# **Machine Learning with BigQuery ML**

# **Task 1. Deploy a Vertex AI Workbench instance**

1. On the **Navigation Menu**, click **More products**, and then **Vertex AI** > **Workbench**.
2. On the Notebook instances page, click **New Notebook** > **TensorFlow Enterprise** > **TensorFlow Enterprise 2.6 (with LTS)** > **Without GPUs**.
3. In the **New notebook dialog**, name the Notebook bqml-notebook.
4. For **Region**, select us-west1.
5. For **Zone**, select a zone within the selected region.
6. Leave all other fields with their default options, and click **Create**.

As the Notebook spins up, the bqml-notebook notebook is listed in the Workbench Notebook list. When it completes, Open Jupyterlab appears inline with the Notebook name.

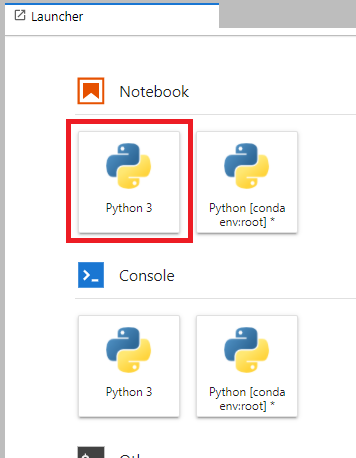
1. Click **Open JupyterLab**.

When the Vertex AI Notebook instance starts you will see a suggestion to start a build to add **tensorflow\_model\_analysis** and **xai\_tabular\_widget**.

1. Click **Build**.

Your notebook is now set up.

1. In the **Notebook** launcher section click **Python 3** to open a new notebook.



To use a Notebook you enter commands into a cell. Be sure you run the commands in the cell by either pressing **Shift + Enter**, or clicking the triangle on the Notebook top menu to **Run selected cells and advance**.

After pasting commands into the Jupyter notebook cell, always run the cell to execute the command and advance to the next cell.

**Task 2. Build logistic regression model**

Create the training dataset

The first step in BigQuery ML is to create the training dataset. We want the three features (departure delay, taxi out time, and distance.) and the label (ontime), so let’s use SQL to craft the dataset just the way we want it.

Enter following code into new cell and then run the cell.

%%bigquery

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance,

IF(is\_train\_day = 'True', False, True) AS is\_eval\_day

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

f.DIVERTED = False

LIMIT 5

Create the model

In this section, you create a logistic regression model where the label column is called ontime, and the remaining columns are used as input features to the model. Here, you do not want to randomly split the flight data – you need to avoid having correlated flights on the same day split between training and test datasets.

That’s why when you created the training data in the previous section, you pre-split the data and created a table that specifies which days should be used for training and which days for evaluation. The resulting model parameters are stored in a BigQuery model object called arr\_delay\_lm in the dataset dsongcp.

You can explicitly tell BigQuery ML to use a column in the training dataset to split the data. Add that column to your SELECT statement as a Boolean value by joining the flight data against the table of prespecified training days.

1. Create a logistic regression model in BigQuery ML by running the following query in new cell.

%%bigquery

CREATE OR REPLACE MODEL dsongcp.arr\_delay\_lm

OPTIONS(input\_label\_cols=['ontime'],

model\_type='logistic\_reg',

data\_split\_method='custom',

data\_split\_col='is\_eval\_day')

AS

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance,

IF(is\_train\_day = 'True', False, True) AS is\_eval\_day

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

f.DIVERTED = False

**Note:** It will take about 10 minutes to complete.

In the above query, BigQuery trains a logistic regression model and puts the weights into the model arr\_delay\_lm. Obtain the training error (called the loss) by querying the result of the special function ML.TRAINING\_INFO:

1. Enter following into new cell and run to obtain training error.

%%bigquery

SELECT \* FROM ML.TRAINING\_INFO(MODEL dsongcp.arr\_delay\_lm)

**Task 3. Evaluate the model**

Evaluate the model by calling ML.EVALUATE in SQL. BigQuery then evaluates the model on the withheld data (where is\_train\_day is False), but use a threshold of 0.7.

Run the query below in the new cell to evaluate the model.

%%bigquery

SELECT \*

FROM ML.EVALUATE(MODEL dsongcp.arr\_delay\_lm,

(

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

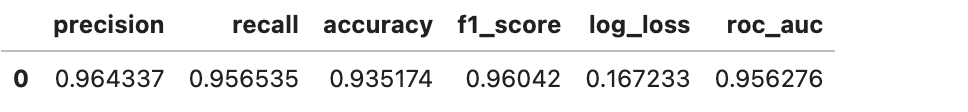
f.DIVERTED = False AND

is\_train\_day = 'False'

),

STRUCT(0.7 AS threshold))

The resulting evaluation statistics are:



The precision is how often the model is right when it reports a flight as being on time. The recall is the fraction of on time flights correctly classified. The Receiver Operating Characteristic (ROC) is a threshold-independent measure of classifier performance.

