Use Dataproc, BigQuery, and Apache Spark ML for Machine Learning

The BigQuery Connector (/hadoop/bigquery-connector) for Apache Spark (http://spark.apache.org/) allows Data Scientists to blend the power of BigQuery (/bigquery/what-is-bigquery)'s seamlessly scalable SQL engine with Apache Spark's Machine Learning (https://spark.apache.org/docs/2.0.0/ml-guide.html) capabilities. In this tutorial, we show how to use Dataproc, BigQuery and Apache Spark ML to perform machine learning on a dataset (http://spark.apache.org/docs/latest/sql-programming-guide.html#datasets-and-dataframes).

Objectives

Use linear regression to build a model of birth weight as a function of five factors:

1. gestation weeks
2. mother's age
3. father's age
4. mother's weight gain during pregnancy
5. Apgar score (https://en.wikipedia.org/wiki/Apgar_score)

BigQuery is used to prepare the linear regression input table, which is written to your Google Cloud Platform project. Python is used to query and manage data in BigQuery. The resulting linear regression table is accessed in Apache Spark, and Spark ML is used to build and evaluate the model. A Dataproc PySpark job is used to invoke Spark ML functions.

Costs

This tutorial uses billable components of Google Cloud Platform, including:

- Compute Engine
- Dataproc

- BigQuery

Use the Pricing Calculator (/products/calculator) to generate a cost estimate based on your projected usage. New Cloud Platform users might be eligible for a free trial (/free-trial).

Before you begin

A Dataproc cluster has the Spark components, including Spark ML, installed. To set up a Dataproc cluster and run the code in this example, you will need to do (or have done) the following:

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Cloud project.

   ★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

   Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboar

3. Enable the Dataproc, BigQuery, Compute Engine APIs.

   Enable the APIs (https://console.cloud.google.com/flows/enableapi?apiid=dataproc,bigquery.googlea

4. Install and initialize the Cloud SDK (/sdk/docs).

5. Create a Dataproc cluster (/dataproc/docs/guides/create-cluster) in your project. Your cluster should be running a Dataproc version (/dataproc/docs/concepts/dataproc-versions) with Spark 2.0 or higher, (includes machine learning libraries).

Create a Subset of BigQuery `natality` data

In this section, you create a dataset in your project, then create a table in the dataset to which you copy a subset of birth rate data from the publicly available natality (/bigquery/sample-tables) BigQuery dataset. Later in this tutorial you will use the subset data in this table to predict birth weight as a function of maternal age, paternal age, and gestation weeks.

You can create the data subset using the Google Cloud Console or running a Python script on your local machine.

ConsolePython (#python)

1. Create a dataset in your project.
   a. Go to the BigQuery Web UI (https://console.cloud.google.com/bigquery).
   b. In the left navigation pane, click your project name, then click **CREATE DATASET**.
   c. In the **Create dataset** dialog:
      i. For **Dataset ID**, enter "natality_regression".
      ii. For **Data location**, you can choose a location (/bigquery/docs/dataset-locations) for the dataset. The default value location is `US multi-region`. After a dataset is created, the location cannot be changed.
      iii. For **Default table expiration**, choose one of the following options:
         - **Never** (default): You must delete the table manually.
         - **Number of days**: The table will be deleted after the specified number days from its creation time.
      iv. For **Encryption**, choose one of the following options:
         - **Google-managed key** (default).
         - **Customer-managed key**: See Protecting data with Cloud KMS keys (/bigquery/docs/customer-managed-encryption).
      v. Click **Create dataset**.

