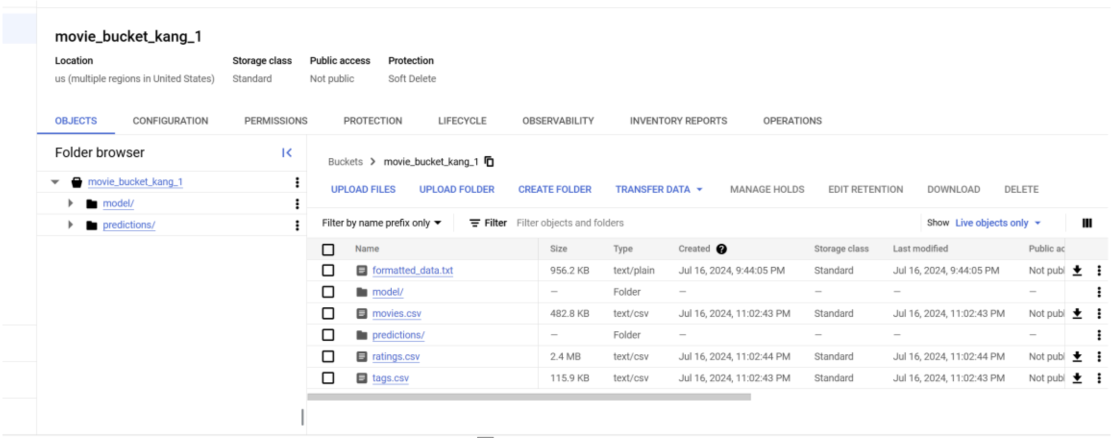
Project: Movie Recommendation with MLlib - Collaborative Filtering (implementaiton 3)

TianzeKang2000/Movie-Recommendation-System (github.com)

Using the GCP Console to Upload the File:
Navigate to Cloud Storage, create a new bucket.
Click the "Upload Files" button and select C:\Users\KANG\Downloads\formatted_data.txt to upload.



movie_bucket_kang_1

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

Name	Size	Type	Created	Storage class	Last modified	Public ac
formatted_data.txt	956.2 KB	text/plain	Jul 16, 2024, 9:44:05 PM	Standard	Jul 16, 2024, 9:44:05 PM	Not publ
model/	–	Folder	–	–	–	–
movies.csv	482.8 KB	text/csv	Jul 16, 2024, 11:02:43 PM	Standard	Jul 16, 2024, 11:02:43 PM	Not publ
predictions/	–	Folder	–	–	–	–
ratings.csv	2.4 MB	text/csv	Jul 16, 2024, 11:02:44 PM	Standard	Jul 16, 2024, 11:02:44 PM	Not publ
tags.csv	115.9 KB	text/csv	Jul 16, 2024, 11:02:43 PM	Standard	Jul 16, 2024, 11:02:43 PM	Not publ

Create a new Python script named recommendation.py:

Import Libraries

```
from pyspark.sql import SparkSession
from pyspark.mllib.recommendation import ALS, Rating
```

- **SparkSession**: Entry point to programming Spark with the DataFrame and SQL API.
- **ALS**: Alternating Least Squares, a collaborative filtering algorithm for recommender systems.
- **Rating**: Class used to store user, product, and rating information.

Initialize Spark Session

```
spark = SparkSession.builder.appName("MovieRecommendation").getOrCreate()
sc = spark.sparkContext
```

- Initializes a Spark session named "MovieRecommendation".
- sc is the SparkContext object, which is the entry point for using Spark functionality.

Load and Parse the Data

```
data = sc.textFile("gs://movie_bucket_kang_1/formatted_data.txt")
ratings = data.map(lambda l: l.split(',')) \
          .map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))
```

- Loads the data from Google Cloud Storage (GCS) as a text file.
- Splits each line of the file by commas and maps it to a Rating object containing user, product, and rating.

Train the Recommendation Model Using ALS

```
rank = 10
```

numIterations = 10
model = ALS.train(ratings, rank, numIterations)

- **rank**: Number of latent factors in the model.
- **numIterations**: Number of iterations to run the ALS algorithm.
- Trains the ALS model using the provided ratings data.

