ISEP - Data Intelligence Engineering

Big Data Project Report

IMDB + Netflix Recommendation Pipeline

Shangyu YANG

ID: 63932

May 29, 2025

Course: Advanced Databases and Big Data
Supervisor: Prof. Hedi YAZID

Contents

1	Project Overview	2
2	Project Architecture	2
3	Data Ingestion (2 pts)	3
4	Data Formatting (4 pts)	3
5	Recommendation Strategies: Heuristic + ALS (2.5 + 1 pts)	4
6	Indexing and Dashboard (2 + 2 pts)	4
7	Airflow DAGs (1 pt)	6
8	Bonus Points Summary (4.5 pts)	7
9	Run Instructions	7
10	Conclusion	7
11	Acknowledgements	7

1 Project Overview

- **Theme:** Movie recommendation
- **Goal:** Build an end-to-end Big Data pipeline using IMDb + Netflix datasets
- **Tools:** Ubuntu, PySpark, Elasticsearch, Airflow, ALS model, Parquet, Pandas
- **Code:** `github.com/Apricity123456/bigdata`

2 Project Architecture

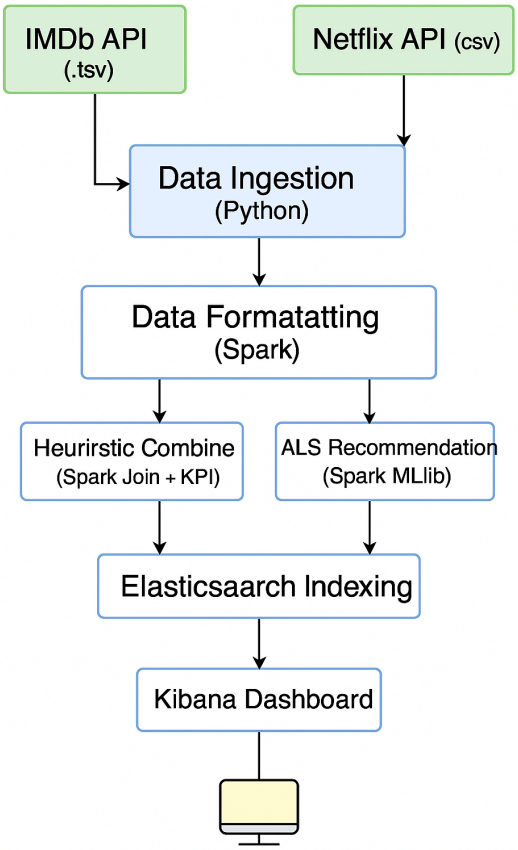

Figure 1: Overall Architecture of the Recommendation Pipeline

As illustrated in Figure 1, the pipeline consists of the following stages:

- **Ingestion**
- **Formatting**
- **Recommendation Generation**:
  - Heuristic Combine (Spark Join + KPI)
  - ALS Recommendation (Spark MLlib)
- **Indexing**
- **Visualization (Kibana)**

Project File Tree

The following image summarizes the project's folder and file layout:

```
.
├── airflow_start.sh
├── dags
│   ├── airflow_dag.py
│   ├── als_pipeline_dag.py
│   └── dual_recommend_dag.py
├── dashboard_screenshots
├── data
│   ├── als
│   │   └── recommendations.parquet
│   ├── combined
│   │   └── recommended_movies.parquet
│   ├── formatted
│   │   ├── imdb
│   │   └── netflix
│   └── raw
│       ├── imdb
│       └── netflix
├── README.md
├── requirements.txt
└── scripts
    ├── als_recommend.py
    ├── combine_recommend.py
    ├── format_imdb.py
    ├── format_netflix.py
    ├── index_als.py
    ├── index_elastic.py
    ├── ingest_imdb_api.py
    └── ingest_netflix_api.py
```

Figure 2: Project Directory Structure (Shell Tree View)

3 Data Ingestion (2 pts)

- IMDb via public .tsv.gz API
- Netflix via Kaggle API
- Scripts: `ingest_imdb_api.py`, `ingest_netflix_api.py`

4 Data Formatting (4 pts)

- Conversion to Parquet (Spark)
- Type cleaning, date normalization
- Scripts: `format_imdb.py`, `format_netflix.py`

5 Recommendation Strategies: Heuristic + ALS (2.5 + 1 pts)

- **Heuristic Recommendation**:
  - Joined `IMDb` and `Netflix` datasets on cleaned titles
  - Aggregated a `recommend_score = rating + log(votes)`
  - Script: `combine_recommend.py`
- **ALS Recommendation (Spark MLlib)**:
  - Built collaborative filtering model using PySpark ALS
  - Generated top-N recommendations
  - Script: `als_recommend.py`

6 Indexing and Dashboard (2 + 2 pts)

- Indexed both ALS and heuristic results into Elasticsearch
- Visualized key KPIs in Kibana via two dashboards:
  - `Movies Recommendation Dashboard` – Heuristic-based insights on top movies, genres, and ratings
  - `ALS Recommendation Overview` – ALS-based metrics including user-wise count and predicted score distribution
- Scripts: `index_als.py`, `index_elastic.py`

See Figures 3, 4, and 5 below for details.

