BDAD MovieLens

```
%sh                                                                                    FINISHED
# wget https://files.grouplens.org/datasets/movielens/ml-25m.zip
# unzip ml-25m.zip
# ls
# hadoop fs -ls

MLens.scala
build.sbt
ml-25m
ml-25m.zip
project
target
zeppelin
```

Took 3 sec. Last updated by anonymous at December 11 2022, 7:54:52 PM. (outdated)

```
%sh                                                                                    FINISHED
spark-shell --deploy-mode client

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/12/12 06:48:51 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/12/12 06:48:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/12/12 06:48:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/12/12 06:48:51 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://nyu-dataproc-m.c.hpc-dataproc-19b8.internal:35917
Spark context available as 'sc' (master = yarn, app id = application_1670432922569_1801).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_322)
```

Took 20 sec. Last updated by anonymous at December 12 2022, 1:49:02 AM.

```
val movieSchema = "movieId INT, title STRING, genres STRING"         ▤ SPARK JOB  FINISHED
val df_movies = spark.read.format("csv").schema(movieSchema).option("header", "true").load("ml-25m/movies.csv")
//df_movies.show(25, false)

val ratingSchema = "userId INT, movieId INT, rating DOUBLE, timestamp LONG"
val df_ratings_timestamp = spark.read.format("csv").schema(ratingSchema).option("header", "true").load("ml-25m/ratings.csv")
val df_ratings = df_ratings_timestamp.drop("timestamp")
//df_ratings.show(25, false)


// sc.stop()



//Commands to try training an ALS model:


val set = df_ratings.randomSplit(Array(0.8, 0.2))
val training = set(0).cache()
val test = set(1).cache()
//training.take(5).foreach(println)

println(s"Training: ${training.count()}, test: ${test.count()}")

Training: 19999829, test: 5000266
movieSchema: String = movieId INT, title STRING, genres STRING
df_movies: org.apache.spark.sql.DataFrame = [movieId: int, title: string ... 1 more field]
ratingSchema: String = userId INT, movieId INT, rating DOUBLE, timestamp LONG
df_ratings_timestamp: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more fields]
df_ratings: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 1 more field]
set: Array[org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]] = Array([userId: int, movieId: int ... 1 more field], [userId: int, movieI
d: int ... 1 more field])
training: org.apache.spark.sq...
```

Took 1 min 3 sec. Last updated by anonymous at December 12 2022, 1:49:46 AM. (outdated)

```
import org.apache.spark.ml.recommendation.ALS                        ▤ SPARK JOB  FINISHED
import org.apache.spark.ml.recommendation.ALSModel
import org.apache.spark.ml.evaluation.RegressionEvaluator


// Build the recommendation model using ALS on the training data
val als = new ALS()
  .setMaxIter(15)
  .setRank(10)
  .setRegParam(0.01)
  .setUserCol("userId")
```

```
    .setItemCol("movieId")
    .setRatingCol("rating")
val model = als.fit(training)

// Evaluate the model by computing the RMSE on the test data
// Note we set cold start strategy to 'drop' to ensure we don't get NaN evaluation metrics
model.setColdStartStrategy("drop")
// model.save("target/tmp/myCollaborativeFilter")
// val model = ALSModel.load("target/tmp/myCollaborativeFilter")
val predictions = model.transform(test)

val evaluator = new RegressionEvaluator()
    .setMetricName("rmse")
    .setLabelCol("rating")
    .setPredictionCol("prediction")
val rmse = evaluator.evaluate(predictions)
println(s"Root-mean-square error = $rmse")
```

```
Root-mean-square error = 0.7990968211519277
import org.apache.spark.ml.recommendation.ALS
import org.apache.spark.ml.recommendation.ALSModel
import org.apache.spark.ml.evaluation.RegressionEvaluator
als: org.apache.spark.ml.recommendation.ALS = als_16d9b48691e4
model: org.apache.spark.ml.recommendation.ALSModel = ALSModel: uid=als_16d9b48691e4, rank=10
predictions: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more fields]
evaluator: org.apache.spark.ml.evaluation.RegressionEvaluator = RegressionEvaluator: uid=regEval_d36884e569d8, metricName=rmse, throughOrigin
=false
rmse: Double = 0.7990968211519277
```

