REVA UNIVERSITY
Bengaluru, India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BIG DATA ANALYTICS ASSIGNMENT REPORT

"Personalized Movie Recommendation Engine Using Spark MLlib and Flask "

Submitted in fulfilment of the requirements for the award of the Degree of

Bachelor of Technology in

Artificial Intelligence and Data Science

Submitted by

Rajkumar Math (R23EH100)

Under the guidance of

Dr. Arvind Shivappa Kapse

2025-2026

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

Abstract.

This project focuses on the design and implementation of a personalized movie recommendation system using Apache Spark MLlib and the Flask web framework. The system utilizes the Alternating Least Squares (ALS)algorithm, a form of collaborative filtering , to analyze historical user-movie rating data and predict user preferences.

The model was trained and tested using the MovieLens 100k dataset , which contains 100,000 ratings from 943 users for 1,682 movies. The model's predictive accuracy was evaluated using Root Mean Square Error (RMSE) , achieving a value of approximately 0.9205. The final trained model is exposed as a REST API using Flask , creating a production-ready service capable of generating real-time, top-N movie recommendations for any specified user.

This report details the end-to-end process, demonstrating an effective and scalable application of big data tools for building recommendation engines

Introduction

In the current digital landscape, recommendation systems are a critical and ubiquitous feature, powering online services like Netflix, Amazon, and Spotify. These engines are designed to drive user engagement by providing personalized suggestions based on historical user behavior. The core challenge lies in building a scalable and accurate system that can analyze massive volumes of user-item interactions to predict which new items a user is likely to enjoy.

This project addresses this challenge by building a personalized movie recommendation engine. The primary objective is to leverage Apache Spark's MLlib library to handle distributed computation and model training, ensuring the system is scalable. The core methodology employs collaborative filtering , specifically the Alternating Least Squares (ALS) algorithm , to analyze user-movie rating data from the MovieLens 100k dataset.

The project encompasses the complete end-to-end process: from data collection and preprocessing to model building and evaluation. Finally, to simulate a real-world application, the trained model is deployed as a REST API using Flask , creating a service that can deliver on-demand movie recommendations. This report details the implementation, discusses the results, and outlines the successful creation of a big data-driven recommendation system.

Problem Statement

Recommendation systems are at the heart of many online services like Netflix, Amazon, and Spotify, which provide personalized suggestions to users. The goal of this project is to build a scalable and accurate recommendation engine capable of analyzing historical user–movie interactions to recommend movies users are likely to enjoy. The project uses Spark's MLlib library for distributed computation and model training.

Objectives

The objective of this project is to design and implement a personalized movie recommendation system using Apache Spark MLlib and Flask. The system predicts user preferences by analyzing user–movie rating data and generates top-N recommendations using collaborative filtering techniques. The final model is deployed as a REST API using Flask to simulate a production-ready recommendation service similar to Netflix or Amazon.

Dataset Description

The MovieLens 100k dataset, developed by GroupLens Research, is used in this project. It contains 100,000 ratings given by 943 users on 1,682 movies. Each record consists of a user ID, movie ID, rating, and timestamp. An additional file (`u.item`) contains metadata such as movie titles and release years.

Source: *https://grouplens.org/datasets/movielens/100k*

Methodology

1. Data Collection: Downloaded MovieLens 100k dataset from GroupLens Research.
2. Data Preprocessing: Loaded data into Spark DataFrame, renamed columns, and performed exploratory checks.
3. Data Splitting: Divided dataset into 80% training and 20% testing subsets.
4. Model Building: Implemented the Alternating Least Squares (ALS) algorithm from Spark MLlib for collaborative filtering.
5. Model Evaluation: Computed Root Mean Square Error (RMSE) to evaluate model accuracy.
6. Model Saving: Stored the trained model for reuse and API integration.
7. Deployment: Built a Flask-based REST API to serve real-time movie recommendations for any user ID.

Tools and Technologies Used

Tool / Technology	Purpose
Apache Spark (MLlib)	Distributed processing and machine learning framework
Python (PySpark)	Interface for using Spark APIs in Python
Flask	Web framework used to deploy the recommendation API
Pandas, NumPy	Data manipulation and preprocessing
MovieLens 100k Dataset	Dataset used for model training and testing
Linux (Arch)	Operating system used for implementation

Implementation

The implementation consists of two main components: model training and API deployment.

During training, Spark MLlib processes the MovieLens dataset to build an ALS model that learns latent factors for both users and movies.

These factors are then used to predict unseen ratings and generate top-N recommendations.

