BQML on Looker

An explanation on LookML / BQ syntax (and not on machine learning)

Lan Tran (lantrann@)

Working In Progress, last updated: Jan 4, 2021

Agenda

BQML — How BigQuery does Machine Learning

BQML on LookML — 5 steps to create and deploy a machine learning model

Next steps — Take-away and further resources

BQML

BQML on LookML

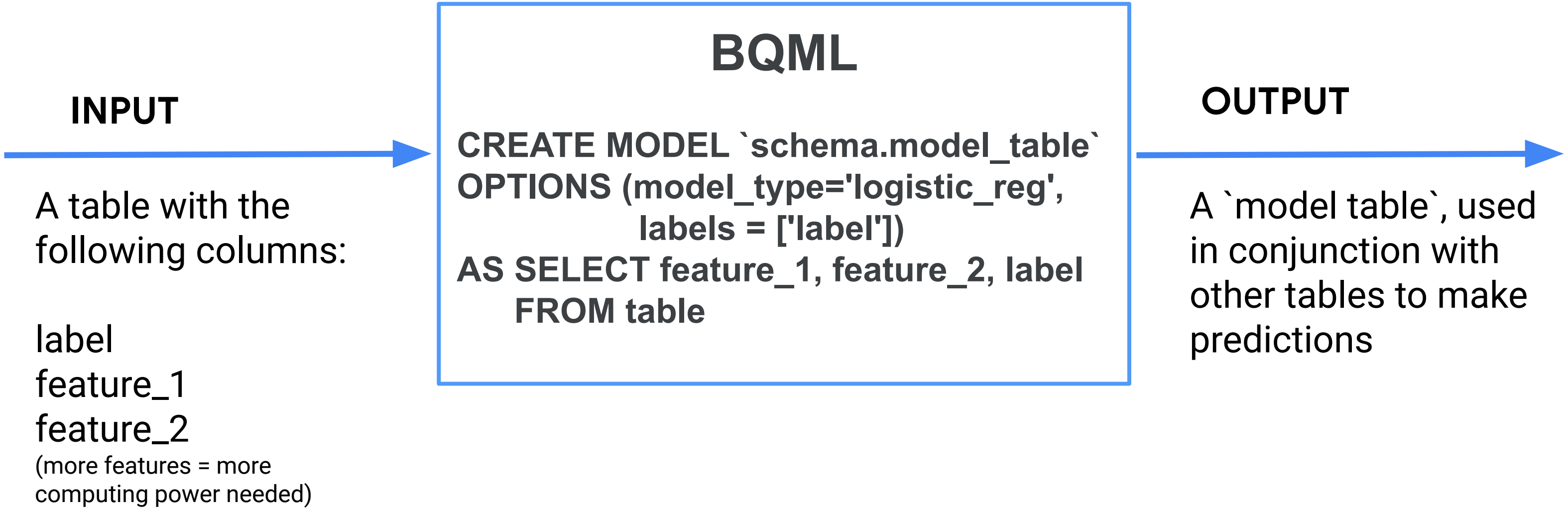
Next steps

How BigQuery does Machine Learning

5 steps to create and deploy a machine learning model

Take-away and further resources

BQML (BigQuery Machine Learning) allows users to create a machine learning model directly in BigQuery using built-in functions in form of a standard SQL query



INPUT

A table with the following columns:

label
feature_1
feature_2
(more features = more computing power needed)

BQML

```
CREATE MODEL `schema.model_table`
OPTIONS (model_type='logistic_reg',
             labels = ['label'])
AS SELECT feature_1, feature_2, label
    FROM table
```

OUTPUT

A `model table`, used in conjunction with other tables to make predictions

BQML (BigQuery Machine Learning) follows a standard data science workflow (create model, evaluate, predict)

BQML Workflow:

- Step 1: Prepare the data (training, testing, evaluation, etc)
- Step 2: Make a model, using `CREATE MODEL`
- Step 3: Evaluate the model, using `ML.EVALUATE`
- Step 3: Use the model to make predictions, using `ML.PREDICT`

