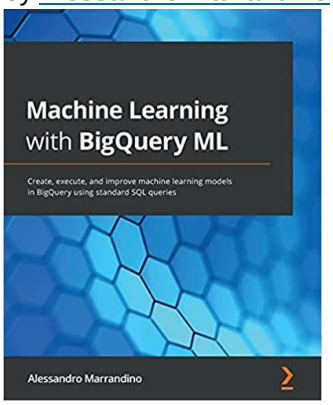
Book review Machine Learning with BigQuery ML: Create, execute, and improve machine learning models in BigQuery using standard SQL queries

by Alessandro Marrandino (Author)



Summary: From Zero to Hero in ML the easy way with BigQuery ML and this book! This is a very easy to read intro. level book to get you started using BQML the easy way. It shows you step by step guide to prep/clean/filter the data, train the model with many different supported algorithms/options, evaluate the model (e.g. no overfitting) and predict with the trained model. It is an excellent book if you like to use standard SQL to do Machine Learning without the need to learn Python, Tensorflow, etc… This will help liberate many business intelligence (BI) specialists upgraded into the next level, i.e. able to predict and classify for good business use cases with BQML machine learning models. A lot of easy to use code to get started. A lot of references to learn more in BQML and other use cases.

Section 1 is more introduction to GCP and BQ environment. You can skip if you know them already.
Section 2 is more on the linear regression, binary and multiclass logistic classifications.
Section 3 is more on Adv. Models, e.g. K-means, matrix factorization, XGBoost, Deep Neural Networks.
Section 4 is more on how to use GCP AI notebooks to run BQML, run TF models with BQML (e.g. import/export BQ <-> TF saved model, BQML Tips and Best Practices, e.g. data quality, prep data, hyperparameters tuning, how to do near real time recommendations, etc.

Suggestions:
1/ there should be a typo in p. 324, "In this case, the MOD function returns a value from 0 to 10", Mod (X, 10) should return from 0-9 (not 10) because it returns the REMINDER of the Mod function after division. See more at https://cloud.google.com/bigquery/docs/reference/standard-sql/mathematical_functions#mod

2/ 2/ Can introduce GCP Data Prep to help prepare and visualize the data in addition to Data Studio and its min/max/Not Null examples as shown in the book, see more at https://cloud.google.com/dataprep

1/ buy it at Amazon: https://www.amazon.com/Machine-Learning-BigQuery-ML-learning/dp/1800560303

2/ sample code in GitHub - https://github.com/PacktPublishing/Machine-Learning-with-BigQuery-ML

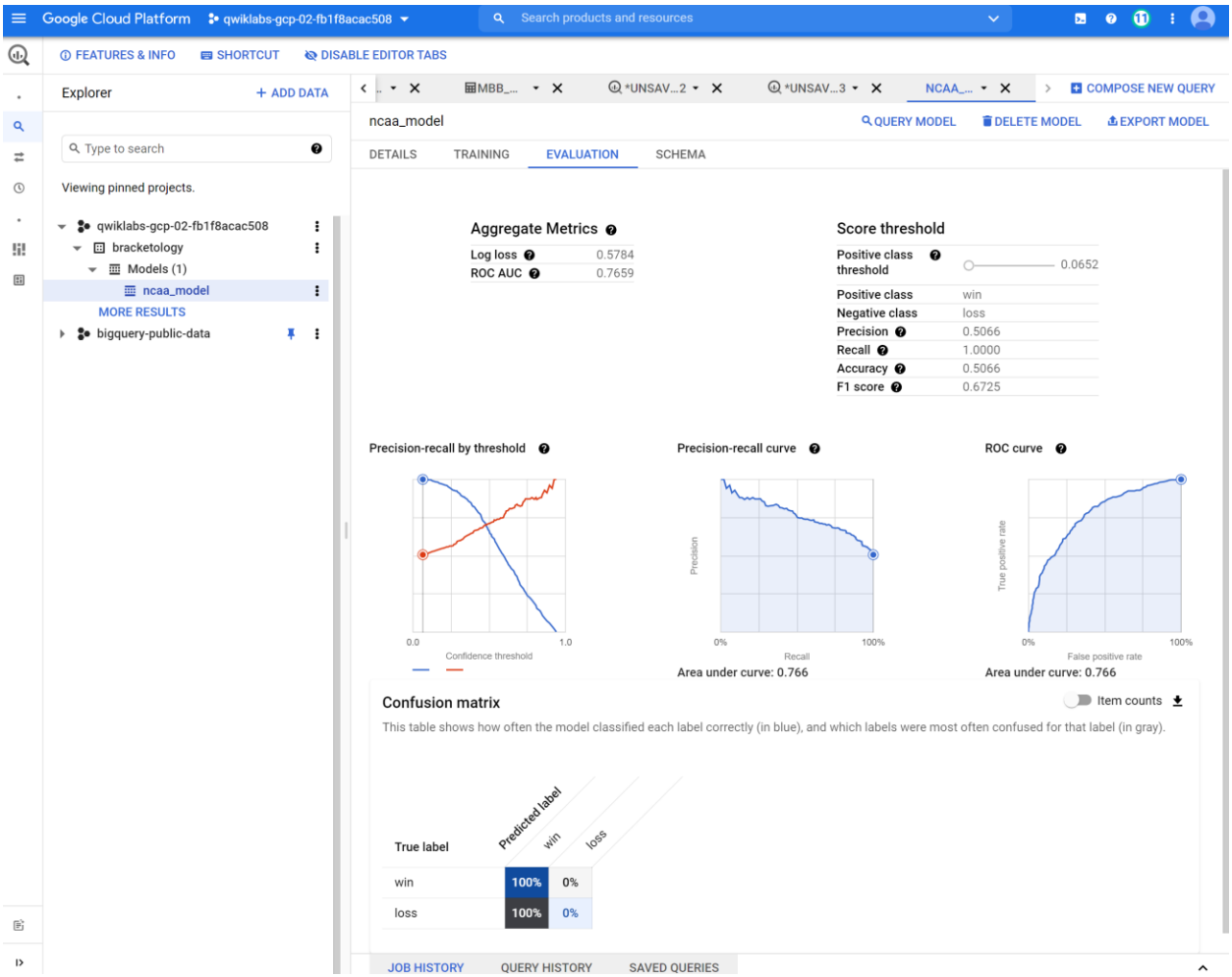
3/ Demo video - https://www.youtube.com/playlist?list=PLeLcvrwLe187Kk5QIqt7Kb8qdQSQd9AcY

4/ lab - https://google.qwiklabs.com/focuses/4337?parent=catalog

Quest - https://google.qwiklabs.com/quests/71 BigQuery for Machine Learning Quest

5/ BQML available models - https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create#model_option_list

```
MODEL_TYPE = { 'LINEAR_REG' | 'LOGISTIC_REG' | 'KMEANS' | 'PCA' |
'MATRIX_FACTORIZATION' | 'AUTOENCODER' | 'TENSORFLOW'
|  'AUTOML_REGRESSOR' |
'AUTOML_CLASSIFIER' | 'BOOSTED_TREE_CLASSIFIER' | 'BOOSTED_TREE_REGRESSOR'
|
'DNN_CLASSIFIER' | 'DNN_REGRESSOR' | 'DNN_LINEAR_COMBINED_CLASSIFIER' |
'DNN_LINEAR_COMBINED_REGRESSOR' | 'ARIMA_PLUS' }
```



5. To keep our dataset consistent, it is better to also drop the view with the `DROP VIEW` statement:

```
DROP VIEW
  `bigqueryml-packt.03_bigquery_syntax.first_view`;
```

   Dropping a view is similar to dropping a table, but this operation affects only the metadata because the view doesn't actually store any records.

In this section of the chapter, we've discovered the main operations that we can do with BigQuery SQL; now it's time to dive into BigQuery ML and its syntax.

Diving into BigQuery ML

Developing an ML model in BigQuery involves three main steps:

1. **Model creation**, where you are required to choose the **features** and **labels** of your ML model and the options to tune the ML model. At this stage, BigQuery runs the training of the ML model on the training set that you've chosen.
2. **Model evaluation** allows you to test the model trained in the previous step on a different set of records to prevent any **overfitting**.
3. **Model use**: when the ML model is ready, we can apply it to a new dataset in order to make predictions or classifications of the labels according to the available features.

In the next paragraphs, we'll take a look at the syntax of these three stages and how these statements are built using stubs of code.

