

1.1 DESCRIPTION OF SET UP

In this task, I am required to analyze and train a model for a game recommendation system using the steam_200k data set which contains data about games, users and purchase/play implicit feedback of the users to an associated game.

I will be making use of the Databricks platform to perform analysis and also apply a machine learning algorithm using pyspark's ALS algorithms for recommendation systems.

Our goal is to develop a recommendation system for the games which will be used to recommend games to users using a collaborative filtering approach, by training our models using ALS, using the Game play time as an implicit user feedback of games played.

PREPARATION OF ENVIRONMENT (COMPUTE CLUSTER AND NOTEBOOK)

In order to complete this task on Databricks, we need to create a machine learning compute cluster which will be responsible for providing computational power and a notebook which will be the IDE where our codes will be written.

1. CREATING A CLUSTER

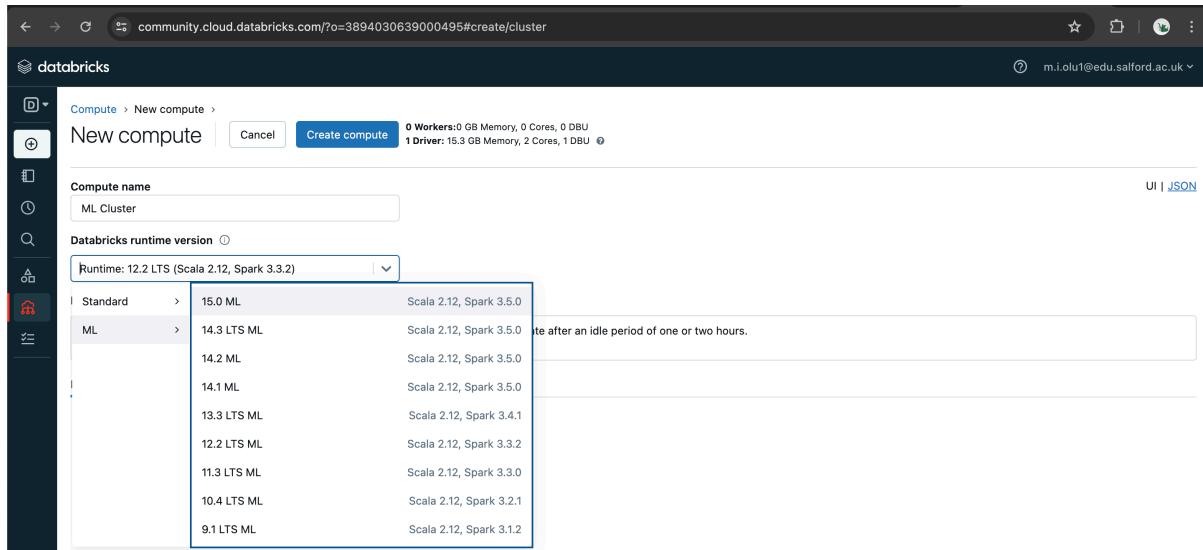


Figure 2.1.1

In figure 2.1.1, I created a machine learning cluster by selecting the ML type when creating the cluster.

In figure 2.1.2 below, I created a notebook to write the codes after creating a cluster in the previous figure 2.1.1.

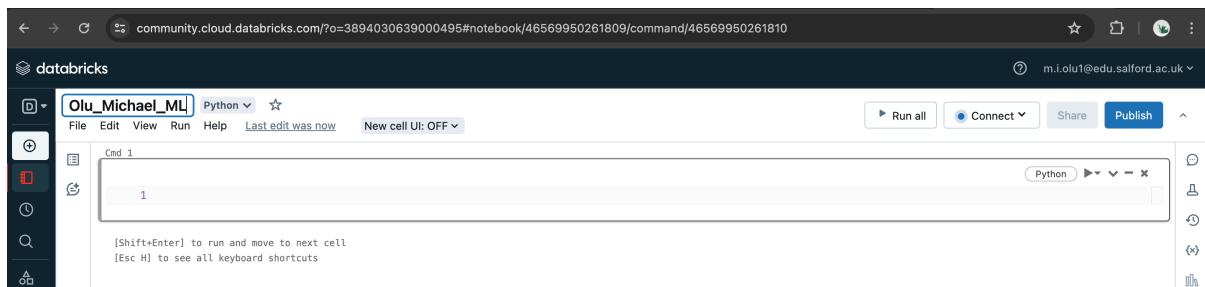


Figure 2.1.2

I created named my notebook Olu_Michael_ML in the figure above where all my codes were written.

2.1 DATA LOADING INTO DATAFRAME

CSV FILE UPLOAD

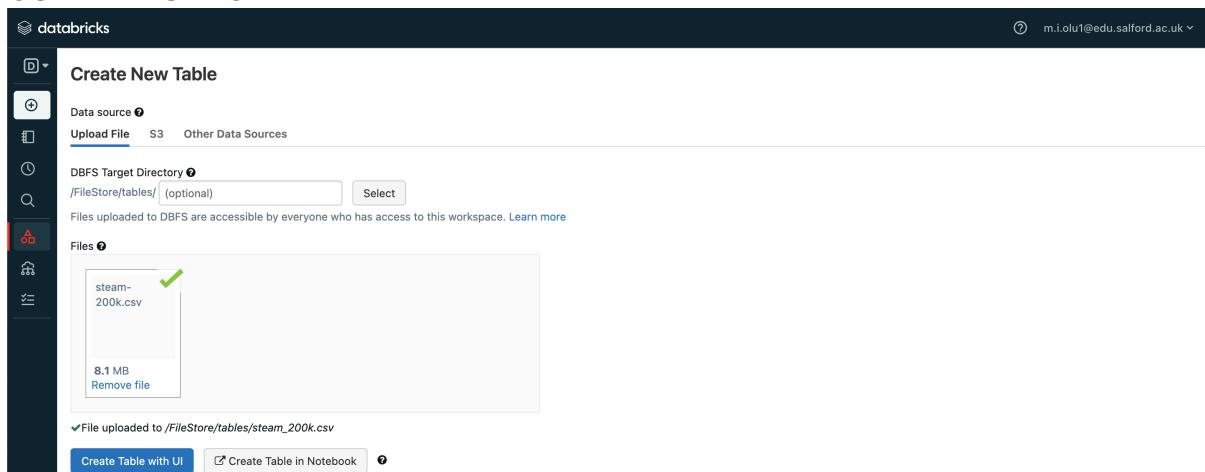


Figure 2.2.1

After our file `steam_200k.csv` has been uploaded, it can be found in the dbfs, the dataframe can now be created from this file.

DATA LOADING INTO PYSPARK DATAFRAME

Before creating the dataframe and running some analysis, I imported the MLflow library and enable autologging.

```
Cmd 2
1 # Import ml flow library to auto-log machine learning runs
2
3 import mlflow
4
5 mlflow.pyspark.ml.autolog()

Command took 0.18 seconds -- by m.i.olul@edu.salford.ac.uk at 24/04/2024, 19:21:24 on ML Cluster
```

