Google Cloud

Effective ML

Advanced ML with TensorFlow on GCP

End-to-End Lab on Structured Data ML

Production ML Systems

Image Classification Models

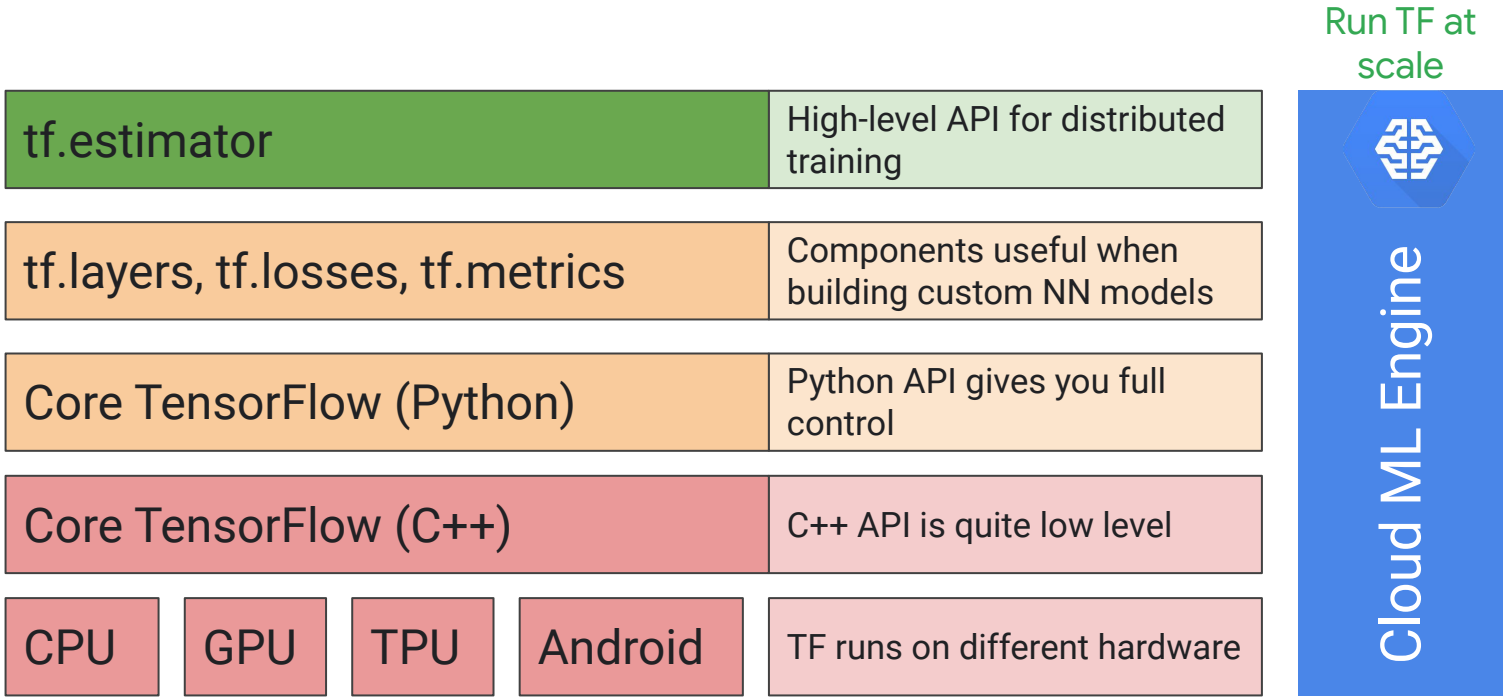
Sequence Models

Recommendation Systems

Steps involved in doing ML on GCP

1. Explore the dataset
2. Create the dataset
3. Build the model
4. Operationalize the model

You use distributed TensorFlow on Cloud ML Engine



Many machine learning frameworks can handle toy problems

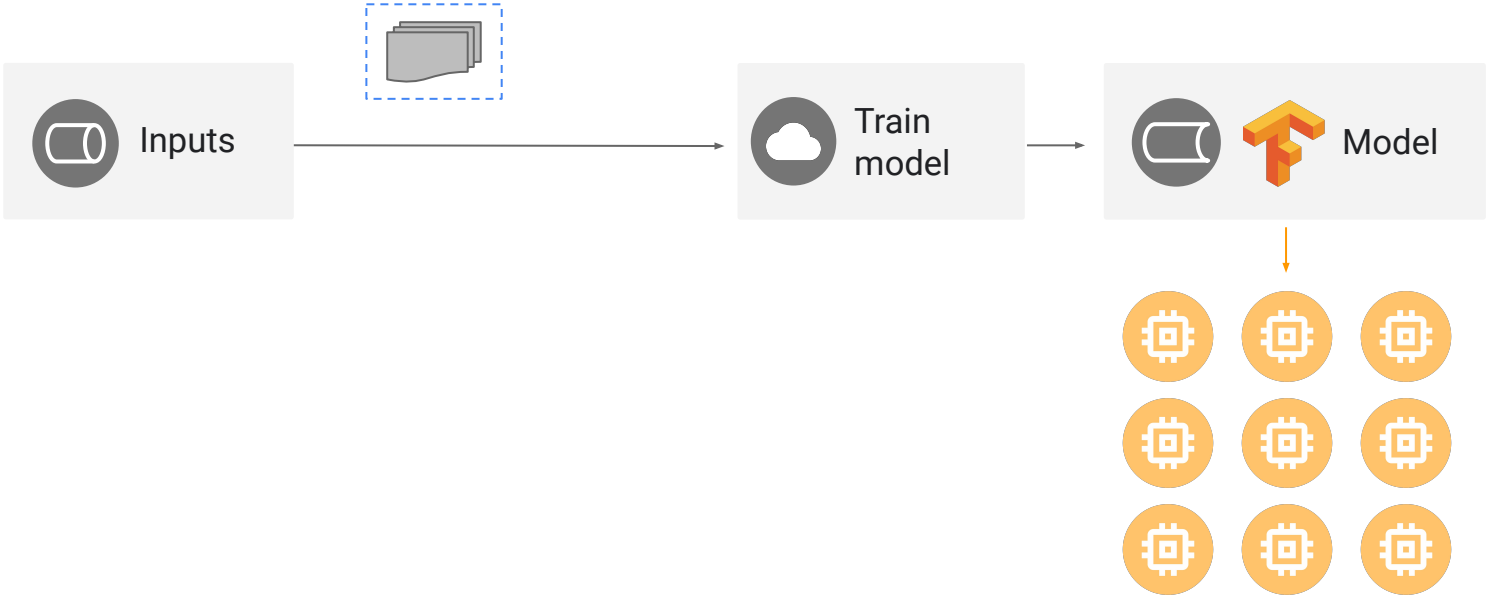
