Google Cloud Machine Learning Engine brings the power and flexibility of TensorFlow to the cloud. You can use its components to select and extract features from your data, train your machine learning models, and get predictions using the managed resources of Google Cloud Platform.

Prerequisite:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Cloud Machine Learning Engine and Compute Engine APIs.

Set up Cloud Shell for working with Cloud ML Engine.:

1. Open the Google Cloud Platform Console.
2. Click the **Activate Google Cloud Shell** button at the top of the console window.
3. Configure the gcloud command-line tool to use your selected project.
   1. Gcloud config set project engaged-context-160605
4. Download and install TensorFlow.
   1. Pip download tensorflow
   2. Pip install –user –U tensorflow
5. Verify the Google Cloud SDK components
   1. Gcloud ml-engine models first ->Returns “Listed 0 items”

Walk through a sample that uses a census dataset to:

* Create a TensorFlow trainer and validate it locally.
* Run your trainer on a single worker instance in the cloud.
* Run your trainer as a distributed training job in the cloud.
* Deploy a model to support prediction.
* Request an online prediction and see the response.
* Request a batch prediction.

The sample builds a wide and deep model for predicting income category based on United States Census Income Dataset.

Wide and deep models use deep neural nets (DNNs) to learn high level abstractions about complex features or interactions between such features. These models then combine the outputs from the DNN with a linear regression performed on simpler features. This provides a balance between power and speed that is effective on many structured data problems.

The sample defines the model using TensorFlow's prebuilt DNNCombinedLinearClassifier class, and need only define the data transformations particular to our dataset before assigning these (potentially) transformed features to either the DNN or the linear portion of the model.

1. Download the data to a local file directory and set variables that point to the downloaded data files.
   1. Mkdir data
   2. Gsutil –m cp gs://cloudml-public/census/data/\* data/
2. Set the TRAIN\_DATA AND EVAL\_DATA variables to your local file paths. For example, the following commands set the variables to local paths.
   1. TRAIN\_DATA=$(pwd)/data/adult.data.csv
   2. EVAL\_DATA=$(pwd)/data/adult.test.csv

