Barcelona School of Economics

Assignment 3

Big Data Management - DSDM - L3-T01

Julian Romero

Moritz Peist

Contents

June 23, 2025

Overview

This report documents the implementation of Parts A (Data Management Backbone) and C (Orchestration Framework) of Lab 3. Our solution implements a comprehensive data lake architecture using PySpark 4.0, Delta Lake, MLFlow, and Apache Airflow 3.0 to process Barcelona's Open Data for real estate and socioeconomic analysis. All outputs are bundled in a comprehensive Streamlit application, providing an interface to our services and analysis (cf. `README.md`). In the following, we sketch our data pipelines (cf. Figure 1):

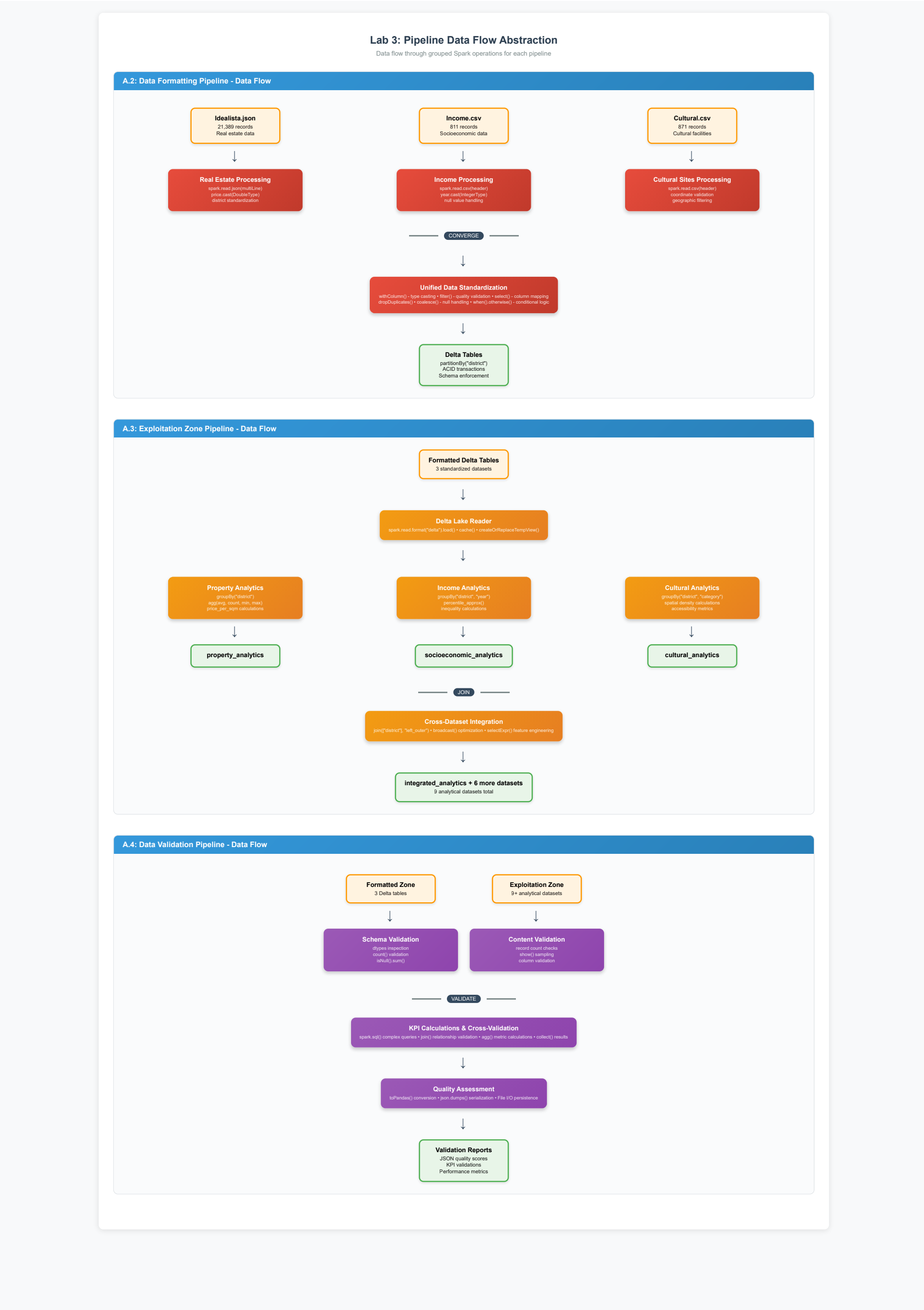


Figure 1: Abstract pipeline sketch

A Data Management Backbone

A.1 Dataset Selection & KPI Definition

We selected three datasets from Barcelona's Open Data portal, satisfying all requirements:

- **Idealista** (JSON, 21,389 records): Real estate listings with comprehensive property characteristics, prices, and geographic coordinates
- **Income** (CSV, 811 records): Socioeconomic data by district/neighborhood (2007-2017) with excellent data quality (0% missing)
- **Cultural Sites** (CSV, 871 records): Geographic distribution of cultural amenities across Barcelona

Our analysis focuses on **10 comprehensive KPIs** spanning housing affordability, socioeconomic equity, and urban quality of life, including Housing Affordability Ratio, Income Inequality Index, Cultural Density, and Neighborhood Attractiveness Score.

A.2 Data Formatting Pipeline

Implementation: `src/airflow/dags/pipelines/a2.py`

The formatting pipeline transforms raw data from Landing Zone into standardized Delta tables:

- **Data Standardization:** Schema unification, data type casting, and column naming conventions
- **Quality Enhancement:** Duplicate removal, missing value imputation, and outlier detection
- **Storage Optimization:** Delta Lake ACID transactions with district-based partitioning for efficient querying
- **Geographic Processing:** Coordinate validation and district/neighborhood mapping integration

Output: `data_zones/02_formatted/` containing three Delta tables with enforced schemas and quality constraints.

A.3 Exploitation Zone Pipeline

Implementation: `src/airflow/dags/pipelines/a3.py`

This pipeline creates analytics-ready datasets through sophisticated transformations:

- **Aggregation Engine:** Property metrics (price/m^2, availability) aggregated by district/neighborhood
- **Feature Engineering:** Income inequality calculations, cultural density metrics, and affordability ratios
- **Cross-Dataset Integration:** Spatial joins enabling composite KPI calculations
- **Analytics Optimization:** Nine specialized datasets created for specific analytical purposes

Key Datasets Created: property_analytics, socioeconomic_analytics, cultural_analytics, and integrated_analytics for comprehensive analysis.

A.4 Data Validation Pipeline

Implementation: `src/airflow/dags/pipelines/a4.py` + validation notebooks

Comprehensive validation framework ensuring data integrity across all zones:

- **Quality Metrics:** Completeness, accuracy, consistency, and uniqueness assessments
- **KPI Validation:** Statistical verification of calculated metrics and cross-zone consistency
- **Performance Monitoring:** Pipeline execution metrics and resource utilization tracking
- **Automated Reporting:** JSON reports for Streamlit dashboard consumption

B Data Analysis Backbone

B.1 Predictive Analysis via Model Training and Management

Implementation: Spark MLlib with MLflow model management and Airflow orchestration

Our solution addresses the task of predictive model training and management as follows:

- **Data Preparation:** We load and preprocess the Exploitation Zone data stored in Delta Lake, including feature engineering with Spark ML transformers (e.g., `StringIndexer`, `VectorAssembler`), and create two datasets (training and validation) via an 80/20 random split.
- **Model Training:** Two regression models are trained using Spark ML, particularly `LinearRegression` and `RandomForestRegressor`. Each model is trained on the training dataset and evaluated on the validation dataset.
- **Evaluation Metrics:** Model performance is measured by the RMSE (Root Mean Squared Error) metric, which is logged alongside hyperparameters and other metadata.
- **Model Management Framework:** MLflow is used extensively to log models, hyperparameters, and evaluation metrics. Each model run is tracked as an experiment under `HousePriceRegression`.
- **Ranking and Selection:** Models are automatically ranked by their RMSE on the validation set, with the best-performing model identified programmatically.
- **Automatic Deployment:** The best model is registered in the MLflow Model Registry and transitioned to the Production stage, archiving previous versions.
- **Orchestration:** Apache Airflow DAG `train_and_deploy_model` automates the full pipeline: model training followed by automatic model registration and deployment, ensuring repeatability and operational robustness.

