Pi Movie Recommendation with MLlib

Natnael Haile
ID: 20007
July 18, 2024

Table of Contents

Introduction

Design

Implementation

Test

Enhancement Ideas

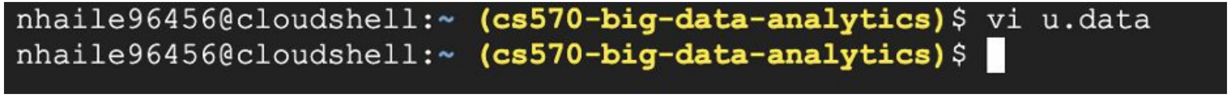
Conclusion

References

Introduction

Objective of the Project:

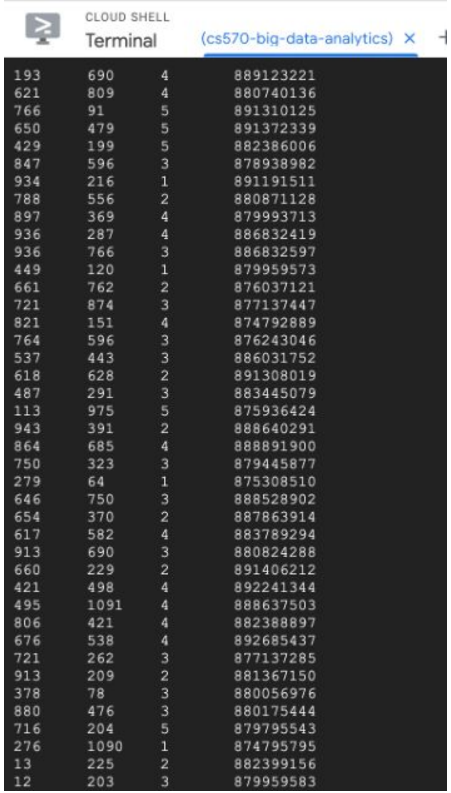
- **Develop a Movie Recommendation System:**
  - Implement a collaborative filtering model using PySpark's MLlib.
  - Provide personalized movie recommendations based on user ratings.
- **Data Preparation:**
  - Load and transform movie rating data to fit the recommendation system's requirements.
  - Ensure data is clean and properly formatted for analysis.
- **Model Training and Evaluation:**
  - Train the recommendation model using the Alternating Least Squares (ALS) algorithm.
  - Evaluate model performance by calculating Root-Mean-Square Error (RMSE).
- **Deployment:**
  - Deploy the model on Google Cloud Platform using Dataproc for scalable processing.
  - Save and manage the trained model in Google Cloud Storage for future use.



```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi u.data
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ 
```

Design: System Overview

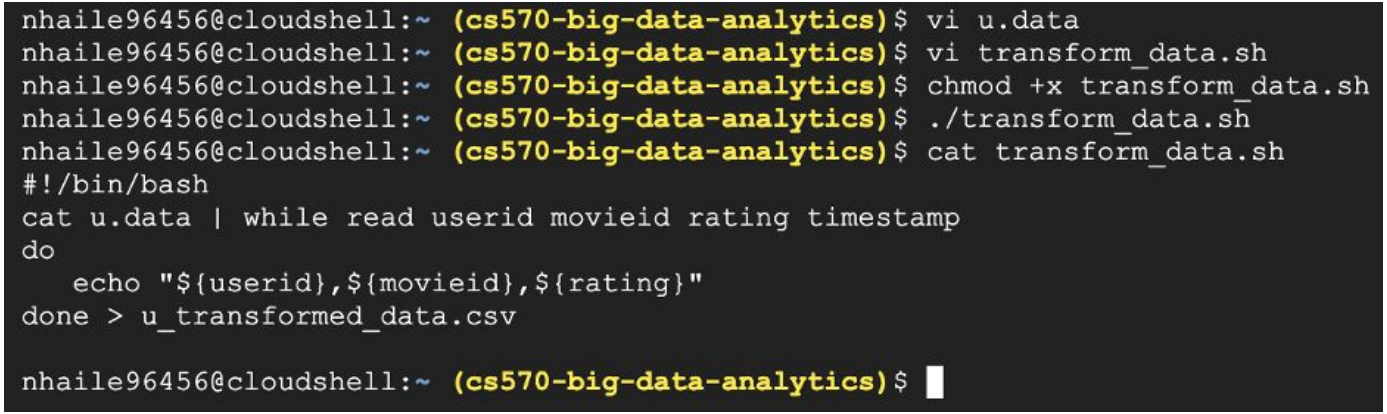
- **Data Collection:**
  - Obtain and store user-movie rating data in the format (UserID, MovieID, rating, Timestamp).
  - Save the initial data to a file named u.data.



```
193     690     4       889123221
621     809     4       880740136
766     91      5       891310125
650     479     5       891372339
429     199     5       882386006
847     596     3       878938982
934     216     1       891191511
788     556     2       880871128
897     369     4       879993713
936     287     4       886832419
936     766     3       886832597
449     120     1       879959573
661     762     2       876037121
721     874     3       877137447
821     151     4       874792889
764     596     3       876243046
537     443     3       886031752
618     628     2       891308019
487     291     3       883445079
113     975     5       875936424
943     391     2       888640291
864     685     4       888891900
750     323     3       879445877
279     64      1       875308510
646     750     3       888528902
654     370     2       887863914
617     582     4       883789294
913     690     3       880824288
660     229     2       891406212
421     498     4       892241344
495     1091    4       888637503
806     421     4       882388897
676     538     4       892685437
721     262     3       877137285
913     209     2       881367150
378     78      3       880056976
880     476     3       880175444
716     204     5       879795543
276     1090    1       874795795
13      225     2       882399156
12      203     3       879959583
```

