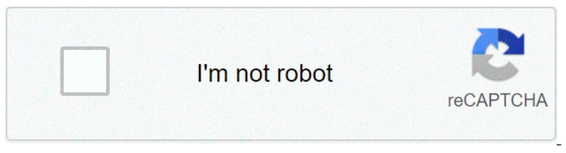

Continue

Clustering in ml

BigQuery ML supports unsupervised learning. You can apply the k-means algorithm to group your data into clusters. Unlike supervised machine learning, which is about predictive analytics, unsupervised learning is about descriptive analytics. It's about understanding your data so that you can make data-driven decisions. In this tutorial, you will use a k-means model in BigQuery ML to build clusters of data in the London Bicycle Hires public dataset. The London Bicycle Hires data contains the number of hires of London's Santander Cycle Hire Scheme from 2011 to present. The data includes start and stop timestamps, station names and ride duration. The queries in this tutorial use Geography Functions available in BigQuery GIS. For more information on BigQuery GIS, see Introduction to BigQuery GIS. Your data may contain natural groupings or clusters of data. You may want to identify these groupings descriptively in order to make data-driven decisions. For example, a retailer may want to identify natural groupings of customers who have similar purchasing habits or locations. This process is known as customer segmentation. Data you use to perform customer segmentation might include the store they visited, what items they bought, how much they paid, and so on. You would create a model to try to understand what these groups of customer personas are like so that you can design items that appeal to group members. You could also find product groups among the items purchased. In this case, you would cluster items based on who purchased them, when they were purchased, where they were purchased, and so on. You would create a model to determine the characteristics of a product group so that you can make informed decisions such as how to improve cross-selling. In this tutorial, you use BigQuery ML to create a k-means model that clusters London Bicycle Hires data based on bike station attributes. Creating your k-means model consists of the following steps. Step one: Create a dataset to store your model. The first step is to create a dataset that stores your model. Note: Because the London Bicycle Hires dataset is stored in the EU multi-region location, your dataset must also reside in the EU. Step two: Examine your training data. The next step is to examine the data you use to train your clustering model by running a query against the london_bicycles table. Because k-means is an unsupervised learning technique, model training does not require labels nor does it require you to split the data into training data and evaluation data. Step three: Create a k-means model. Step three is to create your k-means model. When you create the model, the clustering field is station_name, and you cluster the data based on station attribute, for example the distance of the station from the city center. Step four: Use the ML.PREDICT function to predict a station's cluster. Next, you use the ML.PREDICT function to predict the cluster for a given set of stations. You predict clusters for all station names that contain the string Kennington. Step five: Use your model to make data-driven decisions. The final step is to use the model to make data-driven decisions. For example, based on the model results, you can determine which stations would benefit from extra capacity. Step one: Create your dataset The first step is to create a BigQuery dataset to store your model. To create your dataset: In the Cloud Console, go to the BigQuery page. Go to the BigQuery page In the navigation panel, in the Resources section, click your project name. On the right side, in the details panel, click Create dataset. On the Create dataset page: For Dataset ID, enter bqml_tutorial. For Data location, choose European Union (EU). The London Bicycle Hires public dataset is stored in the EU multi-region location. Your dataset should be in the same location. Leave all of the other default settings in place and click Create dataset. Step two: Examine your training data Next, you examine the data used to train your k-means model. In this tutorial, you cluster bike stations based on the following attributes: Duration of rentals Number of trips per day Distance from city center The following query compiles your training data. This query is included in your CREATE MODEL statement later in this tutorial. WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name) SELECT * FROM stationstats ORDER BY distance_from_city_center ASC Query details This query extracts data on cycle hires, including start_station_name, and duration and joins it against station information, including distance-from-city-center. Then, it computes attributes of the station in stationstats, including the average duration of rides and the number of trips, and passes through the station attribute distance_from_city_center. This query uses the WITH clause to define subqueries. The query also uses the ST_DISTANCE and ST_GEOGPOINT BigQuery GIS functions. For more information on these functions, see Geography Functions. For more information on BigQuery GIS, see Introduction to BigQuery GIS. Run the query To run the query that compiles the training data for your model: In the Cloud Console, click the Compose new query button. Enter the following standard SQL query in the Query editor text area. WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name) SELECT * FROM stationstats ORDER BY distance_from_city_center ASC Click Run. When the query is complete, click the Results tab below the query text area. The results table shows the columns you queried that are used to train your model: station_name, duration, num_trips, distance_from_city_center. The results should look like the following. Step three: Create a k-means model Now that you have examined your training data, the next step is to create a k-means model using the data. You can create and train a k-means model using the CREATE MODEL statement with the option model_type=kmeans. The following query adds a CREATE MODEL statement to the previous query and removes the id fields in the data. CREATE OR REPLACE MODEL bqml_tutorial.london_station_clusters OPTIONS(model_type='kmeans', num_clusters=4) AS WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, isweekday, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name, isweekday) SELECT * EXCEPT(station_name, isweekday) FROM stationstats Query details The CREATE MODEL statement specifies the desired number of