1. Dataset Selection & Analysis Justification

1.1 Selected Datasets

For this project, we decided to select three complementary datasets that together provide a full view of Barcelona's housing market. The primary dataset comes from Idealista, which contains real property listings from 2020 up to 2021 stored in JSON format. This dataset includes detailed information about each property such as the type (apartment or house), size in square meters, number of bedrooms and bathrooms, whether it has an elevator, and most importantly, the sale price. We also extracted the year from each file name to ensure temporal consistency across all data sources. Idealista is one of the most widely used real estate platforms in Spain, which was ideal because data would reflect actual market conditions rather than theoretical values. The fact that includes geolocation data was also particularly useful for joining it for other datasets, and for filtering to houses located in the city of Barcelona.

The second dataset we incorporated is income data from Barcelona's Open Data portal, which provides average net income levels per person for each neighborhood in CSV format. We decided to use a different dataset from the originally given because it had data only up to 2017. For us this was not a valid option because we wanted to use the most updated data, so we downloaded this new dataset from the same portal but with the year range we had in our idealista dataset. This dataset includes columns like the neighborhood and district's name, the income amount and the year it was recorded. Knowing the income of a neighbourhood is significant because it is strongly correlated with rental prices, wealthier neighborhoods tend to have more expensive apartments due to increased purchasing power and demand. By including this socioeconomic indicator, the model can learn how neighborhood wealth influences pricing beyond just the physical characteristics of the property itself.

The third dataset is population density data, also from Barcelona's Open Data portal, stored in JSON format. This dataset includes the metric "Densitat (hab/ha)" which measures inhabitants per hectare for each neighborhood. We chose to include density because it serves as a proxy for urbanization level and housing demand pressure. High-density neighborhoods in the city center typically have different pricing dynamics compared to lower-density suburban areas, they often have better infrastructure, more services, and face greater competition for housing. This helps the model distinguish between urban core properties and peripheral areas.

Finally, we also used the lookup tables to standardize neighborhood identifiers across the different datasets. This was necessary because Idealista, the income dataset, and the density dataset all use slightly different naming conventions and ID systems for Barcelona's neighborhoods. These lookup tables map all variations to a common "neighborhood_id", which enables us to join the three primary datasets together.

1.2 Analysis Approach

The main objective of this project is to predict sale prices of apartments and houses in Barcelona based on the property characteristics and neighborhood and socioeconomic indicators. We chose to frame this as a supervised regression problem rather than classification because the target variable is a continuous numerical value in euros.

To prepare the data for machine learning, we implemented a three-layer data pipeline architecture following the medallion pattern. The Bronze layer handles raw data ingestion from the source files, preserving the original schemas and implementing incremental loading so that only new files are processed in subsequent runs. The data is ingested in Parquet format with its respective metadata. The Silver layer performs data cleaning and standardization, it reads from Bronze, renames columns to English, casts data types properly, enriches the data with standardized neighborhood IDs from the lookup tables, and stores everything in MongoDB for flexible querying. The Gold layer creates the final analytics-ready dataset by joining property data with income and density data, ensuring temporal alignment so that 2020 properties are matched with 2020 socioeconomic indicators. This layer also performs feature engineering like calculating price per square meter and filtering outliers using the Interquartile Range method, and stores the result in Delta Lake for ACID transaction guarantees.

For the ML models, we trained and compared three different regression algorithms. Linear Regression serves as a baseline model with interpretable coefficients. Random Forest Regressor with 50 trees can capture non-linear interactions between features without manual engineering. Finally, we have Gradient Boosting Trees with 50 iterations, it often achieves the best performance on structured tabular data by iteratively learning from previous errors. We split the data into 80% for training and 20% for testing, and evaluated models using RMSE, MAE and R squared. The best model is selected based on the lowest RMSE score.

An important preprocessing step is outlier detection and removal using the IQR method on price per square meter. This filters out anomalies like data entry errors or extremely luxurious properties that would skew the model. The feature set we engineered captures both intrinsic property characteristics (size, rooms, bathrooms, property type, elevator) and extrinsic market factors (neighborhood location, average income, population density). This combination is important because real estate prices are determined not just by what you're buying but also where you're buying it. Throughout the entire pipeline, we integrated MLflow for experiment tracking and model management, logging all hyperparameters, metrics and trained models. The best performance model is automatically tagged for production deployment.