To build effective ML, you need:

Big Data

Feature Engineering

Model Architectures

As your data size increases, batching and distribution become important



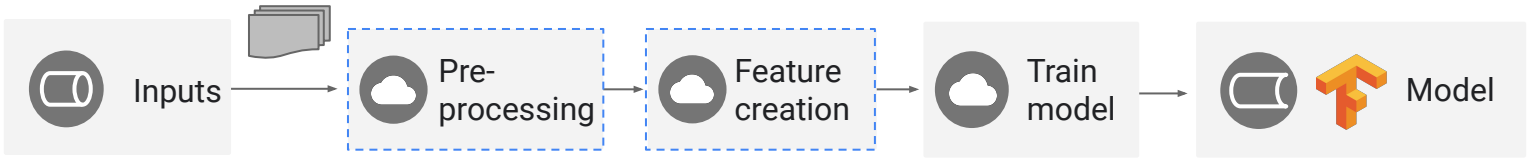
To build effective ML, you need:

Big Data

Feature Engineering

Model Architectures

Input necessary transformations



To build effective ML, you need:

Big Data

Feature Engineering

Model Architectures

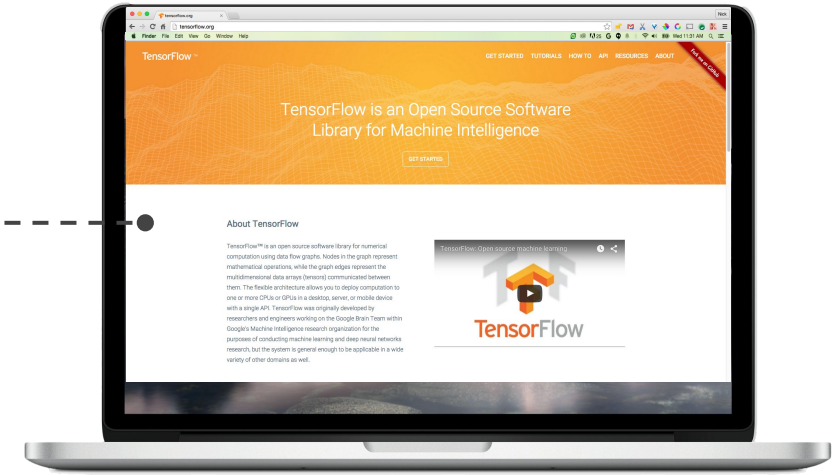
Sharing our tools with researchers and developers around the world