**Task 4. Make prediction from the model**

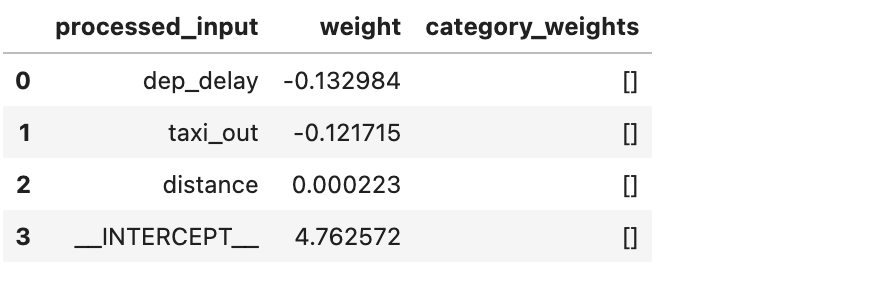
The ML.WEIGHTS function allows you to see the underlying weights used by a model during prediction. This function applies to linear & logistic regression models.

1. Run the following query in new cell to obtain the weights by calling ML.WEIGHTS.

%%bigquery

SELECT \* FROM ML.WEIGHTS(MODEL dsongcp.arr\_delay\_lm)

The output is similar to the following:



However, there is usually no point to getting just the weights. Instead, what you want is the predicted value for some set of inputs.

1. Run the query below in a new cell to carry out a prediction.

%%bigquery

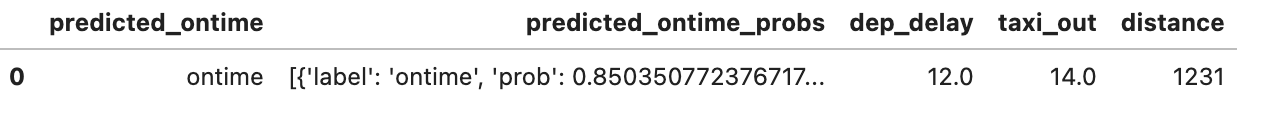
SELECT \* FROM ML.PREDICT(MODEL dsongcp.arr\_delay\_lm,

(

SELECT 12.0 AS dep\_delay, 14.0 AS taxi\_out, 1231 AS distance

))

Output:



While you can use ML.PREDICT to actually carry out predictions, the predictions are subject to the typical BigQuery latency of a second or so. Therefore, ML.PREDICT is typically used for batch predictions over large datasets. For online prediction (i.e. exposing the prediction service a microservice using REST), you can extract the model as a TensorFlow model and deploy it into Vertex AI.

Use ML.PREDICT to actually carry out predictions in case you want to compute some other metric.

1. Run the query below in new cell to compute Root Mean Squared Error (RMSE).

%%bigquery

WITH predictions AS (

SELECT

\*

FROM ML.PREDICT(MODEL dsongcp.arr\_delay\_lm,

(

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

f.DIVERTED = False AND

t.is\_train\_day = 'False'

),

STRUCT(0.7 AS threshold))),

stats AS (

SELECT

COUNTIF(ontime != 'ontime' AND ontime = predicted\_ontime) AS correct\_cancel

, COUNTIF(predicted\_ontime = 'ontime') AS total\_noncancel

, COUNTIF(ontime = 'ontime' AND ontime = predicted\_ontime) AS correct\_noncancel

, COUNTIF(ontime != 'ontime') AS total\_cancel

, SQRT(SUM((IF(ontime = 'ontime', 1, 0) - p.prob) \* (IF(ontime = 'ontime', 1, 0) - p.prob))/COUNT(\*)) AS rmse

FROM predictions, UNNEST(predicted\_ontime\_probs) p

WHERE p.label = 'ontime'

)

SELECT

correct\_cancel / total\_cancel AS correct\_cancel

, total\_noncancel

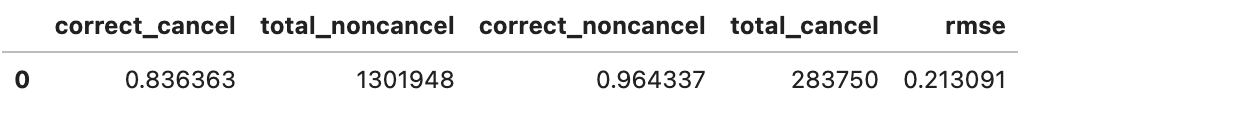
, correct\_noncancel / total\_noncancel AS correct\_noncancel

, total\_cancel

, rmse

FROM stats

Output something similar to the following:



The query above pulls out the probability field from the predictions (it’s an array, one for each category, hence the UNNEST) and uses it to compute the RMSE. The resulting RMSE was 0.2131.