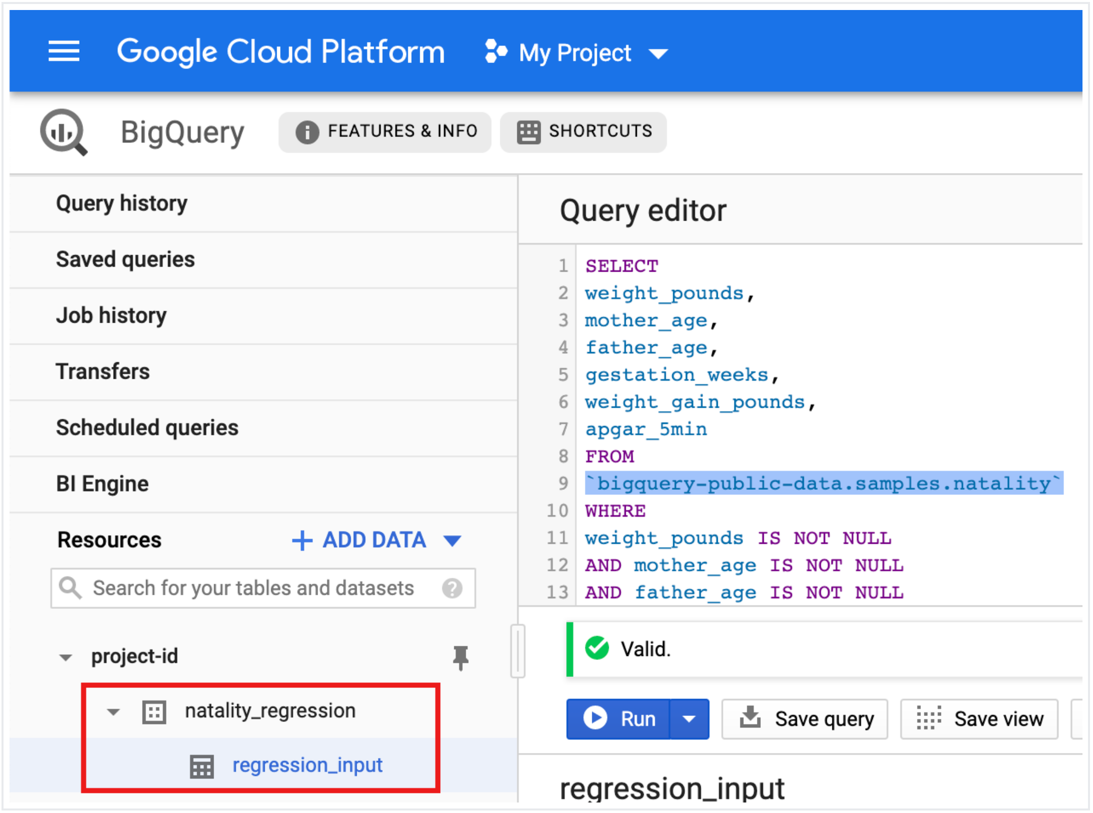
★ You cannot add a description or a label when you create a dataset using the Web UI. After the dataset is created, you can add a description (/bigquery/docs/managing-datasets#update-dataset-description), and add a label (/bigquery/docs/labels#creating_and_updating_dataset_labels).

2. Run a query against the public natality dataset, then save the query results in a new table in your dataset.

    a. Copy and paste the following query into the Query Editor, then click Run.

    ```
    SELECT
    weight_pounds,
    mother_age,
    father_age,
    gestation_weeks,
    weight_gain_pounds,
    apgar_5min
    FROM
    `bigquery-public-data.samples.natality`
    WHERE
    weight_pounds IS NOT NULL
    AND mother_age IS NOT NULL
    AND father_age IS NOT NULL
    AND gestation_weeks IS NOT NULL
    AND weight_gain_pounds IS NOT NULL
    AND apgar_5min IS NOT NULL
    ```

    b. After the query completes (after approximately 1 minute), click **SAVE RESULTS**, then select save options to save the results as a "regression_input" BigQuery table in the `natality_regression` dataset in your project.



```
SELECT
weight_pounds,
mother_age,
father_age,
gestation_weeks,
weight_gain_pounds,
apgar_5min
FROM
`bigquery-public-data.samples.natality`
WHERE
weight_pounds IS NOT NULL
AND mother_age IS NOT NULL
AND father_age IS NOT NULL
```

Run a linear regression

In this section, you'll run a PySpark linear regression by submitting the job to the Dataproc service using the Google Cloud Console or by running the `gcloud` command from a local terminal.

Consolegcloud command (#gcloud-c...

