درس مبانی رایانش ابری

ترم پاییز ۱۴۰۱

تمرین شماره ۴ (امتیازی)

نام و نام‌خانوادگی: نگار کرمی **شماره دانشجویی:** ۹۷۲۲۰۳۹

نام و نام‌خانوادگی: سپهر مقیسه **شماره دانشجویی:** ۹۸۳۱۱۰۳

استاد درس: دکتر جوادی

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

مرحله‌ی ۱: خواندن داده‌ها

```
spark = SparkSession.builder.appName("Game Recommendation System").getOrCreate()
```

```
game_df = spark.read.csv('games.csv', header=True).cache()
rating_df = spark.read.csv('ratings.csv', header=True).cache()
```

```
game_df = game_df.withColumn("game_id", game_df["game_id"].cast('int'))

rating_df = rating_df.withColumn("user_id", rating_df["user_id"].cast('int'))
rating_df = rating_df.withColumn("game_id", rating_df["game_id"].cast('int'))
rating_df = rating_df.withColumn("rating", rating_df["rating"].cast('int'))
```

در ابتدا یک spark session می‌سازیم و از فایل‌های csv که داریم، داده‌هایمان را به صورت Data frame می‌خوانیم.
سپس ستون id آن‌ها را که از نوع string است به نوع int تبدیل می‌کنیم.

مرحله‌ی ۲: تعریف جداول

```
target_id = input('Enter ID: ')
```

```
rating_df.registerTempTable("ratings")
game_df.registerTempTable("games")
```

ابتدا id کاربری که می‌خواهیم بازی به آن پیشنهاد دهیم را به عنوان ورودی دریافت می‌کنیم. سپس داده‌هایی که خوانده بودیم را به عنوان table تعریف و ثبت می‌کنیم.

مرحله‌ی ۳: ساختن مدل ALS

```
# Alternating Least Squares - PySpark Collaborative Filtering
# ALS by default does explicit feedback
als = ALS()
als.setMaxIter(2) # max number of iter to run
als.setRegParam(0.01) # specifies the regularization param in ALS
als.setUserCol('user_id') # users column
als.setItemCol('game_id') # ratings column
als.setRatingCol('rating') # game's ratings
alsModel.setColdStartStrategy('drop') # in order to ensure we wont get NaN (Not a Number) values
```

در این بخش از الگوریتم ALS موجود در Spark.ml استفاده می‌کنیم. ALS یک الگوریتم فاکتورسازی ماتریسی است که از حداقل مربعات متناوب با Weighted-Lambda-Regularizationاستفاده می‌کند.

تعریف هر یک از پارامترهای این تابع در تصویر بالا نوشته شده است.

به طور پیش فرض، Spark پیش بینی‌های NaN را در طول ALSModel اختصاص می‌دهد اما به کاربران این امکان را می دهد که پارامتر coldStartStrategy را روی "Drop" تنظیم کنند تا هر ردیفی در DataFrame پیش بینی‌ها که حاوی مقدار NaN هست، حذف شود. سپس معیار ارزیابی بر روی داده های غیر NaN محاسبه می شود و معتبر خواهد بود.

مرحله‌ی ۴: آموزش مدل ALS

```
(training_data, test_data) = rating_df.randomSplit([0.8, 0.2])

als_model = als.fit(training_data)
```

در این قسمت، ۷۰درصد داده‌ها را برای آموزش و train مدل ALS استفاده می‌کنیم و از ۲۰ درصد دیگر برای پیش‌بینی دقت پیشنهادهایی که بدست می‌آوریم، استفاده می‌کنیم.

مرحله‌ی ۵: بدست آوردن مقدار خطای پیش‌بینی

```
predictions = als_model.transform(test_data)

evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="predictions")
rmse = evaluator.evaluate(predictions)

print("Error = ", str(rmse))
```

در این قسمت با استفاده از داده‌های تست پیش‌بینی را حساب می‌کنیم و با یک ارزیابی regression خطای آن را حساب می‌کنیم و خروجی می‌دهیم.

مرحله‌ی ۶: بدست آوردن پیشنهاد بازی‌ها برای کاربر مورد نظر

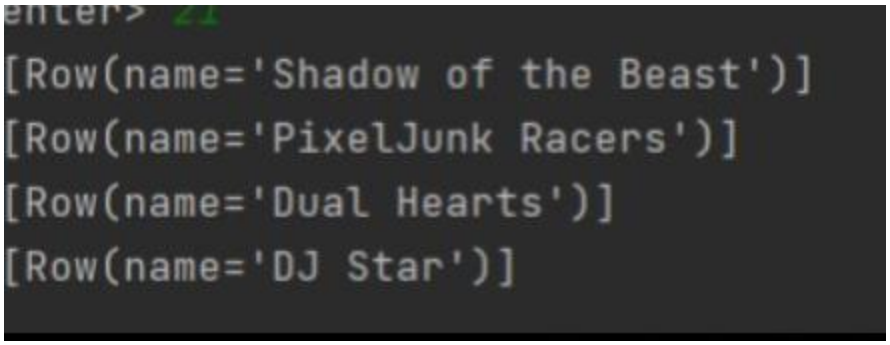
در ابتدا یک sql query برای ترکیب دو جدول games و ratings می‌نویسیم تا بازی‌ها را در کنار امتیازاتی که دریافت کرده‌اند، داشته باشیم.

سپس با استفاده از مدل ALS برای کاربر مورد نظر خود ۵ پیشنهاد برتر را بدست می‌آوریم.

در آخرین مرحله نیز، نتیجه‌ی این ۵ پیشنهاد را به شکل زیر مرتب می‌کنیم و در خروجی نمایش می‌دهیم.

۱) بازی‌ها را بر اساس امتیازهایشان به ترتیب نزولی نمایش می‌دهیم.
۲) در صورتی که دو برنامه امتیاز یکسان داشتند، برنامه با ایندکس کمتر انتخاب می‌شود.

```
[Row(name='The Amazing Spider-Man')]
[Row(name='Burnout Crash!')]
[Row(name='Alien Breed: Impact')]
[Row(name='Spider-Man: Edge of Time')]
```


enter> 21
[Row(name='Shadow of the Beast')]
[Row(name='PixelJunk Racers')]
[Row(name='Dual Hearts')]
[Row(name='DJ Star')]