**Task 5. Create, evaluate and predict the model by adding additional airport information**

Create the BigQuery ML logistic regression model arr\_delay\_airports\_lm by adding airport information to model (note two additional columns: origin and dest). This seemingly simple change adds two categorical variables that, when one-hot-encoded, adds 600+ new columns to the model.

1. To showcase the scalability of BigQuery, add two fields, the origin and destination airport:

%%bigquery

CREATE OR REPLACE MODEL dsongcp.arr\_delay\_airports\_lm

OPTIONS(input\_label\_cols=['ontime'],

model\_type='logistic\_reg',

data\_split\_method='custom',

data\_split\_col='is\_eval\_day')

AS

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance,

origin,

dest,

IF(is\_train\_day = 'True', False, True) AS is\_eval\_day

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

f.DIVERTED = False

Create a logistic regression model by adding additional airport information

**Note:** It will take about 10 minutes to complete.

1. Evaluate the model arr\_delay\_airports\_lm.

%%bigquery

SELECT \*

FROM ML.EVALUATE(MODEL dsongcp.arr\_delay\_airports\_lm,

(

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance,

origin,

dest,

IF(is\_train\_day = 'True', False, True) AS is\_eval\_day

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

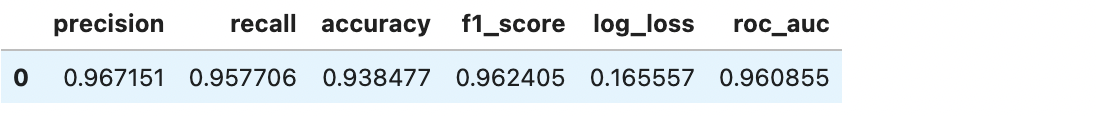
f.DIVERTED = False AND

t.is\_train\_day = 'False'

),

STRUCT(0.7 AS threshold))

Output -



1. Make prediction from the model arr\_delay\_airports\_lm.

%%bigquery

WITH predictions AS (

SELECT

\*

FROM ML.PREDICT(MODEL dsongcp.arr\_delay\_airports\_lm,

(

SELECT

IF(arr\_delay < 15, 'ontime', 'late') AS ontime,

dep\_delay,

taxi\_out,

distance,

origin,

dest,

IF(is\_train\_day = 'True', False, True) AS is\_eval\_day

FROM dsongcp.flights\_tzcorr f

JOIN dsongcp.trainday t

ON f.FL\_DATE = t.FL\_DATE

WHERE

f.CANCELLED = False AND

f.DIVERTED = False AND

t.is\_train\_day = 'False'

),

STRUCT(0.7 AS threshold))),

stats AS (

SELECT

COUNTIF(ontime != 'ontime' AND ontime = predicted\_ontime) AS correct\_cancel

, COUNTIF(predicted\_ontime = 'ontime') AS total\_noncancel

, COUNTIF(ontime = 'ontime' AND ontime = predicted\_ontime) AS correct\_noncancel

, COUNTIF(ontime != 'ontime') AS total\_cancel

, SQRT(SUM((IF(ontime = 'ontime', 1, 0) - p.prob) \* (IF(ontime = 'ontime', 1, 0) - p.prob))/COUNT(\*)) AS rmse

FROM predictions, UNNEST(predicted\_ontime\_probs) p

WHERE p.label = 'ontime'

)

SELECT

correct\_cancel / total\_cancel AS correct\_cancel

, total\_noncancel

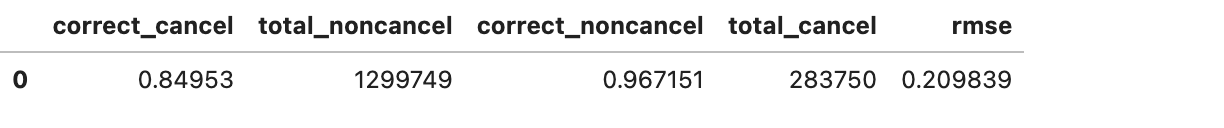
, correct\_noncancel / total\_noncancel AS correct\_noncancel

, total\_cancel

, rmse

FROM stats

The resulting prediction statistics are:



The model that includes the airport information has a RMSE of 0.2098, which is an improvement over the original model created above.