Figure 2.2.2

After initiating autologging, I viewed the contents of the file using dbutils as shown in the figure below.

```
Cmd 1
1 steam_file = "steam_200k.csv"
2
3 dbutils.fs.head(f"/FileStore/tables/{steam_file}")

[Truncated to first 65536 bytes]
151603712,The Elder Scrolls V Skyrim,purchase,1\r\n151603712,Th Elder Scrolls V Skyrim,play,273\r\n151603712,Fallout 4,purchase,1\r\n151603712,Fallout 4,play,12.1\r\n151603712,Left 4 Dead 2,purchase,1\r\n151603712,Left 4 Dead 2,play,8.9\r\n151603712,Fallout New Vegas,purchase,1\r\n151603712,Fallout New Vegas,play,7.5\r\n151603712,Poly Bridge,purchase,1\r\n151603712,Poly Bridge,play,7.5\r\n151603712,HuniePop,purchase,1\r\n151603712,HuniePop,play,8.5\r\n151603712,Path of Exile,purchase,1\r\n151603712,Path of Exile,play,3.3\r\n151603712,Team Fortress 2,purchase,1\r\n151603712,Team Fortress 2,play,2.8\r\n151603712,Tomb Raider,purchase,1\r\n151603712,Tomb Raider,play,2.5\r\n151603712,BioShock Infinite,purchase,1\r\n151603712,BioShock Infinite,play,1.3\r\n151603712,Fallout 3 - Game of the Year Edition,play,0.8\r\n151603712,SEGA Genesis & Mega Drive Classics,purchase,1\r\n151603712,SEGA Genesis & Mega Drive Classics,play,0.8\r\n151603712,Grand Theft Auto IV,purchase,1\r\n151603712,Grand Theft Auto IV,play,0.6\r\n151603712,Realm of the Mad God,purchase,1\r\n151603712,Realm of the Mad God,play,0.5\r\n151603712,Marvel Heroes 2015,purchase,1\r\n151603712,Marvel Heroes 2015,play,0.5\r\n151603712,Eldevin,purchase,1\r\n151603712,Eldevin,play,0.5\r\n151603712,Dota 2,purchase,1\r\n151603712,Dota 2,play,0.5\r\n151603712,BioShock,play,0.5\r\n151603712,Robocraft,purchase,1\r\n151603712,Robocraft,play,0.4\r\n151603712,Garry's Mod,purchase,1\r\n151603712,Garry's Mod,play,0.1\r\n151603712,Jazzpunk,purchase,1\r\n151603712,Jazzpunk,play,0.1\r\n151603712,Alan Wake,purchase,1\r\n151603712,BioShock 2,purchase,1\r\n151603712,Fallen Earth,purchase,1\r\n151603712,Fallout New Vegas Courier's Stash,purchase,1\r\n151603712,Fallout New Vegas Dead Money,purchase,1\r\n151603712,Fallout New Vegas Honest Hearts,purchase,1\r\n151603712,Grand Theft Auto Episodes from Liberty City,purchase,1\r\n151603712,Hitman Absolution,purchase,1\r\n151603712,HuniePop Official Digital Art Collection,purchase,1\r\n151603712,HuniePop Original Soundtrack,purchase,1\r\n151603712,The Banner Saga - Mod Content,purchase,1\r\n151603712,The Elder Scrolls V Skyrim - Dawnguard,purchase,1\r\n151603712,The Elder Scrolls V Skyrim - Dragonborn,purchase,1\r\n151603712,The Elder Scrolls V Skyrim - Hearthfire,purchase,1\r\n187131847,Dota 2,purchase,1\r\n187131847,Dota 2,play,2.3\r\n59945701,Ultra Street Fighter IV,purchase,1\r\n59945701,Ultra Street Fighter IV,play,238\r\n59945701,FINAL FANTASY XIII,purchase,1\r\n59945701,FINAL FANTASY XIII,play,84\r\n59945701,The Elder Scrolls V Skyrim,purchase,1\r\n59945701,The Elder Scrolls V Skyrim,play,58\r\n59945701,Sid Meier's Civilization V,purchase,1\r\n59945701,Sid Meier's Civilization V,play,22\r\n59945701.L.A. Noire,purchase,1\r\n59945701.L.A. Noire,play,13.8\r\n59945701.Company Command took 0.23 seconds -- by m.i.olul@edu.salford.ac.uk at 24/04/2024, 17:13:47 on ML
```

Figure 2.2.3

After viewing file contents and seeing its structure and delimiter, I created the dataframe using pyspark as shown in the figure 2.2.4.

```
Cmd 3
1 # importing spark functions for some dataframe manipulations
2 from pyspark.sql.functions import *
3
4 # function for creating a dataframe using spark
5 def create_dataframe(_file):
6     df = spark.read.options(delimiter =",").csv(f"/FileStore/tables/{_file}")
7     return df
8
9 # calling function for dataframe creation with our csv file as argument
10 steam_dataframe = create_dataframe(steam_file)
11 steam_dataframe.show()

▶ (2) Spark Jobs
▶ steam_dataframe: pyspark.sql.dataframe.DataFrame = [c0: string, c1: string ... 2 more fields]

+---+---+---+---+
| _c0 | _c1 | _c2 | _c3 |
+---+---+---+---+
|151603712|The Elder Scrolls...|purchase| 1|
|151603712|The Elder Scrolls...| play| 273|
|151603712| Fallout 4|purchase| 1|
|151603712| Fallout 4| play| 87|
|151603712| Spore|purchase| 1|
|151603712| Spore| play|14.9|
|151603712| Fallout New Vegas|purchase| 1|
|151603712| Fallout New Vegas| play|12.1|
|151603712| Left 4 Dead 2|purchase| 1|
|151603712| Left 4 Dead 2| play| 8.9|
|151603712| HuniePop|purchase| 1|
|151603712| HuniePop| play| 8.5|
|151603712| Path of Exile|purchase| 1|
|151603712| Path of Exile| play| 8.1|
|151603712| Poly Bridge|purchase| 1|
|151603712| Poly Bridge| play| 7.5|
|151603712| Left 4 Dead|purchase| 1|
|151603712| Left 4 Dead| play| 3.3|
Command took 1.49 seconds -- by m.i.olul@edu.salford.ac.uk at 24/04/2024, 19:23:19 on ML Cluster
```

Figure 2.2.4

The dataframe was created and the schema was inferred automatically. But I have to alter this schema and change the column names to more suitable names for each column.

```

Cmd 5
Python ▶ v - x

1 # Function to alter schema of the dataframe
2 def alter_schema_name(dataframe):
3     # Renaming the Schema
4     df_renamed = dataframe.withColumnRenamed("_c0", "User ID").withColumnRenamed("_c1", "Game Title").withColumnRenamed("_c2", "Member Behaviour").
5     withColumnRenamed("_c3", "Game Play Time")
6     # Altering the datatypes of dataframe schema
7     df = df_renamed.withColumn("User ID", df_renamed["User ID"].cast("int")).withColumn("Game Play Time", df_renamed["Game Play Time"].cast("float"))
8     return df
9
10 steam_dataframe = alter_schema_name(steam_dataframe)
11 steam_dataframe.show(20)

> (1) Spark Jobs
> (1) Spark Jobs
> steam_dataframe: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game Title: string ... 2 more fields]
+-----+-----+-----+-----+
| User ID | Game Title | Member Behaviour | Game Play Time |
+-----+-----+-----+-----+
| 151603712 | The Elder Scrolls... | purchase | 1.0 |
| 151603712 | The Elder Scrolls... | play | 273.0 |
| 151603712 | Fallout 4 | purchase | 1.0 |
| 151603712 | Fallout 4 | play | 87.0 |
| 151603712 | Spore | purchase | 1.0 |
| 151603712 | Spore | play | 14.9 |
| 151603712 | Fallout New Vegas | purchase | 1.0 |
| 151603712 | Fallout New Vegas | play | 12.1 |
| 151603712 | Left 4 Dead 2 | purchase | 1.0 |
| 151603712 | Left 4 Dead 2 | play | 8.9 |
| 151603712 | HuniePop | purchase | 1.0 |
| 151603712 | HuniePop | play | 8.5 |
| 151603712 | Path of Exile | purchase | 1.0 |
| 151603712 | Path of Exile | play | 8.1 |
| 151603712 | Poly Bridge | purchase | 1.0 |
| 151603712 | Poly Bridge | play | 7.5 |
| 151603712 | Left 4 Dead | purchase | 1.0 |
| 151603712 | Left 4 Dead | play | 3.31
Command took 1.43 seconds -- by m.i.olu1@edu.salford.ac.uk at 24/04/2024, 19:21:24 on ML Cluster

```

Figure 2.2.5

Figure 2.2.5 shows the schema which have been renamed using the `withColumnRenamed` methods to alter the dataframe schema, and also `withColumn` method to change the datatypes some columns.

2.2 EXPLORATORY ANALYSIS AND VISUALIZATIONS

Before training the recommendation system, I did some initial exploration of the dataset.

1. Number of unique records in the dataset

Cmd 6

Python ► ▾ ▷ ×

```
1 #Analysis 1: Number of distinct records
2
3 steam_dataframe.distinct().count()
```

▶ (3) Spark Jobs
19923

Command took 6.94 seconds -- by m.i.oluwaseun.edu.salford.ac.uk at 24/04/2024, 19:21:24 on ML Cluster

Figure 2.2.6

The result in figure 2.2.6 shows that there are 199,293 unique records in the dataset.

2. Number of unique users

```
Cmd 7
```

```
Python ► ▾ ▷ ×
```

```
1 #Analysis 2: Unique users
2
3 steam_dataframe.select("User ID").distinct().count()
```

▶ (3) Spark Jobs

12393

Command took 3.38 seconds -- by m.i.olul@edu.salford.ac.uk at 24/04/2024, 19:21:24 on ML Cluster

Figure 2.2.7

Counting the User ID distinctly shows that there are 12,393 different users in the dataset.

