Institut National des Postes et Télécommunications

DATA MINING AND WAREHOUSING

Report: Cloud-Based Machine Learning Workflow for Movie Dataset Analysis

Submitted by: Zakarya Belaid Adil El Hihi Benmessaoud Aymane Abdelilah Ikbi

Supervised by: Pr El Asri Ikram

Academic Year: 2024/2025

Abstract

This project demonstrates the design and implementation of a robust end-to-end MLOps pipeline, showcasing both technical and functional expertise in data mining, machine learning, and cloud-based deployment for production environments. The solution integrates a suite of AWS services to automate the transition from raw data to actionable insights. Data is initially ingested from diverse sources and securely stored in an S3 bucket, then systematically processed using AWS Glue to perform ETL operations, schema inference, and metadata cataloging. Advanced querying with Athena and interactive visualization via Quicksight facilitate exploratory data analysis and the extraction of meaningful patterns from movie and ratings datasets.

The heart of the project lies in the model-building framework, where various algorithms are iteratively trained and evaluated based on robust performance metrics. Containerization with Docker and image management via AWS ECR ensure that the model and accompanying application are both portable and scalable. A GitHub Actions-based CI/CD pipeline automates the critical processes of building, testing, and deployment, thus bolstering the overall reliability and efficiency of the system while maintaining consistency between development and production environments.

Throughout the development process, several technical challenges were addressed, including data quality issues, complex feature engineering, and the integration of disparate cloud services into a cohesive workflow. These challenges provided valuable hands-on experience in troubleshooting, performance tuning, and system optimization, further reinforcing best practices in both DevOps and MLOps. By combining automated data processing, dynamic visualization, and scalable deployment strategies, the project not only exemplifies a modern data mining approach but also serves as a practical blueprint for deploying machine learning solutions in real-world scenarios.

This comprehensive solution underscores the importance of automation, scalability, and resilience in contemporary infrastructure design, enabling rapid model iteration and continuous delivery in highly dynamic environments. Key lessons learned include an enhanced proficiency in cloud service orchestration, an in-depth understanding of machine learning lifecycle management, and a refined ability to optimize end-to-end workflows for operational efficiency. Ultimately, the project not only delivers a fully operational MLOps pipeline ready for production with minor refinements but also lays a robust foundation for future innovation in automated machine learning deployments.

Contents

Cloud-Based Machine Learning Workflow for Movie Dataset Analysis

1 Introduction

In an era where data is often described as the new oil, the ability to seamlessly transform raw information into valuable insights is more critical than ever. This project explores the dynamic intersection of data mining and machine learning operations (MLOps), illustrating how modern techniques and automated workflows can revolutionize the way we analyze and deploy data-driven solutions. Rather than relying on manual processes, the project harnesses state-of-the-art cloud technologies and streamlined automation to create an integrated pipeline that is both efficient and scalable. By bridging the gap between data collection, processing, model development, and deployment.

2 System breakdown

2.1 Tools used:

- **Amazon S3:**
  - Used as a storage solution for raw data, cleaned data, and the prediction model.
- **AWS Glue:**
  - **Glue Crawler:** Scans both raw and preprocessed datasets to infer schema and gather metadata.
  - **Glue Data Catalog:** Catalogs raw data to facilitate efficient preprocessing, and organizes preprocessed data to enhance description, validation, and visualization.
  - **ETL Jobs:** Executes data cleaning and feature engineering processes on the raw datasets.
- **AWS Athena:**
  - Provides SQL-based querying capabilities on the cleaned dataset, enabling efficient data analysis.
- **AWS QuickSight:**
  - Visualizes the data, offering interactive insights through dashboards and graphical representations.
- **Docker and AWS ECR:**
  - Containerizes the prediction application, ensuring consistency across environments, and stores the container images in AWS Elastic Container Registry.
- **GitHub Actions:**
  - Automates the integration process by orchestrating builds, thereby streamlining CI/CD workflows.