Released in Nov. 2015

#1

repository

for “machine learning” category on GitHub



To build effective ML, you need:

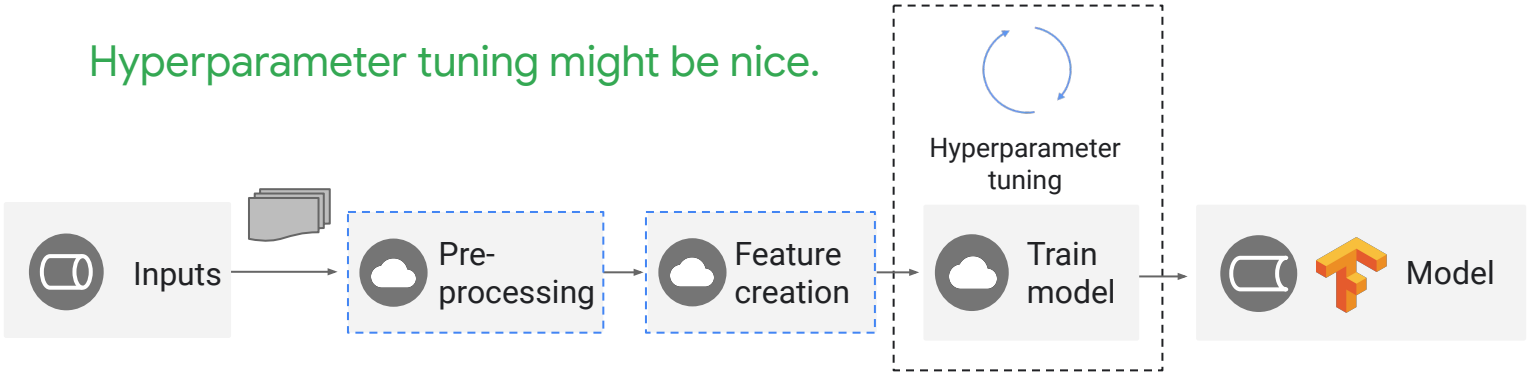
Big Data

Feature Engineering

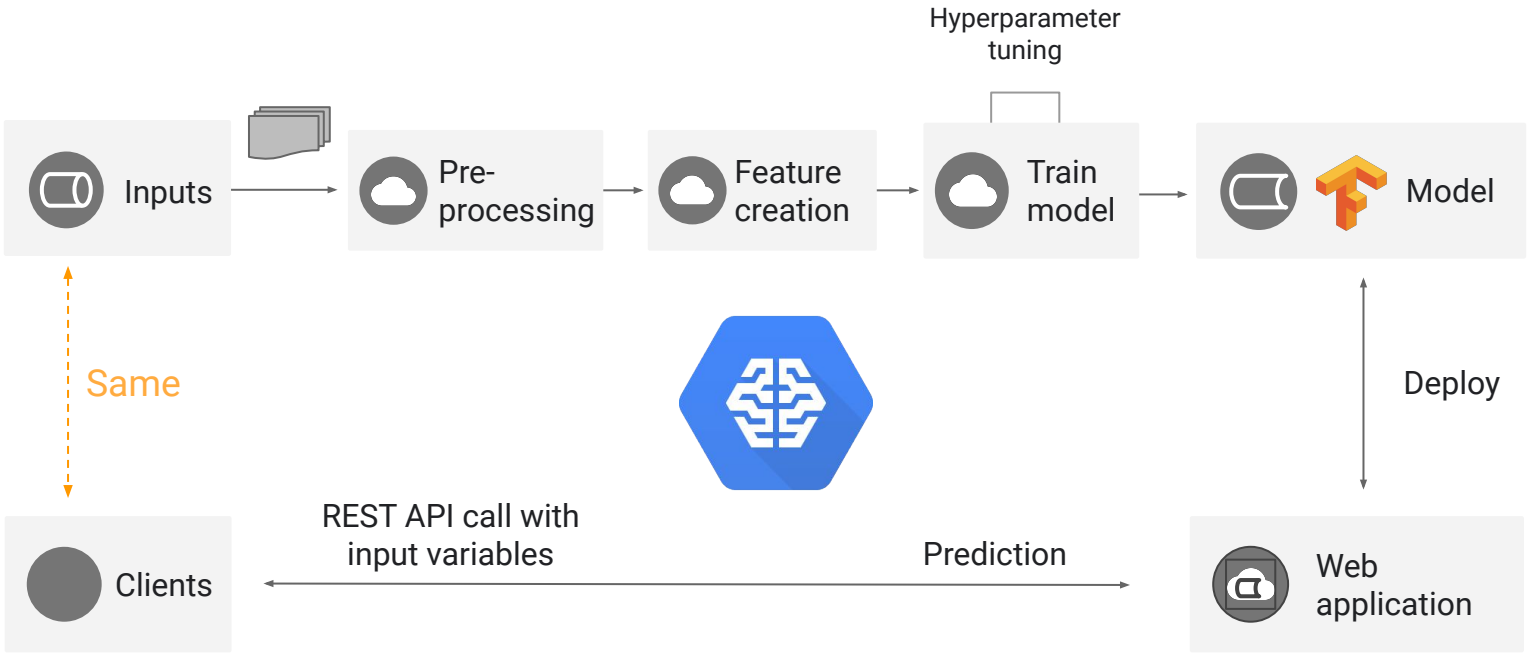
Model Architectures

What else does an ML framework need to provide?

Hyperparameter tuning might be nice.



Cloud machine learning: Repeatable, scalable, tuned



In Cloud Datalab, start locally on a sampled dataset

Google Cloud Datalab tfclassic (unsaved changes)

Notebook Add Code Add Markdown Delete Move Up Move Down Run Clear Reset Session Widgets Navigation Help

2b. Working with low-level TensorFlow

This notebook is Lab2b of CPB 102, Google's course on Machine Learning using Cloud ML.

