Business Proposal Name:
Building a Movie Recommendation Service with Apache Spark & Flask

Source Project URL:

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

Introduction - what is this use case**:**

This project demonstrates the development of a movie recommendation service using Apache Spark and Flask. It employs collaborative filtering techniques to provide personalized movie suggestions based on user ratings, showcasing how machine learning algorithms can be effectively scaled using distributed computing.

Dataset used – name, source address, summary, (list of all fields of dataset, as appendix):

The MovieLens dataset, sourced from GroupLens, was utilized. The small dataset contains 100,000 ratings and 3,600 tags for 9,000 movies by 600 users, while the complete dataset includes 27,000,000 ratings and 1,100,000 tags for 58,000 movies by 280,000 users.

Technical Details: Which aspect of Spark is applied**:**

- SparkContext Configuration: Configured for local mode to enable testing and development.
- Data Loading and Parsing: Utilized RDD (Resilient Distributed Dataset) to load, parse, and process data.
- Collaborative Filtering: Implemented Alternating Least Squares (ALS) algorithm for collaborative filtering to build the recommendation model.
- Model Training and Evaluation: Split the dataset into training, validation, and test sets to train and evaluate the model.
- Performance Metrics: Calculated Root Mean Squared Error (RMSE) to evaluate the accuracy of the recommendation model.

Debugging Details:

Challenges included managing large datasets and optimizing ALS parameters. Modifications involved adjusting dataset splitting ratios and ensuring the system handled new user ratings efficiently. The key achievements included building a scalable recommendation system and achieving a lower RMSE, demonstrating model accuracy improvements. Efficient data handling and parameter tuning were critical coding practices.

Results: Discuss Spark output with above test sets and scenarios

The ALS model was trained and validated, achieving an RMSE of 0.918 on the small dataset and 0.827 on the complete dataset. Personalized recommendations were successfully generated, demonstrating the system's ability to cater to user preferences effectively.

When filtering the full dataset to include only movies with at least 25 ratings, the recommendation system was able to provide a list of top movies that not only had high predicted ratings but also had a sufficient number of user ratings to ensure the recommendations were reliable. This filtering helps to remove obscure or less popular movies that might have high ratings from a few users but do not have widespread appeal.

When the filtering threshold is raised to include only movies with at least 100 ratings, the recommendation list tends to feature more well-known and widely-viewed movies. This ensures the recommendations are not only based on high ratings but also on a significant amount of user feedback, making the recommendations more robust and credible.

Insight:

This project highlights the efficacy of collaborative filtering and distributed computing in creating scalable recommendation systems. Business implications include enhanced user engagement through personalized recommendations, with potential applications in e-commerce and content streaming. Spark's distributed capabilities significantly improve processing efficiency, enabling the handling of large user bases and extensive datasets.

Quality vs. Quantity: Lowering the rating count threshold to 25 allows for discovering hidden gems that might not be widely known but are appreciated by a niche audience. However, raising the threshold to 100 prioritizes well-known, universally acclaimed films that have stood the test of time and user scrutiny.

Recommendation Reliability: Higher rating thresholds generally result in more reliable recommendations. Movies that have been rated by a large number of users are less likely to be anomalies and more likely to be genuinely appreciated.

Diverse Recommendations: The lower threshold (25 ratings) tends to surface more diverse and eclectic recommendations, which can be particularly useful for users looking to explore beyond mainstream options.

Mainstream Favorites: The higher threshold (100 ratings) tends to highlight mainstream favorites, ensuring that the recommendations are more aligned with widely recognized quality.

Reference and Citation:

Blog, Netflix Technology. “System Architectures for Personalization and Recommendation.” Medium, 20 June 2018, <netflixtechblog.com/system-architectures-for-personalization-and-recommendation-e081aa94b5d8>.

Collaborative Filtering - RDD-based API - Spark 3.5.1 Documentation. spark.apache.org/docs/latest/mllib-collaborative-filtering.html.

Dianes, Jose A. “Building a Movie Recommendation Service With Apache Spark &Amp; Flask - Part 1.” Codementor, 14 Sept. 2015, <www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw>.

Lecture 8 – Module 8: “ Data Processing Pattern II”

Lecture 10 – Module 10: “Data Analytics Workflows”