Took 3 min 43 sec. Last updated by anonymous at December 12 2022, 1:52:34 AM.

```
predictions.show(25)
```
                                                                                               ≡ SPARK JOB FINISHED

```
+------+-------+------+----------+
|userId|movieId|rating|prediction|
+------+-------+------+----------+
|   148|     32|   4.0| 3.9667687|
|   148|    111|   4.0| 4.1811914|
|   148|    608|   3.0| 4.1603775|
|   148|    955|   3.5|  3.695281|
|   148|   1089|   4.5| 4.1630325|
|   148|   1196|   3.5|  4.398311|
|   148|   1207|   4.0| 4.1829014|
|   148|   1208|   5.0|  4.356421|
|   148|   1222|   4.0|  4.240112|
|   148|   1250|   4.0| 4.2101173|
|   148|   1254|   4.0|  4.105406|
|   148|   1260|   4.0| 4.1495676|
|   148|   1266|   4.0|  4.224486|
|   148|   1276|   3.5|  4.275051|
|   148|   1293|   3.5|  4.01988|
```

Took 1 min 40 sec. Last updated by anonymous at December 11 2022, 11:23:10 PM.

```
// Generate top 10 movie recommendations for each user
val userRecs = model.recommendForAllUsers(10)
// Generate top 10 user recommendations for each movie
val movieRecs = model.recommendForAllItems(10)
```
                                                                                                          FINISHED

```
userRecs: org.apache.spark.sql.DataFrame = [userId: int, recommendations: array<struct<movieId:int,rating:float>>]
movieRecs: org.apache.spark.sql.DataFrame = [movieId: int, recommendations: array<struct<userId:int,rating:float>>]
```

Took 2 min 49 sec. Last updated by anonymous at December 12 2022, 1:52:35 AM.

```
// For user 148:

// Generate list of movies that he has rated with atleast 4.0 score out of 5. Get this from df_movies
// Check the predictions that i have made for user 148 with atleast 4.0 out of 5. Get this from predictions
// See if those movies are similar based on genre , year? and tags?

// If so, my analytic's goodness is shown!
```
                                                                                                          FINISHED

Took 0 sec. Last updated by anonymous at December 11 2022, 9:03:10 PM. (outdated)

```
// Join predictions.filter(userId === 148)
// with df_movies on movieId

val predictions148 = predictions.filter(predictions("userId") === 148)
val predictions148NoUser = predictions148.drop("userId")
```
                                                                                                          FINISHED

```
predictions148: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [userId: int, movieId: int ... 2 more fields]
predictions148NoUser: org.apache.spark.sql.DataFrame = [movieId: int, rating: double ... 1 more field]
```

Took 0 sec. Last updated by anonymous at December 11 2022, 11:28:19 PM.

```
val predShow = df_movies.join(predictions148NoUser, df_movies("movieId") === predictions148NoUser("movieId"))
```
                                                                                                          FINISHED

```
predShow: org.apache.spark.sql.DataFrame = [movieId: int, title: string ... 4 more fields]
```

Took 0 sec. Last updated by anonymous at December 11 2022, 11:31:13 PM.

```
val predShowFinal = predShow.drop("movieId")
```

FINISHED

```
predShowFinal: org.apache.spark.sql.DataFrame = [title: string, genres: string ... 2 more fields]
```

BDAD MovieLens

Took 1 sec. Last updated by anonymous at December 11 2022, 11:31:27 PM. (outdated)

```
predShowFinal.show(15, false)
```

≡ SPARK JOB FINISHED

```
+-----------------------------------------------+------------------------------+------+---------+
|title                                          |genres                        |rating|prediction|
+-----------------------------------------------+------------------------------+------+---------+
|Twelve Monkeys (a.k.a. 12 Monkeys) (1995)      |Mystery|Sci-Fi|Thriller       |4.0   |3.9667687 |
|Taxi Driver (1976)                             |Crime|Drama|Thriller          |4.0   |4.1811914 |
|Fargo (1996)                                   |Comedy|Crime|Drama|Thriller   |3.0   |4.1603775 |
|Bringing Up Baby (1938)                        |Comedy|Romance                |3.5   |3.695281  |
|Reservoir Dogs (1992)                          |Crime|Mystery|Thriller        |4.5   |4.1630325 |
|Star Wars: Episode V - The Empire Strikes Back (1980)|Action|Adventure|Sci-Fi |3.5   |4.398311  |
|To Kill a Mockingbird (1962)                   |Drama                         |4.0   |4.1829014 |
|Apocalypse Now (1979)                          |Action|Drama|War              |5.0   |4.356421  |
|Full Metal Jacket (1987)                       |Drama|War                     |4.0   |4.240112  |
|Bridge on the River Kwai, The (1957)           |Adventure|Drama|War           |4.0   |4.2101173 |
|Treasure of the Sierra Madre, The (1948)       |Action|Adventure|Drama|Western|4.0   |4.105406  |
|M (1931)                                       |Crime|Film-Noir|Thriller      |4.0   |4.1495676 |
|Unforgiven (1992)                              |Drama|Western                 |4.0   |4.224486  |
|Cool Hand Luke (1967)                          |Drama                         |3.5   |4.275051  |
```

