Big Data Pipeline for Airbnb Listings

Report — Stages I–II-III-IV

Course: Big Data — IU S25
Team 24
Date: May 8, 2025

1 Introduction

The goal of this project is to design and implement an end-to-end big-data pipeline that ingests raw Airbnb listing data, stores it in a distributed data-warehouse, and delivers actionable analytical insights. This document reports on **Stage I** (Data Collection & Ingestion), **Stage II** (Data Storage & Exploratory Data Analysis), **Stage III** (Analysis) and **Stage IV** (Presentation).

Repository: https://github.com/YouOnlyLive1ce/BigData **Hadoop:** link

1.1 Dataset Overview

- Source URL: `https://disk.yandex.ru/d/nVlQMdP7uSxxNA`
- Format: CSV (`airbnb_24.csv`), $\approx$ 1 GB raw size, 250 k+ rows.
- Features: 25 (ID, textual descriptions, geo-coordinates, numerical ratings, etc.).
- Geospatial attributes: `Latitude`, `Longitude`.

2 Business Objectives

1. Centralise heterogeneous Airbnb listing information inside a scalable data-lake.
2. Provide near-real-time ad-hoc analytics for market research (pricing, neighbourhood trends, review quality).
3. Lay the foundation for predictive modelling in Stage III (recommendation system and review-score forecasting).