3. Number of unique games

Cmd 8

```
Python ▶ ▷ ▷ -
```

```
1 # Analysis 3: Number of Unique Games
2
3 steam_dataframe.select("Game Title").distinct().count()
```

▶ (3) Spark Jobs

5155

Command took 2.22 seconds --- by m.i.olui@edu.salford.ac.uk at 24/04/2024, 19:21:24 on ML Cluster

Figure 2.2.8

Counting the Game ID distinctly shows that there are 5,155 different games in the dataset.

4. Top 10 most popularly played games

```
Cmd 9
1 # #Analysis 4: Top 10 Most popularly played games
2
3 steam_dataframe.select("Game Title", "Member Behaviour", "Game Play Time").filter(steam_dataframe["Member Behaviour"] == "play").groupBy("Game Title").count().orderBy('count', ascending=False).limit(10).display()
```

▶ (2) Spark Jobs

Table Visualization 1 +

| Game Title | count |
|-----------------------------------|-------|
| 1 Dota 2 | 4841 |
| 2 Team Fortress 2 | 2323 |
| 3 Counter-Strike Global Offensive | 1377 |
| 4 Unturned | 1069 |
| 5 Left 4 Dead 2 | 801 |
| 6 Counter-Strike Source | 715 |

↓ 10 rows | 3.81 seconds runtime Refreshed 3 minutes ago

Command took 3.81 seconds -- by m.i.olul@edu.salford.ac.uk at 25/04/2024, 16:21:36 on ML Cluster 2

Figure 2.2.9

Figure 2.2.9 shows the 10 most popularly played games based on the number of people who has played the game in descending order with `Dota 2` as the most popular and `Sid Meier's Civilization V` as the 10th most popular. Figure 2.2.10 shows a visualization of this result.

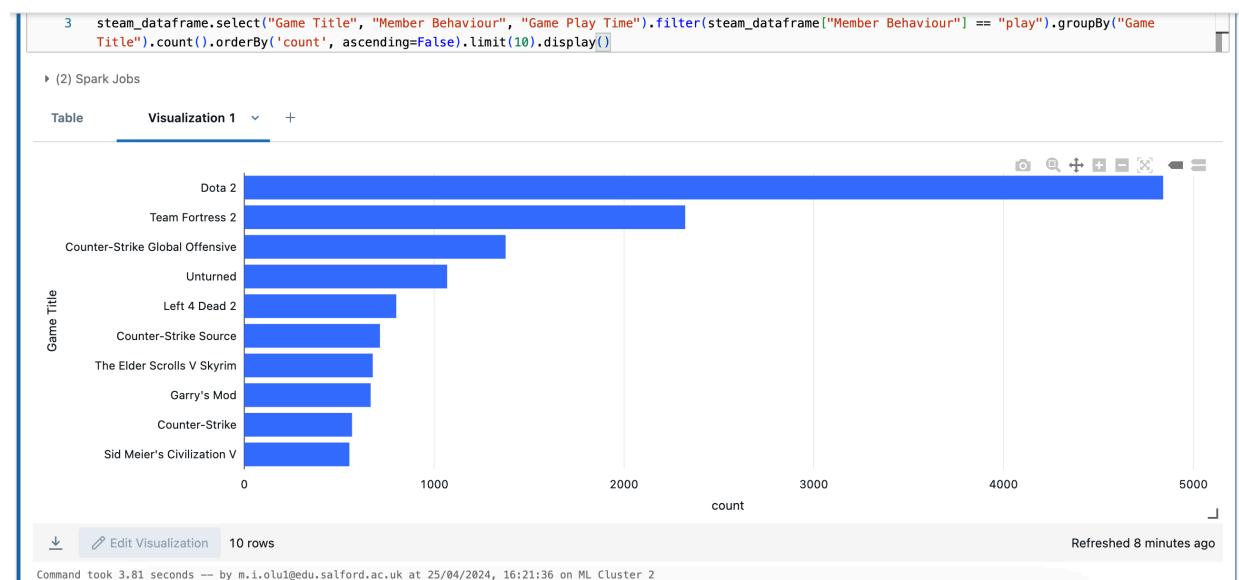


Figure 2.2.10

5. Top 10 games with the highest cumulative play time in hours

```
Cmd 10
Python ▶ v last □ ×

1 #Analysis 5: Top 10 Games with the highest number of play time in hours
2
3 steam_dataframe.select("Game Title", "Member Behaviour", "Game Play Time").filter(steam_dataframe["Member Behaviour"] == "play").groupBy("Game Title").sum("Game Play Time").orderBy('sum(Game Play Time)', ascending=False).limit(10).display()

▶ (2) Spark Jobs

Table +
```

| | Game Title | sum(Game Play Time) |
|---|---------------------------------|---------------------|
| 1 | Dota 2 | 981684.6000046805 |
| 2 | Counter-Strike Global Offensive | 322771.60000587255 |
| 3 | Team Fortress 2 | 173673.30000534654 |
| 4 | Counter-Strike | 134261.1000032574 |
| 5 | Sid Meier's Civilization V | 99821.30000032485 |
| 6 | Counter-Strike Source | 96075.4999980852 |

↓ 10 rows | 3.08 seconds runtime Refreshed 24 minutes ago

Command took 3.08 seconds -- by m.i.olui@edu.salford.ac.uk at 25/04/2024, 16:21:36 on ML Cluster 2

Figure 2.2.11

Figure 2.2.11 displays the top 10 most popular games based on cumulative play time, with Dota 2 having the highest cumulative playtime and Left 4 Dead 2 being the 10th highest, with the results displayed in descending order and visualized in Figure 2.2.12.

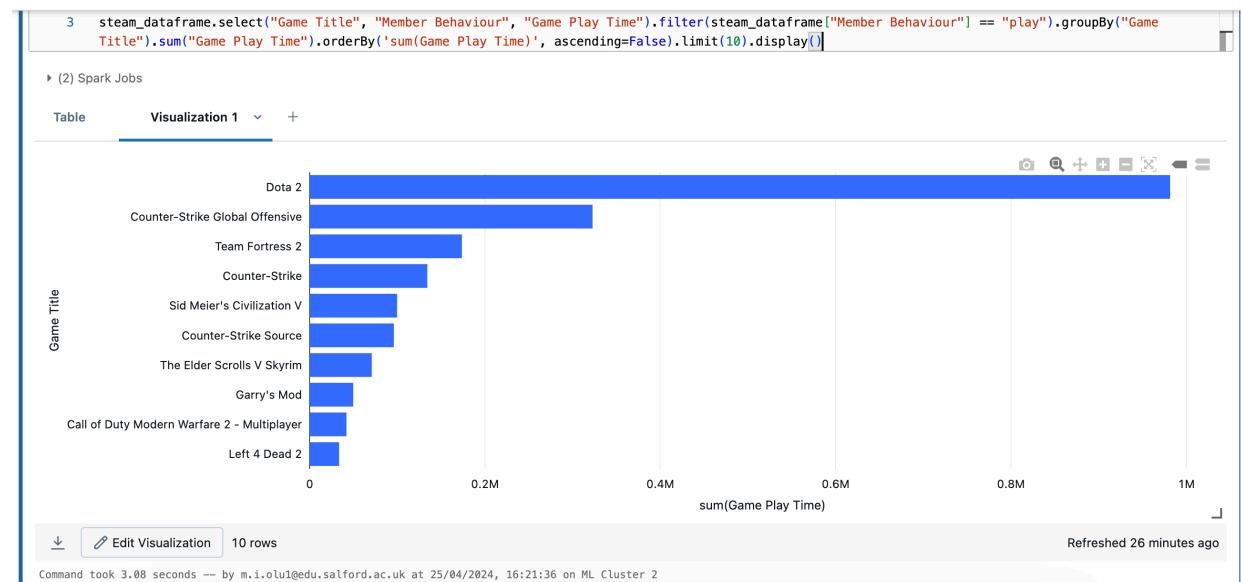


Figure 2.2.12

6. Top 10 most popularly purchased games.

Cmd 11

```

1 # Analysis 6: Top 10 Most popularly purchased games
2
3 steam_dataframe.select("Game Title", "Member Behaviour", "Game Play Time").filter(steam_dataframe["Member Behaviour"] == "purchase").groupBy("Game Title").count().orderBy('count', ascending=False).limit(10).display()

```

▶ (2) Spark Jobs

Table Visualization 1 +

| Game Title | count |
|-----------------------------------|-------|
| 1 Dota 2 | 4841 |
| 2 Team Fortress 2 | 2323 |
| 3 Unturned | 1563 |
| 4 Counter-Strike Global Offensive | 1412 |
| 5 Half-Life 2 Lost Coast | 981 |
| 6 Counter-Strike Source | 978 |

↓ 10 rows | 1.90 seconds runtime Refreshed 1 hour ago

Command took 1.90 seconds -- by m.i.olul@edu.salford.ac.uk at 25/04/2024, 16:21:36 on ML Cluster 2

Figure 2.2.13

Figure 2.2.13 shows the 10 most popularly purchased games in descending order. The result shows Dota 2 as the most purchased game with 4841 purchases and `Half-Life 2 Deathmatch` as the 10th with 823 purchases. Figure 2.2.14 shows a visualization of the result.