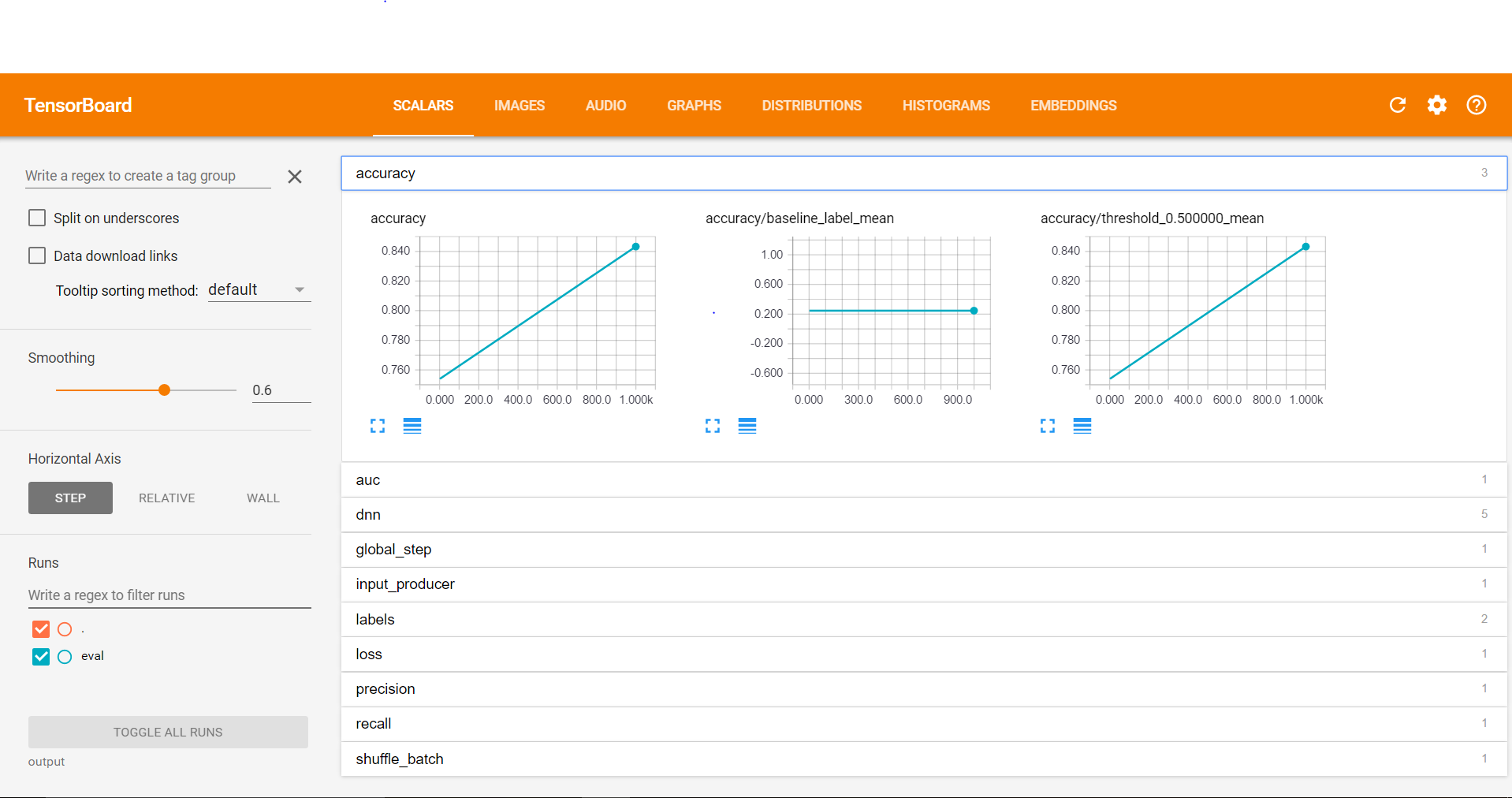
A local trainer loads your Python training program and starts a training process in an environment that's similar to that of a live Cloud ML Engine cloud training job.