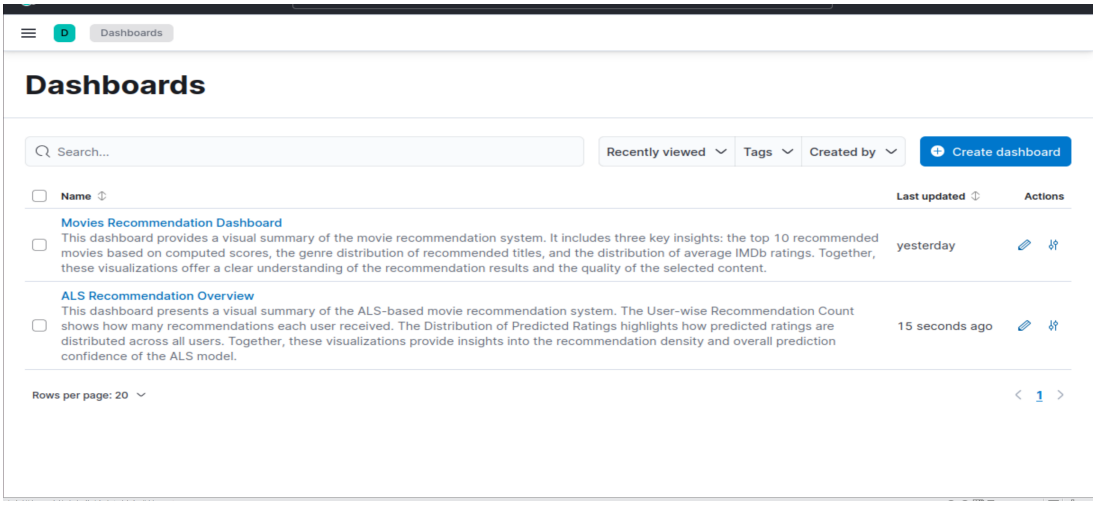

Figure 3: Kibana Dashboards Overview – Includes heuristic-based and ALS-based dashboards

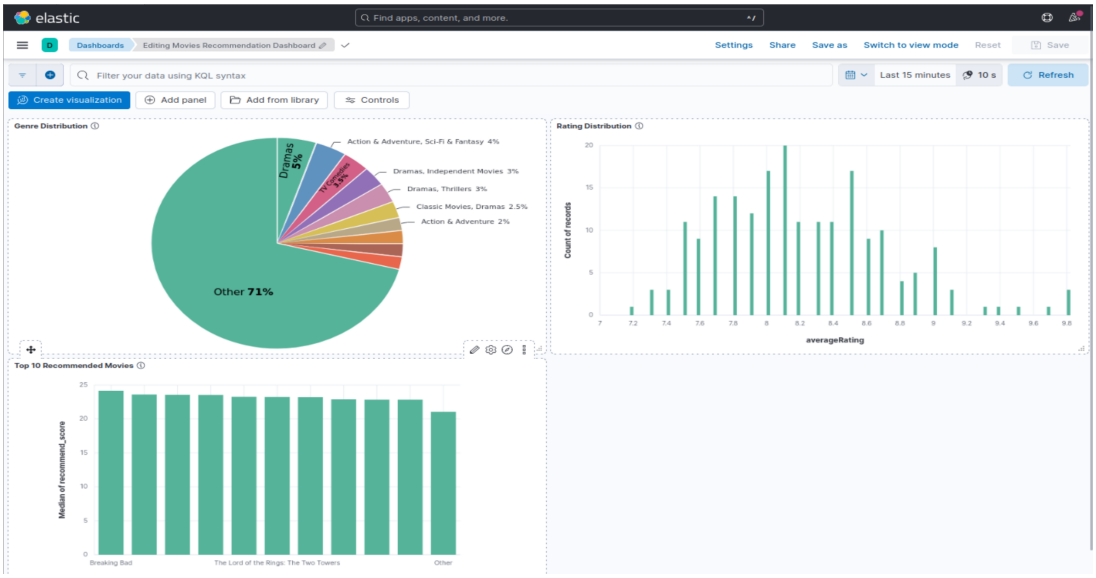

Figure 4: Heuristic Dashboard – Genre distribution, top-10 recommended movies, and rating distribution