In this notebook, we will work with relatively low-level TensorFlow functions to implement a linear regression model. We will use this notebook to demonstrate early stopping -- a technique whereby training is stopped once the error on the validation dataset starts to increase.

```
import datalab.bigquery as bq
import tensorflow as tf
import pandas as pd
import numpy as np
import shutil
```

Code to read data and compute error is the same as Lab2a.

```
def read_dataset(filename):
  return pd.read_csv(filename, header=None, names=['pickuplon','pickuplat','dropofflon','dropofflat','passengers','fare_amount'])

df_train = read_dataset('../labs/taxi-train.csv')
df_valid = read_dataset('../labs/taxi-valid.csv')
df_test = read_dataset('../labs/taxi-test.csv')
df_train[:5]
```

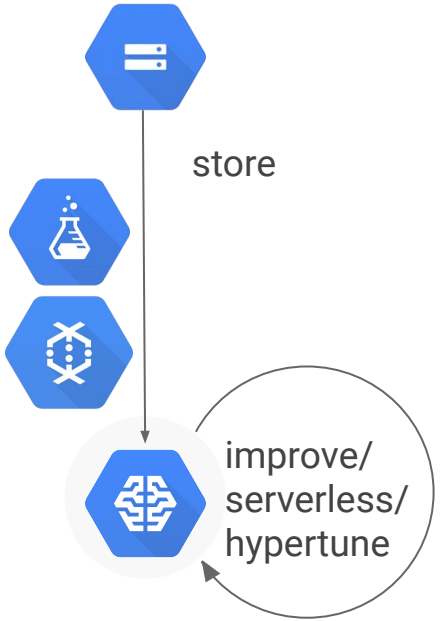
```
FEATURE_COLS = np.arange(0,5)
TARGET_COL   = 'fare_amount'
```