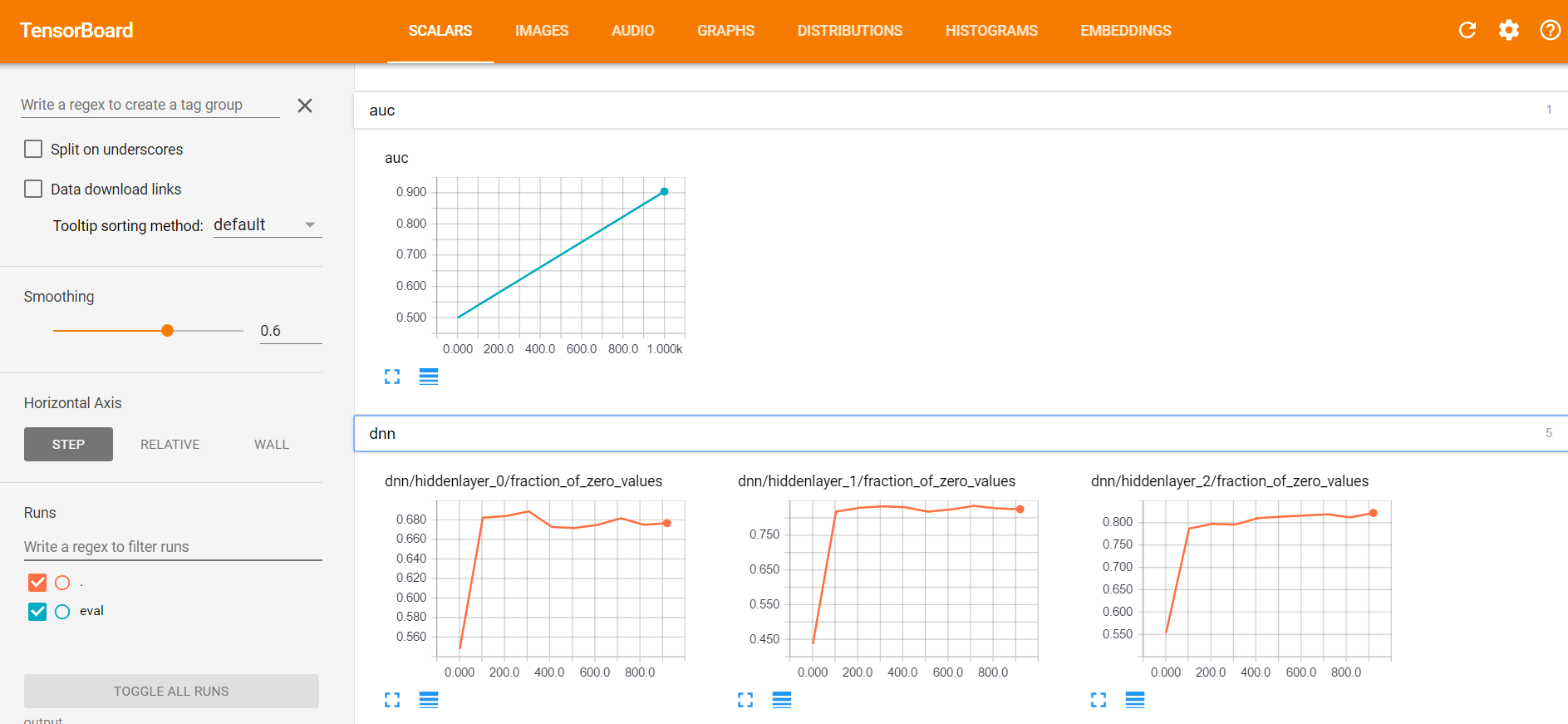
1. Specify an output directory and set a MODEL\_DIR variable.
   1. MODEL\_DIR=output
2. It's a good practice to delete the contents of the output directory in case data remains from a previous training run.
   1. Rm –rf $MODEL\_DIR
3. To run your training locally, run the following command:

gcloud ml-engine local train \  
    --module-name trainer.task \  
    --package-path trainer/ \  
    -- \  
    --train-files $TRAIN\_DATA \  
    --eval-files $EVAL\_DATA \  
    --train-steps 1000 \  
    --job-dir $MODEL\_DIR