1. Copy and paste the following code into a new `natality_sparkml.py` file on your local machine.

```
"""Run a linear regression using Apache Spark ML.

In the following PySpark (Spark Python API) code, we take the following a

  * Load a previously created linear regression (BigQuery) input table
    into our Cloud Dataproc Spark cluster as an RDD (Resilient
    Distributed Dataset)
  * Transform the RDD into a Spark Dataframe
  * Vectorize the features on which the model will be trained
  * Compute a linear regression using Spark ML

"""

from __future__ import print_function
from pyspark.context import SparkContext
from pyspark.ml.linalg import Vectors
from pyspark.ml.regression import LinearRegression
from pyspark.sql.session import SparkSession
# The imports, above, allow us to access SparkML features specific to lin
# regression as well as the Vectors types.


# Define a function that collects the features of interest
# (mother_age, father_age, and gestation_weeks) into a vector.
# Package the vector in a tuple containing the label (`weight_pounds`) fo
# row.
def vector_from_inputs(r):
  return (r["weight_pounds"], Vectors.dense(float(r["mother_age"]),
                                            float(r["father_age"]),
                                            float(r["gestation_weeks"]),
                                            float(r["weight_gain_pounds"]
                                            float(r["apgar_5min"])))

sc = SparkContext()
spark = SparkSession(sc)

# Read the data from BigQuery as a Spark Dataframe.
natality_data = spark.read.format("bigquery").option(
    "table", "natality_regression.regression_input").load()
# Create a view so that Spark SQL queries can be run against the data.
natality_data.createOrReplaceTempView("natality")

# As a precaution, run a query in Spark SQL to ensure no NULL values exis
sql_query = """
SELECT *
from natality
where weight_pounds is not null
and mother_age is not null
and father_age is not null
and gestation_weeks is not null
"""
clean_data = spark.sql(sql_query)
```

```
# Create an input DataFrame for Spark ML using the above function.
training_data = clean_data.rdd.map(vector_from_inputs).toDF(["label",
                                                             "features"])
training_data.cache()

# Construct a new LinearRegression object and fit the training data.
lr = LinearRegression(maxIter=5, regParam=0.2, solver="normal")
model = lr.fit(training_data)
# Print the model summary.
print("Coefficients:" + str(model.coefficients))
print("Intercept:" + str(model.intercept))
print("R^2:" + str(model.summary.r2))
model.summary.residuals.show()
```

2. Copy the local `natality_sparkml.py` file to a Cloud Storage bucket in your project.

```
gsutil cp natality_sparkml.py gs://bucket-name
```

★ Instead of copying the file to a user bucket in your project, you can copy it to the staging bucket (/dataproc/docs/concepts/configuring-clusters/staging-bucket) that Dataproc created when you created your cluster.