Generate Predictions

users_products = ratings.map(lambda r: (r.user, r.product))
predictions = model.predictAll(users_products).map(lambda r: (r.user, r.product, r.rating))

- Creates a list of (user, product) pairs from the ratings data.
- Uses the trained model to predict ratings for all user-product pairs.
- Maps the predictions to include user, product, and predicted rating.

Convert Predictions to DataFrame

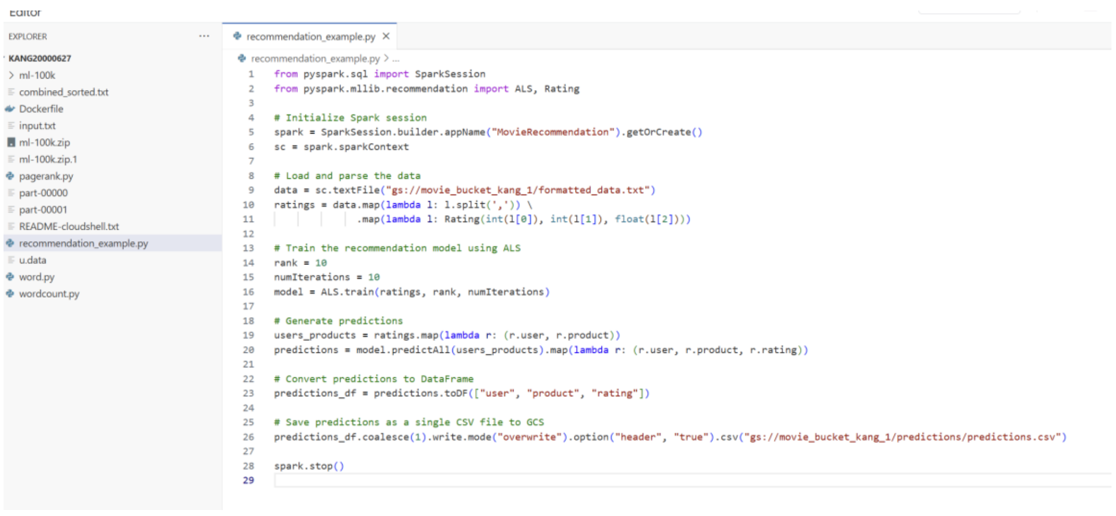
predictions_df = predictions.toDF(["user", "product", "rating"])

- Converts the predictions RDD to a DataFrame with columns "user", "product", and "rating".

Save Predictions as a Single CSV File to GCS

predictions_df.coalesce(1).write.mode("overwrite").option("header", "true").csv("gs://movie_bucket_kang_1/predictions/predictions.csv")

