explicit_recommendation

October 8, 2017

1 Building a Movie Recommendation System with Spark MLlib

1.1 Import required libraries

In [1]:
```
import subprocess
import findspark
findspark.init()
from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.sql import Row

spark = SparkSession \
    .builder \
    .appName("bitcoins") \
    .config("spark.som.config.option", "some-value") \
    .getOrCreate()
```

1.2 Download/Unzip the MovieLens 1M dataset from http://grouplens.org/datasets/movielens

In [2]:
```
subprocess.call(["wget", "http://files.grouplens.org/datasets/movielens/ml-1m.zip"])
subprocess.call(["unzip", "ml-1m.zip"])
```

Out[2]:
```
1
```

1.3 Read and Convert ratings data to a DataFrame

In [3]:
```
lines = spark.read.text("./ml-1m/ratings.dat").rdd
parts = lines.map(lambda row: row.value.split("::"))
ratingsRDD = parts.map(lambda p: Row(userId=int(p[0]), movieId=int(p[1]),
                                     rating=float(p[2]), timestamp=int(p[3])))
ratings = spark.createDataFrame(ratingsRDD)
```

1.4 Show the number of ratings in the dataset

In [4]:
```
print("Number of ratings = " + str(ratings.count()))
```

```
Number of ratings = 1000209
```

1.5 Show a sample of the Ratings DataFrame

```
In [5]: ratings.sample(False, 0.0001, seed=0).show(10)
```

```
+-------+------+---------+------+
|movieId|rating|timestamp|userId|
+-------+------+---------+------+
|   2908|   5.0|977895809|    68|
|   3730|   5.0|978554445|   173|
|   2917|   2.0|976301830|   456|
|    589|   4.0|976161565|   526|
|   2348|   3.0|976207524|   533|
|   1285|   4.0|979154572|   588|
|   1206|   4.0|980628867|   711|
|   3361|   4.0|975510209|   730|
|   3203|   5.0|975435824|   779|
|   1196|   4.0|975356701|   843|
+-------+------+---------+------+
only showing top 10 rows
```

1.6 Show sample number of ratings per user

```
In [6]: grouped_ratings = ratings.groupBy("userId").count().withColumnRenamed("count", "No. of r
        grouped_ratings.show(10)
```

```
+------+--------------+
|userId|No. of ratings|
+------+--------------+
|    26|           400|
|    29|           108|
|   474|           318|
|   964|            78|
|  1677|            43|
|  1697|           354|
|  1806|           214|
|  1950|           137|
|  2040|            46|
|  2214|            81|
+------+--------------+
only showing top 10 rows
```

1.7 Show the number of users in the dataset

```
In [7]: print("Number of users = " + str(grouped_ratings.count()))
```

```
Number of users = 6040
```

1.8 Split Ratings data into Training (80%) and Test (20%) datasets

```
In [8]: (training, test) = ratings.randomSplit([0.8, 0.2])
```

1.9 Show resulting Ratings dataset counts

```
In [9]: trainingRatio = float(training.count())/float(ratings.count())*100
        testRatio = float(test.count())/float(ratings.count())*100

        print("Total number of ratings = " + str(ratings.count()))
        print("Training dataset count = " + str(training.count()) + ", " + str(trainingRatio) +
        print("Test dataset count = " + str(test.count()) + ", " + str(testRatio) + "%")
```

```
Total number of ratings = 1000209
Training dataset count = 800880, 80.07126510559293%
Test dataset count = 199329, 19.928734894407068%
```

1.10 Build the recommendation model on the training data using ALS

```
In [10]: als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="movieId", ratingCol="rat
                  coldStartStrategy="drop")
         model = als.fit(training)
```

1.11 Run the model against the Test data and show a sample of the predictions

```
In [11]: predictions = model.transform(test).na.drop()
         predictions.show(10)
```

```
+-------+------+---------+------+----------+
|movieId|rating|timestamp|userId|prediction|
+-------+------+---------+------+----------+
|    148|   1.0|976295338|   840| 2.9349167|
|    148|   2.0|974875106|  1150| 2.9894443|
|    148|   2.0|974178993|  2456| 3.9975448|
|    463|   5.0|968916009|  3151|  3.967182|
|    463|   3.0|963746396|  4858| 2.0730953|
|    463|   4.0|973625620|  2629| 3.1774714|
|    463|   1.0|966523740|  3683| 1.1212827|
|    463|   2.0|966790403|  3562|  2.780132|
|    463|   4.0|975775726|   721| 3.3978982|
|    463|   3.0|965308300|  4252| 0.9944763|
```

```
+-------+------+---------+------+----------+
only showing top 10 rows
```

