Master in Data Science

P2: Formatted and Exploitation Zones

Big Data Management

Year
2023/24

Date

Team members
Casanova, Adrià
Ferrer, Silvia
Vayá, Gabriel

Contents

1 Introduction

In this work we present a comprehensive approach to a second part of a Big Data Management Backbone. The project starts with a given solution for a Persistent Landing Zone, and is going to implement a Formatted Zone and an Exploitation Zone including the calculation of a set of KPI's and a predictive analysis and the visualization of our overall analysis.

The original code can be found in Github: `https://github.com/silviaferrer/bdm_p2`

2 Start point

2.1 Given Solution for the Persistent Landing Zone

This work starts with a given layout for a Persistent Landing Zone, which ingests and timestamps a given set of data sets.

- **Idealista**: One Parquet file for each JSON.
- **Income**: One MongoDB collection
- **Lookup tables**: Four MongoDB collection (rent_lookup_district, rent_lookup_neighborhood, income_lookup_district and income_lookup_neighborhood) with prefixed attributes.

2.2 Third Dataset

As the additional data set we are choosing another collection from *Open Data Barcelona*, this time about territorial quality of air, between years 2019 and 2023 (4 tables), in CSV format. Here are the columns in this dataset.

- **_id**: The unique identifier for each record.
- **Estacio**: The station name or code.
- **nom_cabina**: The name of the cabin.
- **codi_dtes**: The code for the data set.
- **zqa**: The quality zone area code.
- **codi_eoi**: The EOI code.
- **Longitud**: The longitude coordinate.
- **Latitud**: The latitude coordinate.
- **ubicacio**: The location description.
- **Codi_districte**: The district code.
- **Nom_districte**: The name of the district.
- **Codi_barri**: The neighborhood code.
- **Nom_barri**: The name of the neighborhood.
- **Clas_1**: The first classification.
- **Clas_2**: The second classification.
- **Codi_Contaminant**: The code for the contaminant.

2.3 Logging mechanism

This logging mechanism sets up a *data_management* logger that:

- Writes log messages to a file named main.log.
- Rotates the log file when it reaches 5 MB, keeping up to 5 old log files.
- Captures log messages at the DEBUG level and above.
- Formats log messages with a timestamp, logger name, log level, and the message itself.

This will provide us a history of log files for troubleshooting and analysis.

3 Formatted Zone

3.1 Formatted Loader

The *Formatted Loader* retrieves data from the Persistent Landing Zone and prepares it for processing within the *Data Formatter.* Data from various sources are loaded as Spark DataFrames and organized into a dictionary structure, where each key represents a different data source and its corresponding value is a list of Spark DataFrames.

3.2 Data Formatter

The *Data Formatter* module is designed to format and reconcile heterogeneous datasets using Apache Spark within a Python environment, ensuring seamless integration with MongoDB for efficient data management.
We intend to perform the necessary formatting to the data, obtained by the *Formatted Loader*, reconciliation and data clearance to make the data suitable for the further analysis conducted later on. Our approach was to construct 3 tables, one for **income**, one for **idealista** and one for **airqual**, each containing the reconciled columns from the lookup tables. Instead of having a single joined table with all the columns, it is more flexible to build 3 tables with the necessary columns for the analysts to perform the joins for each particular analysis in the Exploitation Zone.

The data processing workflow involves:

- Reconciliation and integration of disparate datasets:
  - Income Data: Normalization and merging based on predefined rules for districts and neighborhoods.
  - Idealista Data: Column reduction and normalization using lookup tables. In this case, the different DFs contain different columns, and some of them are nested. To solve the latter, we implemented an specialized step for this data and we explode those nested columns before the reconciliation process. The other part of the process is generalized for all DFs ingested by the *Data Formatter*, and implements a join with the lookup tables using a join column specific for each data source, and merges all DFs from the same data source.
  - Air Quality Data: Standardization of metrics by aligning district and neighborhood names with lowercase equivalents from lookup tables.
- Data cleaning operations to ensure data integrity:
  - Removal of duplicate records within each dataset.
  - Filtering out invalid entries from relevant columns.
- Integration with MongoDB for direct storage of formatted datasets, facilitating future retrieval and analysis.
- Robust error handling mechanisms to manage exceptions encountered during data loading processes.

- Structured workflow execution:
  - Initialization of necessary components and configurations at the outset of data processing.
  - Sequential execution of operations including reconciliation, cleaning, and loading.
  - Detailed logging and monitoring throughout each stage to track progress, identify issues, and maintain transparency.

This structured approach ensures efficient data preparation and management for next steps of the *Data Management* workflow.

4 Exploitation Zone

4.1 Descriptive Analysis

We analyzed data related to property listings, air quality, and family income across different neighborhoods using Apache Spark and MongoDB.
Data was loaded from the three MongoDB collections: `airqual`, `idealista`, and `income` into Spark DataFrames for efficient processing. We merged family income data with property listings based on neighborhood, added a year column to the property listings data, and merged air quality data with property listings based on neighborhood and year.

Key Performance Indicators (KPIs) calculated include:

- **Average New Listings per Day**: The average number of new property listings published daily.
- **Top-Selling Neighborhoods in the Last Month**: The five neighborhoods with the highest number of property sales in the most recent month.
- **Most Contaminated Neighborhoods**: The five neighborhoods with the highest number of air contaminant records from 2019 to 2023.
- **Correlation Between Sale Price and Family Income**: The correlation value indicating the relationship between property sale prices and family income.
- **Correlation Between Sale Price and Air Contaminants**: The correlation between property sale prices and air contaminant levels.

