Bright Kyeremeh
DS501
Introduction To Data Science
March 9, 2023

Movie Recommendation System Using Apache Spark and Python

End-to-end Big Data Project To Build A Movie Recommendation System Using Apache Spark and Python

Introduction:

In today's world, recommendation systems are widely used to help people make choices by suggesting items based on their preferences. One of the most popular examples of recommendation systems is the movie recommendation system. Building a movie recommendation system is a big data project that involves collecting data from various sources, preprocessing it, extracting features, building models, evaluating performance, and deploying the system. In this report, we will discuss an end-to-end big data project that involves building a movie recommendation system using Apache Spark and Python.

Data Collection:

The first step of this project is to collect movie data from various sources such as IMDb, Rotten Tomatoes, etc. We used web scraping techniques to extract the required data and stored it in a Hadoop Distributed File System (HDFS). We collected data such as movie title, director, cast, genre, release year, rating, and reviews. The collected data was in unstructured format, so we had to preprocess it before we could extract features.

Data Preprocessing:

The second step was to preprocess the collected data to clean and prepare it for analysis. This involved techniques such as removing duplicates, handling missing values, and performing

feature engineering. We used PySpark, the Python API for Apache Spark, to perform these tasks in a distributed manner. PySpark allowed us to process big data efficiently by distributing the data across a cluster of machines and performing parallel processing.

Feature Extraction:

After the data had been preprocessed, we extracted relevant features from the movie data that could be used for recommendation. We used techniques such as collaborative filtering, content-based filtering, and hybrid filtering to extract features. We used PySpark's MLlib library to perform these tasks. Collaborative filtering is a technique that recommends items based on the similarity between users' preferences. Content-based filtering recommends items based on the similarity between the features of the items. Hybrid filtering combines both collaborative and content-based filtering techniques.

Model Building:

Once the features had been extracted, we built a machine learning model to recommend movies to users. We used algorithms such as Matrix Factorization, Alternating Least Squares, or K-Nearest Neighbors to build and train the model. We used PySpark's MLlib library to build the model. Matrix Factorization is a technique that factors the user-item matrix into two low-rank matrices and uses them to make recommendations. Alternating Least Squares is an iterative optimization algorithm that minimizes the error between the predicted and actual ratings. K-Nearest Neighbors is a technique that recommends items based on the preferences of users with similar tastes.

Model Evaluation:

After the model had been built, we evaluated its performance using metrics such as precision, recall, and F1 score. We used PySpark's MLlib library to perform this task. Precision measures the fraction of recommended items that are relevant, recall measures the fraction of relevant items that are recommended, and F1 score is the harmonic mean of precision and recall. We evaluated the model on a test set to ensure that it was not overfitting the training data.

Deployment and Integration:

Finally, we deployed the model and integrated it with other systems to provide movie recommendations to users. We used tools such as Flask, Docker, and Kubernetes to build scalable and robust applications. Flask is a micro web framework for Python that allows us to build web applications easily. Docker is a platform that allows us to create, deploy, and run applications in

containers. Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

Conclusion:

In conclusion, building a movie recommendation system using Apache Spark and Python involves collecting data, preprocessing it, extracting relevant features, building a machine learning model, evaluating its performance, and deploying and integrating the model with other systems. This end-to-end big data project can be useful for movie companies to promote their content and for users to discover new movies. The project showcases the importance of big data in building accurate and reliable recommendation systems, which can have a significant impact on user behavior and decision-making.

Weekly Schedule:

- Week 1: Data collection
- Week 2 : Data preprocessing
- Week 3: Performing feature extraction
- Week 4: Building machine learning using PySpark's MLlib
- Week 5: Evaluating and tuning the model for best performance
- Week 6: Deployment and integration of the model using Python flask or Heroku cloud or other cloud platforms as described above