Personalized Movie Recommendation Engine

Using the MovieLens Dataset and PySpark

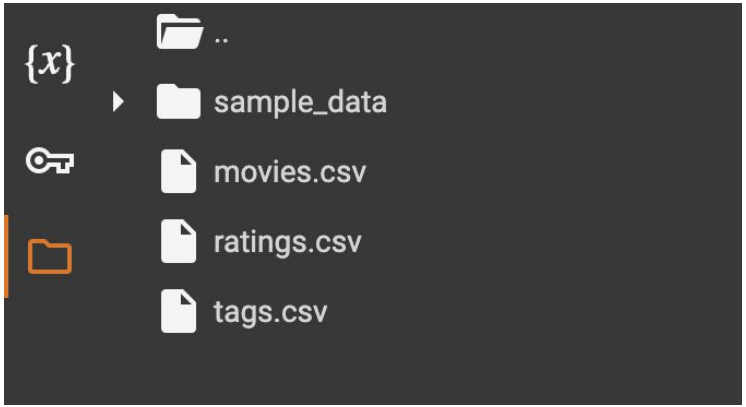
Arsiema Yohannes

Introduction

- "This project focuses on building a personalized movie recommendation engine using the MovieLens dataset and PySpark."
- "The goal is to provide tailored movie recommendations to users based on their preferences."

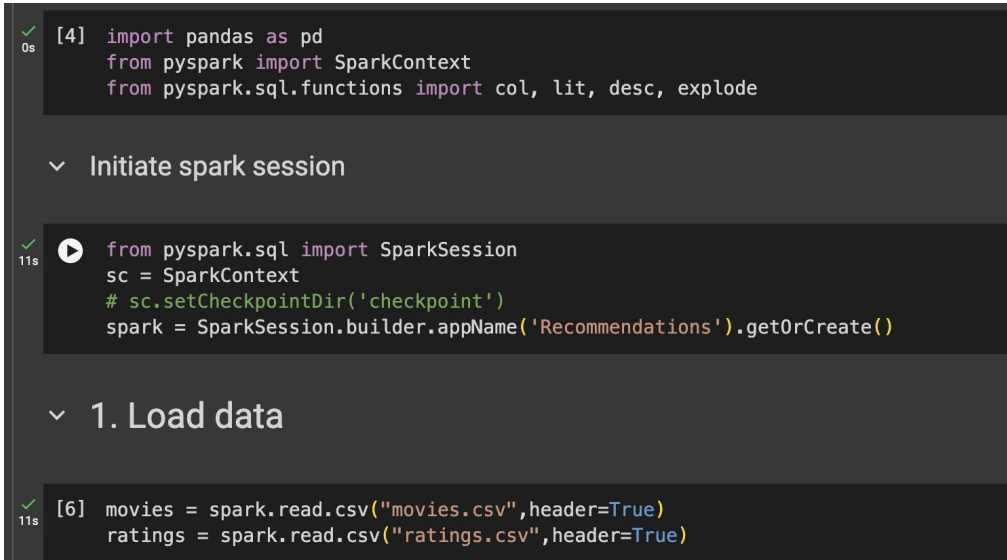
Dataset Overview

- "The MovieLens dataset includes three main files: movies, ratings, and tags."
- "Key columns in the dataset: movieId, userId, rating, title, and genres.":



Data Preprocessing

- "Data preprocessing involves loading the datasets, joining them, and preparing them for model training."



ALS Model Training

- "The ALS (Alternating Least Squares) algorithm is used for collaborative filtering."
- "Hyperparameter tuning is done using CrossValidator with a parameter grid."

```
# Add hyperparameters and their respective values to param_grid
param_grid = ParamGridBuilder() \
            .addGrid(als.rank, [10, 50, 100, 150]) \
            .addGrid(als.regParam, [.01, .05, .1, .15]) \
            .build()
            #             .addGrid(als.maxIter, [5, 50, 100, 200]) \


# Define evaluator as RMSE and print length of evaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="prediction")
print ("Num models to be tested: ", len(param_grid))
```

```
[17] #Fit cross validator to the 'train' dataset
     model = cv.fit(train)

     #Extract best model from the cv model above
     best_model = model.bestModel
```

```
[16] # Build cross validation using CrossValidator
     cv = CrossValidator(estimator=als, estimatorParamMaps=param_grid, evaluator=evaluator, numFolds=5)

     # Confirm cv was built
     print(cv)
```

Model Evaluation

- "The model's performance is evaluated using the RMSE metric."
- "Lower RMSE values indicate better model performance."

```
[19] # View the predictions
     test_predictions = best_model.transform(test)
     RMSE = evaluator.evaluate(test_predictions)
     print(RMSE)

[20] test_predictions.show()
```