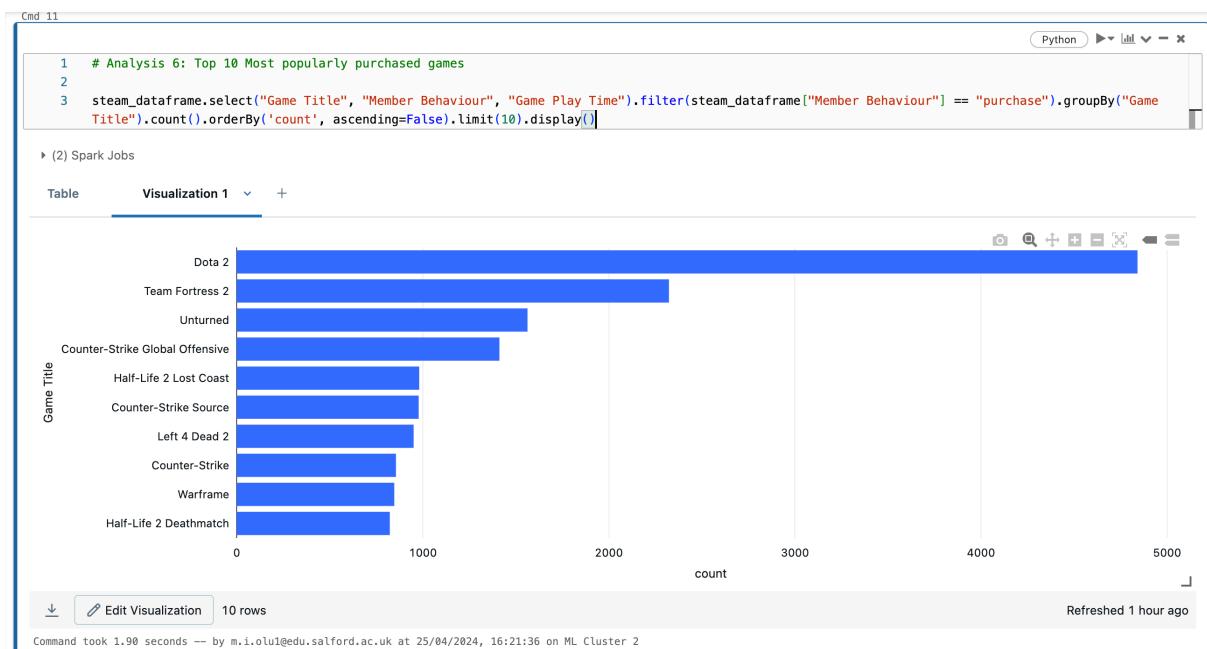


Figure 2.2.14

3 DATA PREPARATION AND PRE-PROCESSING

To train and test a model with ALS, a unique integer ID is required for the item column (Games) and a pre-determined ratio of test and training datasets.

Our original dataset did not have a column for unique IDs for the Games, so I generated Game IDs in a new column and attach this column to our dataset before we split the dataset into the training set and the test sets.

The screenshot shows a Jupyter Notebook cell titled "Cmd 12" containing Python code. The code generates a unique ID for each game in the dataset by selecting the "Game Title" column, applying the distinct function, and then using the monotonically_increasing_id function to create a new column "Game ID". The resulting DataFrame is named "game_id_dataframe". The code is as follows:

```
1 # Preparing data for ML training
2
3 # Generating a unique ID for each game in the dataset
4 game_id_dataframe = steam_dataframe.select("Game Title").distinct().withColumn('Game ID', monotonically_increasing_id()).withColumnRenamed("Game Title", "Game")
5
6 game_id_dataframe.show(truncate=False)
```

Below the code, the output shows the first 20 rows of the "game_id_dataframe" DataFrame, which contains two columns: "Game" and "Game ID". The "Game" column lists various game titles, and the "Game ID" column lists their corresponding unique IDs. The output is as follows:

| Game | Game ID |
|---|---------|
| Dota 2 | 0 |
| METAL GEAR SOLID V THE PHANTOM PAIN | 1 |
| LEGO Batman The Videogame | 2 |
| RIFT | 3 |
| Anodyne | 4 |
| Legend of Grimrock | 5 |
| Divinity Original Sin | 6 |
| Meltdown | 7 |
| SanctuaryRPG Black Edition | 8 |
| Snuggle Truck | 9 |
| Lunar Flight | 10 |
| Dungeons 2 | 11 |
| Zuma's Revenge | 12 |
| HassleHeart | 13 |
| Ihf Handball Challenge 12 | 14 |
| NEON STRUCT Soundtrack & Artbook | 15 |
| Dust An Elysian Tail | 16 |
| Call of Duty Modern Warfare 2 - Multiplayer | 17 |

Command took 1.98 seconds -- by m.i.olul@edu.salford.ac.uk at 25/04/2024, 17:57:51 on ML Cluster 3

Figure 2.3.1

The unique ID for each game was generated by extracting the Game Title column and applying the spark function `monotonically_increasing_id()`, which assigns a unique serial number to each row of the distinct game dataset.

I joined this new dataframe to the main steam_dataframe with the condition where the game title in the main dataframe is equal to the game title in the game_id_dataframe.

Cmd 13

```

1 # Joining the Generated Game ID dataframe with the main Dataframe
2
3 steam_dataframe_id = game_id_dataframe.join(steam_dataframe, game_id_dataframe["GameID"] == steam_dataframe["Game Title"]).drop("Game").select("UserID", "GameID", "Game Title", "Member Behaviour", "Game Play Time")
4 steam_dataframe_id.show()

```

▶ (3) Spark Jobs

steam_dataframe_id: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: long ... 3 more fields]

| User ID | Game ID | Game Title | Member Behaviour | Game Play Time |
|-----------|---------|----------------------|------------------|----------------|
| 151603712 | 2609 | The Elder Scrolls... | purchase | 1.0 |
| 151603712 | 2609 | The Elder Scrolls... | play | 273.0 |
| 151603712 | 410 | Fallout 4 | purchase | 1.0 |
| 151603712 | 410 | Fallout 4 | play | 87.0 |
| 151603712 | 3868 | Spore | purchase | 1.0 |
| 151603712 | 3868 | Spore | play | 14.9 |
| 151603712 | 3820 | Fallout New Vegas | purchase | 1.0 |
| 151603712 | 3820 | Fallout New Vegas | play | 12.1 |
| 151603712 | 69 | Left 4 Dead 2 | purchase | 1.0 |
| 151603712 | 69 | Left 4 Dead 2 | play | 8.9 |
| 151603712 | 3340 | HuniePop | purchase | 1.0 |
| 151603712 | 3340 | HuniePop | play | 8.5 |
| 151603712 | 1562 | Path of Exile | purchase | 1.0 |
| 151603712 | 1562 | Path of Exile | play | 8.1 |
| 151603712 | 4116 | Poly Bridge | purchase | 1.0 |
| 151603712 | 4116 | Poly Bridge | play | 7.5 |
| 151603712 | 1992 | Left 4 Dead | purchase | 1.0 |
| 151603712 | 1992 | Left 4 Dead | play | 3.3 |

Command took 3.09 seconds -- by m.i.olui@edu.salford.ac.uk at 25/04/2024, 17:57:51 on ML Cluster 3

Figure 2.3.2

In the above image, the dataframe has been joined successfully, the next step is splitting the dataframe into the training and the test set.