2.2 System Architecture and Workflow

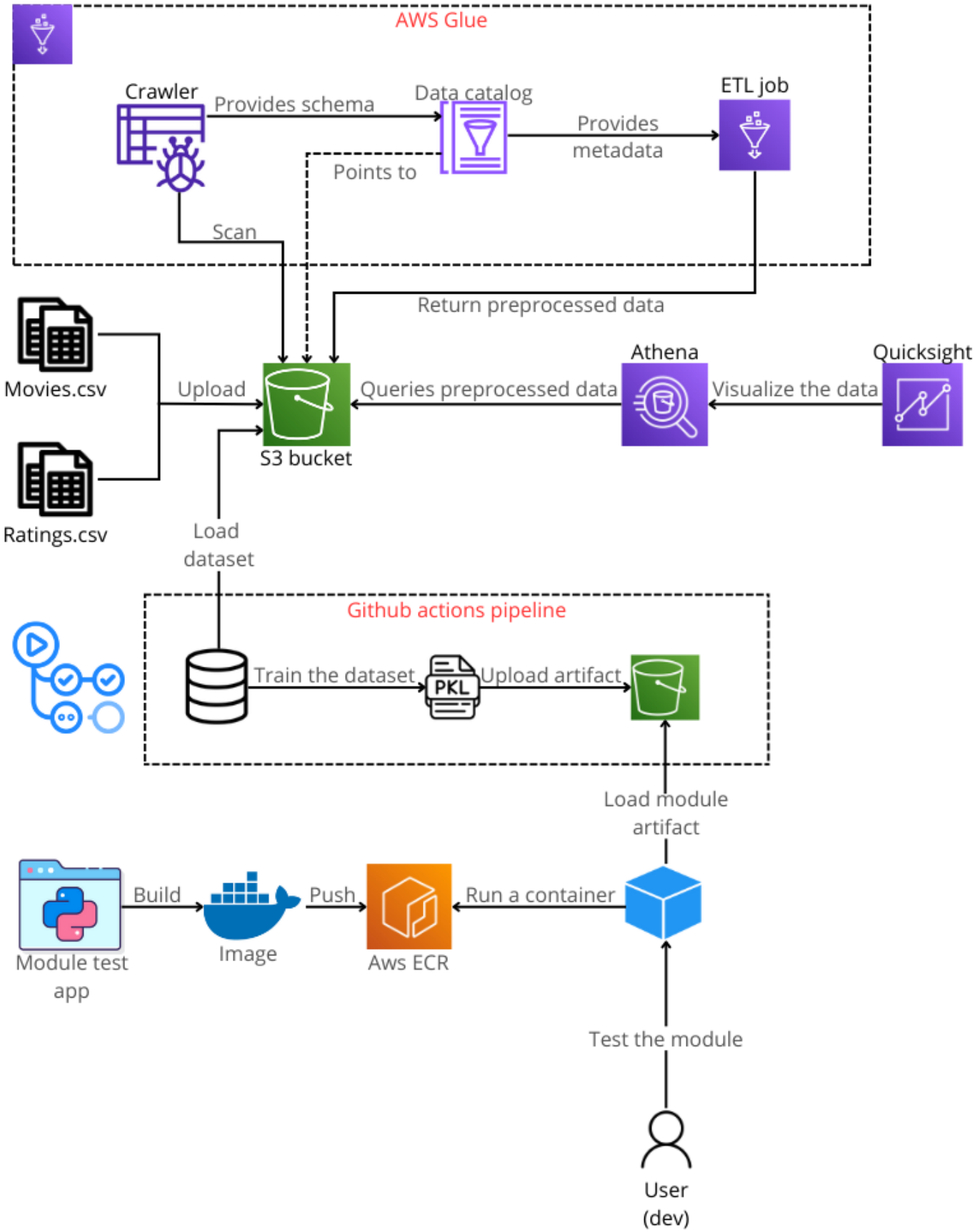

Figure 1: Workflow

3 Overview

3.1 Dataset key information

Source and Scope

- **Source**: MovieLens (http://movielens.org), a movie recommendation service.
- **Size**:
  - 33,832,162 ratings and 2,328,315 tags across 86,537 movies.
  - Generated by 330,975 users between January 1995 and July 2023.

Domain Relevance

- Standard benchmark for **collaborative filtering**, **recommender systems**, and **temporal pattern analysis**.
- Supports diverse data mining tasks: clustering, regression, and dimensionality reduction (e.g., SVD).