Design: System Overview

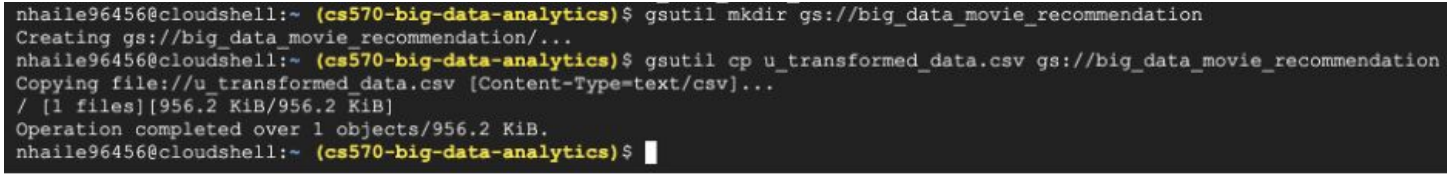
- **Data Transformation:**
  - Use a bash script to preprocess the data, removing the timestamp and converting it to the format (UserID, MovieID, rating).
  - Resulting file: u_transformed_data.csv.



```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi u.data
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi transform_data.sh
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ chmod +x transform_data.sh
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ ./transform_data.sh
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat transform_data.sh
#!/bin/bash
cat u.data | while read userid movieid rating timestamp
do
   echo "${userid},${movieid},${rating}"
done > u_transformed_data.csv

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ 
```

