Big Data Analytics  
June 6, 2025  
Aashma Thapa Magar

**Pumpkinmeter Recommendation System**

**Source Project URL**

<https://www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw>

**Introduction**

This project explores the use of collaborative filtering for personalized movie recommendations using Apache Spark and the Movie Lens dataset. A startup called Ripe Pumpkins aims to introduce a “Pumpkinmeter” score that mirrors the success of recommender systems used by platforms like Netflix. Using Spark’s ALS (Alternating Least Squares) algorithm, the goal is to create a model that delivers personalized top 15 movie recommendations for two new users, under two scenarios: one using the full dataset and the other filtered by movie popularity.

**Dataset Used**

* **Name**: MovieLens 25M Full Dataset
* **Source Address**: <https://grouplens.org/datasets/movielens/25m/>
* **Summary**:The dataset includes 27 million ratings, over 58,000 movies, and tagging information from 280,000 users.
* **Fields** (See appendix):userId, movieId, rating, timestamp, title, genres

**Technical Details**

This project uses Apache Spark’s MLlib library and its implementation of ALS to perform collaborative filtering on large-scale data. Spark was set up on a remote VM using Jupyter Notebook, and all data preprocessing, model training, and recommendations were performed using PySpark.

**Debugging Details – Challenges and Solutions**

* **Challenge 1**: Jupyter port issues → resolved using alternate port 8890
* **Challenge 2**: Java compatibility issue → fixed by upgrading to Java 17
* **Challenge 3**: File path errors → resolved by confirming absolute paths in /home/ubuntu/
* **Challenge 4**: Schema mismatch during .union() → resolved by removing timestamp column
* **Achievement**: Successfully completed all 4 test cases using filtered + full data scenarios, and generated top-15 titles with mapped movie names.

**Results**

| **User** | **Scenario** | **Description** | **Output** |
| --- | --- | --- | --- |
| Aashma | Scenario 1 (≥25) | Diverse + personalized recommendations | Top 15 list |
| Aashma | Scenario 2 (≥100) | Mainstream + reliable suggestions | Top 15 list |
| Friend | Scenario 1 (≥25) | Broader movie list | Top 15 list |
| Friend | Scenario 2 (≥100) | Safe, popular titles | Top 15 list |

Titles were extracted using .join() with movies.csv, and ratings were sorted by predicted score.

**Insight – Business Implications**

Scenario 1 provided more personalized and diverse recommendations but included less reliable data. Scenario 2 delivered mainstream and highly rated content, making it ideal for new users. A hybrid approach is recommended: start with Scenario 2 for trust-building, then shift to Scenario 1 as the user profile matures. This strategy balances personalization with reliability and enhances retention for platforms like Ripe Pumpkins.

**References**

1. Codementor Recommender Tutorial  
   <https://www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw>
2. Grouplens MovieLens Dataset  
   <https://grouplens.org/datasets/movielens/>
3. Apache Spark ALS Docs  
   <https://spark.apache.org/docs/latest/ml-collaborative-filtering.html>
4. In-class Lecture Notes – Spring 2025

**Appendix**

**Dataset Fields**:

* ratings.csv: userId, movieId, rating, timestamp
* movies.csv: movieId, title, genres
* tags.csv: userId, movieId, tag, timestamp

**Jupyter Notebook Paths**:

* /home/ubuntu/ratings.csv
* /home/ubuntu/movies.csv