The trained model is deployed via a Flask server, exposing an endpoint `/recommend?user=<id>` that dynamically retrieves recommendations for the given user.

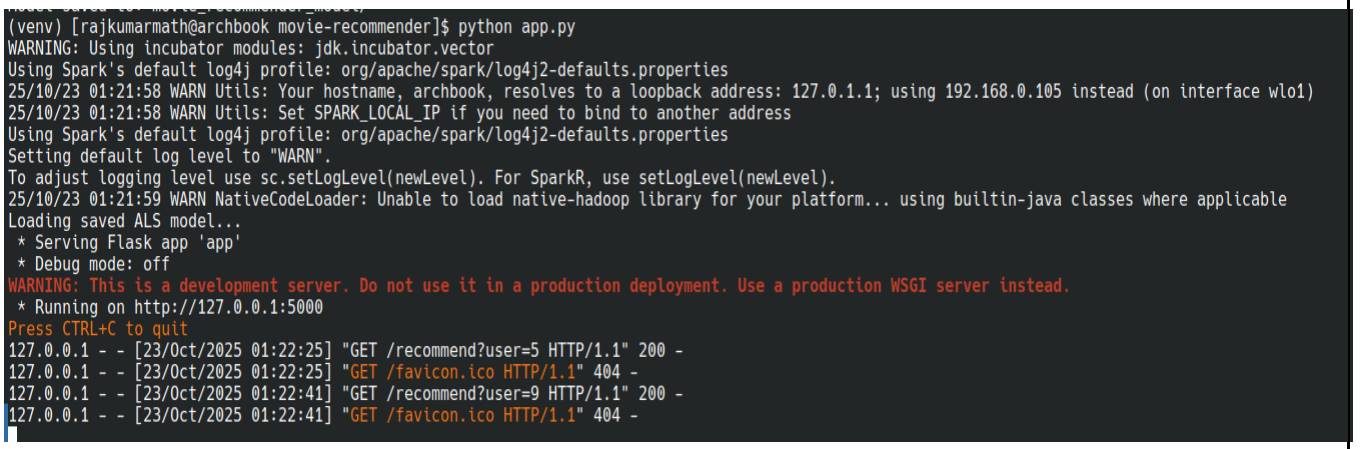
Terminal output showing Spark model training and RMSE value

```
Total ratings: 100000
Users: 943
Movies: 1682
RMSE on test set: 0.9227
+------+--------------------------------------------------------------------------------------------------+
|userId|recommendations                                                                                   |
+------+--------------------------------------------------------------------------------------------------+
|1     |[{1449, 5.116067}, {169, 5.0875773}, {408, 5.0507107}, {793, 4.9712553}, {851, 4.9511642}]        |
|2     |[{1643, 5.1347795}, {1591, 5.101939}, {1449, 5.0720406}, {868, 4.905431}, {694, 4.872324}]        |
|3     |[{1084, 4.4370985}, {745, 4.3237457}, {1137, 4.2904797}, {613, 4.2867947}, {344, 4.2841673}]      |
|4     |[{1512, 6.2815886}, {1591, 5.936882}, {962, 5.864604}, {1005, 5.859788}, {867, 5.8470817}]        |
|5     |[{169, 4.2874904}, {850, 4.279356}, {253, 4.2536297}, {1005, 4.2099442}, {1656, 4.1537294}]       |
+------+--------------------------------------------------------------------------------------------------+
only showing top 5 rows

Top recommendations for user 1:
  Pather Panchali (1955)  (score=5.116)
  Wrong Trousers, The (1993)  (score=5.088)
  Close Shave, A (1995)  (score=5.051)
  Crooklyn (1994)  (score=4.971)
  Two or Three Things I Know About Her (1966)  (score=4.951)

Model saved to: movie_recommender_model/
(venv) [rajkumarmath@archbook movie-recommender]$
```