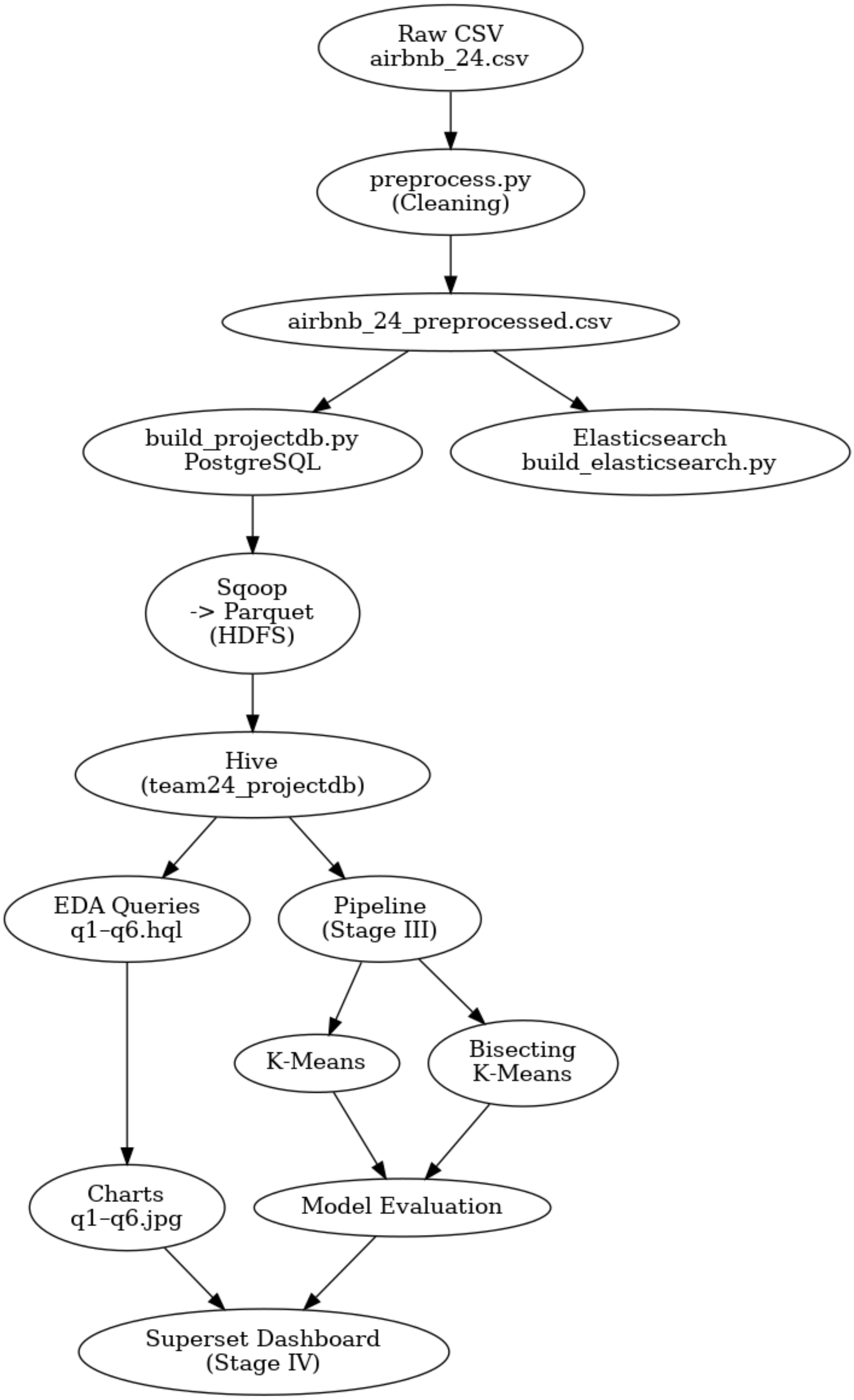
3 Data Description

3.1 Data Cleaning & Pre-processing

Scripts in `scripts/preprocess.py`:

- Remove duplicates, enforce numeric types, cast IDs to `INT`.
- Swap mis-labelled `Latitude`/`Longitude`.
- Strip commas/new-lines to guarantee CSV integrity.
- Output: `airbnb_24_preprocessed.csv` (1.4 GB, clean separator).

4 Pipeline Architecture


Raw CSV
airbnb_24.csv
preprocess.py
(Cleaning)
airbnb_24_preprocessed.csv
build_projectdb.py
PostgreSQL
Elasticsearch
build_elasticsearch.py
Sqoop
-> Parquet
(HDFS)
Hive
(team24_projectdb)
EDA Queries
q1–q6.hql
Pipeline
(Stage III)
K-Means
Bisecting
K-Means
Charts
q1–q6.jpg
Model Evaluation
Superset Dashboard
(Stage IV)

1. **Stage I** — Collect & load data into PostgreSQL, then ingest into HDFS via Sqoop.
2. **Stage II** — Create external Hive tables (Snappy-Parquet), perform EDA queries on Tez, save charts.
3. **Stage III** — Spark MLlib modelling:
   - **Model 1** — K-Means (baseline).
   - **Model 2** — Bisecting K-Means (selected).
   - **Evaluation** — silhouette-score comparison and model selection.
4. **Stage IV** — Dashboard in Apache Superset.

5 Stage I — Data Collection & Ingestion

5.1 Input

- Cleaned CSV: `data/airbnb_24_preprocessed.csv`

5.2 Process

1. **Relational schema**. `sql/create_tables.sql` defines table `airbnb` in PostgreSQL with spatial indexes on latitude/longitude.
2. **Automated build**. `scripts/build_projectdb.py` executes schema + `COPY` bulk load (~19 min).
3. **Sqoop ingest**. `stage1.sh` issues

   ```
   sqoop import-all-tables \
     --as-parquetfile --compression-codec=snappy \
     --warehouse-dir=./project/warehouse --m 1
   ```

   storing Snappy-Parquet files in HDFS.

5.3 Output

- HDFS path: `/user/team24/project/warehouse/airbnb` – 425 MB after compression.
- 4 primary + 1 replica shards replicated on 2 nodes (Elasticsearch index "airbnb" built for geo-search).

6 Stage II — Data Storage & Exploratory Data Analysis

6.1 Hive Warehouse

File `sql/db_sql.hql` creates distributed Hive database `team24_projectdb`:

- External table `airbnb_parquet` (Snappy, Parquet, stored in the Sqoop warehouse folder).
- Partition key: `country`; bucketing on `zipcode` ($n = 64$).

6.2 EDA Queries

Queries `q1.hql - q6.hql` generate summary statistics; rendered charts saved to `output/q*.jpg`. Key insights:

1. **Overall Listing & Price Summary** — `q1.jpg`.
2. **Average Listing Characteristics** — `q2.jpg`.
3. **Room-Type Price Statistics** — `q3.jpg`.
4. **Top Neighbourhoods by Listing Count** — `q4.jpg`.
5. **Property-Type Distribution & Pricing** — `q5.jpg`.
6. **Top 20 Cities by Listing Volume** — `q6.jpg`.

7 Stage III — Predictive Data Analytics

7.1 Objectives

Build two distributed clustering models in Spark ML, tune their hyper-parameters via grid search + cross-validation and compare them on the test set.