Creating the ML model (training)

When you've identified the ML use case and also the set of records to train your model, you can start training the model with the following query:

```
CREATE MODEL`<project_name>.<dataset_name>.<ml_model_name>`
TRANSFORM (<list_of_features_transformed>
OPTIONS(<list_of_options>)
AS <select_statement>;
```

Syntax urls:

Summary

In this chapter, we've learned the main aspects of the BigQuery syntax. After the creation of a dataset, we've discovered how to create tables, insert records, and read the rows stored in a table. You've also learned how to update existing records and how to remove rows and delete objects that are no longer useful, such as tables and views.

Completing the overview of the BigQuery SQL syntax, we dived into the main stages of the life cycle of an ML model. The three main phases to realize a use case are the creation, the evaluation, and the use of the ML model. For the training phase, we have found out how to train and create a new model using SQL. After that, we went through all the functions that can be used to monitor the effectiveness of a trained model, evaluating its key performance indicators. Finally, we saw how to use a trained model on a new dataset to infer the results and get predictions, forecasts, or recommendations. At the end of the chapter, we also learned how to delete BigQuery ML models that are no longer useful.

Now that we have a clear understanding of the syntax and all the capabilities that we can use in BigQuery, it's time to apply all these concepts to our first hands-on use case. In the next chapter, we will develop our first BigQuery ML model to predict the estimated duration of a bike trip for an important bike rental service in New York City.

Further resources

- **BigQuery datasets**: `https://cloud.google.com/bigquery/docs/datasets`
- **BigQuery SQL syntax**: `https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax`
- **BigQuery data types**: `https://cloud.google.com/bigquery/docs/reference/standard-sql/data-types`

- **Create model syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create`
- **Evaluate syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-evaluate`
- **Confusion matrix syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-confusion`
- **ROC curve syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-roc`
- **Predict syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-predict`
- **Forecast syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-forecast`
- **Recommend syntax**: `https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-recommend`

Section 2

In this section, regression machine learning models are explained and presented with real hands-on examples using BigQuery ML.

This section comprises the following chapters:

- *Chapter 4, Predicting Numerical Values with Linear Regression*
- *Chapter 5, Predicting Boolean Values Using Binary Logistic*
- *Chapter 6, Classifying Trees with Multiclass Logistic Regression*

```
    tripduration is not NULL
    AND tripduration>0;
```

While the minimum rental time is an expected value of 1 minute, we can see that the maximum value is not compatible with the normal functionality of a bike sharing service. In fact, the maximum duration is more than 300,000 minutes, which is approximately equivalent to a rental period of more than 225 days.

In the following screenshot, you can see the result of the query:

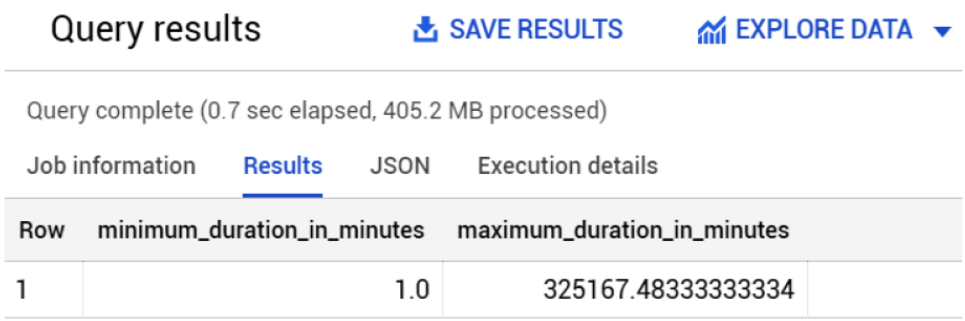


Figure 4.9 – The results of the query, evidencing the presence of outliers in the tripduration column

While preparing our datasets, we'll take all these factors into considerations to avoid any impact on the machine learning model.

3. Then, we can apply a similar check to all the columns that are potential features of our machine learning model, excluding the records that present a non-significant value of `tripduration`:

```
SELECT  COUNT(*)
FROM
  `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE
  (tripduration is not NULL
  AND tripduration>0) AND (
  starttime is NULL
  OR start_station_name is NULL
  OR end_station_name is NULL
  OR start_station_latitude is NULL
  OR start_station_longitude is NULL
  OR end_station_latitude is NULL
  OR end_station_longitude is NULL);
```

Training the linear regression model

Training a BigQuery ML model is not a one-shot operation, but it's a process that can require multiple attempts and recycles to get closer to the final goal of developing an effective asset with good performance, according to the requirements of the business scenario. For our use case, we'll go try to improve the performance of our ML model multiple times. Let's get started:

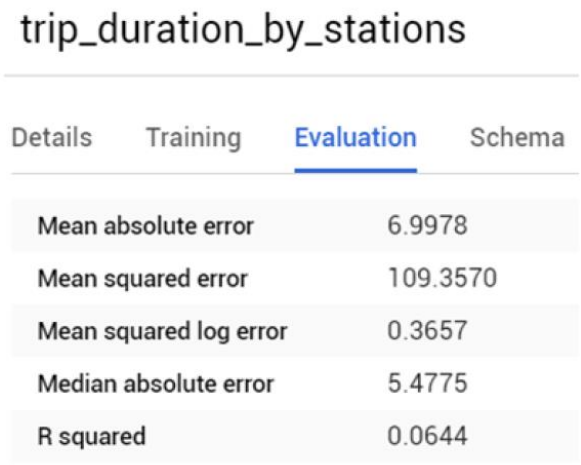
1. First, let's start training a new machine learning model named `trip_duration_by_stations`:

```
CREATE OR REPLACE MODEL `04_nyc_bike_sharing.trip_
duration_by_stations`
OPTIONS
  (model_type='linear_reg') AS
SELECT
  start_station_name,
  end_station_name,
  tripduration as label
FROM
  `04_nyc_bike_sharing.training_table`;
```

After a few seconds, the BigQuery ML model will be created and available in the navigation menu, under the `04_nyc_bike_sharing` dataset. Select the ML model and click on the **Evaluation** tab, where we will find some performance indicators for our brand-new ML model; that is, `trip_duration_by_stations`.

In this case, we'll focus our attention on **Mean absolute error**. This value represents the average distance between the actual value and the predicted value of the label.

As shown in the following screenshot, it's very close to 7 minutes:



Evaluating the linear regression model

For the evaluation stage of our BigQuery ML model, we'll use the `ML.EVALUATE` function and the table that we've expressly created to host the evaluation records. These are completely separate from the rows that are used during the training phase.

Let's execute the following query to evaluate our ML model on the evaluation table:

```
SELECT
  *
FROM
  ML.EVALUATE(MODEL `04_nyc_bike_sharing.trip_duration_by_
stations_and_day`,
    (
    SELECT
          start_station_name,
          end_station_name,

          IF (EXTRACT(DAYOFWEEK FROM starttime)=1 OR
                EXTRACT(DAYOFWEEK FROM starttime)=7,
                true, false) is_weekend,
          tripduration as label
    FROM
          `04_nyc_bike_sharing.evaluation_table`));
```


In the following screenshot, you can see the performance indicators that have been extracted by the evaluation query:

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.1 sec elapsed, cached)

Job information Results JSON Execution details

Row	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
1	7.1240513774200025	119.4608975942807	0.3779480127970511	5.512046726796809	0.09537225481125589	0.0954802008375020

Figure 4.12 – The query results of the EVALUATE function show the ML key performance calculated on the evaluation table

Now that we've trained the model and we're also satisfied with the outcomes of the evaluation stage, let's learn how to apply our machine learning model to other records and get predictions.

Utilizing the linear regression model

To use our BigQuery ML model, we'll use the `ML.PREDICT` function and the table that we've expressly created to host the records that we haven't used yet.

The following query will predict the label using the data in `prediction_table`:

```
SELECT
    tripduration as actual_duration,
    predicted_label as predicted_duration,
    ABS(tripduration - predicted_label) difference_in_min
FROM
  ML.PREDICT(MODEL `04_nyc_bike_sharing.trip_duration_by_
stations_and_day`,
     (
     SELECT
           start_station_name,
           end_station_name,
           IF (EXTRACT(DAYOFWEEK FROM starttime)=1 OR
                 EXTRACT(DAYOFWEEK FROM starttime)=7,
                 true, false) is_weekend,
           tripduration
     FROM
```

5

Predicting Boolean Values Using Binary Logistic Regression

Binary logistic regression is one of the most widely used **Machine Learning** (**ML**) algorithms to predict the classification of future events and behaviors. It's used in different industries and contexts. Some variables that can be predicted with this technique are the propensity to buy a product and the probability of getting positive or negative feedback from customers for a specific service.