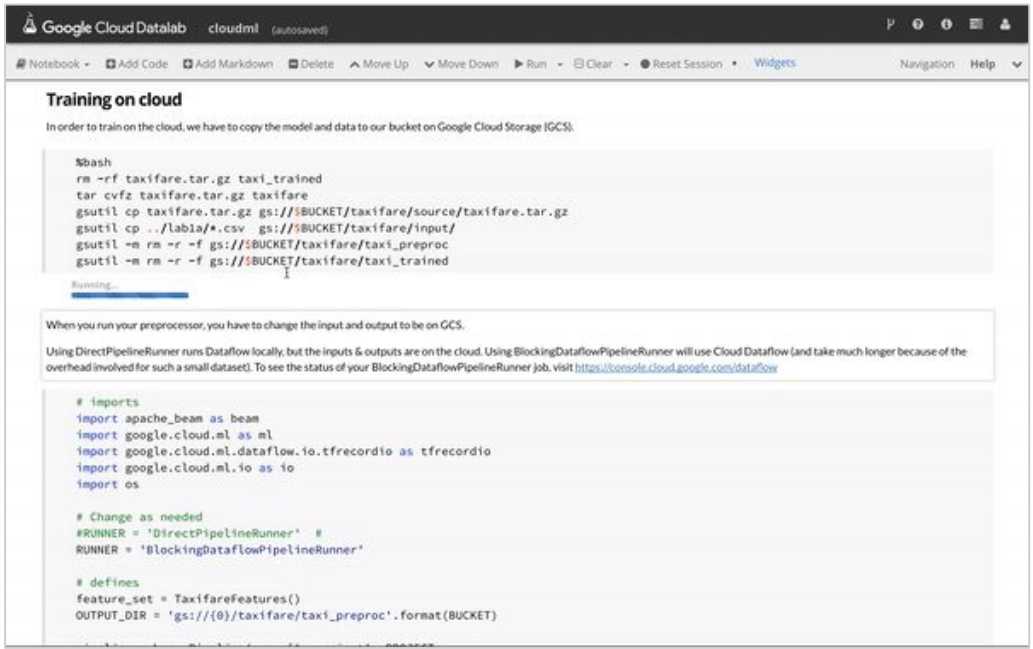
```
def compute_rmse(actual, predicted):
  return np.sqrt(np.mean((actual-predicted)**2))

def print_rmse(model):
  print "Train RMSE = {0}".format(compute_rmse(df_train[TARGET_COL], model.predict(df_train.iloc[:,FEATURE_COLS].values)))
  print "Valid RMSE = {0}".format(compute_rmse(df_valid[TARGET_COL], model.predict(df_valid.iloc[:,FEATURE_COLS].values)))
```

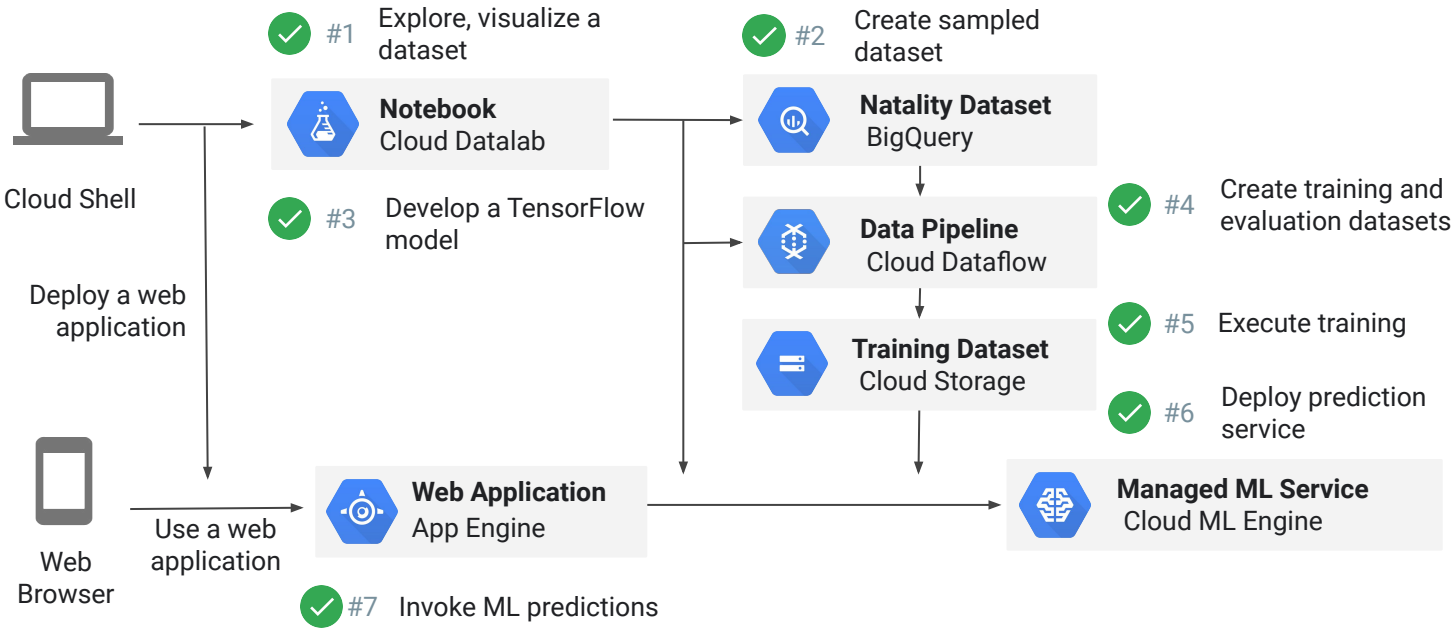
Linear Regression

Then, scale it out to GCP using serverless technology





The end-to-end machine learning set of labs



cloud.google.com