Personalization

- "Personalization is achieved by filtering recommendations based on user-specific preferences and adding personalized messages."

```
tions import col, explode, collect_list

o get personalized recommendations
recommendations(user_id, genre_filter=None):
mendations for the user
ons = best_model.recommendForAllUsers(10).filter(col("userId") == user_id)

ommendations to get individual movie IDs and ratings
ons = user_recommendations.withColumn("rec_exp", explode("recommendations")).select('userId', col("re

s to get titles and genres
ons = user_recommendations.join(movies, on='movieId')

ter if provided

dations = user_recommendations.filter(col("genres").contains(genre_filter))

zed message
ons = user_recommendations.withColumn("personalized_message", lit(f"Hi User {user_id}, based on your

mendations

commendations for a specific user and genre

y"
```

```
+-------+------+---------+--------------------+--------------------+--------------------+
|movieId|userId|   rating|               title|              genres|personalized_message|
+-------+------+---------+--------------------+--------------------+--------------------+
|  67618|   100|5.1655545|Strictly Sexual (...|Comedy|Drama|Romance|Hi User 100, base...|
|  33649|   100|5.0584683|  Saving Face (2004)|Comedy|Drama|Romance|Hi User 100, base...|
+-------+------+---------+--------------------+--------------------+--------------------+
```

Results: Interpret ratings and Predictions.

```
+------+-----+
|userId|count|
+------+-----+
|   414| 2698|
|   599| 2478|
|   474| 2108|
|   448| 1864|
|   274| 1346|
|   610| 1302|
|    68| 1260|
|   380| 1218|
|   606| 1115|
|   288| 1055|
|   249| 1046|
|   387| 1027|
|   182|  977|
|   307|  975|
|   603|  943|
|   298|  939|
|   177|  904|
|   318|  879|
|   232|  862|
|   480|  836|
+------+-----+
only showing top 20 rows
```

```
+-------+-----+
|movieId|count|
+-------+-----+
|    356|  329|
|    318|  317|
|    296|  307|
|    593|  279|
|   2571|  278|
|    260|  251|
|    480|  238|
|    110|  237|
|    589|  224|
|    527|  220|
|   2959|  218|
|      1|  215|
|   1196|  211|
|     50|  204|
|   2858|  204|
|     47|  203|
|    780|  202|
|    150|  201|
|   1198|  200|
|   4993|  198|
+-------+-----+
only showing top 20 rows
```

```
test_predictions.show()
```

```
+------+-------+------+----------+
|userId|movieId|rating|prediction|
+------+-------+------+----------+
|   580|   1580|   4.0| 3.4221864|
|   580|  44022|   3.5| 3.0710287|
|   597|    471|   2.0| 4.0885434|
|   108|   1959|   5.0| 3.8653853|
|   368|   2122|   2.0| 1.8430785|
|   436|    471|   3.0| 3.5769286|
|   587|   1580|   4.0| 3.8385806|
|    27|   1580|   3.0|  3.350423|
|   606|   1580|   2.5|  3.189837|
|   606|  44022|   4.0| 2.8213139|
|    91|   2122|   4.0|  2.240532|
|   157|   3175|   2.0| 3.6166894|
|   232|   1580|   3.5|  3.388801|
|   232|  44022|   3.0|  3.102427|
|   246|   1645|   4.0| 3.8419445|
|   599|   2366|   3.0| 2.8803504|
|   111|   1088|   3.0| 3.1283107|
|   111|   3175|   3.5| 2.9588284|
|    47|   1580|   1.5| 2.6710057|
|   140|   1580|   3.0| 3.3761976|
+------+-------+------+----------+
only showing top 20 rows
```

Results

Make Recommendations

```
# Generate n Recommendations for all users
nrecommendations = best_model.recommendForAllUsers(10)
nrecommendations.limit(10).show()
```

```
+------+--------------------+
|userId|     recommendations|
+------+--------------------+
|     1|[{3379, 5.6955385...|
|     2|[{131724, 4.79241...|
|     3|[{6835, 4.849324}...|
|     4|[{3851, 4.846883}...|
|     5|[{3379, 4.509605}...|
|     6|[{3925, 4.8073444...|
|     7|[{3379, 4.543719}...|
|     8|[{3379, 4.712199}...|
|     9|[{3379, 4.8413095...|
|    10|[{71579, 4.531132...|
+------+--------------------+
```

8s

```
nrecommendations = nrecommendations\
    .withColumn("rec_exp", explode("recommendatio
    .select('userId', col("rec_exp.movieId"), col

nrecommendations.limit(10).show()
```