Cmd 14

```

1 # Splitting data into training and test set
2
3 steam_dataframe_id = steam_dataframe_id.filter(steam_dataframe_id["Member Behaviour"] == "play")
4 training_set, test_set = steam_dataframe_id.randomSplit([0.9, 0.1])
5 training_set.show(truncate=False)

```

▶ (3) Spark Jobs

steam_dataframe_id: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: long ... 3 more fields]

training_set: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: long ... 3 more fields]

test_set: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: long ... 3 more fields]

| | | | | |
|-------|------|---|------|-------|
| 5250 | 1017 | Portal 2 | play | 13.6 |
| 5250 | 3369 | Alien Swarm | play | 4.9 |
| 5250 | 3458 | Cities Skylines | play | 144.0 |
| 5250 | 3893 | Team Fortress 2 | play | 0.8 |
| 76767 | 17 | Call of Duty Modern Warfare 2 - Multiplayer | play | 165.0 |
| 76767 | 44 | Counter-Strike Source | play | 25.0 |
| 76767 | 1487 | Rise of Nations Extended Edition | play | 5.7 |
| 76767 | 635 | Worms Armageddon | play | 0.4 |
| 76767 | 1017 | Portal 2 | play | 15.0 |
| 76767 | 1477 | Total War ATTILA | play | 207.0 |
| 76767 | 1635 | Call of Duty Modern Warfare 3 | play | 15.9 |
| 76767 | 1970 | Call of Duty Black Ops | play | 22.0 |
| 76767 | 2028 | Call of Duty Modern Warfare 3 - Multiplayer | play | 9.7 |
| 76767 | 2307 | Call of Duty Black Ops - Multiplayer | play | 12.5 |
| 76767 | 2332 | The Stanley Parable | play | 1.8 |
| 76767 | 2353 | Counter-Strike Global Offensive | play | 3.5 |
| 76767 | 2373 | Half-Life | play | 1.2 |
| 76767 | 2378 | Call of Duty World at War | play | 271.0 |

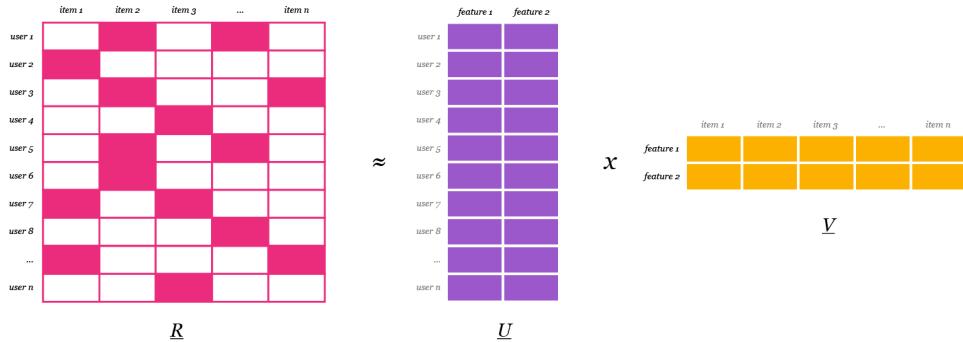
only showing top 20 rows

Command took 4.42 seconds -- by m.i.olui@edu.salford.ac.uk at 25/04/2024, 19:24:57 on ML Cluster 5

Figure 2.3.3

The dataset is divided into a training and test set with a 0.9:0.1 ratio. The dataset is filtered for rows with implicit user behaviour, 'play', to use as implicit user feedback for training the model. The corresponding values (Play time) will be used for the ratingCol parameter during instantiation.

4. MODEL TRAINING USING ALTERNATING LEAST FACTORS ALGORITHM



The ALS algorithm basically tries to get an estimation of a factorized representation of our original data matrix R with each iteration of the optimization process.

We have our users, items, and some kind of feedback data (in this case, 'Game play time') in our original $u \times i$ matrix R . Next, we would like to figure out how to split that into two matrices: one with items and hidden features of size $f \times i$, and another with users and hidden features of size $u \times f$. We've assigned weights to each user/item's relationship to each characteristic in U and V . We compute U and V in such a way that the product of them as nearly resembles R as possible: $R \approx U \times V$. (Victor, 2017).

I made use of the ALS estimator in the MLlib library which uses this factorization and optimization process to train our model. First I imported the ALS class and also the RegressionEvaluator class because we are going to evaluate our model.

I created my own class object which contains various attributes and methods which I will be using to create an object for training, testing and evaluating my model going forward.

The next figure shows where I created the class with the name 'GameRecommenderModel'. I created this class along with some attributes and methods.

```

1 # Training a model using Alternating least squares(ALS) with the training set
2 from pyspark.ml.recommendation import ALS
3 from pyspark.ml.evaluation import RegressionEvaluator
4
5 # Creating a class for recommender system training
6 class GameRecommenderModel:
7     def __init__(self, rank, maxIter, regParam):
8         # Setting ALS object parameters
9         self.rank = rank
10        self.maxIter = maxIter
11        self.regParam = regParam
12        self.userCol = "User ID"
13        self.ratingCol = "Game Play Time"
14        self.itemCol = "Game ID"
15        self.testSet = test_set
16        self.trainningSet = trainning_set
17
18        # ALS object instantiator
19        self.alsObject = ALS(rank=self.rank, maxIter=self.maxIter, regParam=self.regParam, userCol=self.userCol, ratingCol=self.ratingCol,
20        itemCol=self.itemCol, coldStartStrategy="drop", nonnegative=True)
21        # RegressionEvaluator object instantiator
22        self.evaluator = RegressionEvaluator(metricName="rmse", labelCol=self.ratingCol, predictionCol="prediction")
23
24    def trainModel(self): # Model fitting method
25        return self.alsObject.fit(self.trainningSet)
26
27    def getTestSetPredictions(self, model): # Method to get predictions of test
28        return model.transform(self.testSet)
29
30    def getRmse(self, predictions): # Method to get RMSE of the model
31        return self.evaluator.evaluate(predictions)

```

Command took 0.05 seconds -- by m.i.olui@edu.salford.ac.uk at 26/04/2024, 18:43:44 on ML Cluster 7

Figure 2.3.4

The class carries an ALS object and a RegressionEvaluator object inside the init method, and also a, it also carries a trainModel method which will be used to train the model by fitting to the training set, a getTestSetPredictions which will be used to generate predictions for the test set, and finally a getRmse method for evaluating our model.

I made use of classes to create my ALS and Evaluator objects because of code reusability. So that each time I want to train a new model, I can simply create the objects from the class and easily access various characteristics (such as rmse and test predictions dataset) of a model all from one object.

CREATING THE ALS MODEL INSTANCE

```

Cmd 20
1 rank, maxIter, regParam = (15, 15, 0.2) # Defining values for ALS estimator hyperparameters
2
3 gameModel = GameRecommenderModel(rank, maxIter, regParam) # Creating object (ALS instance) with hyper parameters
4 trainedGameModel = gameModel.trainModel() # Calling method for training model

► (9) Spark Jobs
▼ (1) MLflow run
Logged 1 run to an experiment in MLflow. Learn more

2024/05/02 04:05:40 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID 'f6e62f4dc9b94f7e8af627421047830b', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.ml workflow
2024/05/02 04:05:54 WARNING mlflow.utils.autologging_utils: MLflow autologging encountered a warning: "/databricks/python/lib/python3.11/site-packages/mlflow/wrappers/utils.py:393: UserWarning: Hint: Inferred schema contains integer column(s). Integer columns in Python cannot represent missing values. If your input data contains missing values at inference time, it will be encoded as floats and will cause a schema enforcement error. The best way to avoid this problem is to infer the model schema based on a realistic data sample (training dataset) that includes missing values. Alternatively, you can declare integer columns as doubles (float64) whenever these columns may have missing values. See 'Handling Integers With Missing Values <https://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>' for more details."
2024/05/02 04:07:10 WARNING mlflow.pyspark.ml: Model ALS_c52659ad873c will not be autologged because it is not allowlisted or or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistFile Spark conf (see mlflow.pyspark.ml.autolog docs for more info).

Command took 1.55 minutes -- by m.i.olui@edu.salford.ac.uk at 02/05/2024, 04:51:44 on My Cluster 7

```

Figure 2.3.5

The ALS model instance was created using initial values of 15, 15, and 0.2 for the first experiment run, and trained using the trainModel() method from the object

created. The output indicates that the training session has been logged in the experiment UI.

The screenshot shows the Databricks Experiment UI. At the top, it displays 'Experiments > Michael_Olu_mi' and 'Provide Feedback'. Below this, it shows 'Experiment ID: 624775347935368' and 'Artifact Location: dbfs:/databricks/miflow-tracking/624775347935368'. A search bar contains the query 'metrics.rmse < 1 and params.model = "tree"'. The interface includes filters for 'Time created', 'State: Active', 'Datasets', 'Sort: Created', 'Columns', and 'Expand rows'. There are tabs for 'Table', 'Chart', 'Evaluation', and 'Preview'. The main area shows a table of runs with columns: Run Name, Created, Duration, Source, maxIter, rank, and regParam. The first run, 'grandiose-crab-501', is highlighted in red.