1. Inspect the summary logs using Tensorboard

python -m tensorflow.tensorboard --logdir=output --port=8080





Run a local trainer in distributed mode

1. Specify an output directory and set a MODEL\_DIR variable.

MODEL\_DIR=output

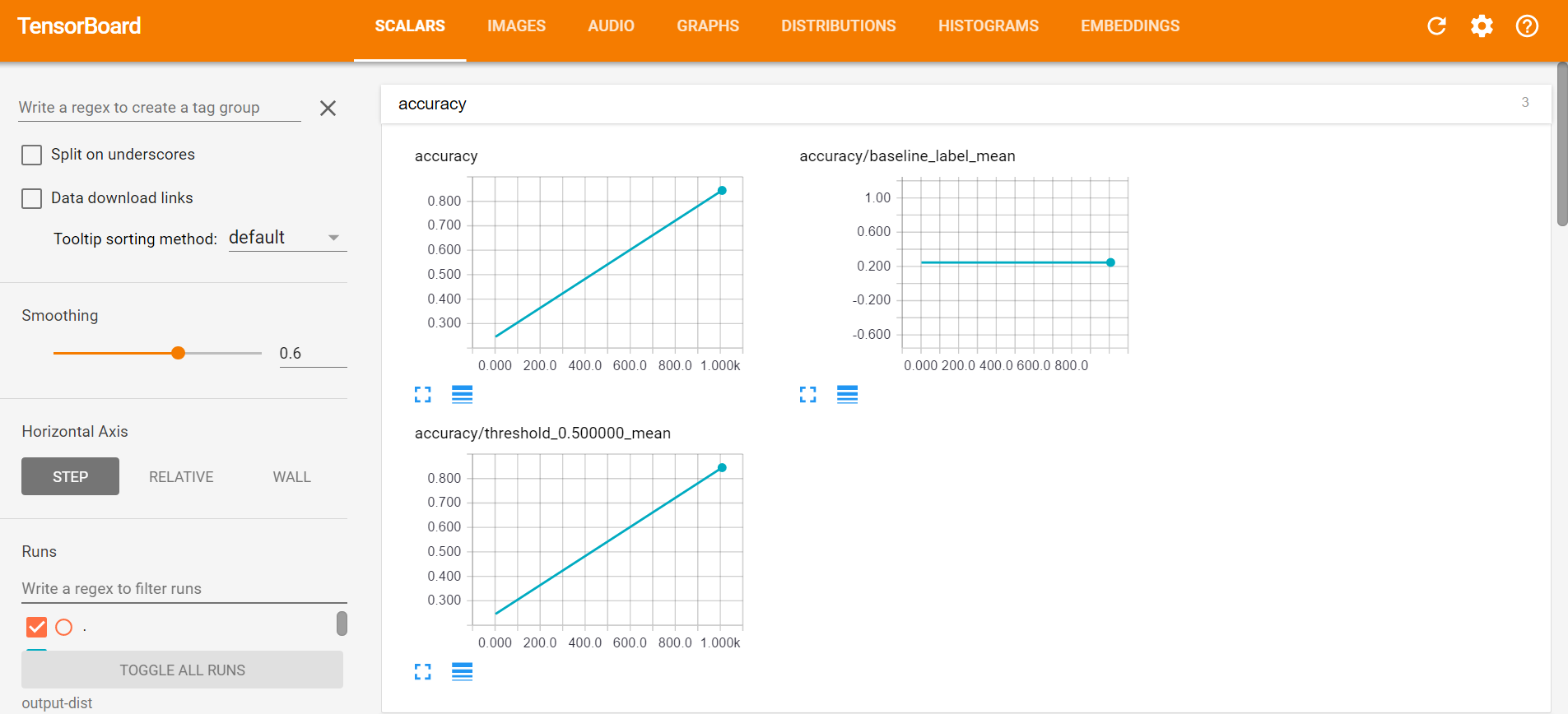
1. delete the contents of the output directory

