End Lab 00:44:11

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Open Google Console

Username
student-02-9f95b284345(

Password
Q7DU9b7DXseg

GCP Project ID
qwiklabs-gcp-01-406a0e(

Movie Recommendations in BigQuery ML

1 hour Free ★★★★½

Overview

BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

BigQuery Machine Learning (BigQuery ML) is a feature in BigQuery where data analysts can create, train, evaluate, and predict with machine learning models with minimal coding.

Collaborative filtering provides a way to generate product recommendations for users, or user targeting for products. The starting point is a table with three columns: a user id, an item id, and the rating that the user gave the product. This table can be sparse -- users don't have to rate all products. Then, based on just the ratings, the technique finds similar users and similar products and determines the rating that a user would give an unseen product. Then, you can recommend the products with the highest predicted ratings to users, or target products at users with the highest predicted ratings.

To illustrate recommender systems in action, you will use the MovieLens dataset. This is a dataset of movie reviews released by GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, through funding by the US National Science Foundation.

Objectives

In this lab, you learn to perform the following tasks:

- Create a BigQuery dataset to store and load MovieLens data
- Explore the MovieLens dataset
- Use a trained model to make recommendations in BigQuery
- Make product predictions for both single users and batch users

Set up your environment

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.

2. Note the lab's access time (for example, 02:00:00), and make sure you can finish within that time. There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **Start lab**.

4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts. If you use other credentials, you'll receive errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

> Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Open BigQuery Console

1. In the Google Cloud Console, on the **Navigation menu** , click **BigQuery**. The **Welcome to BigQuery in the Cloud Console** dialog opens. This dialog provides a link to the quickstart guide and lists UI updates.

2. Click **Done** to close the dialog.

Task 1: Get MovieLens data

In this task you will use the command line to create a BigQuery dataset to store the MovieLens data. The MovieLens data will then be loaded from a Cloud Storage bucket into the dataset.

Start the Cloud Shell Editor

To create a BigQuery dataset and load the MovieLens data the Cloud Shell is used.

1. In the GCP Console, click **Activate Cloud Shell** ().

2. If prompted, click **Continue**.

Create and Load BigQuery Dataset

1. Run the following command to create a BigQuery dataset named `movies`:

```
bq --location=EU mk --dataset movies
```

2. Run the following commands separately in the Cloud Shell:

```
bq load --source_format=CSV \
--location=EU \
--autodetect movies.movielens_ratings \
gs://dataeng-movielens/ratings.csv
```

```
bq load --source_format=CSV \
--location=EU   \
--autodetect movies.movielens_movies_raw \
gs://dataeng-movielens/movies.csv
```

Click *Check my progress* to verify the objective.

Get MovieLens Data

Check my progress

Assessment Completed!

Task 2: Explore the data

In this task you will explore and verify the MovieLens dataset using Query editor.

1. In BigQuery's **Query editor** execute the following query:

```
SELECT
  COUNT(DISTINCT userId) numUsers,
  COUNT(DISTINCT movieId) numMovies,
  COUNT(*) totalRatings
FROM
  movies.movielens_ratings
```

You should confirm that the dataset consists of over 138 thousand users, nearly 27 thousand movies, and a little more than 20 million ratings.

2. Examine the first few movies using the query:

```
SELECT
  *
FROM
  movies.movielens_movies_raw
WHERE
  movieId < 5
```

Row	movieId	title	genres
1	3	Grumpier Old Men (1995)	Comedy|Romance
2	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
3	2	Jumanji (1995)	Adventure|Children|Fantasy
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

3. You can see that the genres column is a formatted string. Parse the genres into an array and rewrite the results into a table named `movielens_movies`.

```
CREATE OR REPLACE TABLE
  movies.movielens_movies AS
SELECT
  * REPLACE(SPLIT(genres, "|") AS genres)
FROM
  movies.movielens_movies_raw
```

Feel free to perform additional queries until you are comfortable with the dataset.

Click *Check my progress* to verify the objective.

Explore the Data

Check my progress

Assessment Completed!

Task 3: Evaluate a trained model created using collaborative filtering

In this task you will view the metrics for a trained model which was generated using matrix factorization.

Matrix factorization is a collaborative filtering technique that relies on two vectors called the user factors and the item factors. The user factors is a low-dimensional representation of a `user_id` and the item factors similarly represents an `item_id`.

To perform a matrix factorization of our data, you use the typical BigQuery ML syntax except that the `model_type` is `matrix_factorization` and you have to identify which columns play what roles in the collaborative filtering setup.

In order to apply matrix factorization to the movie ratings data, the BigQuery ML query needs to be executed to create the model. However, creation of this model type can take up to 40 minutes and requires a Google Cloud project with reservation-oriented resources -- which is unlike those offered by the Qwiklabs environment.

A model has been created in the Cloud Training project's **cloud-training-prod-bucket** BigQuery dataset for use in the rest of the lab.