Code Location: The core model training and MLflow integration are implemented in `src/ml_experiments/house_price_prediction.py`. The Airflow DAG orchestrating the process is in `src/airflow/dags/train_deploy.py`.

Output: The outputs of the experiments are saved in `outputs/mlruns` directory.

C Orchestration Framework (Bonus)

Implementation: Apache Airflow 3.0 with asset-based scheduling

Our orchestration solution provides production-ready pipeline management:

- **DAG Architecture:**
  - `bcn_data_pipeline_with_validation` orchestrates A.2 → A.3 → A.4 pipeline sequence.
  - `train_and_deploy_model` orchestrates B, from training to deployment.
- **Dependency Management:** Asset-based scheduling ensures proper execution order and data availability
- **Error Handling:** Comprehensive failure recovery with email notifications and retry mechanisms
- **Monitoring Integration:** Real-time pipeline status through Airflow UI at `localhost:8080`

Advanced Features: Docker containerization for reproducibility, configuration management through environment variables, and integration with MLflow for model deployment.

Pipeline Flow: Landing Zone (raw data) → Formatted Zone (Delta tables) → Exploitation Zone (analytics datasets) → Validation Reports → Streamlit Dashboards

Technology Integration: Our solution integrates PySpark 4.0 for distributed processing, Delta Lake for ACID transactions, Airflow 3.0 for orchestration, MlFlow for model monitoring/deployment, and Streamlit for interactive visualization.

A Appendix

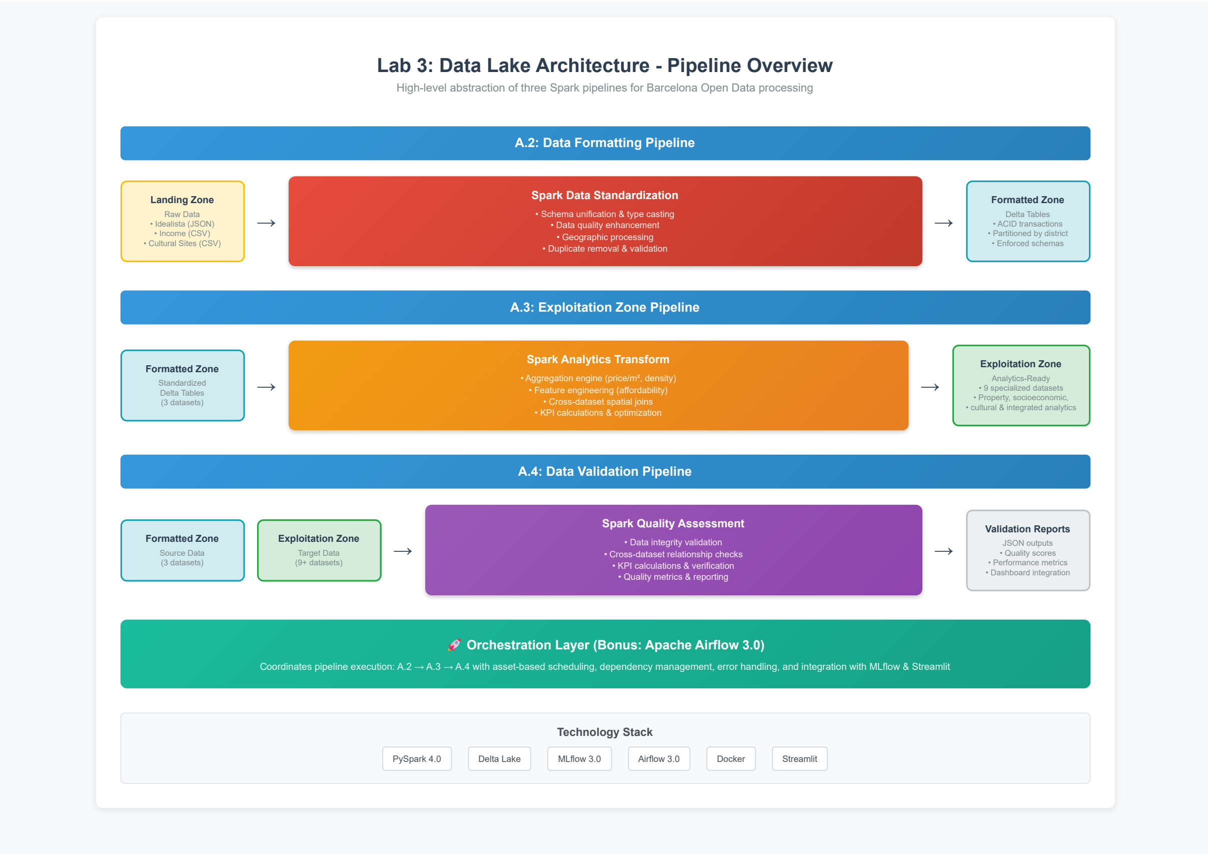
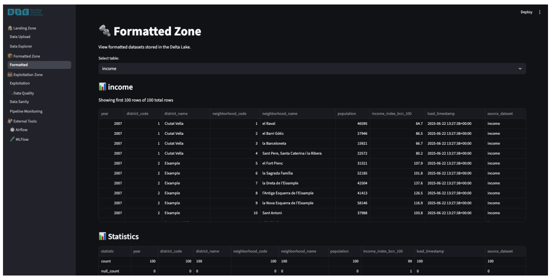
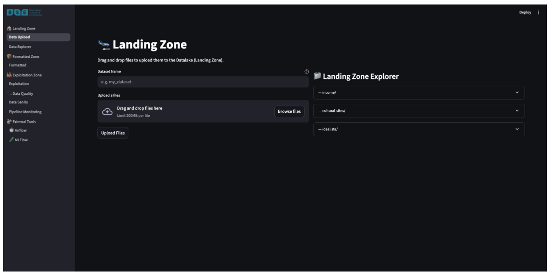