Implementation: Environment Setup

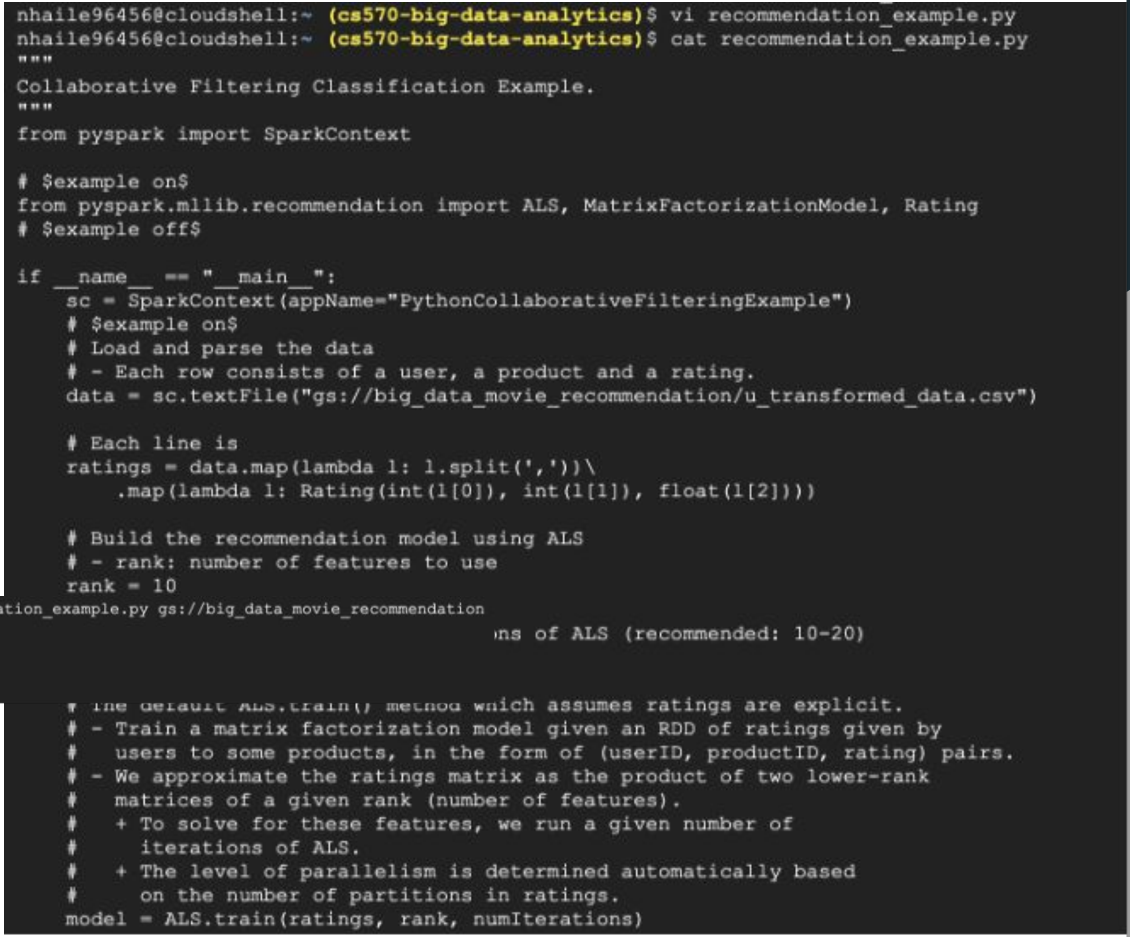
- **Cloud Storage Setup:**
  - Create a Google Cloud Storage bucket for storing the transformed data and model.
  - Upload u_transformed_data.csv to the bucket.



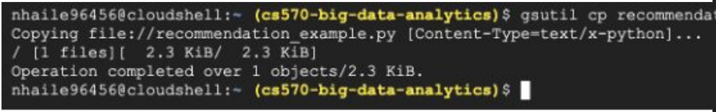
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gsutil mkdir gs://big_data_movie_recommendation
Creating gs://big_data_movie_recommendation/...
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gsutil cp u_transformed_data.csv gs://big_data_movie_recommendation
Copying file://u_transformed_data.csv [Content-Type=text/csv]...
/ [1 files][956.2 KiB/956.2 KiB]
Operation completed over 1 objects/956.2 KiB.
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ 
```

Implementation: Environment Setup

- **PySpark Script Setup:**
  - Create the pyspark script that will perform the collaborative filtering.
  - Upload that file to the cloud storage bucket.



```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi recommendation_example.py
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat recommendation_example.py
"""
Collaborative Filtering Classification Example.
"""
from pyspark import SparkContext

# $example on$
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating
# $example off$

if __name__ == "__main__":
    sc = SparkContext(appName="PythonCollaborativeFilteringExample")
    # $example on$
    # Load and parse the data
    # - Each row consists of a user, a product and a rating.
    data = sc.textFile("gs://big_data_movie_recommendation/u_transformed_data.csv")

    # Each line is
    ratings = data.map(lambda l: l.split(','))\
        .map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))

    # Build the recommendation model using ALS
    # - rank: number of features to use
    rank = 10

                                                   ns of ALS (recommended: 10-20)


