveness of an area. And this can impact the value of the properties.

3.1.2 KPI2:Average house price per neighborhood by year

As for the second KPI, we decided to calculate the average house price per neighborhood by year. This information can be helpful in understanding the overall health of the real state market. It can also serve as a benchmark for property valuation.

3.1.3 KPI3:Property type per district and neighborhood

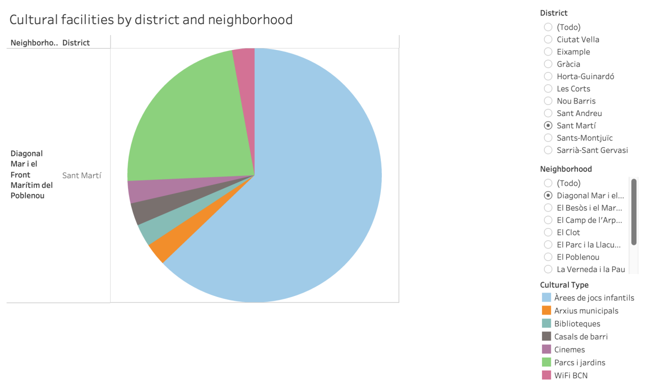
As for the third KPI, we decided to show the different types of properties per neighborhood and district. This KPI can help segment the market and understand buyer preferences based on district or neighborhood preferences. It helps in identifying areas that are more popular for specific property types.

3.1.4 KPI4: Comparison between avg family income(2007-2017) and avg listing price(2020-2021

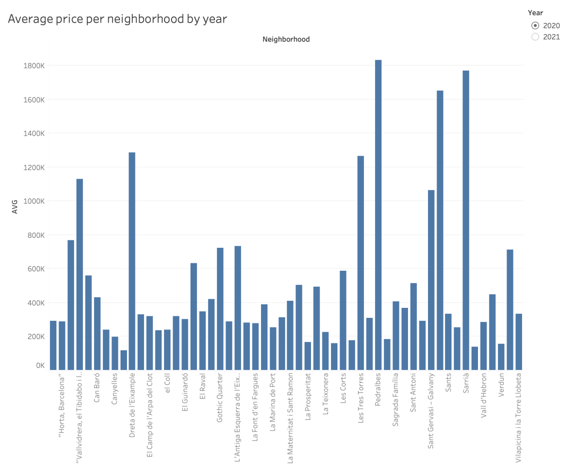
As mentioned before it was difficult to perform an analysis based on the Idealista and Income dataset due to the fact that they do not coincide in the dates. So we decided to show a comparison between the average family income for the years from 2007 to 2017 and the average listing price for the years from 2020 to 2021. Thus, althought this isn´t a year-to-year comparison, it can give an idea of the general differences.

3.1.5 Visualization

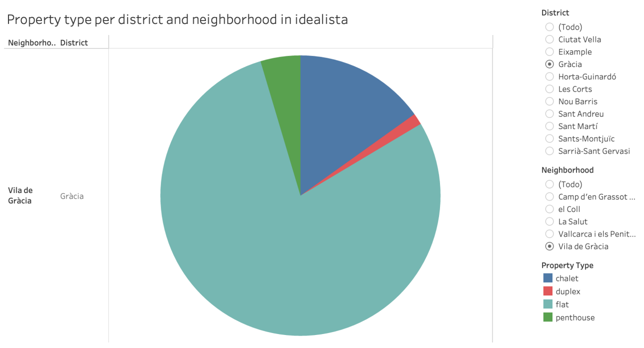
To visualize the different KPI's we used Tableau. In order for us to add to the project the Dashboard in Tableu and don´t have the need to have the database credentials to properly visualize it (we used local Mongo so we can´t share the credentials), we decided to save in .csv format all the KPIS and read the data from there in Tableu. But, as mentioned before we did store the exploitation zone in Mongo. The CVS files are generated every time the exploitation zone is run. To ensure effective visualization and avoid clustering all the KPIs together, we developed separate dashboards for each KPI.
Figure 2 depicts the different visualizations. The dashboard can be found in the project delivery under the name Dashboard.twb.



