BigQuery
Machine Learning cheat sheet

55 the data company

Created by Bastien Chappis

Main advantages of using BigQuery ML for model training, tuning and prediction:

- **Model training, evaluation and inference using SQL only**
- **Easy Access to data stored in BigQuery without the need to export it to local memory**
- **Centralized, ready to use model end-points, facilitating deployement and prototyping**
- **Distributed processing through BigQuery processing engine**

MANUAL DATA PREPROCESSING — TRANSFORM()

- Manual preprocessing can be added using the **TRANSFORM** clause
- Features listed in the **TRANSFORM** clause are the only one used to train the model
- Preprocessing parameters are saved within the model for prediction

Bucketing: Separate numerical values into buckets or groups	`ML.BUCKETIZE(numerical_expression, array_split_points[, exclude_boundaries])`
MinMax Scaler: Transform all the values into the range [0,1]	`ML.MIN_MAX_SCALER(numerical_expression) OVER()`
Standard Scaler: Transform all the values to center the mean to 0 and standard deviation to 1	`ML.STANDARD_SCALER(numerical_expression) OVER()`
Other transformations	Other preprocessing functions

```
CREATE MODEL `mydataset.mymodel`
    TRANSFORM(
      column1,
      ML.BUCKETIZE(column2, [1, 2, 3]) AS column2,
      ML.MIN_MAX_SCALER(column3) AS column3
      label
    )
    OPTIONS(...)
AS SELECT
    column1, column2, column3, label
FROM `mydataset.mytable
```

CREATE MODEL — CREATE MODEL

Create ML model: Generate a model to be stored in BigQuery	`CREATE MODEL <model_name>`
	`CREATE MODEL IF NOT EXISTS <model_name>`
	`CREATE OR REPLACE MODEL <model_name>`

Define ML model type: Type of model to be trained	`OPTIONS(`
	`MODEL_TYPE = 'LINEAR_REG'`
	`)`

Define Hyperparameters (optional): List of hyperparameters used for training	`OPTIONS(`
	list of hyperparameters
	`)`

Define training data: Select the label and features from the data-source used to train the model	`AS SELECT * FROM `my-project.my_dataset.my_table``

```
CREATE MODEL `mydataset.mymodel`
OPTIONS
  ( MODEL_TYPE='LINEAR_REG',
    LS_INIT_LEARN_RATE=.15,
    L1_REG=1,
    MAX_ITERATIONS=5 ) AS
SELECT
  column1, column2, column3, label
FROM `mydataset.mytable`
WHERE
  column4 < 10
```

```
SELECT
  *
FROM ML.PREDICT(
    MODEL `mydataset.mymodel`,
    (
    SELECT
      label,
      column1,
      column2
    FROM `mydataset.mytable`)
)
```

INFERENCE — ML.PREDICT(), ML.FORECAST(), ML.RECOMMEND()

Predict: Predict outcomes using the trained ML model	`SELECT *`
	`FROM ML.PREDICT(MODEL `my_project`.my_dataset.my_model,`
	`(SELECT * FROM input_data))`

Forecast (time series): Forecasts time series values from a trained ARIMA_PLUS or ARIMA model	`SELECT *`
	`FROM`
	`ML.FORECAST(MODEL `mydataset.my_model`,`
	`STRUCT(30 AS horizon, 0.8 AS confidence_level))`

Recommend: Generates a predicted rating for every user-item row combination for a matrix factorization model	`SELECT *`
	`FROM`
	`ML.RECOMMEND(MODEL `mydataset.mymodel`,`
	`(SELECT user,item FROM `mydataset.mytable`))`

DEFAULT DATA PREPROCESSING

Missing data	Null values are: • Replaced by the mean in numerical columns • Mapped to an additional one-hot-encoded category in non numerical columns
Standardization	Numerical columns are scaled using a standard scaler
One-hot encoding	Non-numerical non-array columns are one-hot-encoded

EVALUATION — ML.EVALUATE()

General evaluation: Return a single row containing common evaluation metrics corresponding to the trained model	`ML.EVALUATE(MODEL `mydataset.mymodel`)`
Classification models evaluation: • ROC curve • Confusion matrix	`ML.ROC_CURVE(MODEL `mydataset.mymodel`)` `ML.CONFUSION_MATRIX(MODEL `mydataset.mymodel`)`
Time series models evaluation:	`ML.ARMIMA_EVALUATE(MODEL `mydataset.mymodel`)`
Training iterations details:	`ML.TRAINING_INFO(MODEL `mydataset.mymodel`)`

```
SELECT
  *
FROM
  ML.EVALUATE(MODEL `mydataset.mymodel`)
```

MODEL LEARNING FLOW

Model Category	Model Type	Model Creation	Preprocessing	Feature and training infos	Evaluation	Inference	Explanation
Supervised learning	**Linear and logistic regression**	`LINEAR_REG, LOGISTIC_REG`			`ML.EVALUATE, ML.CONFUSION_MATRIX, ML.ROC_CURVE`	`ML.PREDICT`	`ML.EXPLAIN_PREDICT, ML.GLOBAL_EXPLAIN, ML.ADVANCED_WEIGHT`
	Deep Neural Network	`DNN_CLASSIFIER, DNN_REGRESSOR`					`ML.EXPLAIN_PREDICT, ML.GLOBAL_EXPLAIN, ML.FEATURE_IMPORTANCE`
	Boosted Tree	`BOOSTED_TREE_CLASSIFIER`					
Unsupervised learning	**Kmeans**	`KMEANS`	`TRANSFORM`	`ML.TRAINING_INFO`	`ML.EVALUATE`	`ML.PREDICT, ML.DETECT_ANOMALIES`	
	Matrix Factorization	`MATRIX_FATCTORIZATION`				`ML.RECOMMEND`	
	Autoencoder	`AUTOENCODER`		`ML.FEATURE_INFO`		`ML.PREDICT. ML.DETECT_ANOMALIES, ML.RECONSTRUCTION_LOSS`	
Time Series	**ARIMA +**	`ARIMA_PLUS`			`ML.EVALUATE, ML.ARIMA_EVALUATE`	`ML.FORECAST, ML.DETECT_ANOMALIES`	`ML.EXPLAIN_FORECAST`
Imported Models	**Tensorflow**	`TENSORFLOW`				`ML.PREDICT`	