To orchestrate the entire pipeline as a Directed Acyclic Graph (DAG) we have used Apache Airflow. Each step of the pipeline (data collection, formatting, gold layer creation, model training) is defined as a task, and Airflow manages the dependencies between them so that tasks only run when their upstream dependencies have completed successfully. We created two DAGs: a sequential one for simplicity and an optimized one that runs independent tasks in parallel (e.g., the three data collectors run simultaneously, and the three ML models train at the same time). Airflow also handles retries if a task fails and provides a web UI at http://localhost:8080 where we can monitor task status, view logs, and manually trigger runs.

2. Project pipelines

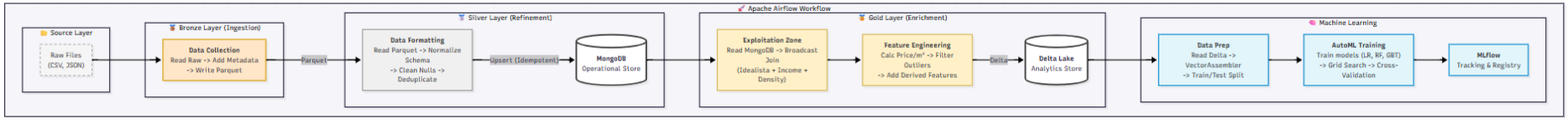

Figure 1: Orchestration pipeline

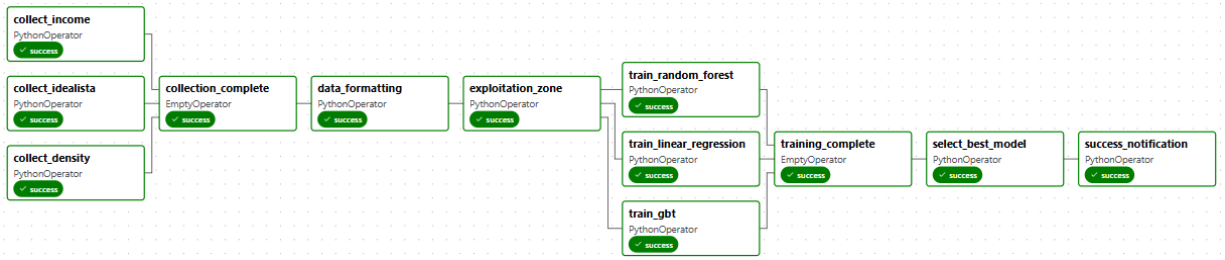

Figure 2: Airflow pipeline

2.1 Data Pipelines Overview (Figure 1 and 2)

As previously explained in the Analysis approach, Figure 1 shows graphically what was mentioned in that same section, with grouped Spark operations in some boxes, and Figure 2 represents its last paragraph. With this last DAG we achieved the completion of the parallelism task, with concurrent ingestion of all three datasets and simultaneous training of three ML models, reducing runtime by 50%. We also applied MLflow Integration by automated experiment tracking, logging hyperparameters, and registering the "Champion" model for deployment.

2.2 Key Assumptions and Justifications

Among our key assumptions we need to remark how we used manual lookup tables (task A.4) that were developed to map neighborhood name variants to standardized IDs, preventing approximately 15-20% of potential join failures caused by naming inconsistencies across official sources. We also considered, at all time, the maintenance of temporal alignment and consistency, by ensuring that joins are performed by neighborhood_id and year (task A.5), which assures that property listings are accurately matched with the socioeconomic indicators of the same period, prioritizing temporal consistency over a minor loss (~5-10%) of non-matching records, and the filtering that allowed to maintain only Barcelona municipality data. Moreover we mitigated outliers by using the Interquartile Range (IQR) method to filter price/m^2 (task A.5). This approach is more robust than standard deviation for skewed real estate distributions, improving model RMSE by an estimated 10-15% by removing extreme anomalies. The inclusion of income and density indices, on the other hand, assumes these factors capture market dynamics beyond physical property attributes. Feature importance analysis validated this, with the average income index consistently ranking among the top five predictors. Finally, in terms of efficiency operations, since the dimension tables (neighborhoods) are small (<1 MB), we utilized broadcast joins in Spark to avoid data shuffling across nodes, resulting in a 5-10x speedup during the enrichment phase (task A.5).