Most digital native companies offer their services in subscription mode. In streaming video services, telco operators, and pay TVs, the binary logistic regression technique is widely used to predict the probability of churn of a customer. Predicting this kind of information is fundamental to target marketing campaigns and special offers to customers with the highest propensity to buy and increase revenue.

In this chapter, we'll see all the stages necessary to implement a binary logistic regression model leveraging BigQuery ML.

In order to predict one of the two labels, this ML algorithm calculates the probability of two different outcomes and allows us to choose a probability threshold to get the final classification of the binary variable.

Since this is an algorithm based on a regression technique, the prediction of the label is based on a set of independent variables called features that are used to predict the dependent variable, called a label.

This ML technique can be used to answer relevant business questions across different industries, such as the following:

- Will this customer buy my product?
- Is my customer satisfied with my service?
- Will my customer unsubscribe from my service in the next months?
- Will this student pass the next exam?
- Will this person develop diabetes in the next year?

In our business scenario, the possibility of getting a tip at the end of a taxi ride can be predicted by leveraging binary logistic regression. In fact, we're interested in predicting whether a certain event will happen or not. If the taxi driver will get a tip, the binary categorical variable will be valued with 1, otherwise 0.

Training a binary logistic regression model means trying to find the values of the coefficients that can be used in the equation between the input variables, called features, and the binary output variable, called the label.

After the training, we'll leverage a **confusion matrix** to evaluate the performance of our binary logistic regression model. In this matrix, the rows represent the predicted value of the label while the columns are used to store the actual values.

The following figure represents a confusion matrix that is used to evaluate the performances of the binary logistic regression:

		Actual Value	
		Positive	**Negative**
Predicted Value	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 5.2 – Confusion matrix

In the same tab, we can also see the ROC curve:

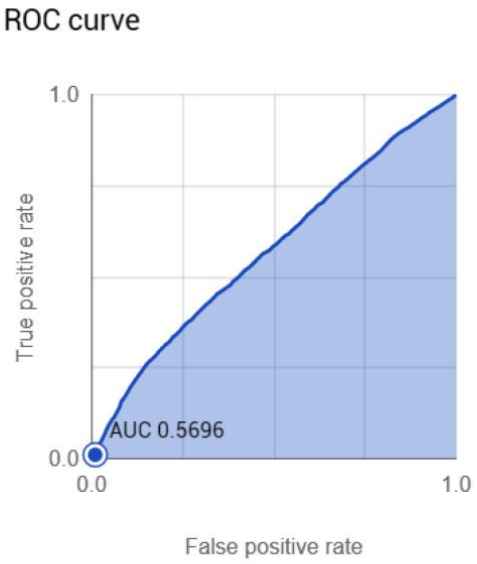


Figure 5.9 – In the Evaluation tab, it is also possible to graphically analyze the ROC curve and see the blue area under the ROC curve

As we can see from the preceding diagram, the ROC curve, which expresses the rate between the true positive and the false positive, is not close to 1. The blue area under the curve is about 50% of the entire square.

As shown in the following screenshot, we can also leverage the confusion matrix in the same tab to experiment with the outcome of the ML model according to different thresholds:

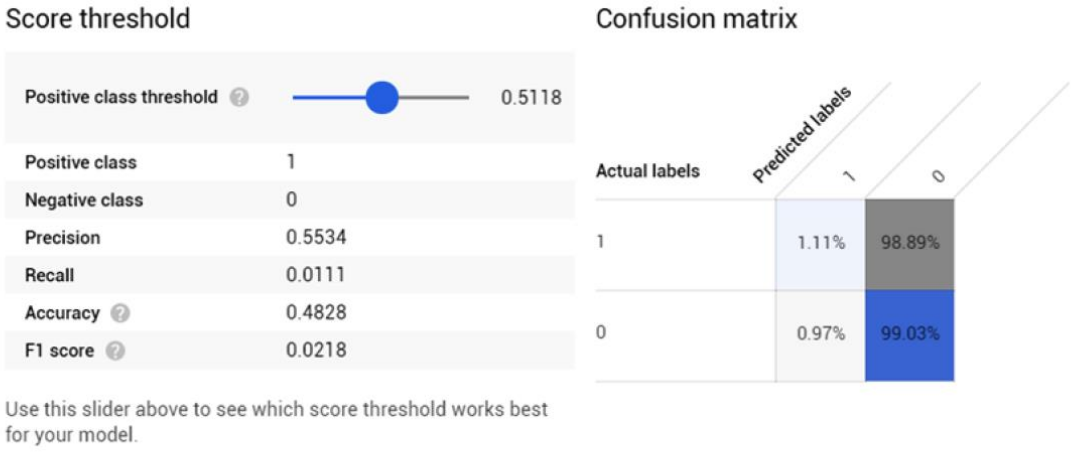


Figure 5.10 – In the Evaluation tab, it is also possible to see the confusion matrix of the classification model

In this section, we've trained some binary logistic regression ML models leveraging the features available in our dataset. To proceed with the evaluation stage, we choose to pick up the `binary_classification_version_4` model, which showed the best performance. Now, let's see how to start the evaluation phase.

Evaluating the binary logistic regression model

To evaluate our BigQuery ML model, we'll use the `ML.EVALUATE` function and the table that we've expressly created as the evaluation dataset.

The following query will tell us whether the model is suffering from overfitting or is also able to perform well on new data:

```
SELECT
  roc_auc,
  CASE
    WHEN roc_auc > .9 THEN 'EXCELLENT'
    WHEN roc_auc > .8 THEN 'VERY GOOD'
    WHEN roc_auc > .7 THEN 'GOOD'
    WHEN roc_auc > .6 THEN 'FINE'
    WHEN roc_auc > .5 THEN 'NEEDS IMPROVEMENTS'
  ELSE
  'POOR'
END
  AS model_quality
FROM
  ML.EVALUATE(MODEL `05_chicago_taxi.binary_classification_
version_5`,
    (
    SELECT
        trip_seconds,
        fare,
        tolls,
        company,
        payment_type,
        pickup_location,
        dropoff_location,
```

6
Classifying Trees with Multiclass Logistic Regression

Multiclass logistic regression is the **Machine Learning** (**ML**) algorithm used to classify events, entities, and behaviors into a fixed number of categories. It can be used across different industries and business scenarios when it's necessary to predict the classification of an entity into multiple groups. A typical classification use case is represented by the desire to segment the customer base of a company according to their profitability and preferences in order to target the right customers with the most effective marketing campaigns.

This kind of technique is an extension of the binary logistic regression that allows us to overcome the limits of two possible labels and opens the applicability to other contexts where we can find multiple categories to identify.

In this chapter, we'll see all the stages necessary to implement, evaluate, and test a multiclass logistic regression model leveraging BigQuery ML.

Training the multiclass logistic regression model

Now that we've clearly understood the structure of the data and we've segmented it into multiple tables to support the different stages of the ML model life cycle, let's focus on the training of our multiclass logistic regression model. We'll execute the SQL queries to create our multiclass logistic regression models:

1. Let's start creating the first version of our ML model:

```
CREATE OR REPLACE MODEL `06_nyc_trees.classification_
model_version_1`
OPTIONS
  ( model_type='LOGISTIC_REG',
    auto_class_weights=TRUE
  ) AS
SELECT
  zip_city,
  tree_dbh,
  spc_latin as label
FROM
  `06_nyc_trees.training_table` ;
```

The query used to create the `classification_model_version_1` model is based only on two features: the zip area and the diameter of the tree.

To have an idea of the effectiveness of our ML model, we can look at the **ROC AUC** value of **0.7383**.

By scrolling down with the mouse in the **Evaluation** tab, we can take a look at the confusion matrix of our multiclass logistic regression model.