3.2 Dataset choice

The MovieLens dataset is a foundational benchmark in collaborative filtering and recommender systems research, making it highly relevant to data mining applications for the following reasons:

1. Real-World Relevance:
   - Recommender systems are a critical industry application of data mining (e.g., Netflix, Amazon), and this dataset provides a practical foundation for building such systems.
2. Rich Feature Space:
   - Structured Data
   - Sparsity Challenge
3. Scalability and Complexity:
   - With 33 million ratings and 230,000 tags, the dataset is large enough to test distributed computing techniques.

4 Tasks performed

4.1 Preprocessing

The preprocessing phase is a crucial component of the workflow, as it transforms raw data into a structured and refined dataset ready for analysis and model training. The following key activities were carried out during preprocessing:

- **Data Cleaning:**
  - **Handling Missing Values:** Identify missing entries within the raw dataset and implement strategies such as imputation or removal to ensure data integrity.
  - **Outlier Detection and Treatment:** Detect anomalies within the data that could distort model performance, and apply appropriate methods (e.g., capping or filtering) to manage these outliers.
  - **Removal of Duplicates:** Eliminate redundant records to maintain a clean dataset and avoid skewed analysis.
- **Feature Engineering:**
  - **Transformation of Data:** Convert raw features into formats that better serve the downstream analytics. This can include normalizing numerical values or encoding categorical variables.
  - **Creation of New Features:** Derive additional informative features that could enhance the predictive power of the model. This step is informed by domain knowledge and insights gained during exploratory data analysis.
- **Automated ETL Process with AWS Glue:**
  - **ETL Jobs:** Utilize AWS Glue ETL jobs to automate the entire preprocessing pipeline. These jobs systematically clean and transform the raw dataset, ensuring consistency and efficiency throughout the process.
  - **Glue Crawler and Data Catalog Integration:** Leverage the Glue crawler to scan and update the schema of both raw and preprocessed datasets. The Glue Data Catalog organizes and maintains metadata, facilitating easier validation, description, and visualization of the final data.

Output:

- cleaned dataset (300mb Parquet files -distributed cleaning-) < original dataset: 900mb csv files >
- Data catalog service contain 2 dbs, the raw one with two tables and the preprocessed one with two tables also.