Figure 2.3.6

We will use this UI as tool to evaluate and compare our model, trained with different hyper parameters for subsequent runs towards a goal to obtain the best model from our dataset.

VIEWING PREDICTIONS FOR TEST SET

The screenshot shows a Jupyter Notebook cell with Python code. The code is:

```

1 # Calling the getTestSetPredictions method to view predictions made from trained model
2 predictions = gameModel.getTestSetPredictions(trainedGameModel)
3 predictions.display()

```

Below the code, it says '(2) Spark Jobs' and 'predictions: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: integer ... 4 more fields]'. A 'Table' button is shown. The resulting table has columns: User ID, Game ID, Game Title, Member Behaviour, Game Play Time, and prediction. The data includes:

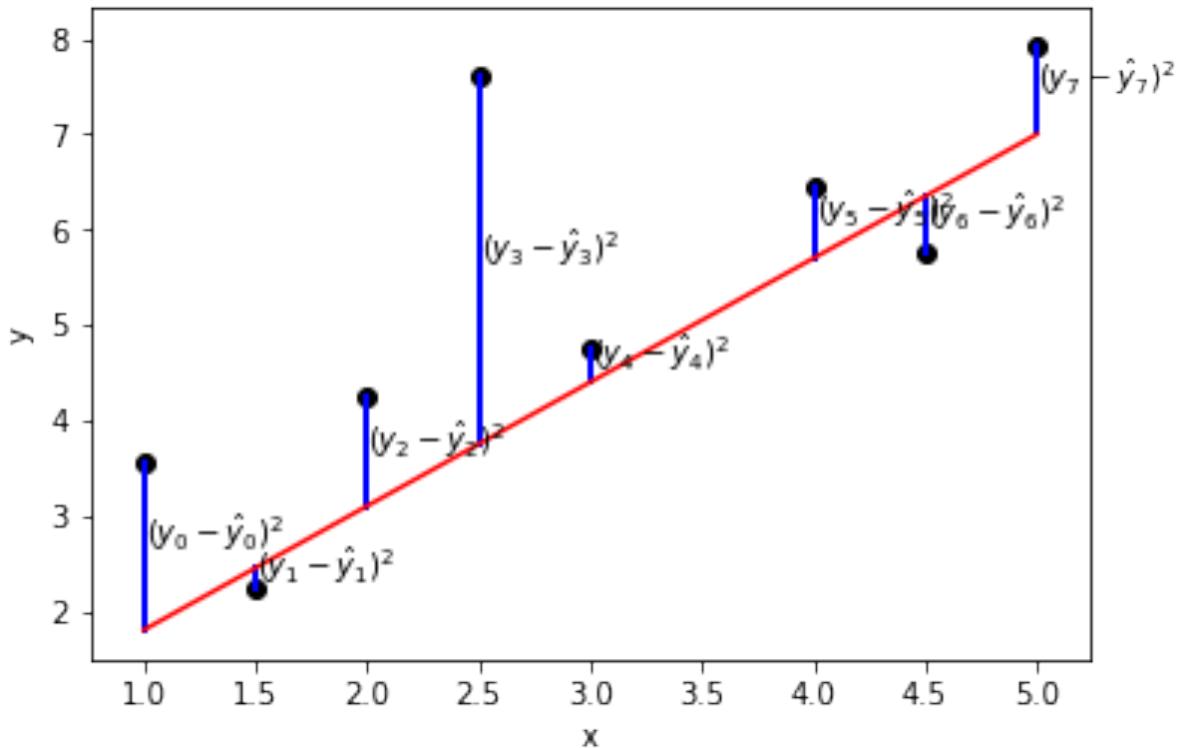
| User ID | Game ID | Game Title | Member Behaviour | Game Play Time | prediction |
|---------|---------|---------------------------------------|------------------|----------------|------------|
| 1 | 1017 | Portal 2 | play | 15 | 16.148693 |
| 2 | 3917 | Banished | play | 24 | 28.640377 |
| 3 | 653 | Dragon Age Origins - Ultimate Edition | play | 0.2 | 3.4520493 |
| 4 | 962 | Sid Meier's Civilization V | play | 135 | 166.31955 |
| 5 | 1329 | Shadow Warrior | play | 0.2 | 7.2455206 |
| 6 | 1918 | The Wolf Among Us | play | 8.2 | 3.8144841 |

At the bottom, it says '6,209 rows | 15.02 seconds runtime' and 'Refreshed 15 hours ago'. A note at the bottom left says 'Command took 15.02 seconds -- by m.i.oluwafemi.saf@edu.salford.ac.uk at 26/04/2024, 18:05:41 on ML Cluster 7'.

Figure 2.3.7

Figure 2.3.7 illustrates the model's predictions vs actual values. To assess the model's performance, we evaluate its RMSE and train more models with different hyper parameters. This process, known as hyper parameter tuning, aims to identify the model with the best performance by generating the lowest RMSE.

The RMSE (root mean squared error) is the root value of the average of the squared sum of differences between the predictions and the actual values.



According to the figure above, the root mean squared error can be calculated by the equation:

$$\text{RMSE} = \sqrt{\frac{(y_0 - \hat{y}_0)^2 + (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots + (y_n - \hat{y}_n)^2}{n}}$$

Where y_n is the actual implicit feedback value, \hat{y}_n is the predicted value and n is the total number of items in the dataset.

The RMSE of the predictions in the dataset in figure 2.3.8 is computed by the RegressionEvaluator object and shown below.

```
Cmd 23
1 # Viewing Root mean squared error for gameModel
2 print(f'Rmse for gameModel: {gameModel.getRmse(predictions)}')

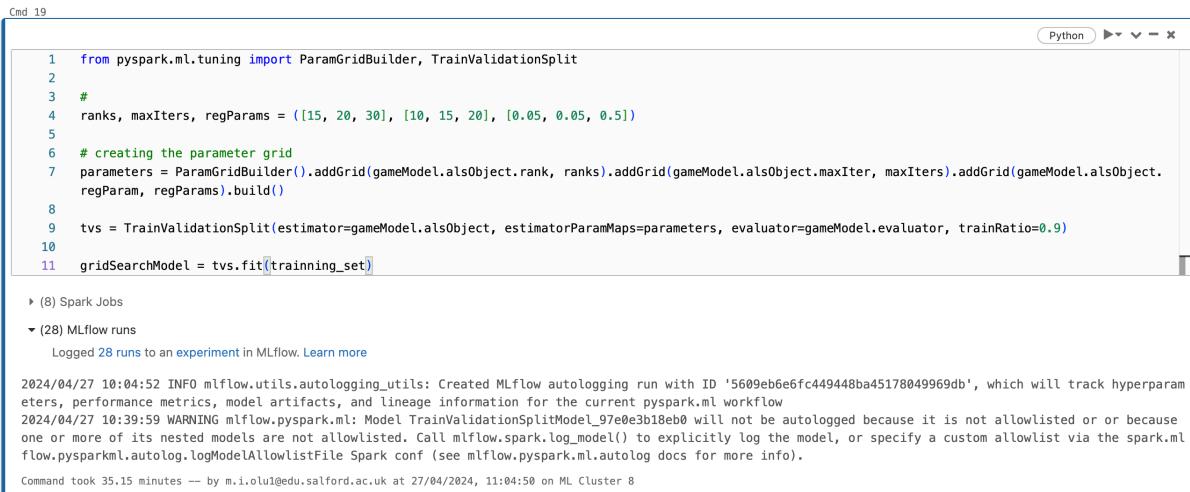
▶ (5) Spark Jobs
Rmse for gameModel: 234.12696611401253
Command took 27.93 seconds -- by m.i.olul@edu.salford.ac.uk at 02/05/2024, 04:51:44 on My Cluster 7
```