In the following figure, the confusion matrix shows the percentage of predicted and actual labels on the training dataset:

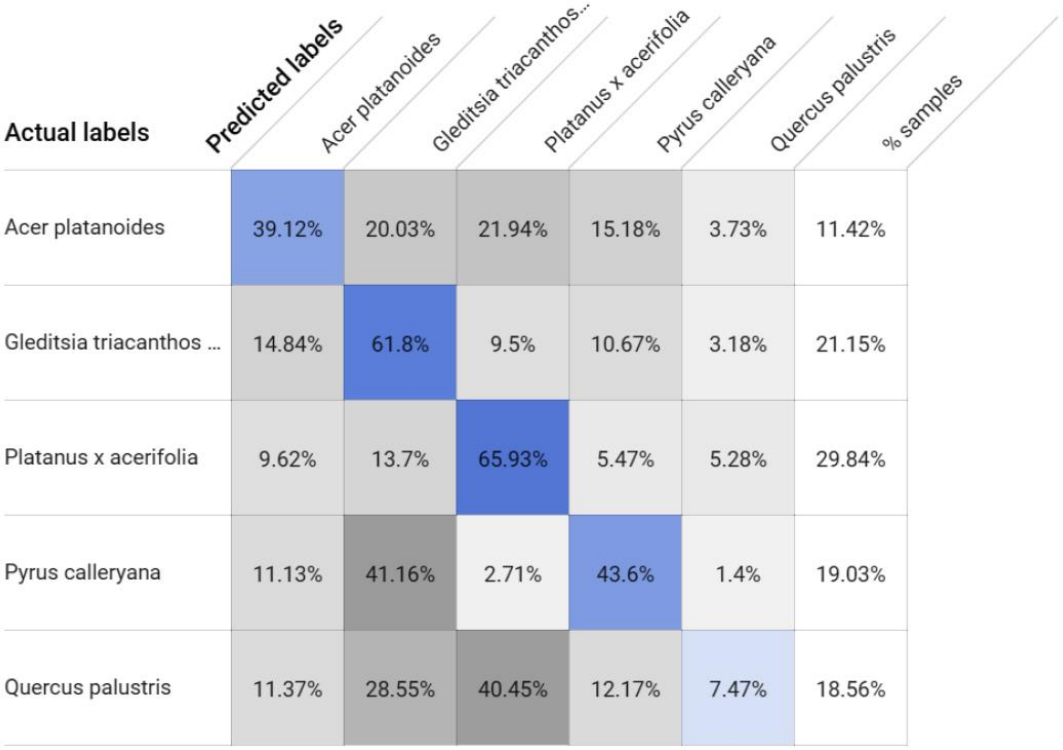


Figure 6.7 – The Evaluation tab shows the confusion matrix related to the selected BigQuery ML model

Looking at the confusion matrix, we can visually notice that our ML model works quite well for some species but performs very poorly for others. For example, when the actual label is **Quercus palustris**, in 40% of the cases the ML model suggests a different species: **Platanus x acerifolia**.

3. Let's try to improve our model by adding new features with the following BigQuery ML SQL statement:

```
CREATE OR REPLACE MODEL `06_nyc_trees.classification_
model_version_2`
OPTIONS
  ( model_type='LOGISTIC_REG',
```

Considering that the total size of the `classification_table` table is 27,182, we can declare that in 49% of cases, our ML model is able to predict the right species of tree based on its characteristics and its position.

This could seem like a bad result, but we need to consider that multiclass logistic regression is more complex than a binary one because there are multiple options that could deceive the results of our model.

Summary

In this chapter, we've built our first multiclass classification model. After a brief introduction to the use case, we discovered what multiclass logistic regression is and how it can be used to classify events, behaviors, and objects according to their features into more than two categories.

Before diving into the development of the ML model, we analyzed the schema of the dataset related to the trees in New York City and applied some data quality checks necessary to build an effective ML model.

During the training stage, we trained three different ML models using different features to gradually improve the performance of the BigQuery ML model.

Then, we chose the third ML model and we evaluated it against the evaluation dataset. In this phase, we noticed that the ML model was able to maintain its performance on new records also and was ready to pass to the next phase.

In the last step, we used our ML model to classify the trees in New York City into five different categories and leveraged their characteristics, such as size, health status, and position in the city.

We also calculated that our classification model is able to classify the right species of tree in 49% of cases.

In the next chapter, we'll introduce unsupervised ML and the K-Means clustering technique.

Section 3: Advanced Models with BigQuery ML

In this section, additional and advanced machine learning models are explained and presented with real hands-on examples using BigQuery ML.

This section comprises the following chapters:

- *Chapter 7, Clustering Using the K-Means Algorithm*
- *Chapter 8, Forecasting Using Time Series*
- *Chapter 9, Suggesting the Right Product by Using Matrix Factorization*
- *Chapter 10, Predicting Boolean Values Using XGBoost*
- *Chapter 11, Implementing Deep Neural Networks*

With an incremental approach, we'll go through the following topics:

- Introducing the business scenario
- Discovering K-Means clustering
- Exploring and understanding the dataset
- Training a K-Means clustering model
- Evaluating a K-Means clustering model
- Using a K-Means clustering model
- Drawing business conclusions

Discovering K-Means clustering

In this section, we'll understand what **unsupervised learning** is and we'll learn the basics of the **K-Means** clustering technique.

K-Means is an **unsupervised learning** algorithm that solves clustering problems. This technique is used to classify data into a set of classes. The letter *k* represents the number of clusters that are fixed *a priori*. For our business scenario, we'll use three different clusters.

> **Important note**
>
> While supervised learning is based on a prior knowledge of what the output values of labels should be in a training dataset, unsupervised learning does not leverage labeled datasets. Its goal is to infer the structure of data within a training dataset, without any prior knowledge of it.

Each cluster of data is characterized by a **centroid**. The centroid represents the midpoint of the cluster and is identified during the training stage and according to the features of the model.

After the training of the K-Means clustering model, each entity can be associated with the nearest centroid and included in one of the *k* clusters.

In the following diagram, you can take a look at the graphical representation of a simple clustering model based on two features and a value of *k* equals to `3`:

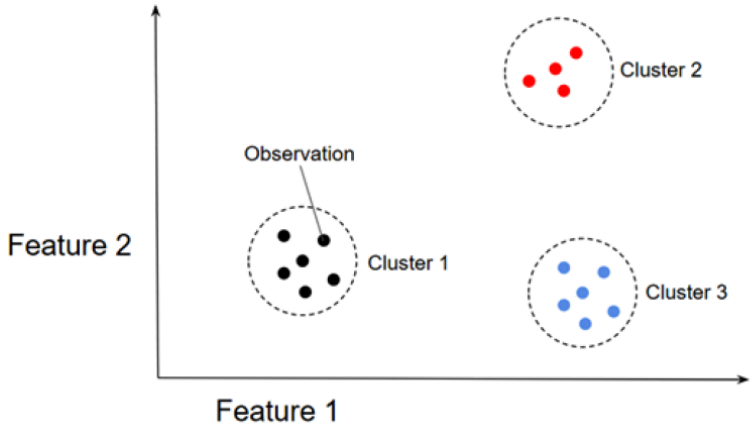


Figure 7.1 – Graphical representation of K-Means clustering

In the preceding Cartesian diagram, you can see some observations represented by dots. The diagram is composed of two axes that correspond to the features used to train the K-Means clustering machine learning model.

According to the values of the two features, some observations are closer than others. Assuming that we need to cluster the observations into three different clusters, the K-Means model is trained to find the three areas that divide the observations into different classes.

We'll not go through all the details of the K-Means clustering algorithm in this book, but we can mention some examples of use cases where this kind of algorithm is applicable. In real life, we can find a lot of scenarios that can be addressed with a clustering model, such as the following:

- **Customer segmentation**: Finding similar customers in the customer base of a company to improve the effectiveness of marketing campaigns and promotions
- **Employee segmentation**: Identifying employees with the best performance
- **Document classification**: Clustering documents into multiple categories according to tags, topics, authors, and publishing dates

Now, let's take a look at the options that we've used to train the machine learning model. The selected model type is `'kmeans'`. This option describes the technique that we're using to train the model. The `num_clusters` option is valued at `3` because we're trying to classify observations into three different clusters: *Top*, *Good*, and *Neutral*.

By default, BigQuery ML starts the training of the K-Means clustering algorithm with a random starting point. The quality of the machine learning model also depends on this point, which is randomly chosen by BigQuery. By using the `kmeans_init_method = 'KMEANS++'` option, the point is initialized leveraging the **K-Means++** algorithm. This type of algorithm is able to produce better and repeatable results by selecting the starting points for the training stage. It's always recommended to use 'KMEANS++' as the initialization method.

The training of the model is based on all the columns of the `taxi_miles_per_minute` table except for the `taxi_id` column, which will only be used during the prediction phase.

2. After the training of the first machine learning model, let's train a second one, which also includes the `tot_income` value of the taxi driver during the year, as illustrated in the following code snippet:

```
CREATE OR REPLACE MODEL `07_chicago_taxi_drivers.
clustering_by_speed_and_income`
OPTIONS(model_type='kmeans', num_clusters=3, standardize_
features = true, kmeans_init_method = 'KMEANS++') AS
  SELECT * EXCEPT (taxi_id)
  FROM `07_chicago_taxi_drivers.taxi_speed_and_income`;
```

This query is very similar to the SQL statement executed for the creation of the previous K-Means clustering model, but we can immediately notice a relevant difference. The `clustering_by_speed_and_income` model is trained using an additional option, `standardize_features = true`. This option is particularly useful when you have numeric features with different orders of magnitude. In this case, the model is using the `speed_mph` field (which goes from 0 to 50) and the `tot_income` field, which can reach a value of 150,000.