3. Run the regression from the Dataproc Submit a job (https://console.cloud.google.com/dataproc/jobsSubmit) page.

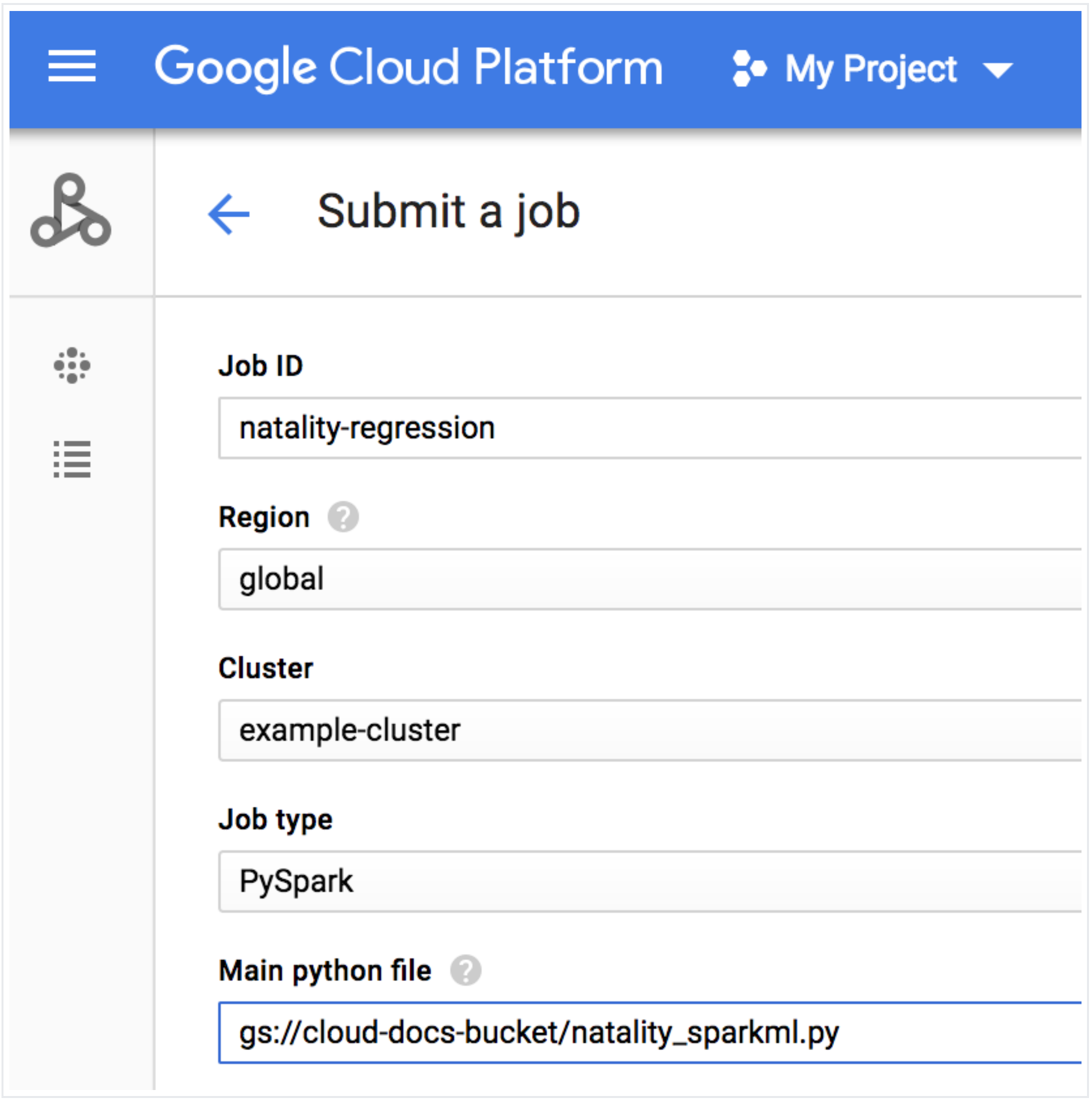
   a. In the **Main python file** field, insert the `gs://` URI of the Cloud Storage bucket where your copy of the `natality_sparkml.py` file is located.

   b. Select `PySpark` as the **Job type**.

   c. Insert `gs://spark-lib/bigquery/spark-bigquery-latest.jar` in the **Jar files** field. This makes the spark-bigquery-connector available to the PySpark application at runtime to allow it to read BigQuery data into a Spark DataFrame.

   d. Fill in the **Job ID**, **Region**, and **Cluster** fields.

   e. Click **Submit** to run the job on your cluster.