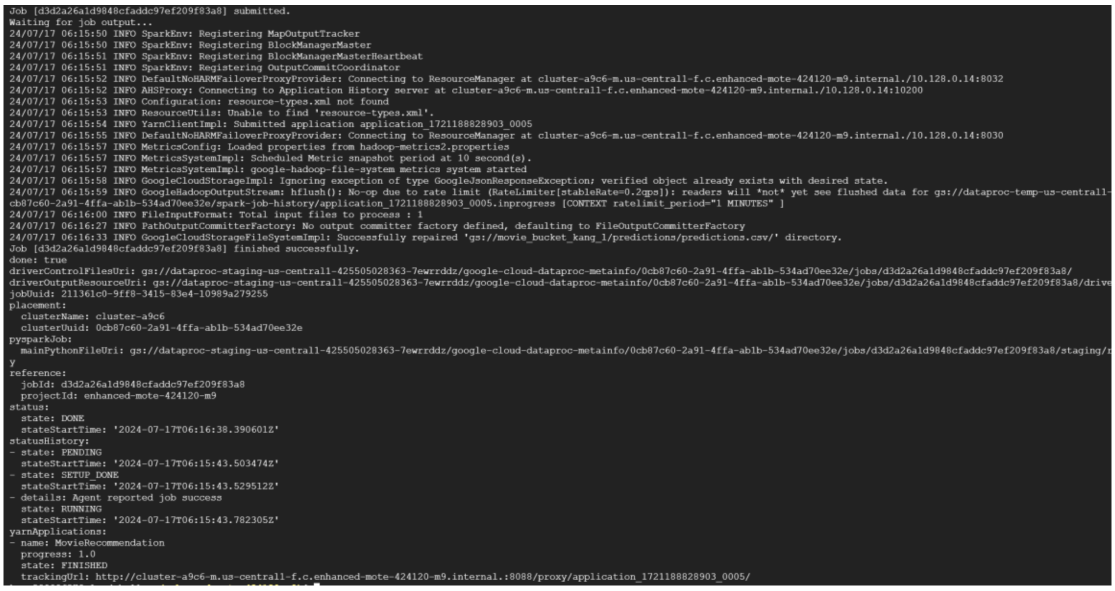
- **coalesce(1)**: Ensures the DataFrame is written to a single CSV file instead of multiple part files.
- **mode("overwrite")**: If the file already exists, it will be overwritten.
- **option("header", "true")**: Writes the DataFrame with a header row.
- **csv("gs://movie_bucket_kang_1/predictions/predictions.csv")**: Specifies the GCS path where the CSV file will be saved.



```python
from pyspark.sql import SparkSession
from pyspark.mllib.recommendation import ALS, Rating

# Initialize Spark session
spark = SparkSession.builder.appName("MovieRecommendation").getOrCreate()
sc = spark.sparkContext

# Load and parse the data
data = sc.textFile("gs://movie_bucket_kang_1/formatted_data.txt")
ratings = data.map(lambda l: l.split(',')) \
              .map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))

# Train the recommendation model using ALS
rank = 10
numIterations = 10
model = ALS.train(ratings, rank, numIterations)

# Generate predictions
users_products = ratings.map(lambda r: (r.user, r.product))
predictions = model.predictAll(users_products).map(lambda r: (r.user, r.product, r.rating))

# Convert predictions to DataFrame
predictions_df = predictions.toDF(["user", "product", "rating"])

# Save predictions as a single CSV file to GCS
predictions_df.coalesce(1).write.mode("overwrite").option("header", "true").csv("gs://movie_bucket_kang_1/predictions/predictions.csv")

spark.stop()
```