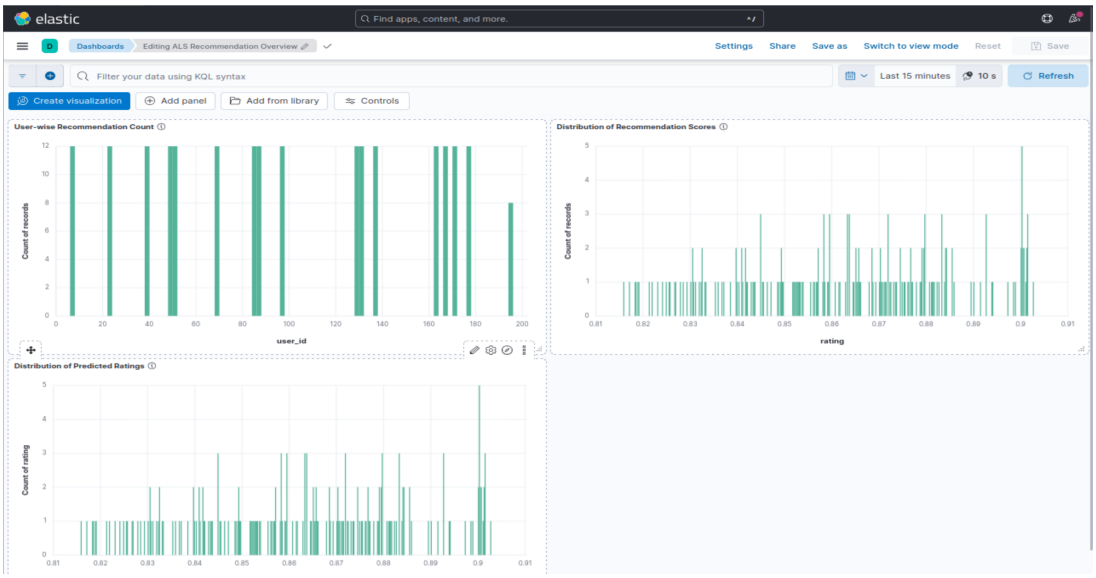

Figure 5: ALS Dashboard – User-wise recommendation count and predicted score distribution

7 Airflow DAGs (1 pt)

- Managed full pipeline automation using Apache Airflow.
- Defined and scheduled the following DAGs:
  - `airflow_dag.py` – Minimal test DAG for pipeline setup
  - `als_pipeline_dag.py` – DAG for ALS-based recommendation flow
  - `dual_recommend_dag.py` – DAG combining heuristic and ALS recommendation in parallel
- *See Figures 6, 7, and 8 for the visual DAG layouts.*

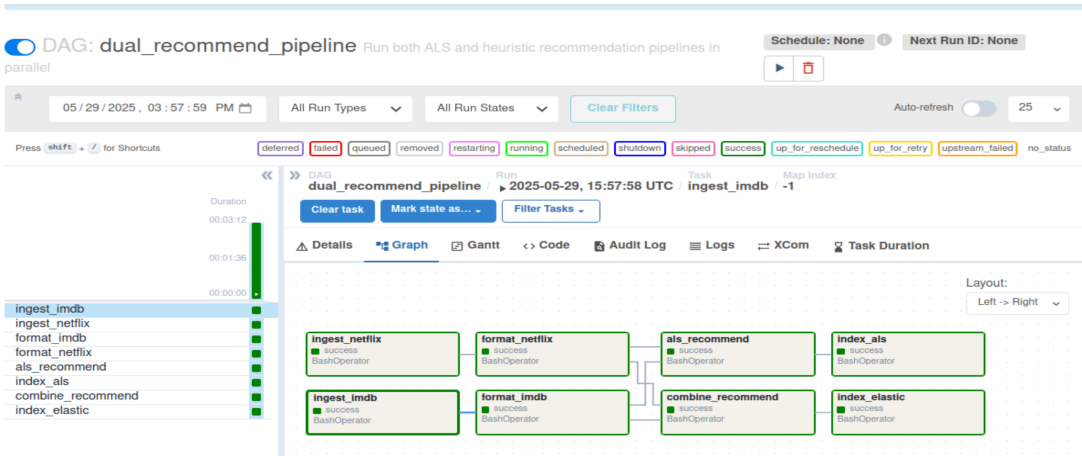

Figure 6: Airflow DAG: `dual_recommend_dag.py` – Run ALS and heuristic pipelines in parallel