1.12 Evaluate the model by computing the RMSE on the test data

```
In [12]: predictions = model.transform(test)
         evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                         predictionCol="prediction")
         rmse = evaluator.evaluate(predictions)
         print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 0.8908929362860674
```

1.13 Show that a smaller value of rmse is better

This is obviously the case since RMSE is an aggregation of all the error. Thus evaluator.isLargerBetter should be 'false'.

```
In [13]: evaluator.isLargerBetter()
```

```
Out[13]: False
```

1.14 Make movie recommendations

```
In [14]: # Generate top 10 movie recommendations for each user
         userRecs = model.recommendForAllUsers(10)
         # Generate top 10 user recommendations for each movie
         movieRecs = model.recommendForAllItems(10)
```

2 Show sample recommendations per user

```
In [15]: userRecs.sample(False, 0.01).show(10, False)
```

```
+------+-----------------------------------------------------------------------------------------------
|userId|recommendations
+------+-----------------------------------------------------------------------------------------------
|148   |[[1780,7.2854385], [1369,6.99533], [666,6.6703053], [2892,6.5549903], [1741,6.528875], [
|5173  |[[3245,7.7563887], [1038,7.52281], [3867,7.2047706], [632,7.0838833], [37,7.0073814], [7
|5695  |[[1458,9.663776], [3855,9.074218], [3106,9.053921], [2837,9.043263], [2192,8.797422], [2
|1863  |[[962,6.392259], [2175,6.2921085], [2984,6.027778], [759,5.9641767], [3737,5.929455], [2
|1924  |[[1038,8.618518], [219,7.9083204], [131,7.871811], [632,7.788521], [1458,7.681244], [157
|4610  |[[3670,6.8609476], [1117,6.645418], [2994,6.6018786], [2830,6.596518], [2934,6.505612],
|4104  |[[649,7.115762], [1421,6.597936], [3885,6.493393], [1585,6.441885], [1741,6.0131593], [5
|1249  |[[3636,8.443559], [1420,7.907082], [1664,7.8959613], [3456,7.7776465], [2697,7.7743106],
|855   |[[3670,5.6403356], [557,5.452341], [503,4.9971642], [3338,4.9897413], [3012,4.9187536],
```

```
|5361  |[[3523,8.854711], [1662,7.23445], [2388,7.046192], [1780,6.9375844], [2892,6.929945], [2
+------+----------------------------------------------------------------------------------------
only showing top 10 rows
```

3 Show sample recommendations per user

```
In [16]: movieRecs.sample(False, 0.01).show(10, False)
```

```
+-------+----------------------------------------------------------------------------------------
|movieId|recommendations
+-------+----------------------------------------------------------------------------------------
|3844   |[[1213,7.3201046], [2441,6.9640417], [5297,6.8789372], [2549,6.8698826], [2816,6.507644
|1031   |[[1070,5.9382234], [4143,5.8492775], [3897,5.841146], [2755,5.6947303], [4282,5.6827908
|26     |[[1213,7.0531287], [2640,6.3756685], [879,6.1351347], [2502,6.0931673], [5298,5.9518814
|626    |[[4504,9.705521], [3222,8.426963], [1713,8.153491], [5863,7.892766], [4583,7.852765], [
|3752   |[[5670,6.538592], [21,5.9881763], [5258,5.949679], [4393,5.7138], [4028,5.6019115], [10
|2256   |[[745,7.8676734], [2469,7.4058766], [906,7.213084], [2431,7.1617584], [1754,7.1158795],
|3793   |[[640,5.7342196], [5218,5.440282], [1673,5.2526026], [947,5.2225814], [2694,5.2105126],
|2867   |[[745,5.992924], [2534,5.8074617], [527,5.6805005], [2755,5.653826], [283,5.3882546], [
|846    |[[4008,10.775237], [4504,10.658872], [3222,9.88133], [399,9.678963], [5240,9.402692], [
|729    |[[665,11.115968], [1459,9.497441], [5803,7.76634], [1384,7.726793], [4317,7.657247], [6
+-------+----------------------------------------------------------------------------------------
only showing top 10 rows
```