Submit a job

Job ID
natality-regression

Region
global

Cluster
example-cluster

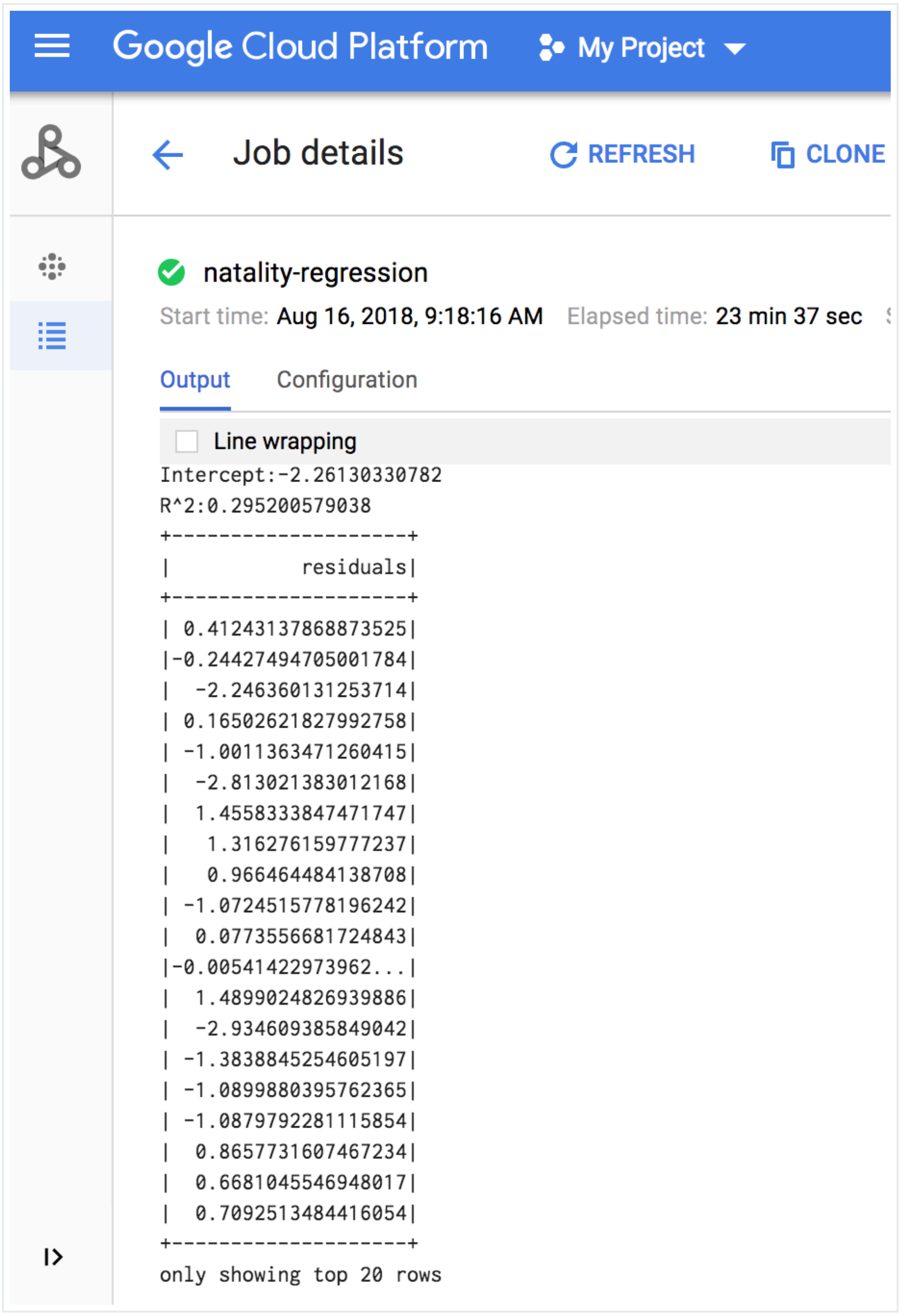
Job type
PySpark

Main python file
gs://cloud-docs-bucket/natality_sparkml.py

When the job completes, the linear regression output model summary appears in the Dataproc Job details window.

Model Summary Terminology:

- **R^2:** a measure for how much of the "signal" the model captures.
- **Residuals:** the difference between the prediction of the model and the actual value for each point that was used to fit the model.



```
Intercept:-2.26130330782
R^2:0.295200579038
+--------------------+
|           residuals|
+--------------------+
| 0.41243137868873525|
|-0.24427494705001784|
|  -2.246360131253714|
| 0.16502621827992758|
| -1.0011363471260415|
|  -2.813021383012168|
|  1.4558333847471747|
|   1.316276159777237|
|   0.966464484138708|
| -1.0724515778196242|
|  0.0773556681724843|
|-0.00541422973962...|
|  1.4899024826939886|
|  -2.934609385849042|
| -1.3838845254605197|
| -1.0899880395762365|
| -1.0879792281115854|
|  0.8657731607467234|
|  0.6681045546948017|
|  0.7092513484416054|
+--------------------+
only showing top 20 rows
```

Cleaning up

After you've finished the Use , , and Spark ML for Machine Learning tutorial, you can clean up the resources that you created on Google Cloud so they won't take up quota and you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

Deleting the project

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

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.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to the Manage resources page (https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project that you want to delete and then click **Delete** 🗑.
3. In the dialog, type the project ID and then click **Shut down** to delete the project.

Deleting the Dataproc cluster

See Delete a cluster (/dataproc/docs/guides/manage-cluster#delete_a_cluster).

Last updated 2020-07-23 UTC.