    # The default ALS.train() method which assumes ratings are explicit.
    # - Train a matrix factorization model given an RDD of ratings given by
    #   users to some products, in the form of (userID, productID, rating) pairs.
    # - We approximate the ratings matrix as the product of two lower-rank
    #   matrices of a given rank (number of features).
    #   + To solve for these features, we run a given number of
    #     iterations of ALS.
    #   + The level of parallelism is determined automatically based
    #     on the number of partitions in ratings.
    model = ALS.train(ratings, rank, numIterations)
```

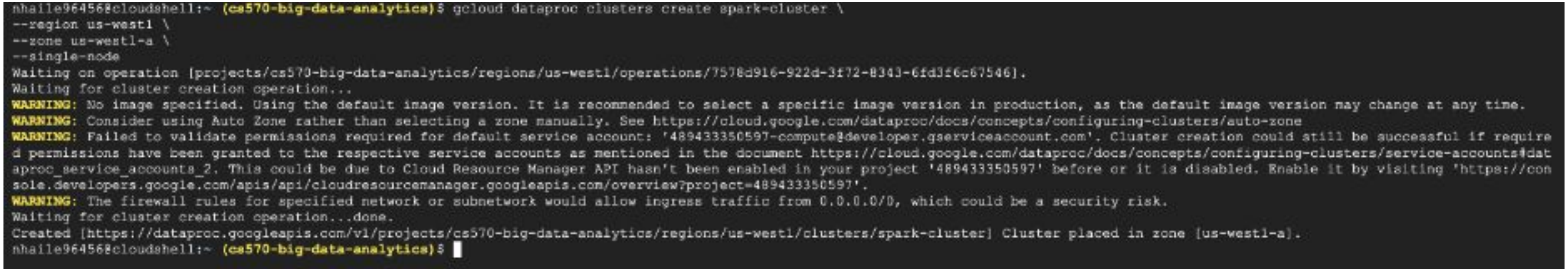


```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gsutil cp recommendation_example.py gs://big_data_movie_recommendation
Copying file://recommendation_example.py [Content-Type=text/x-python]...
/ [1 files][  2.3 KiB/  2.3 KiB]
Operation completed over 1 objects/2.3 KiB.
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ █
```

Implementation: Environment Setup

- **Dataproc Cluster Setup:**
    - Create a dataproc cluster