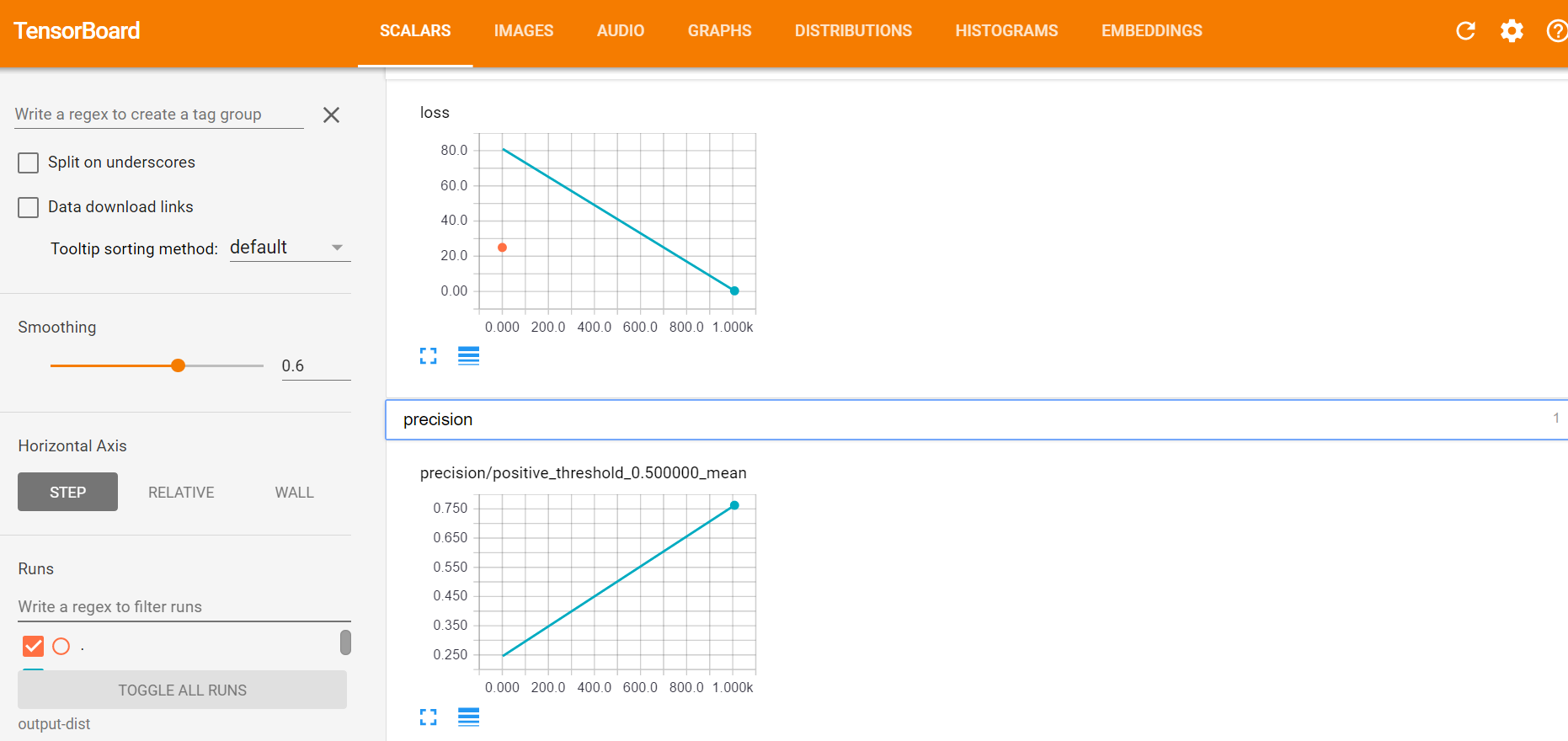
rm -rf $MODEL\_DIR

1. Run the local train command using the --distribued option

gcloud ml-engine local train \  
    --module-name trainer.task \  
    --package-path trainer/ \  
    --distributed \  
    -- \  
    --train-files $TRAIN\_DATA \  
    --eval-files $EVAL\_DATA \  
    --train-steps 1000 \  
    --job-dir $MODEL\_DIR

1. ls -R output-dist/





Set up your Cloud Storage bucket

The Cloud ML Engine services need to access Cloud Storage locations to read and write data during model training and batch prediction.

Create a [Google Cloud Storage](https://cloud.google.com/storage) bucket for reading and writing data during model training and batch prediction:

1. Set a name for your new bucket.

PROJECT\_ID=$(gcloud config list project --format "value(core.project)")  
BUCKET\_NAME=${PROJECT\_ID}-mlengine

2.echo $BUCKET\_NAME

3. Select a region for your bucket and set a `REGION` environment variable.

REGION=us-central1

1. Create the new bucket:

gsutil mb -l $REGION gs://$BUCKET\_NAME

Next, upload the data files to your Cloud Storage bucket.