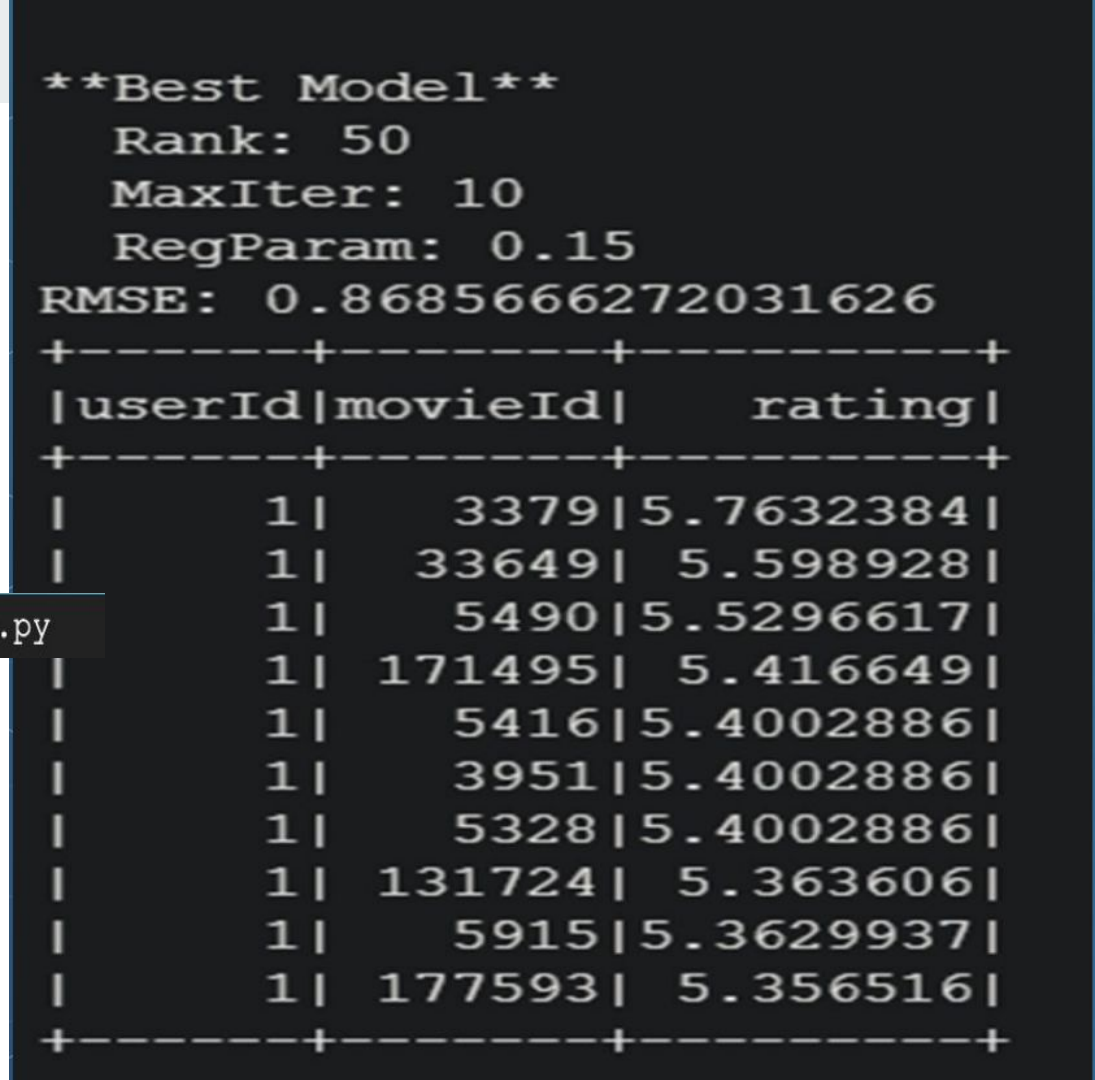
```
+------+-------+---------+
|userId|movieId|   rating|
+------+-------+---------+
|     1|   3379|5.6955385|
|     1|  33649|5.5589347|
|     1|   5490| 5.546711|
|     1| 171495|5.4176087|
|     1|   5915|5.3804026|
|     1|   5416|5.3631263|
|     1|   5328|5.3631263|
|     1|   3951|5.3631263|
|     1|  78836| 5.342607|
|     1| 184245|5.3101015|
+------+-------+---------+
```

```
ratings.join(movies, on='movieId').filter('userId = 100').sort('rating', ascending=False).limit(10).show()
```

```
+-------+------+------+--------------------+--------------------+
|movieId|userId|rating|               title|              genres|
+-------+------+------+--------------------+--------------------+
|   1101|   100|   5.0|      Top Gun (1986)|      Action|Romance|
|   1958|   100|   5.0|Terms of Endearme...|        Comedy|Drama|
|   2423|   100|   5.0|Christmas Vacatio...|              Comedy|
|   4041|   100|   5.0|Officer and a Gen...|       Drama|Romance|
|   5620|   100|   5.0|Sweet Home Alabam...|      Comedy|Romance|
|    368|   100|   4.5|     Maverick (1994)|Adventure|Comedy|...|
|    934|   100|   4.5|Father of the Bri...|              Comedy|
|    539|   100|   4.5|Sleepless in Seat...|Comedy|Drama|Romance|
|     16|   100|   4.5|       Casino (1995)|         Crime|Drama|
|    553|   100|   4.5|    Tombstone (1993)|Action|Drama|Western|
+-------+------+------+--------------------+--------------------+
```

GCP

```
spark submit recommendation_engine_movielens.py
```



Conclusion

- "This project demonstrates the creation of a personalized movie recommendation engine using the MovieLens dataset and PySpark."
- "Future improvements could include incorporating more user-specific features and real-time recommendations."