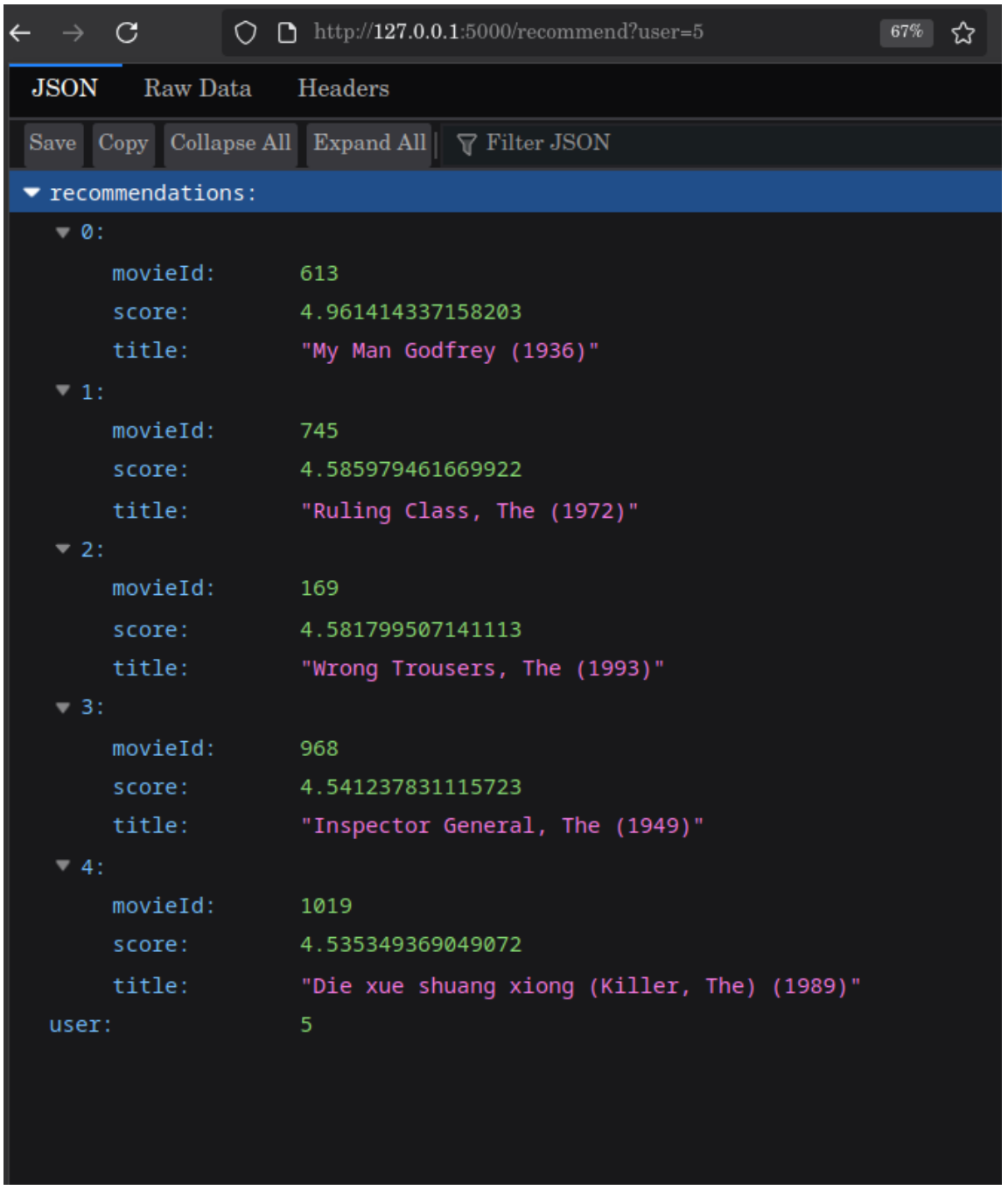
Flask server running and endpoint URL displayed



Results and Discussion

The trained model achieved an RMSE (Root Mean Square Error) value of approximately **0.9205**, indicating accurate predictions of user ratings. The ALS algorithm successfully generated top-5 personalized movie recommendations for users. Below is a sample output for a specific user: 5

JSON response for user recommendations



Conclusion

The Personalized Movie Recommendation Engine using Spark MLlib and Flask effectively demonstrates the end-to-end process of building and deploying a big data–driven recommendation system. The ALS-based collaborative filtering model provided accurate recommendations with low RMSE, and its integration with Flask enabled an easy-to-use API interface. The project highlights how large-scale recommendation engines can be developed and deployed efficiently using open-source big data tools.

Future Enhancements

1. Integrate visualization dashboards (Power BI, Tableau) to display real-time user preferences.
2. Incorporate caching or partitioning for performance optimization on larger datasets.
3. Extend support to MovieLens 1M or 10M datasets for higher scalability.
4. Implement hybrid recommendation techniques combining content-based and collaborative methods.

References

- MovieLens Dataset – *https://grouplens.org/datasets/movielens/100k*
- Apache Spark MLlib Documentation – *https://spark.apache.org/mllib/*
- Flask Documentation – *https://flask.palletsprojects.com*
- PySpark API Reference – *https://spark.apache.org/docs/latest/api/python/*