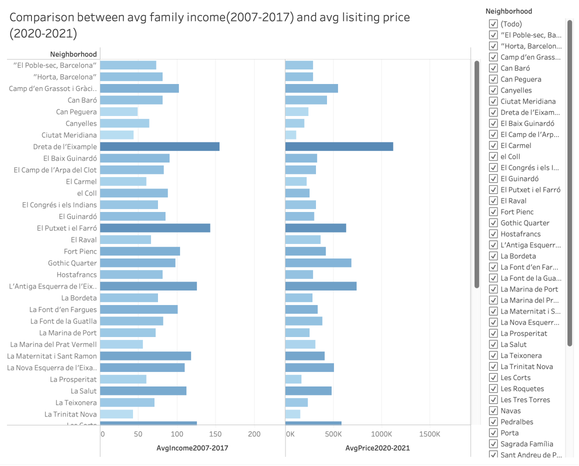
(a) KPI1



(b) KPI2



(c) KPI3



(d) KPI4

Figure 2: KPIs

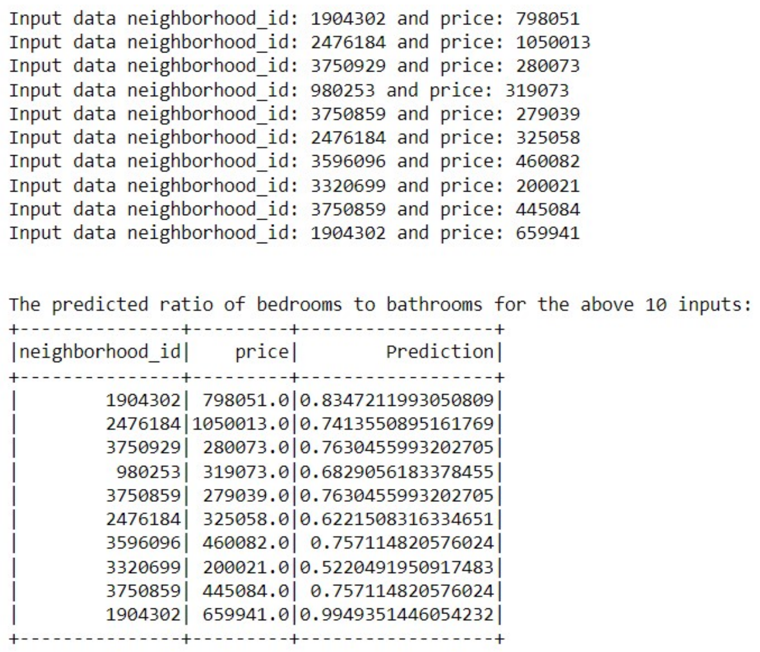
4 Data Streaming

The data Streaming pipeline we implemented consists basically of the following steps:

- **Data source**: we read the streaming from the Kafka endpoint provided, where the messages we obtained are composed of date, timestamp, neighborhood id, and price(we assumed it as the price of one particular house).
- **Model Training**: as from the Kafka messages the only two attributes we can use for data stream analysis are the neighborhood id and the price, this lack of data did not allow us to apply some sophisticated analysis algorithms, so we finally decided to train a Regression model that can predict the ratio of bathrooms to bedrooms given those two attributes as input. For the model training, we used the Idealista data stored in Mongo(Formatted zone). We read and extract from Mongo only the features that we are interested in for the training, which are neighborhood id, price, and the number of rooms and bathrooms. The rooms number were used to compute the bathroom-bedroom ratio. Usually, the bigger the ratio value is the higher would be the house price. Those features will be combined into a single feature vector in order to train the chosen ML model. We assumed that each message we read from Kafka corresponds to the click a user would make when opening a specific house description link to get more detailed information. Thus, this real-time

prediction of the bathrooms-to-bedrooms can be leveraged for targeted marketing efforts and to get to know the preferences of specific customer segments.

- **Real-time prediction**: Finally we trained the model with the Idealista data stored in our Formatted Zone and we performed the real-time prediction by feeding to the trained model each in bulk. Batching messages(10) allows us to improve the throughput and the performance of the pipeline and reduce resource utilization.



```
Input data neighborhood_id: 1904302 and price: 798051
Input data neighborhood_id: 2476184 and price: 1050013
Input data neighborhood_id: 3750929 and price: 280073
Input data neighborhood_id: 980253 and price: 319073
Input data neighborhood_id: 3750859 and price: 279039
Input data neighborhood_id: 2476184 and price: 325058
Input data neighborhood_id: 3596096 and price: 460082
Input data neighborhood_id: 3320699 and price: 200021
Input data neighborhood_id: 3750859 and price: 445084
Input data neighborhood_id: 1904302 and price: 659941


The predicted ratio of bedrooms to bathrooms for the above 10 inputs:
+---------------+---------+------------------+
|neighborhood_id|    price|        Prediction|
+---------------+---------+------------------+
|        1904302| 798051.0|0.8347211993050809|
|        2476184|1050013.0|0.7413550895161769|
|        3750929| 280073.0|0.7630455993202705|
|         980253| 319073.0|0.6829056183378455|
|        3750859| 279039.0|0.7630455993202705|
|        2476184| 325058.0|0.6221508316334651|
|        3596096| 460082.0| 0.757114820576024|
|        3320699| 200021.0|0.5220491950917483|
|        3750859| 445084.0| 0.757114820576024|
|        1904302| 659941.0|0.9949351446054232|
+---------------+---------+------------------+
```

Figure 3: Streaming data prediction

5 How to run the code

In order to run the code, it is necessary to modify the MongoDB connection string in each .py file.
After this, create firstly the Formatted zone by running the following scripts (in no particular order):

- `formatted_cultural_eq.py`: This creates the cultural collection.
- `formatted_idealista.py`: This creates idealista collection.
- `formatted_income.py`: This creates income collection.

Then, create the exploitation zone by running the script called `exploitation.py`
The Streaming data prediction can be done by running the following scripts:

- **StreamingModelTraining.py**: It read the Idealista data from Mongo and then uses it to train a Regression model. The model will be used later on in the streaming data prediction. This step is optional as there is already a pre-trained model in the directory `trained_model`.
- **Streaming.py**: It performs the real-time prediction after loading the stored model from the above-mentioned directory.