NOTE: The query below is for reference only. Please **DO NOT EXECUTE** this query in your project.

```
CREATE OR REPLACE MODEL movies.movie_recommender

OPTIONS (model_type='matrix_factorization', user_col='userId', item_col='movieId', rating_col='rating', l2_reg=0.2, num_factors=16) AS

SELECT userId, movieId, rating

FROM movies.movielens_ratings
```

Note, the `num_factors` and `l2_reg` options have been selected after much experimentation to speed up training of the model.

1. To view metrics for the trained model, run the following query:

```
SELECT * FROM ML.EVALUATE(MODEL `cloud-training-prod-bucket.movies.movie_recommender`)
```

Task 4: Make Recommendations

In this task you will use the trained model to provide recommendations.

1. Let's find the best comedy movies to recommend to the user whose `userId` is 903. Enter the query below:

```
SELECT
  *
FROM
  ML.PREDICT(MODEL `cloud-training-prod-
bucket.movies.movie_recommender`,
    (
    SELECT
      movieId,
      title,
      903 AS userId
    FROM
      `movies.movielens_movies`,
      UNNEST(genres) g
    WHERE
      g = 'Comedy' ))
ORDER BY
  predicted_rating DESC
LIMIT
  5
```

Row	predicted_rating	movieId	title	userId
1	6.305484877897655	82978	Neighbors (1920)	903
2	5.659955887029915	26136	Hallelujah Trail, The (1965)	903
3	5.608127858593018	69075	Trojan War (1997)	903
4	5.423441457257417	3337	I'll Never Forget What's'isname (1967)	903
5	5.301408212165985	6167	Stand-In (1937)	903

2. This result includes movies the user has already seen and rated in the past. Let's remove them:

```
SELECT
  *
FROM
  ML.PREDICT(MODEL `cloud-training-prod-
bucket.movies.movie_recommender`,
    (
    WITH
      seen AS (
      SELECT
        ARRAY_AGG(movieId) AS movies
      FROM
        movies.movielens_ratings
      WHERE
        userId = 903 )
    SELECT
      movieId,
      title,
      903 AS userId
    FROM
      movies.movielens_movies,
      UNNEST(genres) g,
      seen
    WHERE
      g = 'Comedy'
      AND movieId NOT IN UNNEST(seen.movies) ))
ORDER BY
  predicted_rating DESC
LIMIT
  5
```

For this user, this happens to yield the same set of movies -- the top predicted ratings didn't include any of the movies the user has already seen.

Click *Check my progress* to verify the objective.

Making Recommendations

Check my progress

Task 5:Apply customer targeting

In this task you will look at how to identify the top-rated movies for a specific user. Sometimes, you have a product and have to find the customers who are likely to appreciate it.

1. You wish to get more reviews for `movieId=96481` which has only one rating and you wish to send coupons to the 100 users who are likely to rate it the highest. Identify those users using:

```
SELECT
  *
FROM
  ML.PREDICT(MODEL `cloud-training-prod-
bucket.movies.movie_recommender`,
    (
    WITH
      allUsers AS (
      SELECT
        DISTINCT userId
      FROM
        movies.movielens_ratings )
    SELECT
      96481 AS movieId,
      (
      SELECT
        title
      FROM
        movies.movielens_movies
      WHERE
        movieId=96481) title,
      userId
    FROM
      allUsers ))
ORDER BY
  predicted_rating DESC
LIMIT
  100
```

The result gives us 100 users to target, the top 5 of whom are:

Row	predicted_rating	movieId	title	userId
1	6.000193988615432	96481	American Mullet (2001)	104104
2	5.92811262777923	96481	American Mullet (2001)	57703
3	5.902559169949699	96481	American Mullet (2001)	22625
4	5.882101585633906	96481	American Mullet (2001)	118093
5	5.740621111206273	96481	American Mullet (2001)	37594

Click *Check my progress* to verify the objective.

Customer Targeting

Check my progress

Assessment Completed!

Task 6: Perform Batch predictions for all users and movies

In this task you will perform a query to obtain batch predictions for users and movies.

What if you wish to carry out predictions for every user and movie combination? Instead of having to pull distinct users and movies as in the previous query, a convenience function is provided to carry out batch predictions for all `movieId` and `userId` encountered during training.

1. Enter the following query to obtain batch predictions:

```
SELECT
  *
FROM
  ML.RECOMMEND(MODEL `cloud-training-prod-bucket.movies.movie_recommender`)
LIMIT
  100000
```

Without the LIMIT command the results would be too large to return given the default settings. But the output provides you a sense of the type of predictions that can be made with this model.

As seen in a section above, it is possible to filter out movies the user has already seen and rated in the past. The reason already seen movies aren't filtered out by default is that there are situations (think of restaurant recommendations, for example) where it is perfectly expected that you would need to recommend restaurants the user has liked in the past.

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.