Now that we've trained two different machine learning models based on the K-Means clustering algorithm, let's take a look at how we can evaluate them leveraging BigQuery ML SQL syntax and the BigQuery **user interface** (**UI**).

8

Forecasting Using Time Series

Predicting future trends using historical data is one of the most fascinating activities that we can do with machine learning.

Making predictions based on historical data points and time series is particularly interesting and can be very useful in different industries. Forecasting can help us in predicting the future, but also in identifying anomalies in data that don't respect the expected pattern.

In this chapter, we'll focus on time series forecasting by using the ARIMA Plus algorithm. This technique can be used to predict numerical values in different fields, such as the sales of a company, the customers in a restaurant, stock prices, and the electricity consumption of a building.

To understand how we can use BigQuery ML to forecast trends and to effectively present our results, we'll go through the following topics:

- Introducing the business scenario
- Discovering time series forecasting
- Exploring and understanding the dataset
- Training the time series forecasting model
- Evaluating the time series forecasting model

Training the time series forecasting model

In this section, we'll train the BigQuery ML time series forecasting model.

Let's start training the machine learning model `liquor_forecasting`, executing the following SQL statement:

```
CREATE OR REPLACE MODEL `08_sales_forecasting.liquor_
forecasting`
OPTIONS
 (model_type = 'ARIMA',
  time_series_timestamp_col = 'date',
  time_series_data_col = 'total_sold_liters',
  auto_arima = TRUE,
  data_frequency = 'AUTO_FREQUENCY'
) AS
SELECT *
FROM
 `08_sales_forecasting.iowa_liquor_sales`;
```

The SQL statement is composed of the following parts:

- The first lines of the query start with the keywords `CREATE OR REPLACE MODEL`, followed by the identifier of the machine learning model, `` `08_sales_forecasting.liquor_forecasting` ``, and by `OPTIONS`.
- Now let's focus on the options that we've used to train our machine learning model. The selected model type is `'ARIMA'`. This option describes the algorithm that we're using to train the BigQuery ML forecasting model.
- `time_series_timestamp_col = 'date'` specifies the column that is used to host the time of the data points in the time series.
- The next option selects the column `total_sold_liters` as the column that stores the value of each data point and it's represented by the clause `time_series_data_col = 'total_sold_liters'`.
- The property `auto_arima`, set to `TRUE`, allows BigQuery ML to automatically identify the parameters `p`, `d`, and `q` of the model.
- With the last parameter, `'AUTO_FREQUENCY'`, BigQuery automatically infers the frequency of the time series. In this case, the frequency is daily. The other options are `'HOURLY'`, `'DAILY'`, `'WEEKLY'`, `'MONTHLY'`, `'QUARTERLY'`, and `'YEARLY'`.

- A `STRUCT` that includes the `horizon` of the forecast: `30` days and the `confidence_level` chosen for the prediction – in this case, 80%:

```
SELECT
  *
FROM
  ML.FORECAST(MODEL `08_sales_forecasting.liquor_
forecasting`,
              STRUCT(30 AS horizon, 0.8 AS confidence_
level));
```

The execution of the query generates the records shown in the following screenshot:

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 23.4 KB processed)

Job information Results JSON Execution details

Row	forecast_timestamp	forecast_value	standard_error	confidence_level	prediction_interval_lower_bound	prediction_interval_upper_bound	confidence_interval_lower_bound	confidence_interval_upper_bound
1	2020-01-01 00:00:00 UTC	49516.648901194705	0.0	0.8	49516.648901194705	49516.648901194705	49516.648901194705	49516.648901194705
2	2020-01-02 00:00:00 UTC	22738.593972943178	2138.4352565226172	0.8	19996.507004484145	25480.68094140221	19996.507004484145	25480.68094140221
3	2020-01-03 00:00:00 UTC	17721.442759603313	2238.6046293658883	0.8	14850.909942211307	20591.97557699532	14850.909942211307	20591.97557699532
4	2020-01-04 00:00:00 UTC	41521.169448279164	2646.587193130179	0.8	38127.4860383947	44914.85285816363	38127.4860383947	44914.85285816363
5	2020-01-05 00:00:00 UTC	53200.9074424626	2656.9814339309707	0.8	49793.89563722159	56607.91924770361	49793.89563722159	56607.91924770361

Figure 8.14 – The results generated by the forecast function

We can notice in *Figure 8.14* that the predictions are chronologically ordered according to the date in the field **forecast_timestamp**. Each row represents a day and its related field **forecast_value** predicted by the BigQuery ML model. Since we've selected a horizon of 30 days, our result set is composed of 30 rows that go from January 1, 2020 to January 30 of the same year. The `confidence_level` value of 0.8 means that 80% of the predicted values should fall into the prediction interval identified by the fields **prediction_interval_lower_bound** and **prediction_interval_upper_bound**.

Now that we've applied our model, let's understand how we can effectively present the results using Data Studio.

According to different business scenarios, this feedback can be considered explicit or implicit:

- **Explicit feedback** is available when the user voluntarily rates a specific item, such as on a review website.
- If explicit feedback is not available, **implicit feedback** can be calculated and inferred by the developer of the recommendation system. For example, if a customer has bought a product, we can assume they would give positive feedback for that item.

Often, in e-commerce data, feedback is not explicitly given by the users but can be extracted from other information that's collected in the process, such as the number of clicks, the time spent on a specific page, or the quantity of a specific product bought by the user.

Matrix factorization algorithms are widely used in real-life scenarios. Some examples are as follows:

- The suggested books that we see in an online bookshop.
- The recommended TV series that we can see on video streaming services.
- The posts that are highlighted in our social media feeds.
- The products suggested by internet advertising systems.

In this section, we learned about the basics of matrix factorization. Now, let's configure BigQuery Flex Slots.

Configuring BigQuery Flex Slots

In this chapter, we'll understand how to configure BigQuery **Flex Slots** to train our ML model.

A BigQuery **Slot** is a unit of BigQuery analytics capacity that's used to execute SQL queries and train BigQuery ML models. One BigQuery slot represents the compute capacity of a **Virtual Compute Processing Unit** (**VCPU**).

Flex Slots allow us to buy BigQuery analytics capacity for short periods. They are usually used to quickly satisfy sudden demands for resources with a minimum duration of 60 seconds.

Enabling Flex Slots is mandatory to train a matrix factorization model; otherwise, BigQuery will return an error during the training stage.

Let's see how we can enable BigQuery Flex Slots if we're using an on-demand plan:

1. If you haven't enabled BigQuery Reservations yet, we need to access **Reservations** from the BigQuery menu on the left:

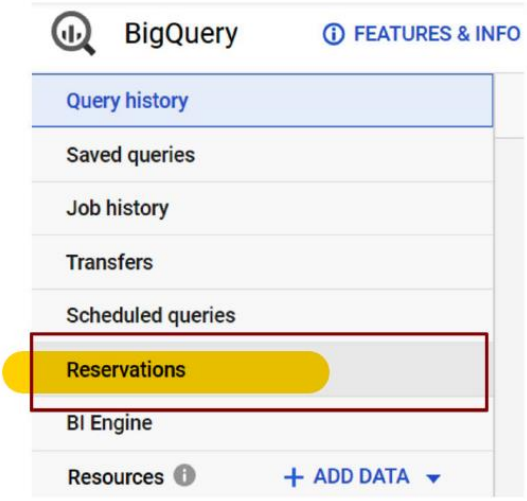


Figure 9.3 – Accessing Reservations from the BigQuery navigation menu

2. Click on the **BUY SLOTS** button to initialize the buying process for BigQuery Flex Slots:

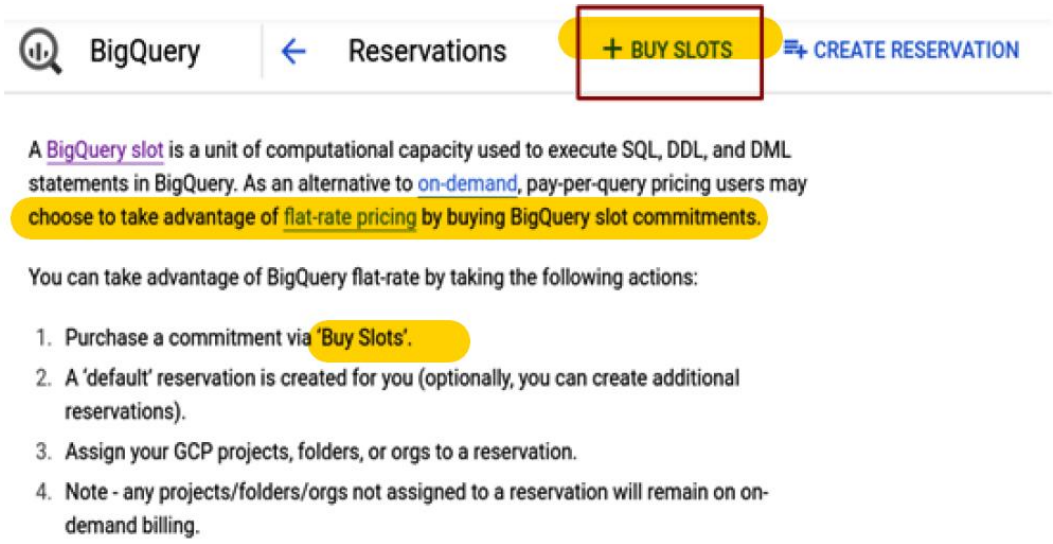


Figure 9.4 – Screenshot of the Reservations page

3. Choose the minimum number of Flex Slots to buy; that is, **100**. Then, click on the **NEXT** button:

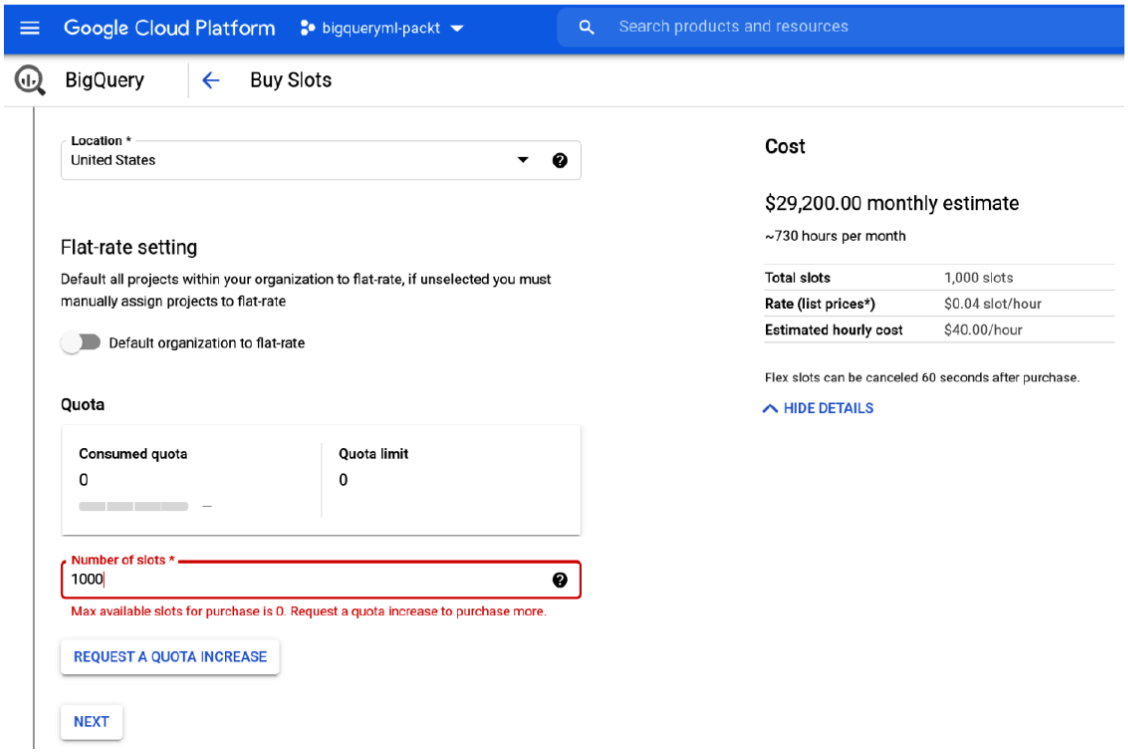


Figure 9.5 – BigQuery Buy Slots process

4. Confirm your choice by writing **CONFIRM** in the confirmation text box and clicking on the blue **PURCHASE** button:

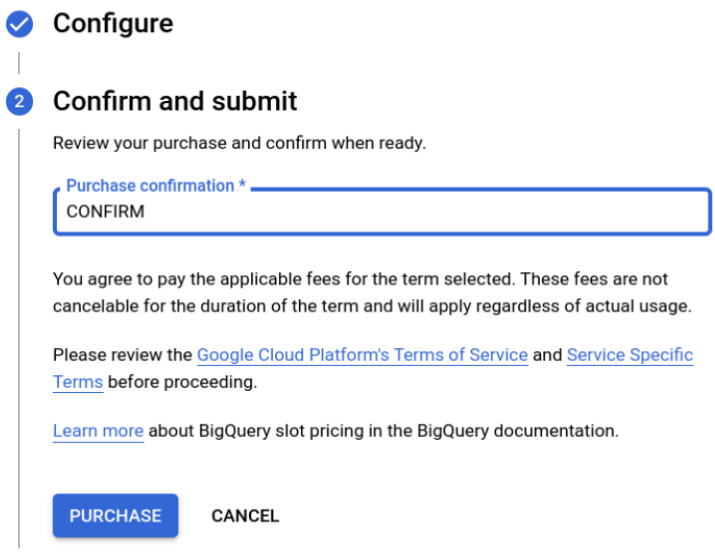


Figure 9.6 – Confirming your BigQuery Slots purchase

...

Collaborating with BigQuery

In this section, we'll discover how to use BigQu
maximum value from both technologies.

The following diagram shows the interactions b
scientists using BigQuery ML and TensorFlow:

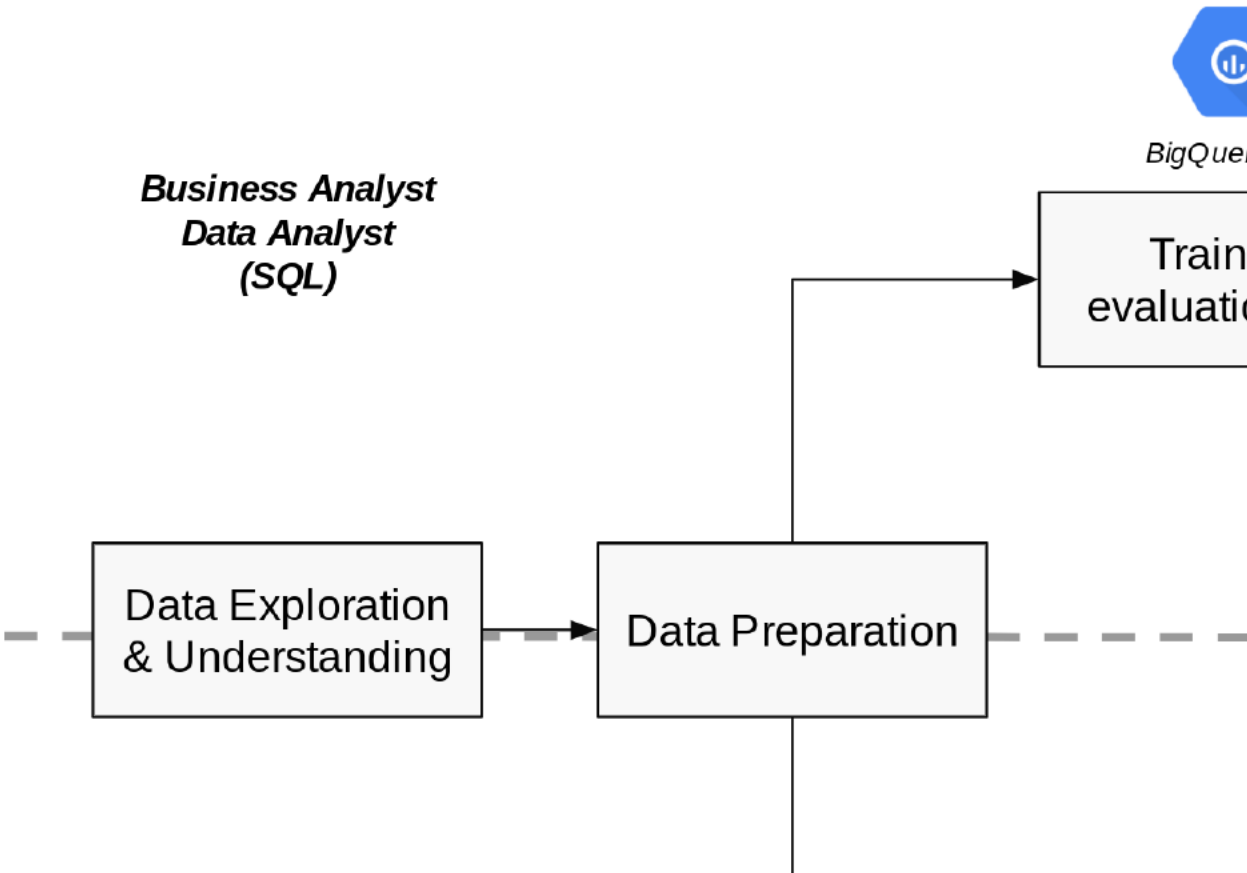


Figure 13.7 – Google Cloud Stora

Upon accessing the Google Cloud storage buck
the `bqml_exported_model` subfolder, we'll
BigQuery ML model, as shown in the following