1.gsutil cp -r data gs://$BUCKET\_NAME/data

2. TRAIN\_DATA=gs://$BUCKET\_NAME/data/adult.data.csv  
EVAL\_DATA=gs://$BUCKET\_NAME/data/adult.test.csv

3.gsutil cp ../test.json gs://$BUCKET\_NAME/data/test.json

4.TEST\_JSON=gs://$BUCKET\_NAME/data/test.json

Run a single-instance trainer in the cloud:

1.JOB\_NAME=census\_single\_1

2.OUTPUT\_PATH=gs://$BUCKET\_NAME/$JOB\_NAME

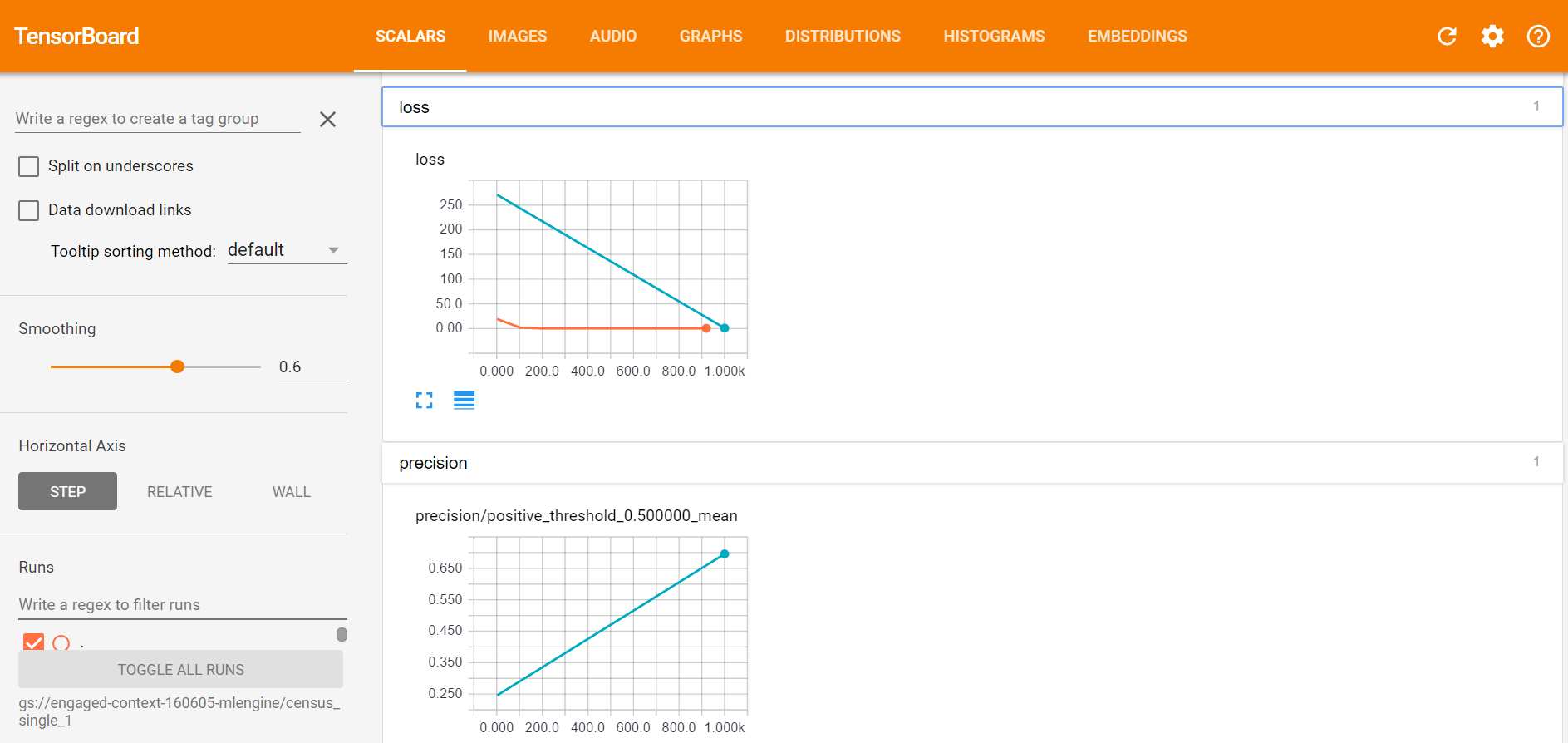
3. gcloud ml-engine jobs submit training $JOB\_NAME \  
--job-dir $OUTPUT\_PATH \  
--runtime-version 1.0 \  
--module-name trainer.task \  
--package-path trainer/ \  
--region $REGION \  
-- \  
--train-files $TRAIN\_DATA \  
--eval-files $EVAL\_DATA \  
--train-steps 1000 \  
--verbose-logging true