These results were saved to MongoDB collections: `ExploitationZone_KPIs`, `ExploitationZone_topSellers`, and `ExploitationZone_topContaminants`.

4.2 Predictive Analysis

Predictive Analysis integrates data processing and modeling using Apache Spark, DuckDB, and Python. It connects to DuckDB for local data storage, interfaces with MongoDB for data loading and model storage, and utilizes Spark for distributed data processing. The analysis pipeline includes:

- Initialization:
  The Predictive Analysis integrates various data processing and modeling functionalities using Apache Spark, DuckDB, and Python libraries. It establishes connections to DuckDB for local data storage and retrieval, ensuring seamless integration with Spark for distributed data processing. It also interfaces with MongoDB for data loading and model storage.
- Feature Selection and Data Preparation:
  Data from multiple sources (airqual, idealista, income) is loaded into Spark DataFrames. The module performs a join operation using reconciled columns (e.g., neighborhood) to create a unified DataFrame with all relevant features, including the target variable (price). Before model building, it verifies the absence of null values and selects required columns for analysis.

- Data Cleaning:
  The module identifies and handles null values in the DataFrame, ensuring data integrity and accuracy. Columns with null values are dropped to maintain data quality.

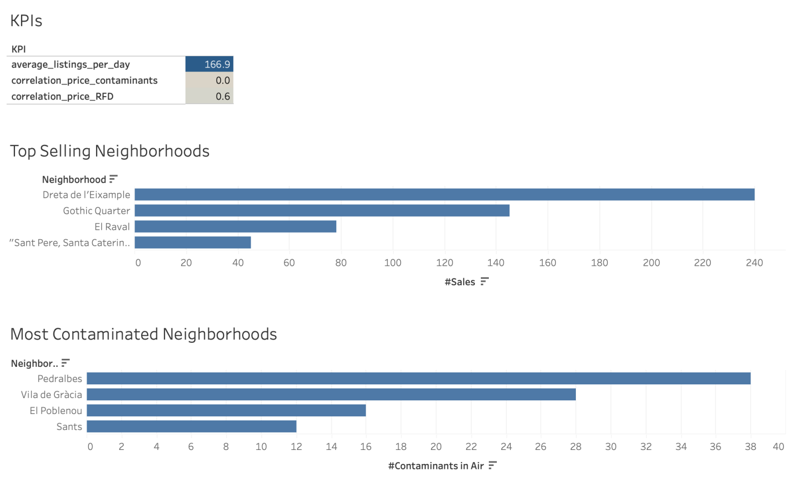
- Model Building and Evaluation:
  Using Spark's MLlib, the module builds a linear regression model to predict real estate prices. Categorical variables are encoded using one-hot encoding, and features are assembled into a vector format for model training. The model's performance is evaluated using metrics like Root Mean Squared Error (RMSE) and R-squared (R2).

- Model Serialization and Storage:
  Trained models are serialized into bit arrays and stored in DuckDB for persistence. The serialization process involves saving the model to a temporary directory before loading it into DuckDB, ensuring model availability and integrity.

- Data and Model Storage in MongoDB:
  Processed data and the trained predictive model are stored in MongoDB collections. The module writes the cleaned DataFrame and serialized model to designated collections in MongoDB, enabling easy access for subsequent analyses or deployment.

4.3 Data visualization

Key Performance Indicators (KPIs) extracted from the Exploitation Zone in MongoDB are directly connected to Tableau for graphical visualization, utilizing an Atlas connector between Tableau and MongoDB. We create interactive charts and dashboards that visually present the results of our analysis in a clear and effective manner. These visualizations enable stakeholders to gain insights quickly.
This process is shown in detail in the provided video *Tableau_demonstration.mov*. Due to the custom connector we have used, we have not been able to share the Tableau dashboard in web format. Nevertheless, we provide its Tableau Workbook, *BDM_P2.twb*, and show it in figure 1.


KPIs

KPI	
average_listings_per_day	166.9
correlation_price_contaminants	0.0
correlation_price_RFD	0.6

Figure 1: KPIs visualization in Tableau.

5 Conclusions

Throughout this project, we have undertaken a comprehensive approach to big data management, focusing on the creation of Formatted and Exploitation Zones, and subsequent analysis using advanced analytics techniques.

When preparing the data and creating the KPIs, our approach gives valuable insights into real estate market dynamics, neighborhood air quality, and how they relate, which is key for making informed decisions in urban planning, public health policies, and real estate strategies.

With predictive analysis it ensures robust data handling and modeling capabilities, supporting informed decision-making and enabling efficient data-driven insights across diverse domains. This cohesive workflow highlights the module's versatility in handling complex data operations and its contribution to enhancing predictive analytics in real-world applications.

In conclusion, this project underscores the importance of robust data management, advanced analytics, and data visualization in extracting actionable insights from our datasets. By adopting a structured approach to data integration, processing, and analysis, we have not only enhanced our understanding of our dataset but also equipped decision-makers with the tools to drive positive outcomes, policy formulation, and strategic planning.