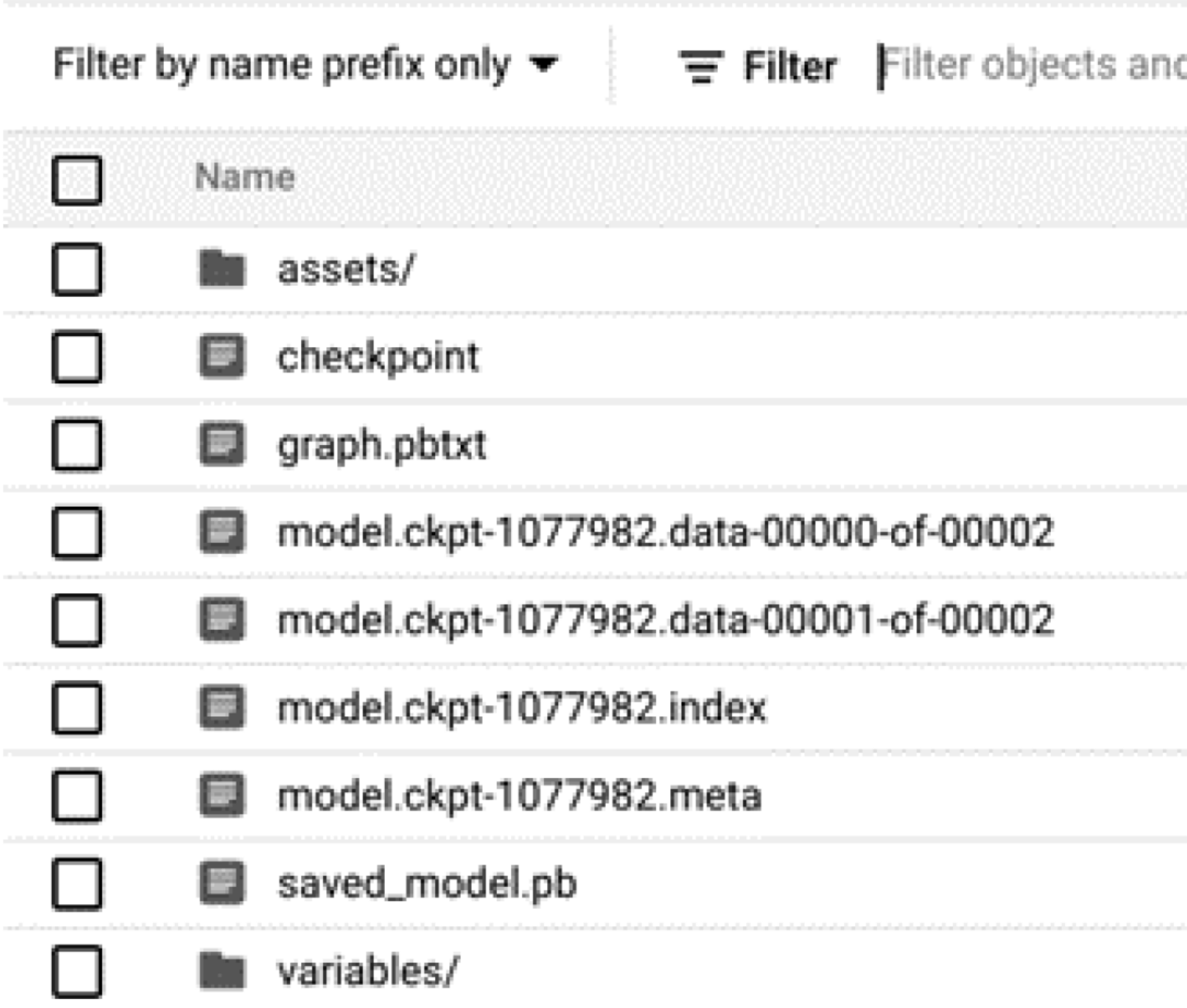


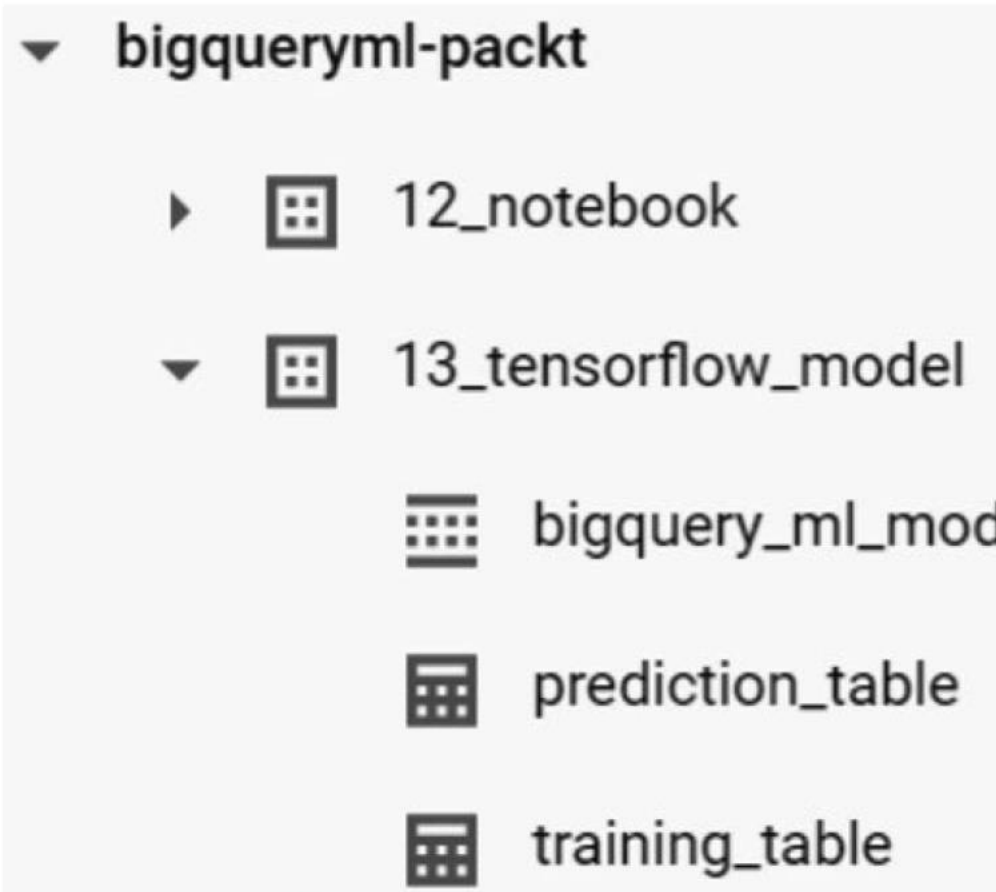
Figure 13.8 – The BigQuery ML model exported into t

`tensorflow_model.trip_duration_tf_`

In the `OPTIONS` clause, we've specified `'tenso` option. Using the `model_path` parameter, we've TensorFlow SavedModel will be stored in the Go

2. To verify that BigQuery ML successfully loaded t browse the BigQuery navigation menu and check `13_tensorflow_model` dataset.