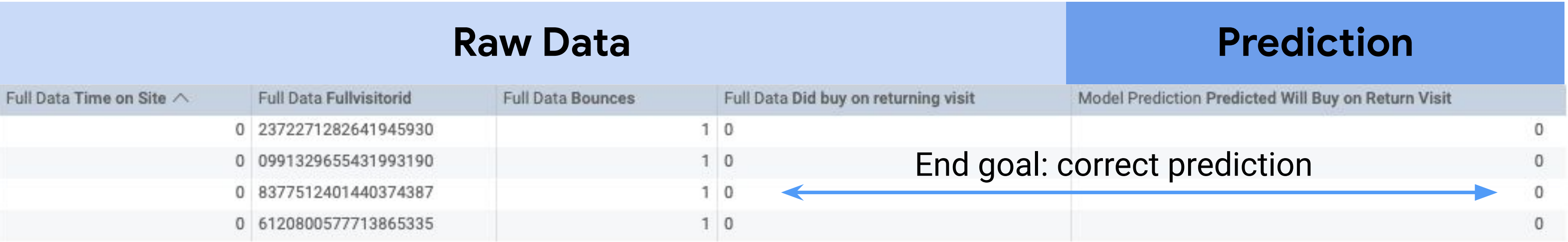
BQML in Looker:

- Read the card "How does Looker work with BQML"
- TLDR: **We define BQML functions inside Looker's PDT to execute ML models**

Build a model that uses “bounce” and “time_on_site” to predict “will_buy_on_return_visit” (if the same user will purchase in their next visit)

- “will_buy_on_return_visit” is a ***label*** because we want to predict its value
- “bounce” and “time_on_site” are ***features*** because we use them to predict another value

How do we know which machine learning model and which features to use?
⟶ That’s the job of a data scientist: They create multiple models to find the most optimal one



Raw Data				Prediction
Full Data Time on Site	Full Data Fullvisitorid	Full Data Bounces	Full Data Did buy on returning visit	Model Prediction Predicted Will Buy on Return Visit
0	2372271282641945930	1	0	0
0	0991329655431993190	1	0	0
0	8377512401440374387	1	0	0
0	6120800577713865335	1	0	0

End goal: correct prediction

Bounce: Visitors who enter the site and then leave ("bounce") (yes = 1)
Time on Site: Total time of the session expressed in seconds.

BQML

BQML on LookML

Next steps

How BigQuery does Machine Learning

5 steps to create and deploy a machine learning model

Take-away and further resources

STEP 1: PREPARE THE DATA

We split the dataset into ***training data*** and ***testing data***. BQML will make a model from learning the training data. We then use the model to make a prediction on the testing data (“Does this model make accurate predictions?). For demo purposes, we split data on “date”. Example Code

FULL DATA

```
view: full_data {
  derived_table:{
    sql: WITH first_time_visitor as (SELECT
          fullVisitorId,
          date,
          IFNULL(totals.bounces, 0) AS bounces,
          IFNULL(totals.timeOnSite, 0) AS time_on_site
        FROM
          `data-to-insights.ecommerce.web_analytics`
        WHERE
          totals.newVisits = 1)
       SELECT * FROM first_time_visitor
       JOIN (SELECT
          fullvisitorid,
          IF(COUNTIF(totals.transactions > 0 AND totals.newVisits IS NULL) > 0, 1, 0) AS
will_buy_on_return_visit
        FROM
           `data-to-insights.ecommerce.web_analytics`
        GROUP BY fullvisitorid)
        USING (fullVisitorId);;
  }
```

TRAINING DATA

```
view: training_input {
  derived_table: {
    sql: SELECT * FROM ${full_data.SQL_TABLE_NAME}
          WHERE date BETWEEN '20160801' AND '20170430';; }
}
```

TESTING DATA

```
view: training_input {
  derived_table: {
    sql: SELECT * FROM ${full_data.SQL_TABLE_NAME}
          WHERE date BETWEEN '20170501' AND '20170630';; }
}
```

STEP 2: CREATE OR REPLACE MODEL

Make a PDT, using datagroup_trigger or sql_trigger_value.

Use `sql_create` so Looker doesn't check the syntax

Use the "training" view (step 1). Include **features** (columns used to make prediction: "bounces" and "time_on_site") and **labels** (columns we want to predict: "will_buy_on_return_visit")

```
view: future_purchase_model {
  derived_table: {
    datagroup_trigger: bqml_datagroup
    sql_create:
      CREATE OR REPLACE MODEL ${SQL_TABLE_NAME}
      OPTIONS(model_type='logistic_reg'
        , labels=['will_buy_on_return_visit'])
     AS
        SELECT bounces, time_on_site,
                  will_buy_on_return_visit
        FROM ${training_input.SQL_TABLE_NAME};;}}
```