```
gcloud dataproc clusters create spark-cluster \

--region us-west1 \

--zone us-west1-a \

--single-node
```

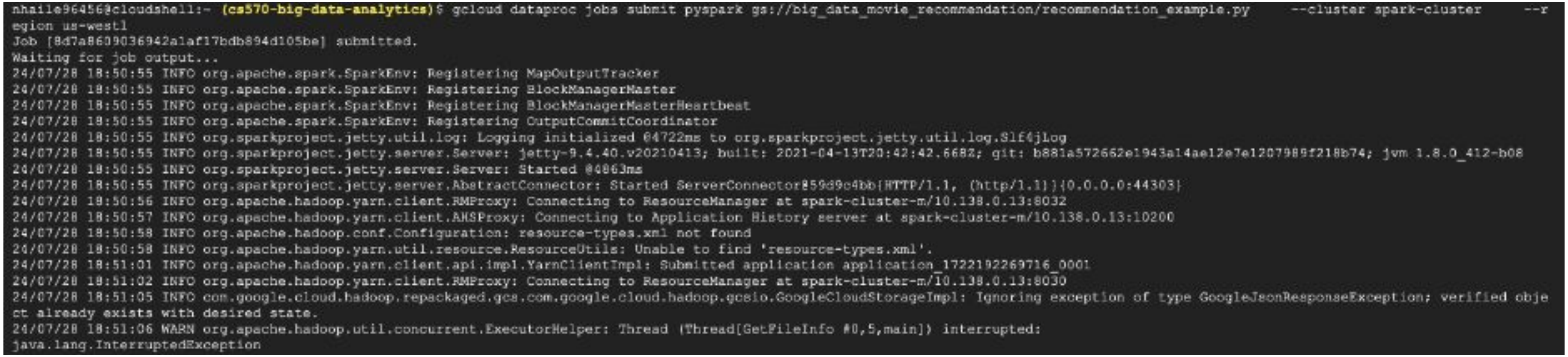


```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gcloud dataproc clusters create spark-cluster \
--region us-west1 \
--zone us-west1-a \
--single-node
Waiting on operation [projects/cs570-big-data-analytics/regions/us-west1/operations/7578d916-922d-3f72-8343-6fd3f6c67546].
Waiting for cluster creation operation...
WARNING: No image specified. Using the default image version. It is recommended to select a specific image version in production, as the default image version may change at any time.
WARNING: Consider using Auto Zone rather than selecting a zone manually. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/auto-zone
WARNING: Failed to validate permissions required for default service account: '489433350597-compute@developer.gserviceaccount.com'. Cluster creation could still be successful if require
d permissions have been granted to the respective service accounts as mentioned in the document https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/service-accounts#dat
aproc_service_accounts_2. This could be due to Cloud Resource Manager API hasn't been enabled in your project '489433350597' before or it is disabled. Enable it by visiting 'https://con
sole.developers.google.com/apis/api/cloudresourcemanager.googleapis.com/overview?project=489433350597'.
WARNING: The firewall rules for specified network or subnetwork would allow ingress traffic from 0.0.0.0/0, which could be a security risk.
Waiting for cluster creation operation...done.
Created [https://dataproc.googleapis.com/v1/projects/cs570-big-data-analytics/regions/us-west1/clusters/spark-cluster] Cluster placed in zone [us-west1-a].
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Implementation: Environment Setup

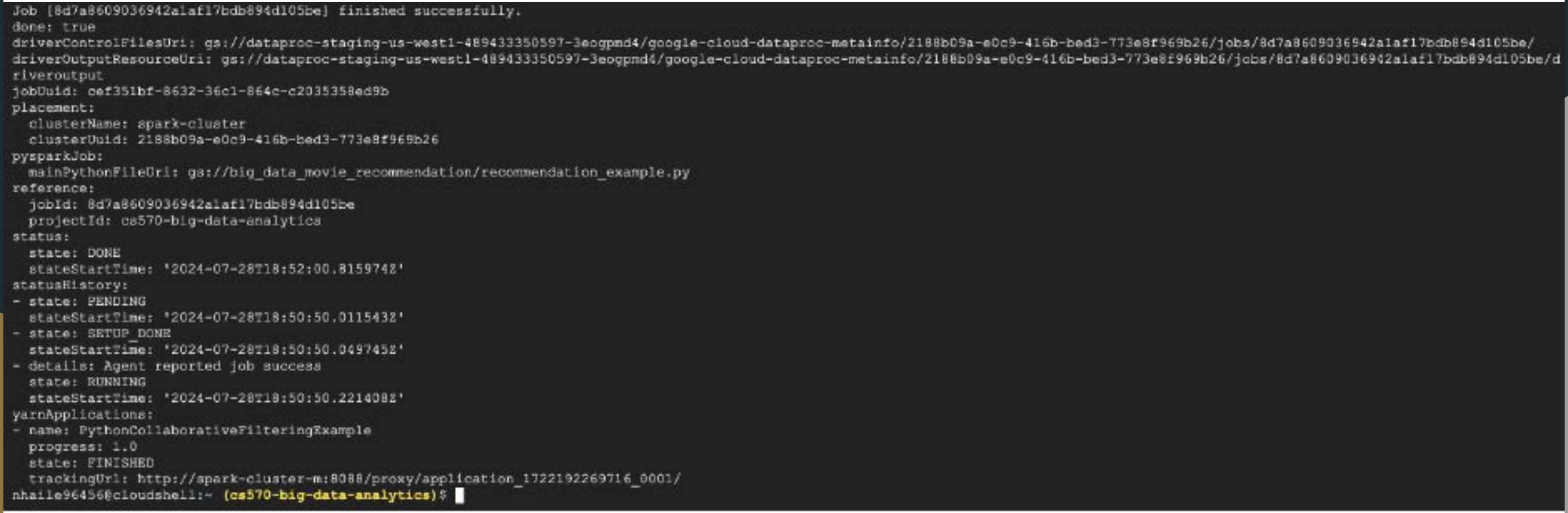
- **Submit PySpark Job:**

```
gcloud dataproc jobs submit pyspark
gs://big_data_movie_recommendation/recommendation_example.py \
    --cluster spark-cluster \
    --region us-west1
```