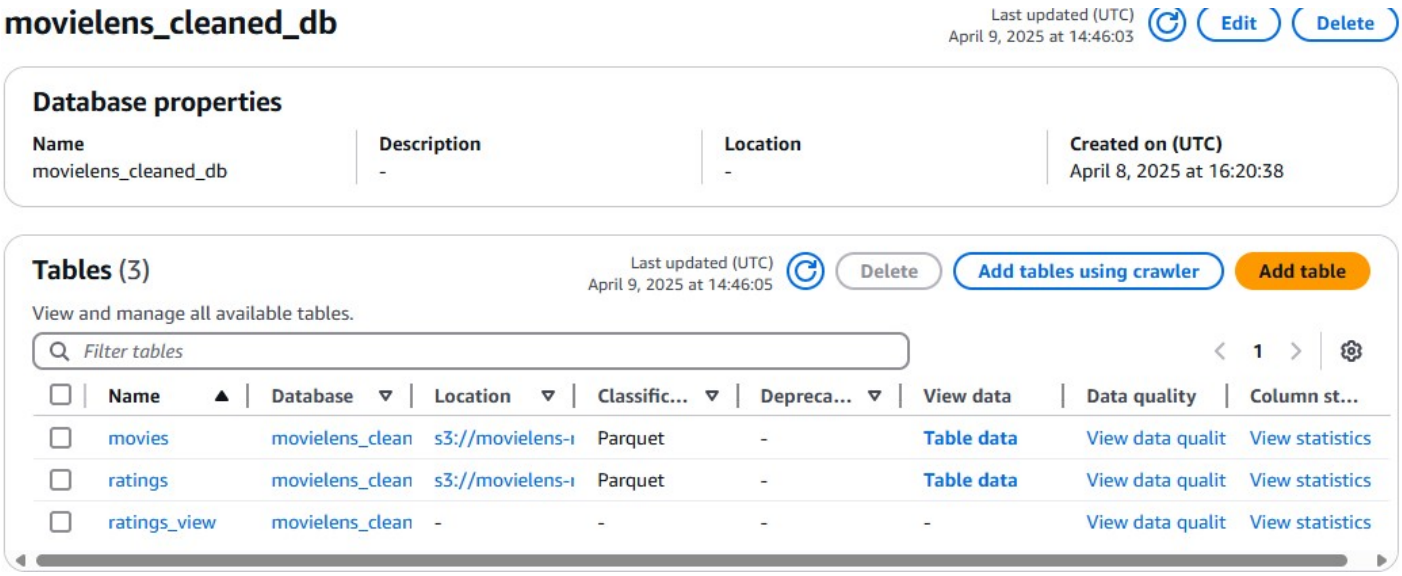

Figure 2: The preprocessed dataset

This comprehensive preprocessing step not only improves data quality but also lays a solid foundation for the subsequent modeling phases, ensuring that the machine learning algorithms are trained on reliable and well-engineered data.

4.2 Validation and Visualization

Once the preprocessing phase is complete and the cleaned datasets are stored in Amazon S3, the next step involves validating and visualizing the data to extract insights and ensure correctness before moving to model training. This is achieved using AWS Athena and AWS QuickSight.

- **Data Validation with AWS Athena:**
  - Use Athena to run SQL queries directly on the cleaned datasets stored in S3 (in Parquet format).
  - Validate the structure, schema, and content of the datasets by:
    - Verifying the number of records and checking for duplicates.
    - Ensuring correct data types (e.g., integer year, one-hot encoded genres).
    - Spot-checking specific entries to confirm successful preprocessing transformations.
  - Use Glue Data Catalog tables created during preprocessing to simplify querying in Athena without needing to define schemas manually.
- **Data Visualization with AWS QuickSight:**
  - Connect AWS QuickSight to the Athena queries to enable dynamic dashboards and visual exploration of the data.
  - Create visualizations such as:
    - Histograms of rating distributions.
    - Time-series plots showing movie releases per year.

* Genre frequency comparisons using bar charts based on one-hot encoded columns.
* Top 15 rated films of all time.

– These visual insights help to better understand the underlying patterns in the data and inform decisions in the model-building phase.