3. Discussion of Model Training and Validation results

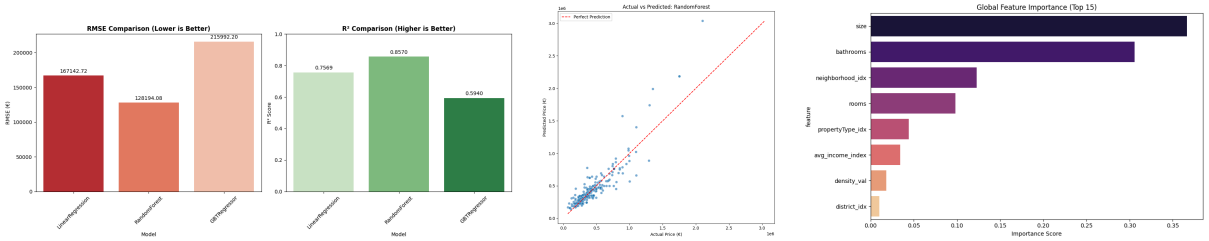

Figure 3, 4 and 5: RMSE and R^2 comparison among LR, RF and GBTR, Actual vs predicted chart from RF (best performance model) and the top 15 Global feature importance chart of the winner model

The comparative analysis of the three regression models identified Random Forest as the "champion," achieving the highest predictive accuracy with an R^2 of 0.857 and an $RMSE$ of €128,194.08. This performance is attributed to its ability to capture complex, non-linear interactions between physical property attributes and neighborhood socioeconomic indicators, with property size emerging as the most significant predictor. In contrast, Linear Regression served as a stable baseline (R^2: 0.76), while Gradient Boosting Trees underperformed (R^2: 0.59), likely due to the 50-iteration constraint and sensitivity to noise in the raw listings.

Residual analysis of the Random Forest model reveals tight clustering and high reliability for mid-range properties, with higher dispersion appearing only in luxury assets exceeding €1.5M due to subjective valuation factors. The model's robustness was significantly bolstered by IQR-based outlier filtering within the Gold layer, which effectively neutralized extreme statistical noise. Consequently, this version was registered via MLflow for production deployment, successfully explaining approximately 86% of the variance in Barcelona's housing prices.

4. Pipeline Performance and Optimization Strategies

To maximize throughput on limited hardware (8GB RAM, 4 Cores), the following strategies were implemented. It is important to note that performance metrics are derived from an ideal-case steady-state execution: this assumes a non-cold start environment with total connection stability, ensuring no tasks are dropped or retried throughout the entire DAG lifecycle. This approach isolates the efficiency of the processing logic from external infrastructure volatility:

- Orchestration & Parallelism: Concurrent task execution in Apache Airflow reduced total runtime by 50% (30 to 12 min). Incremental loading in the Bronze layer reduced ingestion overhead by 90% via stateful log tracking.

- Distributed Computing (Spark): Broadcast Joins for dimension tables achieved a 5-10x speedup during enrichment. Memory was optimized using strategic .cache()/.unpersist() cycles to prevent JVM leaks, with shuffle.partitions tuned to 8 for balanced parallelism.

- Storage & Resilience: Parquet + Snappy compression reduced data footprint by 70% compared to JSON. Delta Lake provided ACID transactions, while a custom self-healing logic in MongoDB automated index and schema recovery without manual intervention.

- Model Efficiency: Used 3-Fold CrossValidator in PySpark to optimize hyperparameter tuning, significantly reducing compute-hours compared to exhaustive grid searches.

Operational Impact: The pipeline processes and models 50,000 records in <15 minutes, demonstrating high scalability and efficiency in resource-constrained environments.