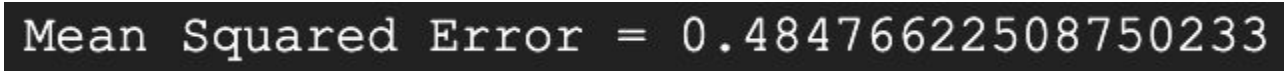


```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gcloud dataproc jobs submit pyspark gs://big_data_movie_recommendation/recommendation_example.py     --cluster spark-cluster     --r
egion us-west1
Job [8d7a8609036942a1af17bdb894d105be] submitted.
Waiting for job output...
24/07/28 18:50:55 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/07/28 18:50:55 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/07/28 18:50:55 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/07/28 18:50:55 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/07/28 18:50:55 INFO org.sparkproject.jetty.util.log: Logging initialized @4722ms to org.sparkproject.jetty.util.log.Slf4jLog
24/07/28 18:50:55 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_412-b08
24/07/28 18:50:55 INFO org.sparkproject.jetty.server.Server: Started @4863ms
24/07/28 18:50:55 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@59d9c4bb{HTTP/1.1, (http/1.1)}{0.0.0.0:44303}
24/07/28 18:50:56 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at spark-cluster-m/10.138.0.13:8032
24/07/28 18:50:57 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at spark-cluster-m/10.138.0.13:10200
24/07/28 18:50:58 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
24/07/28 18:50:58 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
24/07/28 18:51:01 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1722192269716_0001
24/07/28 18:51:02 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at spark-cluster-m/10.138.0.13:8030
24/07/28 18:51:05 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified obje
ct already exists with desired state.
24/07/28 18:51:06 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #0,5,main]) interrupted:
java.lang.InterruptedException
```

Test



```
Job [8d7a8609036942a1af17bdb894d105be] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-489433350597-3eogpmd4/google-cloud-dataproc-metainfo/2188b09a-e0c9-416b-bed3-773e8f969b26/jobs/8d7a8609036942a1af17bdb894d105be/
driverOutputResourceUri: gs://dataproc-staging-us-west1-489433350597-3eogpmd4/google-cloud-dataproc-metainfo/2188b09a-e0c9-416b-bed3-773e8f969b26/jobs/8d7a8609036942a1af17bdb894d105be/d
riveroutput
jobUuid: cef351bf-8632-36c1-864c-c2035358ed9b
placement:
  clusterName: spark-cluster
  clusterUuid: 2188b09a-e0c9-416b-bed3-773e8f969b26
pysparkJob:
  mainPythonFileUri: gs://big_data_movie_recommendation/recommendation_example.py
reference:
  jobId: 8d7a8609036942a1af17bdb894d105be
  projectId: cs570-big-data-analytics
status:
  state: DONE
  stateStartTime: '2024-07-28T18:52:00.815974Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-07-28T18:50:50.011543Z'
- state: SETUP_DONE
  stateStartTime: '2024-07-28T18:50:50.049745Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-07-28T18:50:50.221408Z'
yarnApplications:
- name: PythonCollaborativeFilteringExample
  progress: 1.0
  state: FINISHED
  trackingUrl: http://spark-cluster-m:8088/proxy/application_1722192269716_0001/
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```



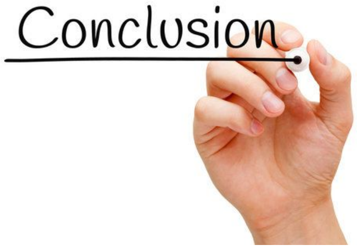
```
Mean Squared Error = 0.48476622508750233
```

Enhancement Ideas

- **Additional Features:** Integrate demographic and movie metadata for improved recommendations.
- **Hybrid Model:** Combine collaborative and content-based filtering for better accuracy.
- **Real-Time Processing:** Use Apache Kafka and PySpark Streaming for real-time recommendations.
- **Enhanced Evaluation:** Implement metrics like Precision, Recall, and F1-score; conduct A/B testing.
- **User Feedback:** Collect user feedback to continuously retrain and improve the model.
- **Scalability:** Optimize ALS parameters and scale the system with more Dataproc nodes.

Conclusion

- A movie recommendation system was created using PySpark's MLlib.
- The collaborative filtering model was trained and evaluated, achieving effective recommendations.
- The system was deployed using Google Dataproc and Cloud Storage for scalable processing and model storage.
- Future work includes enhancements like hybrid models, real-time processing, and improved evaluation metrics.
- This project established a scalable, adaptable recommendation system foundation.



References

Movie Recommendation with Spark MLlib

Collaborative Filtering for Movie Recommendations

Movie Recommendation with Collaborative Filtering in …

Collaborative Filtering - Spark 2.2.0 Documentation

GitHub Link

- https://github.com/cur10usityDrives/Big-Data/tree/main/PySpark/Movie-Recommendation-with-MLlib

THANK YOU