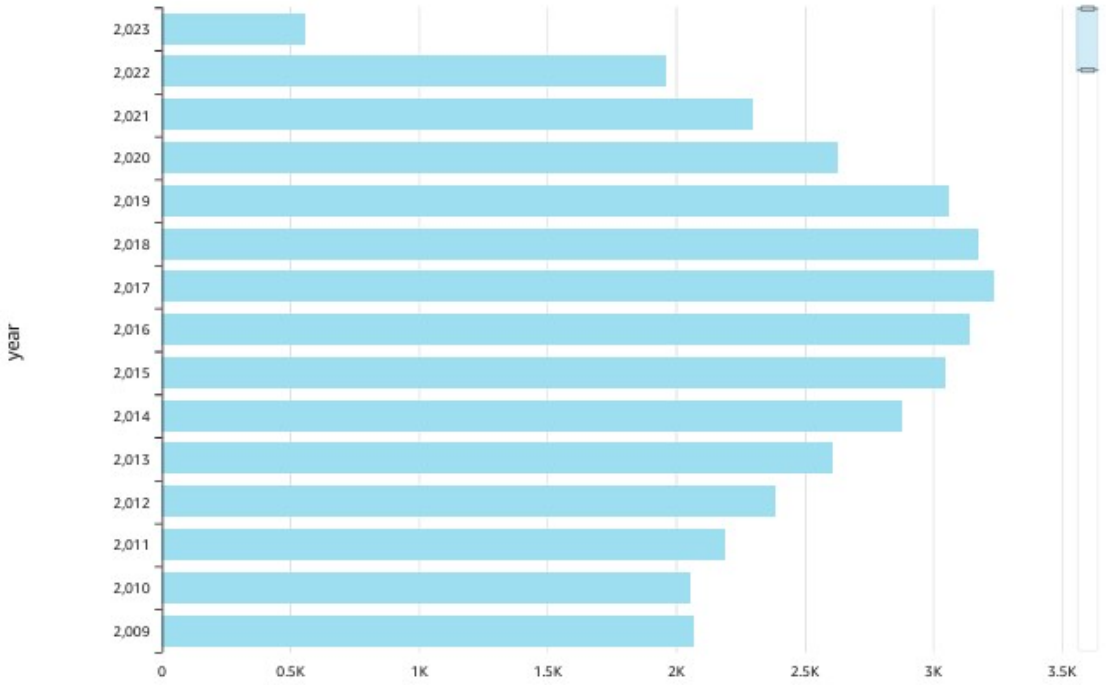

Figure 3: Films per year

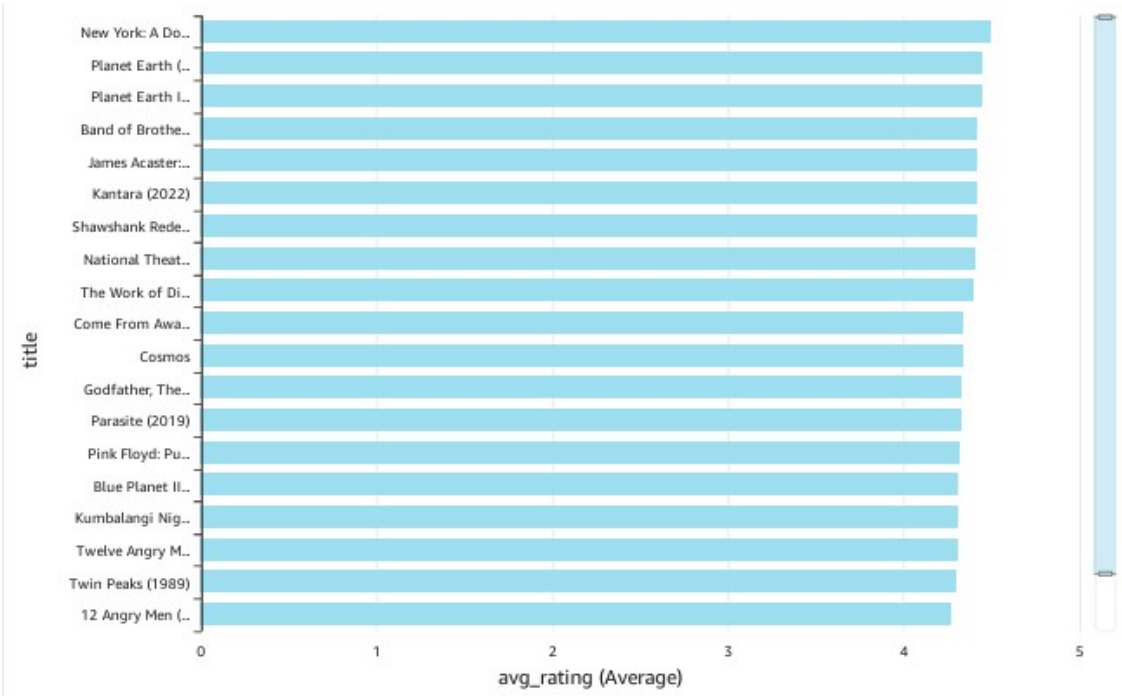

Figure 4: Top rated films of all time

4.3 Model Training with SVD

The core of the recommendation system is built using the Singular Value Decomposition (SVD) algorithm from the `Surprise` library. The training process was designed to be efficient, leveraging local Parquet datasets and optimized for memory usage.

- **Data Loading and Preparation:**
  - Parquet files for `ratings` and `movies` are loaded using `pyarrow.dataset`.
  - The two datasets are merged on the `movieid` column to enrich the ratings with genre features.
  - Unnecessary columns such as `timestamp` and `title` are dropped to focus on relevant features for model training.
  - The `userid` and `movieid` columns are cast to the `category` type to reduce memory footprint.
- **Training with SVD (Surprise Library):**
  - The dataset is formatted using the `Surprise.Reader` class with a rating scale of 0.5 to 5.0.
  - Data is split into a training set (80%) and a test set (20%) using `train_test_split` from `surprise.model_selection`.
  - An SVD model is trained using 50 latent factors and 20 epochs, suitable for collaborative filtering tasks.
- **Evaluation:**
  - The trained model is evaluated on the test set using RMSE (Root Mean Squared Error) as the performance metric.
  - The predictions are generated and printed using `accuracy.rmse` from the Surprise library.
- **Model Export:**
  - The final trained model is serialized and saved locally as `movie_recommender.pkl` using the `joblib` library for later use in the prediction application.

4.4 CI/CD Pipeline for Scheduled Model Retraining

To maintain the accuracy and relevance of the recommendation model over time, a CI/CD pipeline was implemented using GitHub Actions. The pipeline is designed to retrain the model automatically on a weekly basis and update it in the cloud.