clusters — four. In the SELECT statement, the EXCEPT clause excludes the station_name column because station_name is not a feature. The query creates a unique row per station_name, and only the features are mentioned in the SELECT statement. If you omit the num_clusters option, BigQuery ML will choose a reasonable default based on the total number of rows in the training data. You could also perform hyperparameter tuning to find a good number. To determine an optimal number of clusters, you would run the CREATE MODEL query for different values of num_clusters, find the error measure, and pick the point at which the error measure is at its minimum value. You can obtain the error measure by selecting your model and clicking the Evaluation tab. This tab shows the Davies–Bouldin index. Run the CREATE MODEL query To run the query that creates your k-means model: In the Cloud Console, click the Compose new query button. Enter the following standard SQL query in the Query editor text area. CREATE OR REPLACE MODEL bqml_tutorial.london_station_clusters OPTIONS(model_type='kmeans', num_clusters=4) AS WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, isweekday, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name, isweekday) SELECT * EXCEPT(station_name, isweekday) FROM stationstats Click Run. In the navigation panel, in the Resources section, expand your project name, click bqml_tutorial, and then click london_station_clusters. Click the Schema tab. The model schema lists the four station attributes that BigQuery ML used to perform clustering. The schema should look like the following. Click the Evaluation tab. This tab displays visualizations of the clusters identified by the k-means model. Under Numerical features, bar graphs display up to 10 of the most important numerical feature values for each centroid. You can select which features to visualize from the drop-down menu. Step four: Use the ML.PREDICT function to predict a station's cluster To identify the cluster to which a particular station belongs, use the ML.PREDICT function. The following query predicts the cluster of every station that has the string "Kennington" in its name. WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name) SELECT * EXCEPT(nearest_centroids_distance) FROM ML.PREDICT(MODEL bqml_tutorial.london_station_clusters, (SELECT * FROM stationstats WHERE REGEXP_CONTAINS(station_name, 'Kennington'))) Query details This query uses the REGEXP_CONTAINS function to find all entries in the station_name column that contain the string "Kennington". The ML.PREDICT function uses those values to predict which clusters would contain those stations. Run the ML.PREDICT query To run the ML.PREDICT query: In the Cloud Console, click the Compose new query button. Enter the following standard SQL query in the Query editor text area. WITH hs AS (SELECT h.start_station_name AS station_name, IF (EXTRACT(DAYOFWEEK FROM h.start_date) = 1 OR EXTRACT(DAYOFWEEK FROM h.start_date) = 7, "weekend", "weekday") AS isweekday, h.duration, ST_DISTANCE(ST_GEOGPOINT(s.longitude, s.latitude), ST_GEOGPOINT(-0.1, 51.5))/1000 AS distance_from_city_center FROM `bigquery-public-data.london_bicycles.cycle_hire` AS h JOIN `bigquery-public-data.london_bicycles.cycle_stations` AS s ON h.start_station_id = s.id WHERE h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP) AND CAST('2016-01-01 00:00:00' AS TIMESTAMP)), stationstats AS (SELECT station_name, AVG(duration) AS duration, COUNT(duration) AS num_trips, MAX(distance_from_city_center) AS distance_from_city_center FROM hs GROUP BY station_name) SELECT * EXCEPT(nearest_centroids_distance) FROM ML.PREDICT(MODEL bqml_tutorial.london_station_clusters, (SELECT * FROM stationstats WHERE REGEXP_CONTAINS(station_name, 'Kennington'))) Click Run. When the query is complete, click the Results tab below the query text area. The results should look like the following. Step five: Use your model to make data-driven decisions Evaluation results can help you to interpret the different clusters. In this example, Cluster#3 shows a busy city station that is close to the city center. Cluster#2 shows the second city station which is less busy. Cluster#1 shows a less busy suburban substation, with longer duration rentals. Cluster#4 shows another suburban station with trips that are shorter. Based on these results, you can use the data to inform your decisions. For example: Assume that you need to experiment with a new type of lock. Which cluster of stations should you choose as a subject for this experiment? The stations in Cluster#1, Cluster#2 or Cluster#4 seem like logical choices because they are not the busiest stations. Assume that you want to stock some stations with racing bikes. Which stations should you choose? Cluster#1 is the group of stations that are far from the city center, and they have the longest trips. These are likely candidates for racing bikes. To avoid incurring charges to your Google Cloud account for the resources used in this tutorial, either delete the project that contains the resources, or keep the project and delete the individual resources. You can delete the project you created. Or you can keep the project and delete the dataset. Deleting your dataset Deleting your project removes all datasets and all tables in the project. If you prefer to reuse the project, you can delete the dataset you created in this tutorial: If necessary, open the BigQuery page in the Cloud Console. Go to the BigQuery page In the navigation, click the bqml_tutorial dataset you created. Click Delete dataset on the right side of the window. This action deletes the dataset and the model. In the Delete dataset dialog box, confirm the delete command by typing the name of your dataset (bqml_tutorial) and then click Delete. Deleting your project To delete the project: Caution: Deleting a project has the following effects: Everything in the project is deleted. If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project. Custom project IDs are lost. When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an appspot.com URL, delete selected resources inside the project instead of deleting the whole project. If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits. In the Cloud Console, go to the Manage resources page. Go to Manage resources In the project list, select the project that you want to delete, and then click Delete. In the dialog, type the project ID, and then click Shut down to delete the project.