monitor the progress of your trainer by watching the command-line output or in **ML Engine** > **Jobs** on [Google Cloud Platform Console](https://console.cloud.google.com/ml/jobs?_ga=1.2862768.1047576418.1488688364).

4.gsutil ls -r $OUTPUT\_PATH

5. Inspect the Stackdriver logs

6. Inspect the summary logs using Tensorboard



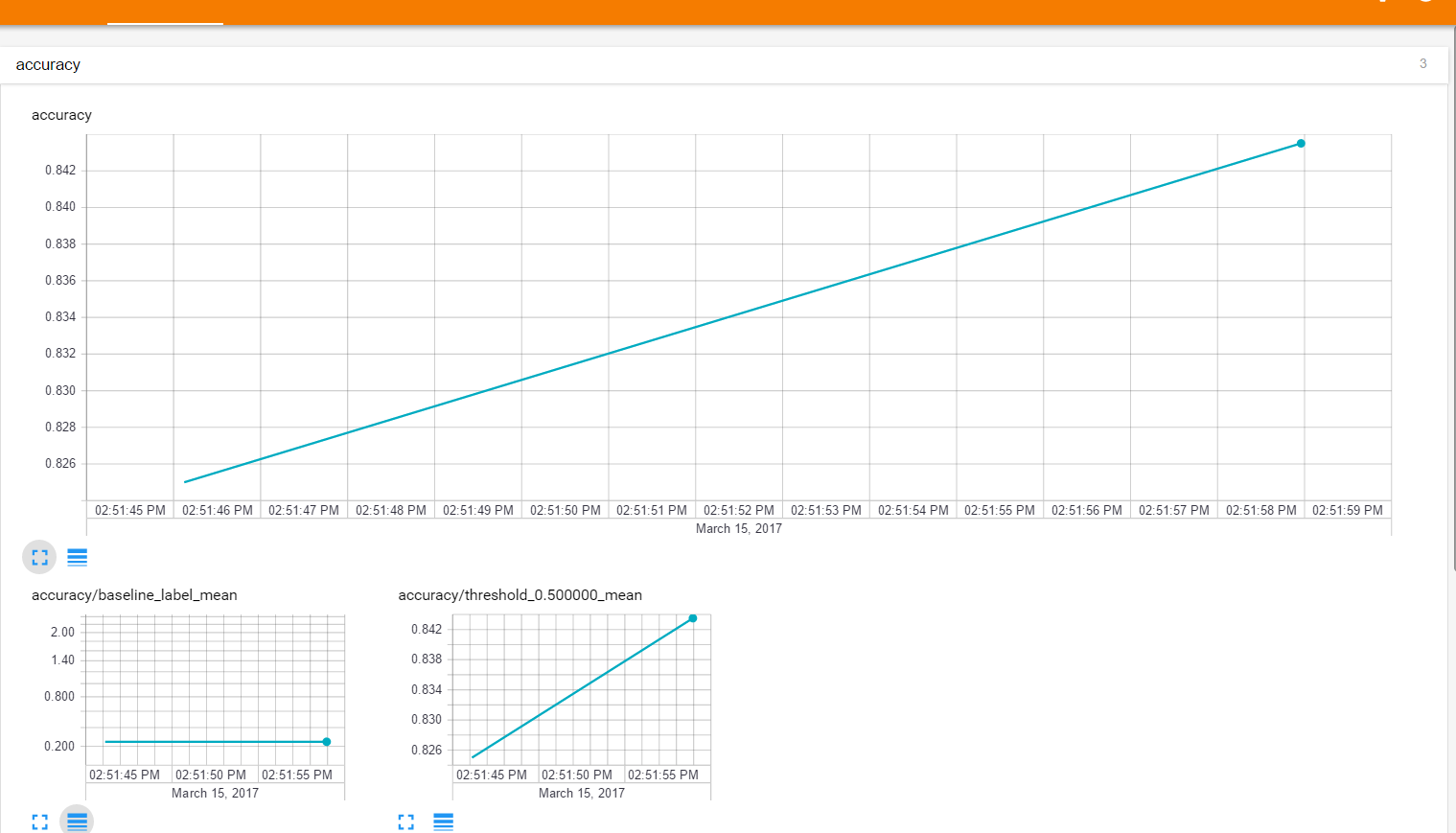


Run distributed training in the cloud:

1.JOB\_NAME=census\_dist\_1

2. OUTPUT\_PATH=gs://$BUCKET\_NAME/$JOB\_NAME

3. gcloud ml-engine jobs submit training $JOB\_NAME \  
    --job-dir $OUTPUT\_PATH \  
    --runtime-version 1.0 \  
    --module-name trainer.task \  
    --package-path trainer/ \  
    --region $REGION \  
    --scale-tier STANDARD\_1 \  
    -- \  
    --train-files $TRAIN\_DATA \  
    --eval-files $EVAL\_DATA \  
    --train-steps 1000 \  
    --verbose-logging true



Deploy a model to support prediction

1.MODEL\_NAME=census

2. gcloud ml-engine models create $MODEL\_NAME --regions=$REGION

3. OUTPUT\_PATH=gs://$BUCKET\_NAME/census\_dist\_1

4. gsutil ls -r $OUTPUT\_PATH/export

5. MODEL\_BINARIES=gs://$BUCKET\_NAME/census\_dist\_1/export/Servo/**1487877383942**/

6. gcloud ml-engine versions create v1 \  
--model $MODEL\_NAME \  
--origin $MODEL\_BINARIES \  
--runtime-version 1.0

7. gcloud ml-engine models list

Send a prediction request to a deployed model:

gcloud ml-engine predict \  
--model $MODEL\_NAME \  
--version v1 \  
--json-instances \  
../test.json

response:

CLASSES LOGISTIC LOGITS PROBABILITIES

0 [0.003707568161189556] [-5.593664646148682] [0.9962924122810364, 0.003707568161189556]