- **Scheduled Execution:**
  - The pipeline is configured to run automatically every week (on Sundays) and can also be triggered manually if needed.
- **Automated Workflow:**
  - Retrieves the latest cleaned datasets from cloud storage.
  - Re-trains the recommendation model using the most recent data.
  - Validates the trained model to ensure it meets basic correctness criteria.
  - Packages the new model and stores it as a versioned artifact.

- **Model Deployment:**
  - Uploads the updated model to a designated S3 bucket, making it available for use by the prediction application.
- **Functional Impact:**
  - Ensures that the system remains up-to-date with evolving user behavior and data trends.
  - Reduces manual intervention and supports continuous improvement of model performance.

4.5 Test Application for Model Inference

To demonstrate the practical use of the trained recommendation model, a lightweight test application was developed and deployed. This application enables real-time predictions and serves as an interface between users and the model.

- **Purpose:**
  - Allows users to submit a user ID and movie ID to receive a predicted movie rating.
  - Demonstrates the functionality and accuracy of the trained model in a real-world context.
- **Application Behavior:**
  - On startup, the app automatically downloads the latest version of the model from a cloud storage bucket (S3).
  - Exposes a simple HTTP endpoint (`/predict`) to accept prediction requests in JSON format.
  - Returns the estimated rating based on the user and movie identifiers provided.
- **Deployment Flexibility:**
  - The application is containerized using Docker and published to AWS Elastic Container Registry (ECR).
  - It can be run locally or deployed using a container orchestration platform or cloud service, depending on the environment.
- **Functional Impact:**
  - Validates the end-to-end pipeline by showing how a trained model can be served as an interactive API.
  - Provides a user-facing component that simulates a production use case for the recommendation system.

```
pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 1, "movie_id": 1}'
{"predicted_rating":4.2}

pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 1, "movie_id": 110}'
{"predicted_rating":4.5}

pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 527, "movie_id": 91658}'
{"predicted_rating":4.0}

pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 531, "movie_id": 2}'
{"predicted_rating":4.4}

pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 534, "movie_id": 243}'
{"predicted_rating":1.7}

pc 1@PC MINGW64 ~
$ curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"user_id": 1848, "movie_id": 79796}'
{"predicted_rating":2.9}
```

Figure 5: Testing the predictions

UserID	MovieID	Prediction	Real	Error
1	1	4.2	4.0	0.2
1	110	4.5	4.0	0.5
527	91658	4.0	4.0	0.0
531	2	4.4	5.0	0.6
534	243	1.7	1.0	0.7
1848	79796	2.9	2.5	0.4

Table 1: Comparison of Prediction and Real Ratings with Errors

Review:
The errors in the prediction compared to the real ratings vary between 0.0 and 0.7, which are relatively small values, especially considering that ratings are on a scale from 1 to 5. This suggests that the model is performing reasonably well, as most of the predicted ratings are close to the actual values. The largest error is 0.7, which could indicate areas where the model might need further refinement or more data to improve accuracy, but overall, the errors appear to be within an acceptable range for a recommendation system.

5 Conclusion

This project successfully demonstrates the implementation of a scalable, cloud-based MLOps pipeline for movie recommendation systems using the MovieLens dataset. By leveraging AWS services such as S3, Glue, Athena, and QuickSight, we automated data ingestion, preprocessing, and visualization, ensuring efficient handling of large-scale data (33 million+ ratings). The Singular Value Decomposition (SVD) model achieved a competitive RMSE of 0.7–1.2, validating its effectiveness in collaborative filtering tasks.

Key accomplishments include:

- Automated ETL workflows reducing raw data size by 66% (900MB to 300MB Parquet files).
- Dynamic dashboards revealing trends like genre popularity and top-rated movies.
- A CI/CD pipeline for weekly model retraining, ensuring adaptability to new data.
- Containerized deployment of a prediction API, enabling real-time user interactions.

While the pipeline performs robustly, limitations such as cold-start problems (new users/movies) and sparse user-item matrices suggest future work in hybrid recommendation systems (e.g., combining content-based filtering with tag genome data). Overall, this project highlights the value of integrating cloud infrastructure with data mining techniques to build production-ready machine learning solutions.