Figure 2.3.8

The result above shows the rmse of our first model as ~234.126966

HYPER PARAMETER TUNING

The hyper parameter tuning for my models was done using the `ParamGridBuilder` library from the pyspark ml library. This tool uses a pre-defined range of hyper parameter values and performs multiple runs with different combinations to achieve the best performing models.



```
Cmd 19
Python ▶ v - x

1 from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
2
3 #
4 ranks, maxIters, regParams = ([15, 20, 30], [10, 15, 20], [0.05, 0.05, 0.5])
5
6 # creating the parameter grid
7 parameters = ParamGridBuilder().addGrid(gameModel.alsObject.rank, ranks).addGrid(gameModel.alsObject.maxIter, maxIters).addGrid(gameModel.alsObject.regParam, regParams).build()
8
9 tvs = TrainValidationSplit(estimator=gameModel.alsObject, estimatorParamMaps=parameters, evaluator=gameModel.evaluator, trainRatio=0.9)
10
11 gridSearchModel = tvs.fit(training_set)

▶ (8) Spark Jobs
▶ (28) MLflow runs
Logged 28 runs to an experiment in MLflow. Learn more

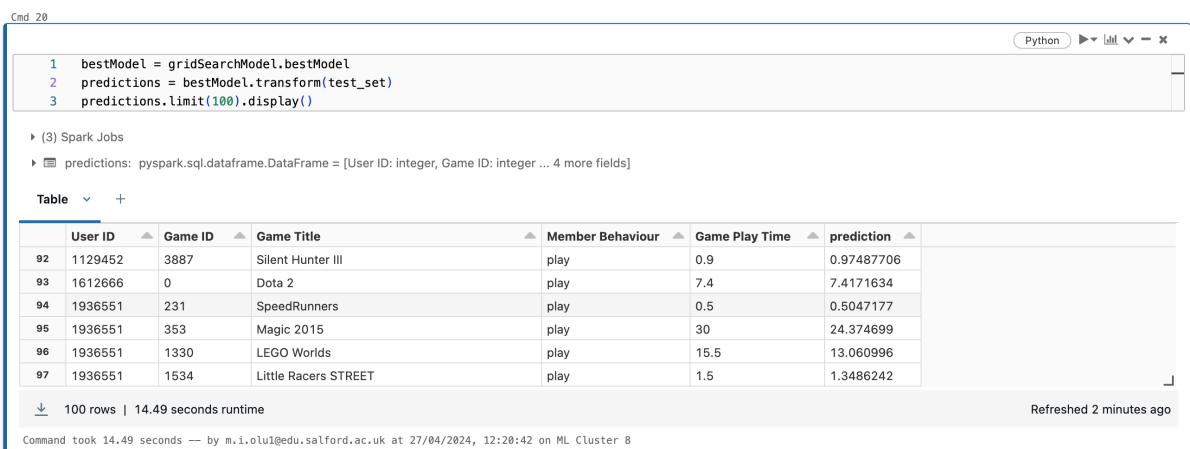
2024/04/27 10:04:52 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID '5609eb6e6fc449448ba45178049969db', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.ml workflow
2024/04/27 10:39:59 WARNING mlflow.pyspark.ml: Model TrainValidationSplitModel_97e0e3b18eb0 will not be autologged because it is not allowlisted or or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistfile Spark conf (see mlflow.pyspark.ml.autolog docs for more info).

Command took 35.15 minutes -- by m.i.olui@edu.salford.ac.uk at 27/04/2024, 11:04:50 on ML Cluster 8
```

Figure 2.3.9

I defined each of the parameters as a list of values: 15, 20 and 30 for the rank, 10, 15 and 20 for the maxIters and 0.005, 0.05 and 0.5 for the regParams.

Figure 2.3.10 shows the predictions made from the best model after running multiple times with the ParamGridBuilder.



```
Cmd 20
Python ▶ v - x

1 bestModel = gridSearchModel.bestModel
2 predictions = bestModel.transform(test_set)
3 predictions.limit(100).display()

▶ (3) Spark Jobs
▶ predictions: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: integer ... 4 more fields]

Table ▾ +
User ID Game ID Game Title Member Behaviour Game Play Time prediction
92 1129452 3887 Silent Hunter III play 0.9 0.97487706
93 1612666 0 Dota 2 play 7.4 7.4171634
94 1936551 231 SpeedRunners play 0.5 0.5047177
95 1936551 353 Magic 2015 play 30 24.374699
96 1936551 1330 LEGO Worlds play 15.5 13.060996
97 1936551 1534 Little Racers STREET play 1.5 1.3486242

↓ 100 rows | 14.49 seconds runtime
Refreshed 2 minutes ago

Command took 14.49 seconds -- by m.i.olui@edu.salford.ac.uk at 27/04/2024, 12:20:42 on ML Cluster 8
```

Figure 2.3.10

The result shows that the model has improved its predictions, having the predictions closer to the Game Play Time.

We can also use some attributes of the bestModel to extract the hyper parameters which was used to train that model.



```
Cmd 21
Python ▶ v - x

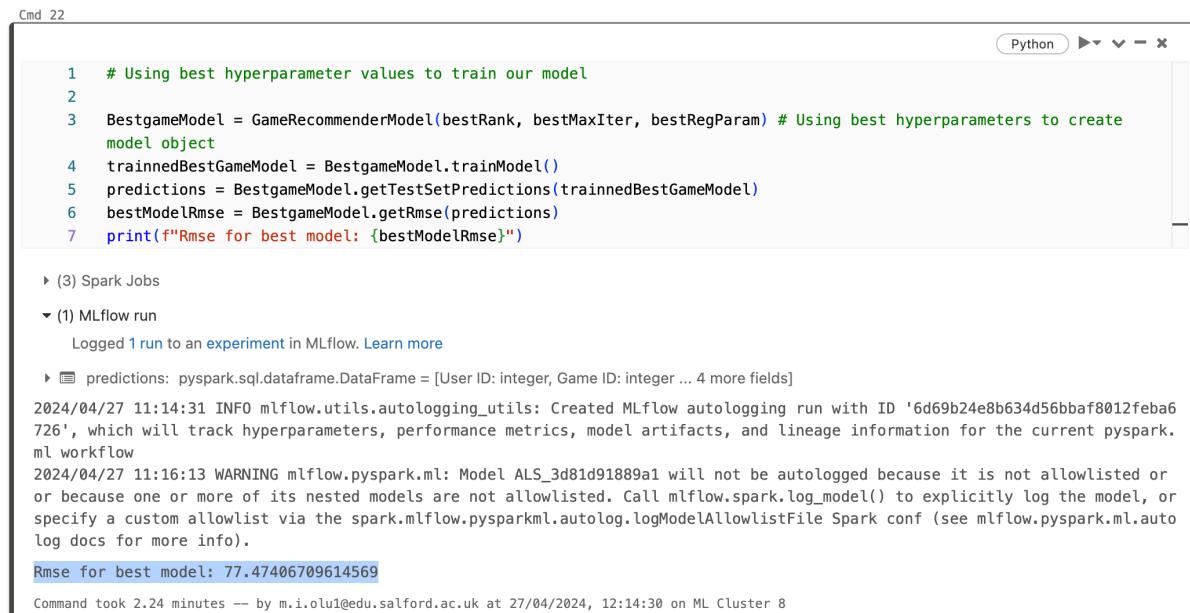
1 bestRank, bestRegParam, bestMaxIter = (bestModel.rank, bestModel._java_obj.parent().getRegParam(), bestModel._java_obj.parent().getMaxIter())
2 print('Hyper parameters for best model: ')
3 print(f'Rank Parameter: {bestRank}')
4 print(f'regParam Parameter: {bestRegParam}')
5 print(f'maxIter: {bestMaxIter}')


Hyper parameters for best model:
Rank Parameter: 30
regParam Parameter: 0.5
maxIter: 20
Command took 0.08 seconds -- by m.i.olul@edu.salford.ac.uk at 27/04/2024, 12:14:19 on ML Cluster 8
```

Figure 2.3.11

The figure above shows the parameters has been extracted and assigned to new variables and printed out to the console. It gave 30 for the Rank value, 0.5 for the regParam and 20 for the maxIter.

We can now use these hyper parameter values to train our own model and check for the rmse as shown below.



```
Cmd 22
Python ▶ v - x

1 # Using best hyperparameter values to train our model
2
3 BestgameModel = GameRecommenderModel(bestRank, bestMaxIter, bestRegParam) # Using best hyperparameters to create
model object
4 trainnedBestGameModel = BestgameModel.trainModel()
5 predictions = BestgameModel.getTestSetPredictions(trainnedBestGameModel)
6 bestModelRmse = BestgameModel.getRmse(predictions)
7 print(f"Rmse for best model: {bestModelRmse}")


▶ (3) Spark Jobs
▼ (1) MLflow run
    Logged 1 run to an experiment in MLflow. Learn more
    ▶ predictions: pyspark.sql.dataframe.DataFrame = [User ID: integer, Game ID: integer ... 4 more fields]
2024/04/27 11:14:31 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID '6d69b24e8b634d56bbaf8012feba6
726', which will track hyperparameters, performance metrics, model artifacts, and lineage information for the current pyspark.
ml workflow
2024/04/27 11:16:13 WARNING mlflow.pyspark.ml: Model ALS_3d81d91889a1 will not be autologged because it is not allowlisted or
or because one or more of its nested models are not allowlisted. Call mlflow.spark.log_model() to explicitly log the model, or
specify a custom allowlist via the spark.mlflow.pysparkml.autolog.logModelAllowlistFile Spark conf (see mlflow.pyspark.ml.auto
log docs for more info).
Rmse for best model: 77.47406709614569
Command took 2.24 minutes -- by m.i.olul@edu.salford.ac.uk at 27/04/2024, 12:14:30 on ML Cluster 8
```

Figure 2.3.12

Getting the rmse of the Best model shows the rmse as 77.474, which is more than 3 times better than the first model which was ran with random hyper parameter values.