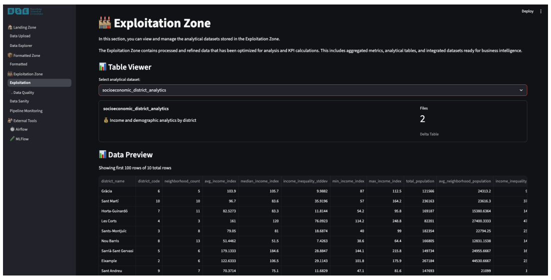
Figure 2: Project abstract overview



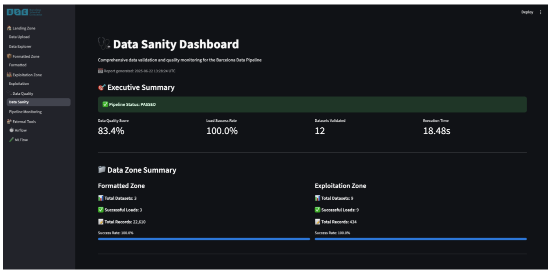
(a) Streamlit - Formatted zone



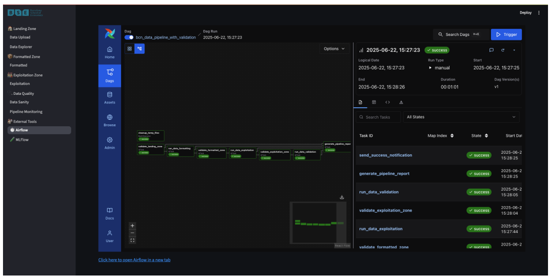
(b) Streamlit - Landing zone



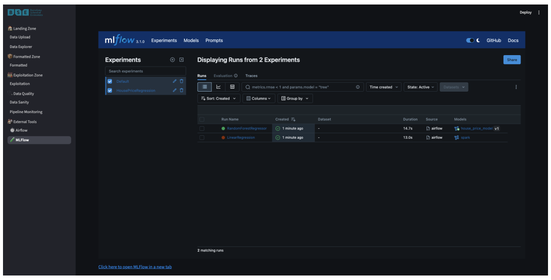
(c) Streamlit - Exploitation zone



(d) Streamlit - Data validation



(e) Streamlit - Airflow DAGs



(f) Streamlit - MLFlow

Figure 3: Streamlit Application Screenshots