   The following screenshot shows that the TensorF BigQuery ML. Its name is `trip_duration_tf`



BigQuer
and Best

BigQuery ML has the great advantage of democrat
(**ML**) for data and business analysts. In fact, BigQu
programming experience to implement advanced
ML is designed to simplify and automatize the cre
best practices and tips that should be adopted duri
algorithm to obtain an effective performance from

Having a background in data science can help us i
of our ML models and in avoiding pitfalls during t
chapter, we'll learn how to choose the right techni
and will learn about the tools we can leverage to in

Following a typical ML development life cycle, we'

In the following table, we can see a summary of a
can use to develop our ML models:

Labelled Dataset Required	Expected Output
Yes	Continuous real numbers
Yes	Two discrete classes
Yes	Multiple discrete classes
No	Grouping of the input items
Yes	Couples of linked input items
Not applicable	Continuous real numbers
Yes	Continuous real numbers
Yes	Multiple discrete classes
Yes	Continuous real numbers
Yes	Multiple discrete classes

In the following diagram, you can see the different
quality of the data:

- The remaining 10% for the test set.
- If we're working on large amounts of data observations used for the validation and

In the following screenshot, you can see a graph splitting strategy:

Figure 14.8 – The 80/10/10 s

In order to achieve the best results, the split pro

In the following code block, you can see how to `MOD` function:

```
SELECT
  CASE
    WHEN MOD(<RECORD_KEY>, 10) <
    WHEN MOD(<RECORD_KEY>, 10) =
    WHEN MOD(<RECORD_KEY>, 10) =
```

One typo – p.324 – 0-9 in Mod function, see https://cloud.google.com/bigquery/docs/reference/standard-sql/mathematical_functions#mod

In the following code block, you can see how to appl
`MOD` function:

```
SELECT
   CASE
      WHEN MOD(<RECORD_KEY>, 10) < 8 TH
      WHEN MOD(<RECORD_KEY>, 10) = 8 TH
      WHEN MOD(<RECORD_KEY>, 10) = 9 TH
   END AS dataframe
FROM
   <TABLE_NAME>
```

In the example, the records stored in the table repres
`<TABLE_NAME>`, are split into three different sets ac
dataframe. In this case, the MOD function returns a
function allows us to group the records into three di
clauses `MOD(<RECORD_KEY>, 10)<8`, `MOD(<R`
`MOD(<RECORD_KEY>, 10) = 9`, we can split the
`'evaluation'` and `'test'` sets.

Now that we've understood how to segment the data
for training, validation, and testing, let's look at the

- `ML.FEATURE_CROSS`: This is used to com
  For example, if in a dataset we have the gen
  can combine these two features to simplify
  is particularly indicated when we've correla
  the information in our ML model.
- `ML.QUANTILE_BUCKETIZE`: This is very
  In this case, the function is analytic and ap
  splitting of records into buckets is based on

> **Important note**
>
> The **Quantile** is the specific portion of a data
> many values are above or below a certain thr

- `ML.MIN_MAX_SCALER`: This is an analytic
  to one according to the distribution of the
- `ML.STANDARD_SCALER`: This is an analyt
  standard deviation and mean of the record

For an entire list of feature engineering and prepr

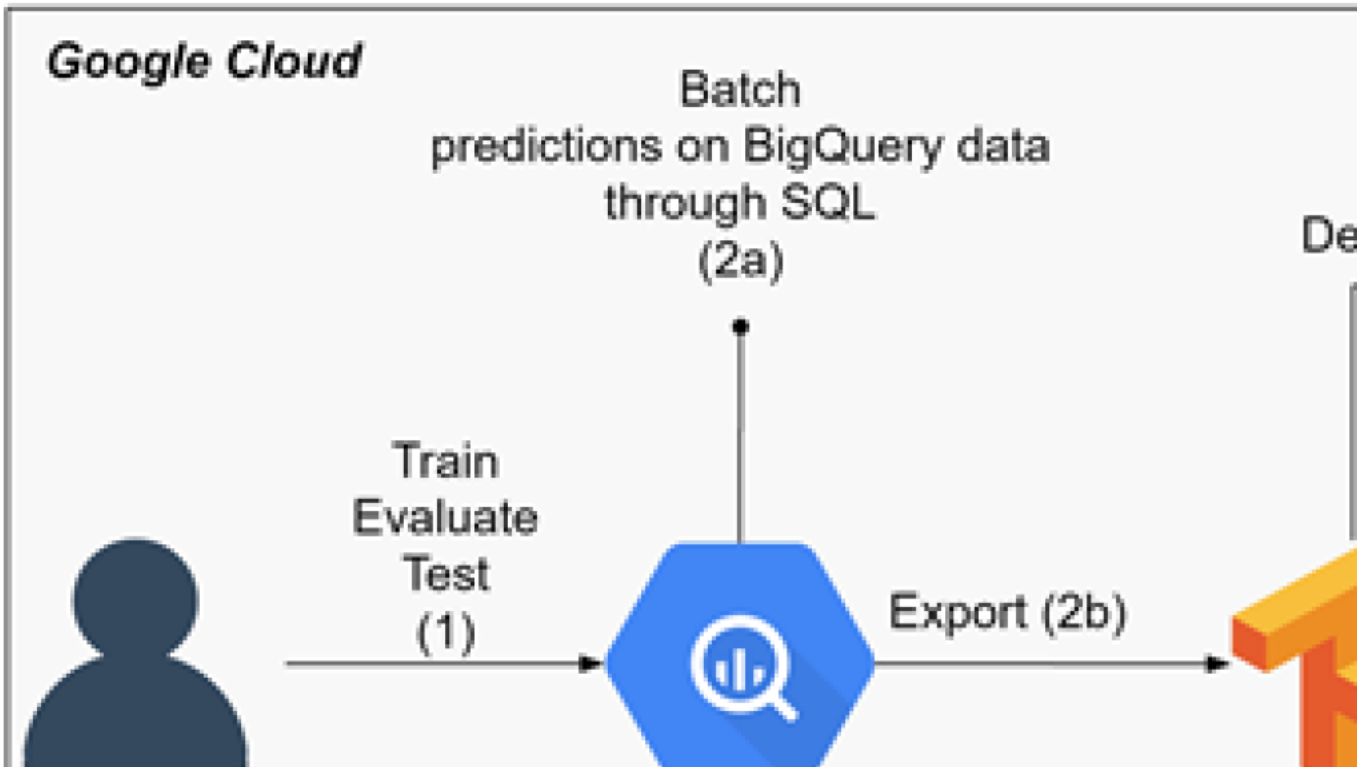
In BigQuery ML, we can specify the hyperparam
parameters. The most relevant hyperparameters,
change before starting the training of a BigQuery

- `L1_REG`: This is a regularization paramete
by keeping the weights of the model close
- `L2_REG`: This is a second regularization pa
overfitting.
- `MAX_ITERATIONS`: This represents the m
BigQuery ML will perform to train the mo
- `LEARN_RATE`: This is a parameter that affe
according to the error of the previous itera
- `MIN_REL_PROGRESS`: This is the minimu
continue the training after an iteration.
- `NUM_CLUSTERS`: This is used for K-Mean
of clusters that the model will create.
- `HIDDEN_UNITS`: This is used in **Deep Ne**
the number of hidden layers in a network.

- We use **online prediction** when we want to e
  and when getting an immediate prediction is
- We adopt **batch prediction** to process large v
  immediate predictions—for example, schedu
  predictions on the data collected since the las

While using BigQuery SQL statements is more suit
number of records stored in a BigQuery table, the p
models into TensorFlow opens new opportunities i

In the following diagram, you can see the life cycle
training phase to the deployment phase:



Amazon review :

Walter Lee Edit

Create Review

Machine Learning with BigQuery ML: Create, execute, and improve machine learnin…

Overall rating

Clear

Add a headline

From Zero to Hero in ML the easy way with GCP BigQuery ML and this book!

Add a photo or video

Shoppers find images and videos more helpful than text alone.

Add a written review

This is a very easy to read intro. level book to get you started using BQML the easy way. It shows you step by step guide to prep/clean/filter the data, train the model with many different supported algorithms/options, evaluate the model (e.g. no overfitting) and predict with the trained model. It is an excellent book if you like to use standard SQL to do Machine Learning without the need to learn Python, Tensorflow, etc… This will help liberate many business intelligence (BI) specialists upgraded into the next level, i.e. able to predict and classify for good business use cases with BQML machine learning models. A lot of easy to use code to get started. A lot of references to learn more in BQML and other use cases.

Section 1 is more introduction to GCP and BQ environment. You can skip if you know them already.
Section 2 is more on the linear regression, binary and multiclass logistic classifications.
Section 3 is more on Adv. Models, e.g. K-means, matrix factorization, XGBoost, Deep Neural Networks.
Section 4 is more on how to use GCP AI notebooks to run BQML, run TF models with BQML