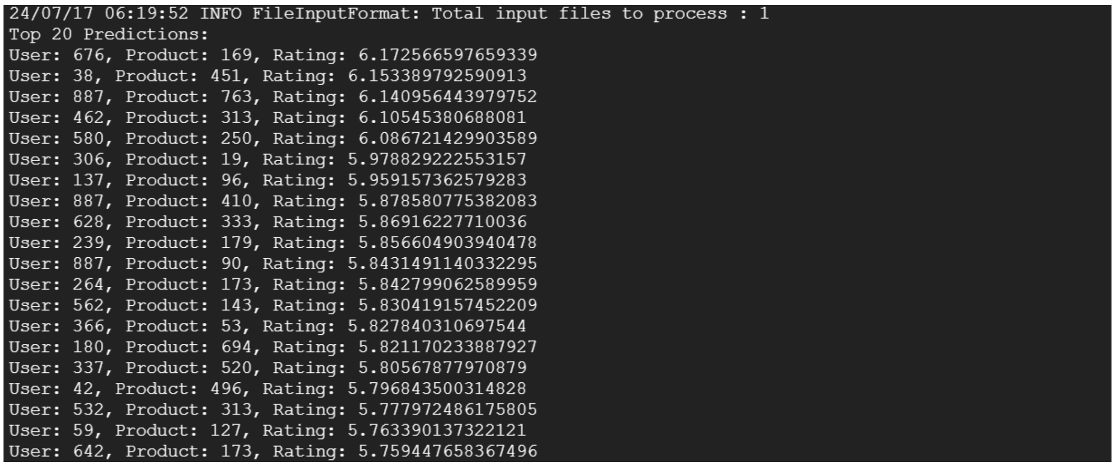
Submit the PySpark Job

In Cloud Shell, submit the PySpark job to your Dataproc cluster using the following command:
gcloud dataproc jobs submit pyspark recommendation_example.py --cluster=cluster-a9c6 --region=us-central1



```
Job [d3d2a26a1d9848cfaddc97ef209f83a8] submitted.
Waiting for job output...
24/07/17 06:15:50 INFO SparkEnv: Registering MapOutputTracker
24/07/17 06:15:50 INFO SparkEnv: Registering BlockManagerMaster
24/07/17 06:15:51 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/07/17 06:15:51 INFO SparkEnv: Registering OutputCommitCoordinator
24/07/17 06:15:52 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-a9c6-m.us-central1-f.c.enhanced-mote-424120-m9.internal./10.128.0.14:8032
24/07/17 06:15:52 INFO AHSProxy: Connecting to Application History server at cluster-a9c6-m.us-central1-f.c.enhanced-mote-424120-m9.internal./10.128.0.14:10200
24/07/17 06:15:53 INFO Configuration: resource-types.xml not found
24/07/17 06:15:53 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/07/17 06:15:54 INFO YarnClientImpl: Submitted application application_1721188828903_0005
24/07/17 06:15:55 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-a9c6-m.us-central1-f.c.enhanced-mote-424120-m9.internal./10.128.0.14:8030
24/07/17 06:15:57 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/07/17 06:15:57 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/07/17 06:15:57 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/07/17 06:15:58 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/07/17 06:15:59 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-us-central1-
cb87c60-2a91-4ffa-ab1b-534ad70ee32e/spark-job-history/application_1721188828903_0005.inprogress [CONTEXT ratelimit_period="1 MINUTES" ]
24/07/17 06:16:00 INFO FileInputFormat: Total input files to process : 1
24/07/17 06:16:27 INFO PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
24/07/17 06:16:33 INFO GoogleCloudStorageFileSystemImpl: Successfully repaired 'gs://movie_bucket_kang_1/predictions/predictions.csv/' directory.
Job [d3d2a26a1d9848cfaddc97ef209f83a8] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-425505028363-7ewrrddz/google-cloud-dataproc-metainfo/0cb87c60-2a91-4ffa-ab1b-534ad70ee32e/jobs/d3d2a26a1d9848cfaddc97ef209f83a8/
driverOutputResourceUri: gs://dataproc-staging-us-central1-425505028363-7ewrrddz/google-cloud-dataproc-metainfo/0cb87c60-2a91-4ffa-ab1b-534ad70ee32e/jobs/d3d2a26a1d9848cfaddc97ef209f83a8/drive
jobUuid: 211361c0-9ff8-3415-83e4-10989a279255
placement:
  clusterName: cluster-a9c6
  clusterUuid: 0cb87c60-2a91-4ffa-ab1b-534ad70ee32e
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-425505028363-7ewrrddz/google-cloud-dataproc-metainfo/0cb87c60-2a91-4ffa-ab1b-534ad70ee32e/jobs/d3d2a26a1d9848cfaddc97ef209f83a8/staging/r
y
reference:
  jobId: d3d2a26a1d9848cfaddc97ef209f83a8
  projectId: enhanced-mote-424120-m9
status:
  state: DONE
  stateStartTime: '2024-07-17T06:16:38.390601Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-07-17T06:15:43.503474Z'
- state: SETUP_DONE
  stateStartTime: '2024-07-17T06:15:43.529512Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-07-17T06:15:43.782305Z'
yarnApplications:
- name: MovieRecommendation
  progress: 1.0
  state: FINISHED
  trackingUrl: http://cluster-a9c6-m.us-central1-f.c.enhanced-mote-424120-m9.internal.:8088/proxy/application_1721188828903_0005/
```

Output



```
24/07/17 06:19:52 INFO FileInputFormat: Total input files to process : 1
Top 20 Predictions:
User: 676, Product: 169, Rating: 6.172566597659339
User: 38, Product: 451, Rating: 6.153389792590913
User: 887, Product: 763, Rating: 6.140956443979752
User: 462, Product: 313, Rating: 6.10545380688081
User: 580, Product: 250, Rating: 6.086721429903589
User: 306, Product: 19, Rating: 5.978829222553157
User: 137, Product: 96, Rating: 5.959157362579283
User: 887, Product: 410, Rating: 5.878580775382083
User: 628, Product: 333, Rating: 5.86916227710036
User: 239, Product: 179, Rating: 5.856604903940478
User: 887, Product: 90, Rating: 5.8431491140332295
User: 264, Product: 173, Rating: 5.842799062589959
User: 562, Product: 143, Rating: 5.830419157452209
User: 366, Product: 53, Rating: 5.827840310697544
User: 180, Product: 694, Rating: 5.821170233887927
User: 337, Product: 520, Rating: 5.80567877970879
User: 42, Product: 496, Rating: 5.796843500314828
User: 532, Product: 313, Rating: 5.777972486175805
User: 59, Product: 127, Rating: 5.763390137322121
User: 642, Product: 173, Rating: 5.759447658367496
```