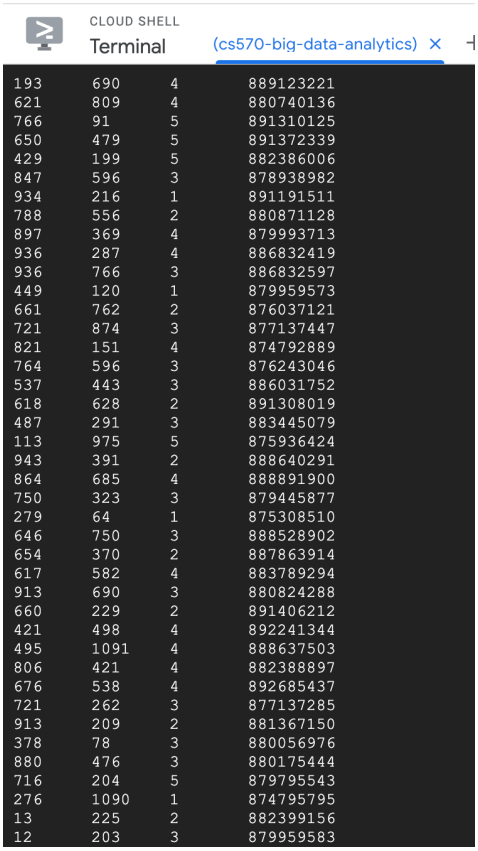
Week 8: Homework 1: Project: Movie Recommendation with MLlib - Collaborative Filtering (implementaiton 2)

Step1: Preparing and Transform data

For this part of the project, we need to load the data we need for the recommendation system and save it in our cloud.

1. **Create a file named u.data** and load the rating data into this file. The data has an initial format of (UserID, MovieID, rating, Timestamp)

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



2. **Transform the data**: next we need to convert the initial format to (UserID, MovieID, rating) this format. So, we can use a bash script to do that.
   The script can be formatted as follows:
   - First copy this bash script into a script file called ***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)$ 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)$
```

Step2: Uploading the data to cloud bucket

Next, we will create a new bucket and upload the transformed data that we saved in the u_transformed_data.csv to our new cloud storage bucket.

```
gsutil mkdir gs://big_data_movie_recommendation
```

```
gsutil cp u_transformed_data.csv
gs://big_data_movie_recommendation
```

```
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)$
```

Step 3: Create and upload the PySpark script

In this step, we will create the pyspark script that will perform the collaborative filtering and upload that file to the cloud storage bucket.

1. Create a file named **recommendation_example.py**
   `vim recommendation_example.py`
2. **Copy this script** in the file but replace the path to the text file since currently it is working from another link. Replace it with the correct path to your pyspark script in the cloud 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

    # - iterattions: number of iterations of ALS (recommended: 10-20)
    numIterations = 10

    # 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)
```

3. **Upload the script to the bucket:**
   `gsutil cp recommendation_example.py gs://big_data_movie_recommendation`

   This PySpark script implements collaborative filtering using the Alternating Least Squares (ALS) algorithm to build a movie recommendation system. It loads user-movie rating data, trains a recommendation model with the ALS algorithm, evaluates the model by calculating the Mean Squared Error (MSE) of predictions, and saves the trained model for future use.

```
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)$
```

Step 4: Submit the pyspark job to google Dataproc

To submit the pyspark script, first we need to create a Dataproc cluster and submit the script

1. **Create a Dataproc cluster** with this command

```
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)$
```

2. Next, submit the pyspark job to the Dataproc cluster

```
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
```

```
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
```