Submit a batch prediction job:

1. JOB\_NAME=census\_prediction\_1
2. OUTPUT\_PATH=gs://$BUCKET\_NAME/$JOB\_NAME
3. gcloud ml-engine jobs submit prediction $JOB\_NAME \  
   --model $MODEL\_NAME \  
   --version v1 \  
   --data-format TEXT \  
   --region $REGION \  
   --input-paths $TEST\_JSON \  
   --output-path $OUTPUT\_PATH/predictions
4. Check the progress of the job and wait for it to finish:

gcloud ml-engine jobs describe $JOB\_NAME

You should see state: SUCCEEDED once the job completes; this may take several minutes.

1. Read the output summary.

gsutil cat $OUTPUT\_PATH/predictions/prediction.results-00000-of-00001

{"probabilities": [0.9962924122810364, 0.003707568161189556], "logits": [-5.593664646148682], "classes": 0, "logistic": [0.003707568161189556]}

Compared to online prediction, batch prediction:

* Is slower for this small number of instances (but is more suitable for large numbers of instances).
* Could return output in a different order than the input (but the numeric index allows each output to be matched to its corresponding input instance; this is not necessary for online prediction since the outputs are returned in the same order as the original input instances).

After the predictions are available, the next step is usually to ingest these predictions into a database or data processing pipeline.

## **Clean up**

If you are done analyzing the output from your training and prediction runs, you can avoid incurring additional charges to your Google Cloud Platform account for the Cloud Storage directories used in this guide:

gsutil rm -r gs://$BUCKET\_NAME/$JOB\_NAME