The multiple runs made by the ParamGridBuilder were logged and we can view this from the experiment UI. We can also compare different hyper parameters with their corresponding rmse values.

The screenshot shows the Databricks interface for the 'TASK 2 MACHINE LEARNING' experiment. The table view lists 10 different model runs. Each row includes the run name, creation time, duration, source, and various evaluation metrics such as rmse, rmse_test_set, rmse_unknown, maxIter, rank, and regParam.

| Run Name | Created | Duration | Source | Metrics | | Parameters | | | |
|--------------------|----------------|----------|------------|--------------|---------------|--------------|---------|------|----------|
| | | | | rmse | rmse_test_set | rmse_unknown | maxIter | rank | regParam |
| languid-lamb-627 | 14 minutes ago | 1.7min | TASK 2 ... | - | 77.474067... | 20 | 30 | 0.5 | |
| delicate-hare-405 | 2 hours ago | 59.1s | TASK 2 ... | - | 272.93074... | 15 | 15 | 0.2 | |
| colorful-foal-879 | 2 hours ago | 1.3min | TASK 2 ... | - | 552.53781... | 15 | 10 | 0.05 | |
| able-wasp-697 | 18 hours ago | 2.0min | TASK 2 ... | - | 256.147190... | 20 | 30 | 0.05 | |
| exultant-perch-986 | 18 hours ago | 1.6min | TASK 2 ... | - | 210.718238... | 20 | 15 | 0.5 | |
| burly-jay-925 | 19 hours ago | 1.9min | TASK 2 ... | - | 225.78229... | 20 | 30 | 0.05 | |
| gaudy-chimp-298 | 19 hours ago | 1.3min | TASK 2 ... | - | - | 20 | 30 | 0.05 | |
| wistful-yak-278 | 20 hours ago | 2.8min | TASK 2 ... | 280.85189... | - | 20 | 30 | 0.05 | |
| sassy-perch-412 | 1 day ago | 52.4s | TASK 2 ... | 259.86246... | - | 10 | 20 | 0.2 | |

Figure 2.3.13

Each row in the table corresponds to each model ran. It shows that the gridSearchModel ran multiple times using different maxIter, rank and regParam values. The lowest rmse appears to be the model with 77.47067, with its corresponding hyper parameters as shown, which are same values which we extracted from the bestModel in the notebook.

RECOMMENDING GAMES FOR USERS

The model can be used to create a recommendation system using the recommendForAllUsers method on our best model. The parameter 10 is passed to extract the top 10 game recommendations based on predicted implicit behaviour and game play time for each user in the dataset, labelled as 'rating' in each dictionary.

The screenshot shows the Databricks notebook environment. The code cell contains two lines of Python code: `userRecommendations = trainedBestGameModel.recommendForAllUsers(10)` and `userRecommendations.display()`. The output section shows the resulting DataFrame, which contains User ID and a list of recommendations (Game IDs and ratings). The DataFrame has 9,966 rows and was refreshed 11 hours ago.

| User ID | recommendations |
|---------|--|
| 1 | [{"Game ID": 3102, "rating": 986.06146}, {"Game ID": 2803, "rating": 916.41956}, {"Game ID": 4124, "rating": 883.9304}, {"Game ID": 2294, "rating": 780.27057}, {"Game ID": 2341, "rating": 682.7263}, {"Game ID": 3176, "rating": 681.3969}, {"Game ID": 962, "rating": 651.90314}, {"Game ID": 2202, "rating": 612.48083}, {"Game ID": 3436, "rating": 549.2946}, {"Game ID": 232, "rating": 450.056}] |
| 2 | [{"Game ID": 3102, "rating": 0.17412502}, {"Game ID": 2202, "rating": 0.1297716}, {"Game ID": 2803, "rating": 0.12966798}, {"Game ID": 2341, "rating": 0.11970982}, {"Game ID": 4124, "rating": 0.119170524}, {"Game ID": 3436, "rating": 0.11292331}, {"Game ID": 3176, "rating": 0.107010305}, {"Game ID": 3689, "rating": 0.09996688}, {"Game ID": 962, "rating": 0.08503229}, {"Game ID": 1896, "rating": 0.06499595}] |
| 3 | [{"Game ID": 1097, "rating": 678.03235}, {"Game ID": 225, "rating": 635.47205}, {"Game ID": 2202, "rating": 554.82184}, {"Game ID": 3029, "rating": 459.97812}, {"Game ID": 2294, "rating": 388.75948}, {"Game ID": 3441, "rating": 388.64038}, {"Game ID": 3772, "rating": 314.63315}, {"Game ID": 2508, "rating": 295.50174}, {"Game ID": 4124, "rating": 270.84573}, {"Game ID": 962, "rating": 264.78647}] |
| 4 | [{"Game ID": 1240, "rating": 107.40031}, {"Game ID": 2105, "rating": 98.31627}, {"Game ID": 1223, "rating": 87.27834}, {"Game ID": 44, "rating": 82.84897}, {"Game ID": 2428, "rating": 78.01639}, {"Game ID": 1585, "rating": 68.52529}, {"Game ID": 1779, "rating": 64.417274}, {"Game ID": 2587, "rating": 61.669327}, {"Game ID": 1292, "rating": 57.578045}, {"Game ID": 1389, "rating": 55.209137}] |
| 975449 | [{"Game ID": 2294, "rating": 1367.4384}, {"Game ID": 1223, "rating": 1131.8398}, {"Game ID": 3772, "rating": 996.6625} ...] |

Figure 2.3.13

I wrote a query to extract the recommendations for a particular user using their User ID and also matched the Game ID to the game Titles by joining the extracted user row from the recommendation table to the game_id_dataframe which I created earlier on.

```

Cmd 24
Python ▶ v ↻ - x
1 #Final top 10 Recommendations for 1 user
2 singleUserRecommendation = userRecommendations.filter(userRecommendations["User ID"] == 26333936).select("recommendations").withColumn(
3   "recommendations", explode("recommendations")).select("recommendations.Game ID", "recommendations.rating").join(game_id_dataframe, ["Game ID"])
4 singleUserRecommendation.display()

```

(4) Spark Jobs

singleUserRecommendation: pyspark.sql.dataframe.DataFrame = [Game ID: integer, rating: float ... 1 more field]

Table +

| | Game ID | rating | Game |
|---|---------|-----------|-----------------------------|
| 1 | 2202 | 1156.7416 | Football Manager 2010 |
| 2 | 2294 | 728.744 | X-Plane 10 Global - 64 Bit |
| 3 | 466 | 630.20654 | Crusaders of the Lost Idols |
| 4 | 232 | 562.9908 | Football Manager 2011 |
| 5 | 3772 | 530.57465 | Football Manager 2012 |
| 6 | 1506 | 473.58014 | Football Manager 2015 |

↓ 10 rows | 54.41 seconds runtime Refreshed 22 hours ago

Command took 54.41 seconds -- by m.i.olul@edu.salford.ac.uk at 27/04/2024, 21:28:18 on ML Cluster 9

Figure 2.3.14

From the results in figure 2.3.14 above, we can see that same types of games were recommended for user `26333936`. It can be deduced that this user top choice of games is Football Manager for various years.

5. DISCUSSION OF RESULTS.

The model's initial training yielded an RMSE of ~234.126966, with hyper parameters of 15 for rank, 15 for maxIter, and 0.2 for regParam. After multiple experiments using ParamGridBuilder, the best RMSE was ~77.5, with parameters of 30, 20, and 0.5 for rank, maxIter, and regParam. This best model was used to create recommendations for all users, and further filtered to check the recommendations for one particular user and revealed this user's top preference to be the `Football Manager` game.

Bibliography

Victor. (2017, August 23). *ALS Implicit Collaborative Filtering*. Retrieved from medium: <https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe>