CREATE OR REPLACE MODEL, a BQML function to make ML model

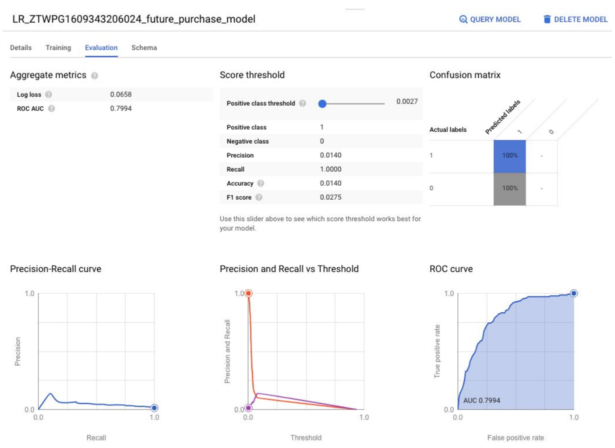
${SQL_TABLE_NAME}: LookML syntax for the current PDT

OPTIONS: The two important ones are **model_type** and **labels.**
→ How do we know which options to use? That's the job of a data scientist (not supported on chat)

STEP 2 (cont): MODEL TABLE

The model table will be made and saved in Looker's schema for PDT (usually `looker_scratch`):

- **The model table can not be queried in a Looker explore thread** (*"Model XXX cannot be scanned as a table."* or *"invalidQuery: Name xxx not found inside model*) → Use `datagroup_trigger` or `sql_trigger_value` so Looker uses the generator to build the model table
- Information about the "model" is available inside BQ
- We use this model table to evaluate the performance of the model and to make predictions on other tables



LR_ZTWPG1609343206024_future_purchase_model

Aggregate metrics

Log loss	0.0658
ROC AUC	0.7994

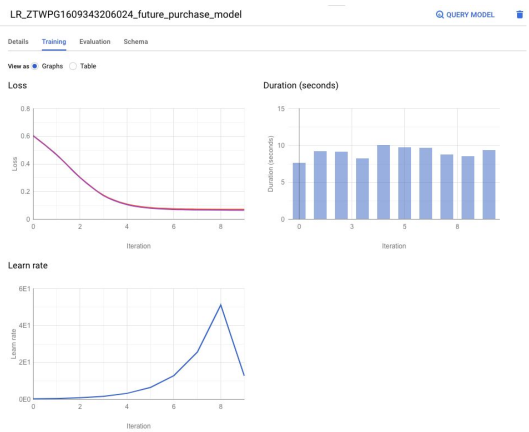
Score threshold

Positive class threshold	0.0027
Positive class	1
Negative class	0
Precision	0.0140
Recall	1.0000
Accuracy	0.0140
F1 score	0.0275

Use this slider above to see which score threshold works best for your model.

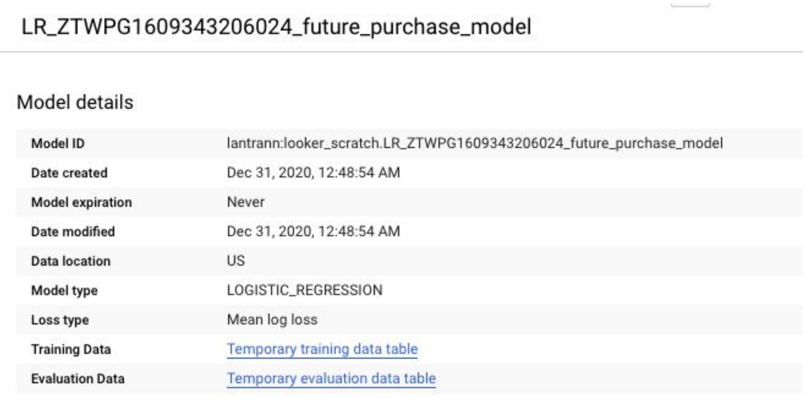
Confusion matrix

Actual labels \ Predicted labels	1	0
1	100%	-
0	100%	-



LR_ZTWPG1609343206024_future_purchase_model

View as ● Graphs ○ Table



LR_ZTWPG1609343206024_future_purchase_model

Model details

Model ID	lantrann:looker_scratch.LR_ZTWPG1609343206024_future_purchase_model
Date created	Dec 31, 2020, 12:48:54 AM
Model expiration	Never
Date modified	Dec 31, 2020, 12:48:54 AM
Data location	US
Model type	LOGISTIC_REGRESSION
Loss type	Mean log loss
Training Data	Temporary training data table
Evaluation Data	Temporary evaluation data table

STEP 3: EVALUATE THE MODEL - ML.EVALUATE

Make a LookML view, and an explore to run the query. The table doesn't have to be a PDT because we run it on to fly.