Took 2 sec. Last updated by anonymous at December 11 2022, 11:32:17 PM.

```
val arrMovies148 = model.transform (test.where(test("userId") === 148))
    .select ('movieId, 'prediction)
    .orderBy('prediction.desc)
    .limit(10)
    .toDF()
```

FINISHED

```
arrMovies148: org.apache.spark.sql.DataFrame = [movieId: int, prediction: float]
```

Took 2 sec. Last updated by anonymous at December 12 2022, 1:52:36 AM.

```
arrMovies148.show(10)
```

≡ SPARK JOB FINISHED

```
+-------+----------+
|movieId|prediction|
+-------+----------+
|   1203|  4.539253|
|   1221| 4.5276504|
|   2019|  4.464319|
|   1233|  4.381468|
|    904|   4.30421|
|   1204| 4.2515717|
|   3462|  4.235806|
|     47|  4.234333|
|   2028| 4.2043543|
|   2324| 4.1968174|
+-------+----------+
```

Took 9 sec. Last updated by anonymous at December 12 2022, 1:43:24 AM.

```
val movies148 = df_movies.join(arrMovies148, df_movies("movieId") === arrMovies148("movieId")).drop("movieId")
```

FINISHED

```
movies148: org.apache.spark.sql.DataFrame = [title: string, genres: string ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 12 2022, 1:52:36 AM.

```
movies148.show(10, false)
```

≡ SPARK JOB FINISHED

```
+-----------------------------------------------+------------------------+----------+
|title                                          |genres                  |prediction|
+-----------------------------------------------+------------------------+----------+
|Star Wars: Episode IV - A New Hope (1977)      |Action|Adventure|Sci-Fi |4.4641266 |
|Pulp Fiction (1994)                            |Comedy|Crime|Drama|Thriller|4.26713 |
|Silence of the Lambs, The (1991)               |Crime|Horror|Thriller   |4.2729945 |
|Godfather, The (1972)                          |Crime|Drama             |4.637894  |
|Dial M for Murder (1954)                       |Crime|Mystery|Thriller  |4.2225246 |
|Monty Python and the Holy Grail (1975)         |Adventure|Comedy|Fantasy|4.224548  |
|Star Wars: Episode V - The Empire Strikes Back (1980)|Action|Adventure|Sci-Fi|4.4889464 |
|Goodfellas (1990)                              |Crime|Drama             |4.2514267 |
|Amadeus (1984)                                 |Drama                   |4.204793  |
|Rope (1948)                                    |Crime|Drama|Thriller    |4.2580085 |
+-----------------------------------------------+------------------------+----------+
```

Took 25 sec. Last updated by anonymous at December 12 2022, 1:55:12 AM.

```
val ratings148 = df_ratings.filter(df_ratings("userId") === 148 && df_ratings("rating") === 5.0)
```

FINISHED

```
ratings148: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [userId: int, movieId: int ... 1 more field]
```

Took 0 sec. Last updated by anonymous at December 12 2022, 2:01:10 AM.

BDAD MovieLens

```
ratings148.join(df_movies, df_movies("movieId") === ratings148("movieId")).drop("movieId")...  FINISHED
```

```
+------+--------------------------------------------------------------------+------------------------------+
|rating|title                                                               |genres                        |
+------+--------------------------------------------------------------------+------------------------------+
|5.0   |Pulp Fiction (1994)                                                 |Comedy|Crime|Drama|Thriller   |
|5.0   |Shawshank Redemption, The (1994)                                    |Crime|Drama                   |
|5.0   |Schindler's List (1993)                                             |Drama|War                     |
|5.0   |Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)|Comedy|War              |
|5.0   |2001: A Space Odyssey (1968)                                        |Adventure|Drama|Sci-Fi        |
|5.0   |Paths of Glory (1957)                                               |Drama|War                     |
|5.0   |Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966)|Action|Adventure|Western  |
|5.0   |12 Angry Men (1957)                                                 |Drama                         |
|5.0   |Apocalypse Now (1979)                                               |Action|Drama|War              |
|5.0   |Chinatown (1974)                                                    |Crime|Film-Noir|Mystery|Thriller|
+------+--------------------------------------------------------------------+------------------------------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at December 12 2022, 2:02:13 AM.

READY