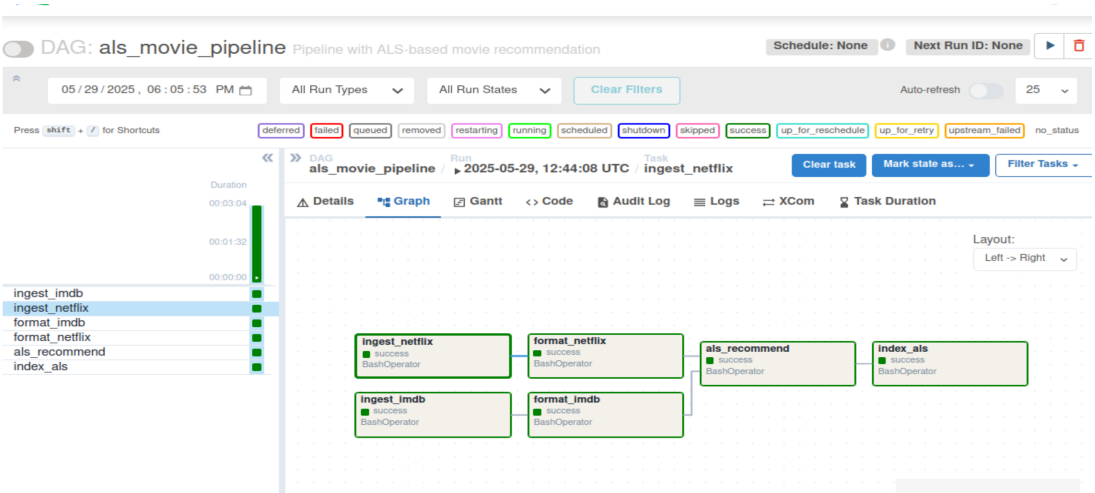

Figure 7: Airflow DAG: `als_pipeline_dag.py` – ALS-based recommendation pipeline

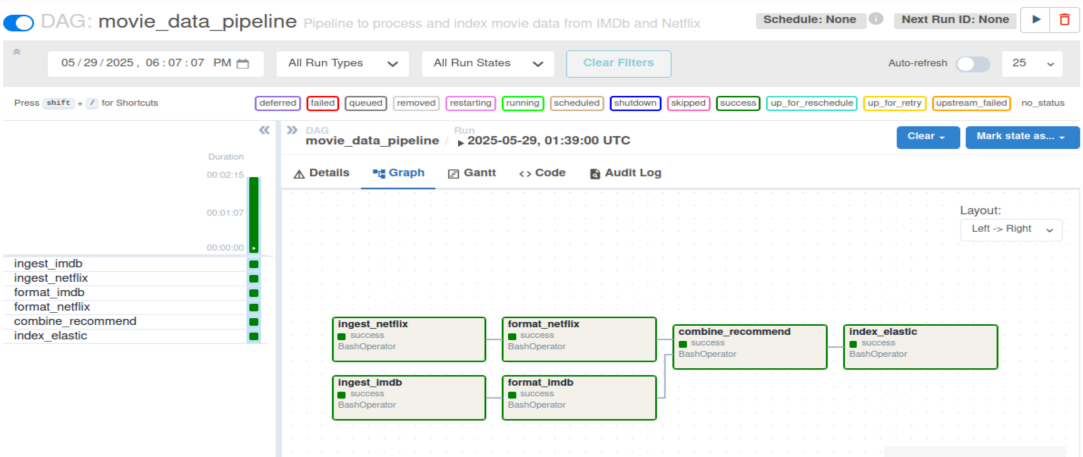

Figure 8: Airflow DAG: `airflow_dag.py` – Basic data processing and indexing pipeline

8 Bonus Points Summary (4.5 pts)

- **Clean folder structure (1 pt):** Clearly structured project layout with separated directories like `scripts/`, `dags/`, and `data/`, as shown in Section 2 and Figure 2.
- **Meaningful recommendation formula (1.5 pts):** Heuristic recommendation formula defined as `recommend_score = rating + log(votes)`, detailed in Section 5.
- **Machine learning implementation (1 pt):** ALS collaborative filtering implemented using PySpark MLlib (Section 5), with visualization in Figures 5 and 4.
- **Technical blog published (1 pt): Link**: `ClickHere`

9 Run Instructions

- Run the pipeline: `bash airflow_start.sh`
- Ensure `Elasticsearch` and `Spark` are running
- Open Airflow UI at `http://localhost:8080`
- Open Kibana UI at `http://localhost:5601`
- Access Elasticsearch at `http://localhost:9200`

10 Conclusion

- Successfully built and executed a Spark-based recommendation pipeline
- Learned orchestration with Airflow and indexing with Elasticsearch

11 Acknowledgements

I sincerely thank **Professor Hedi YAZID** for his guidance and support throughout this course. His insights on recommendation systems and Airflow orchestration were invaluable to my work.