Calling ***ML.EVALUATE*** will generate a table consisting of the following columns, used to evaluate the performance of the model

```
view: model_evaluation {
  derived_table: {
    sql: SELECT *
         FROM ML.EVALUATE(MODEL
${future_purchase_model.SQL_TABLE_NAME},
(SELECT * FROM ${testing_input.SQL_TABLE_NAME})) ;;
  }

dimension: roc_auc {}
dimension: log_loss {}
dimension: accuracy {}
dimension: recall {}
dimension: f1_score {}
dimension: precision {}
}
```

ML.EVALUATE is a BQML function

ML.EVALUATE(MODEL <model_table>, <testing_data>)

Model table: *${future_purchase_model.SQL_TABLE_NAME}* -- made in step 2

Testing data: *(SELECT * FROM ${testing_input.SQL_TABLE_NAME}*
Use the `testing table` in step 1

STEP 4: USE THE MODEL TO PREDICT - ML.PREDICT

Make a LookML view to run the query.

Calling ***ML.PREDICT*** will make a table with predicted values for the label.

Define the primary key to join this view to the main explore (next step)

```
view: model_prediction {
  derived_table: {
    sql: SELECT * FROM ML.PREDICT(
   MODEL ${future_purchase_model.SQL_TABLE_NAME},
   (SELECT * FROM ${full_data.SQL_TABLE_NAME}));;
  }

dimension: predicted_will_buy_on_return_visit {type:number}

dimension: fullVisitorId {hidden:yes}

}
```

ML.PREDICT is a BQML function

ML.EVALUATE(MODEL <model_table>, <data_to_predict>)

Model table: *${future_purchase_model.SQL_TABLE_NAME}* -- made in step 2

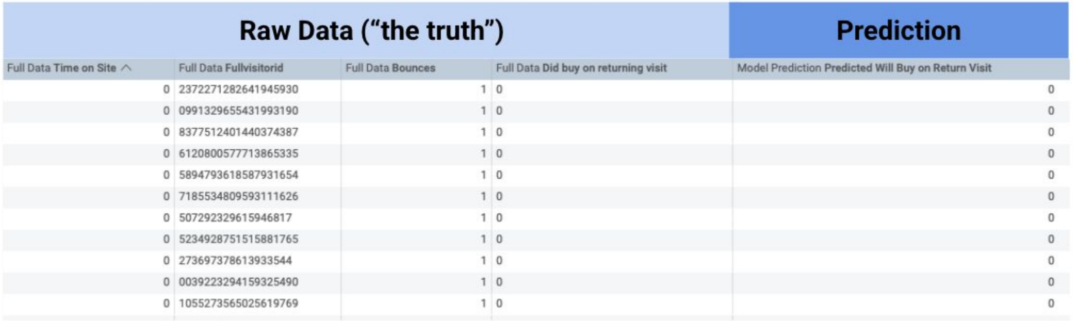
Data to predict: *(SELECT * FROM ${full_data.SQL_TABLE_NAME}* (For demonstration purposes, we use the `full_data` table).

STEP 5: JOIN BACK TO THE MAIN EXPLORE

`full_data` is the data table

`model_prediction` includes a column for prediction.

```
explore: full_data {
  label: "Data and Prediction"
  join: model_prediction {
    relationship: one_to_one
    type: inner
    sql_on: ${model_prediction.fullVisitorId} = ${full_data.fullvisitorid} ;;
  }
}
```



Raw Data ("the truth")				Prediction
Full Data Time on Site ^	Full Data Fullvisitorid	Full Data Bounces	Full Data Did buy on returning visit	Model Prediction Predicted Will Buy on Return Visit
0	2372271282641945930	1	0	0
0	0991329655431993190	1	0	0
0	8377512401440374387	1	0	0
0	6120800577713865335	1	0	0
0	5894793618587931654	1	0	0
0	7185534809593111626	1	0	0
0	507292329615946817	1	0	0
0	5234928751515881765	1	0	0
0	273697378613933544	1	0	0
0	0039223294159325490	1	0	0
0	1055273565025619769	1	0	0

Why do we join back?

From Guru: JOIN that prediction into existing Explores, Looks and dashboards within Looker - this way business users can easily access them within the environments they are used to using. It operationalizes the data science workflow.

BQML

How BigQuery does Machine Learning

BQML on LookML

5 steps to create and deploy a machine learning model

Next steps

Take-away and further resources

PERSONAL TAKE

1 Looker is not an optimal tool to do BQML

- Doing machine learning needs iteration (changing models, changing parameters). Each time we change these params, Looker will create a new “model table” as a PDT inside `looker_scratch` ⟶ There is a possibility of clogging the data warehouse.
- Suggested use case for BQML in Looker: a production model. Data scientists create and test models using other tools that allows quick iteration, decide on a finalized models/params, and then define the production model to be used in for end-users in Looker.

2 Other resources:

- Looker’s block: Google Analytics with BQML (same dataset with this presentation)
- Coursera: BigQuery for Machine Learning