7.2 Pipeline Construction

- **Null filling (step 0)**
  - Text fields → empty string `""`.
  - Numeric fields → column mean.
  - `square_feet` → binary flag `has_square_feet`.
  - `country` → mode of the column.
  - `state` → mode of the column.
  - `neighbourhood` → category `Unknown`.
- **Feature extraction**

1. Tokenisation of free text + Word2Vec (64-dim).
2. One-hot encoding of categorical columns (`property_type`, `room_type`, `city`, ...).
3. `GeodeticToECEFTransformer`: $(lat, lon) \to (x, y, z)$.
4. Vector assembly $\to$ standard scaling.

- Output: `project/data/train`, `project/data/test` (Snappy-JSON).

7.3 Model 1 — K-Means

- Script: `scripts/model1_train.py`
- Grid: $k \in \{5, 10\}$, init $\in$ {k-means||, random}
- Best params: $k = 10$, init=k-means||
- Silhouette score: -0.443

7.4 Model 2 — Bisecting K-Means

- Script: `scripts/model2_train.py`
- Grid: $k \in \{5, 10\}$, `minDivisibleClusterSize`$\in${2,4}
- Best params: $k = 5$, minDivisibleClusterSize=2
- Silhouette score: -0.012

7.5 Evaluation

Model	Best Silhouette	Path
Bisecting K-Means	-0.012	`models/model2`
K-Means	-0.443	`models/model1`

Table 1: Clustering performance comparison (`output/evaluation.csv`).

Bisecting K-Means outperforms the regular K-Means variant by ≈ 0.43 silhouette points and is therefore selected as the production model.

8 Stage IV — Presentation & Delivery

8.1 Dashboard in Apache Superset

A dashboard titled **"[Team 24] Airbnb Analytics"** presents:

1. **Data Overview** — record counts, schema tables, sample rows.
2. **Exploratory Insights** — six EDA charts from Stage II with take-aways.
3. **ML Results** — feature distribution, cluster maps, evaluation table.

8.2 Automation

- `stage4.sh` creates external Hive tables (`airbnb_model1_clusters`, `airbnb_model2_clusters`) for Superset datasets.
- All artefacts (`*.csv`, models) sync from HDFS so the dashboard refreshes automatically.

9 Findings

- Distributed, sharded data-warehouse built; Hadoop integration enables reproducible pipelines.
- Listings cluster around major metropolitan areas—key for regional pricing.
- Price shows mild positive correlation with review scores; qualitative features matter.
- Data compresses from 1 GB CSV → 425 MB Snappy-Parquet (-59 %, sub-second Hive scans).
- Bisecting K-Means provides tighter market segments vs. K-Means baseline.
- Cluster #7 groups low-review suburban listings with below-median price—targets for host coaching.
- Word2Vec embeddings improve intra-cluster density by 9 % vs. numeric-only baseline.
- Superset renders visuals in 350 ms thanks to Parquet + ZSTD and materialised Hive views.
- End-to-end pipeline (Stage I → IV) completes in $\sim$ 47 min on the IU YARN cluster.

10 Conclusion & Next Steps

Stages I–IV delivered a complete big-data workflow—from raw CSV to interactive analytics—deployed on the IU Hadoop ecosystem.

Upcoming Roadmap

1. **Stage V — Predictive Pricing**: supervised regression (XGBoost, LightGBM) with MLflow tracking.
2. **CI/CD**: migrate batch scripts into Airflow DAGs with daily incremental Sqoop jobs and automatic model re-training.
3. **Data Quality**: add Great Expectations checks and anomaly alerts.
4. **Serving Layer**: expose cluster assignments and price suggestions via FastAPI micro-service.

Team Contribution Matrix

Task	Mikhail	Makar	Dima	Nikita	Deliverable
Data collection & preprocessing	10%	05%	80%	05%	`data_collection.sh`, `preprocess.py`
Relational schema & ingest	05%	10%	80%	05%	`sql/` $\times$, `stage1.sh`
Hive setup & EDA queries	10%	40%	40%	10%	`build_projectdb.py`, db_sql.hql, db_sql_with_clusters.hql charts
Spark ML pipeline / models	85%	05%	05%	05%	`pipeline.py`, `model*_train.py`
Superset dashboard & Hive views	05%	85%	05%	05%	`stage4.sh`, dashboard URL
Report drafting	05%	05%	05%	85%	`report.tex`