Assignment 7: optional catchup assignment 2 - VERTEX AI - for midterm and quiz - this will catch up midterm.

a) Custom training job and prediction using managed datasets

Reference: https://codelabs.developers.google.com/codelabs/vertex-ai-custom-code-training#0

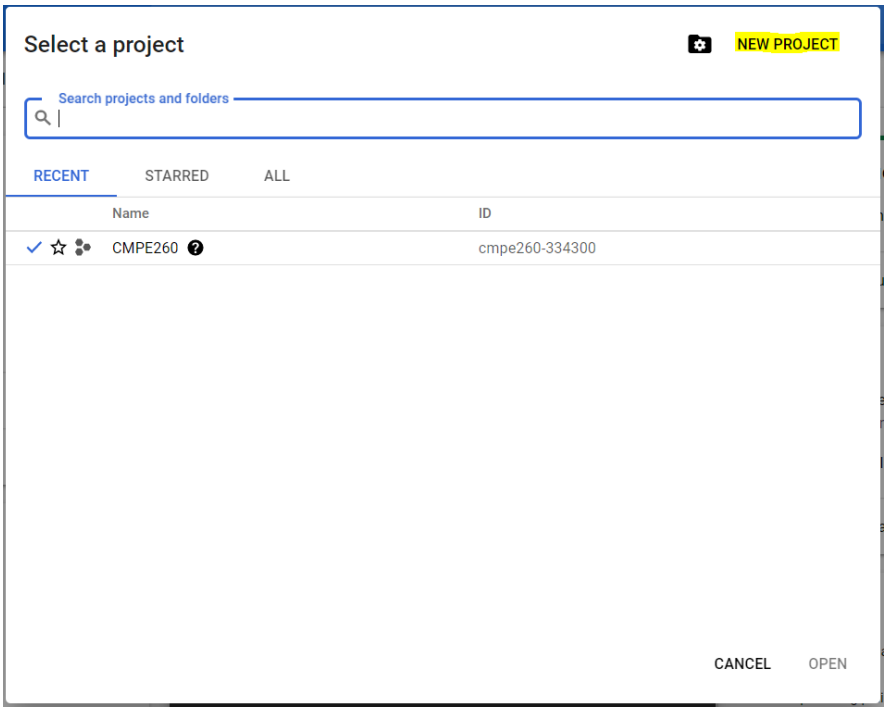
Objectives:

- Use Vertex AI to train and deploy a ML model
- Use Datasets for dataset creation and management, and custom model for training a Scikit Learn model.
- Deploy the trained model and get online predictions

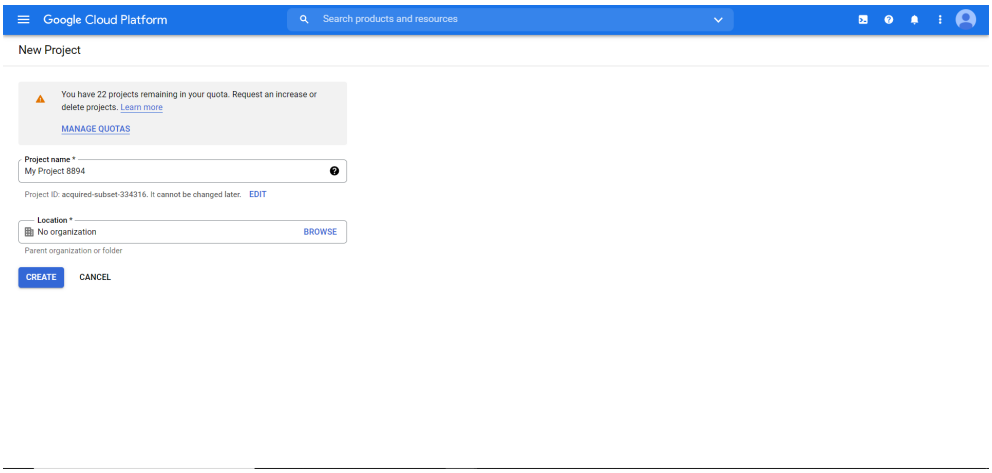
Setup the environment

Create a project

- To create a project, check if our Role has the resourcemanager.projects.create permission
- Login to Cloud Console.
- On the **Select organization** drop-down list at the top of the page, select the organization in which you want to create a project. If you are a free trial user, skip this step, as this list does not appear.
- Click **Create Project**.



- In the **New Project** window that appears, enter a project name and select a billing account as applicable. A project name can contain only letters, numbers, single quotes, hyphens, spaces, or exclamation points, and must be between 4 and 30 characters.



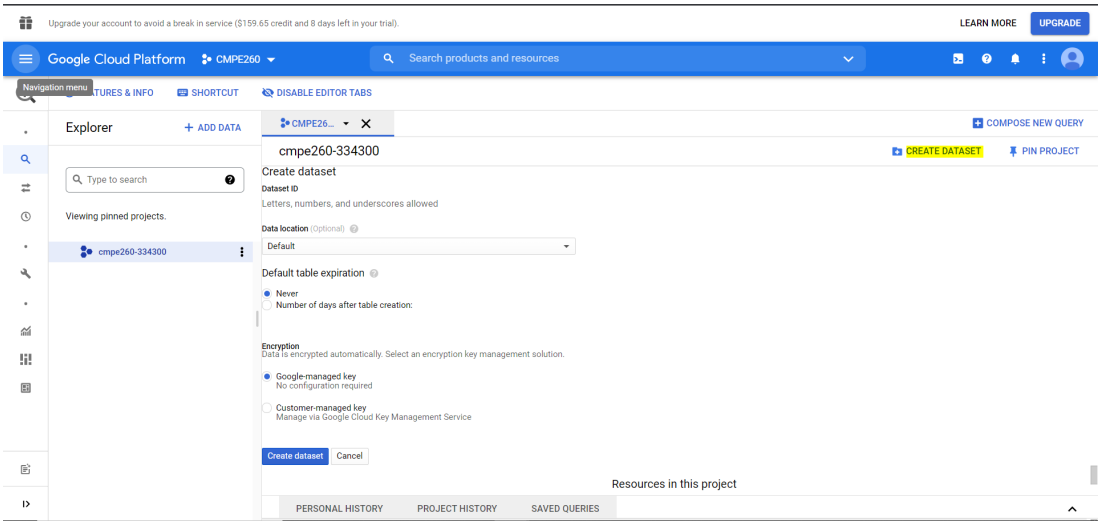
- Enter the parent organization or folder in the **Location** box. That resource will be the hierarchical parent of the new project.
- When you're finished entering new project details, click **Create**.

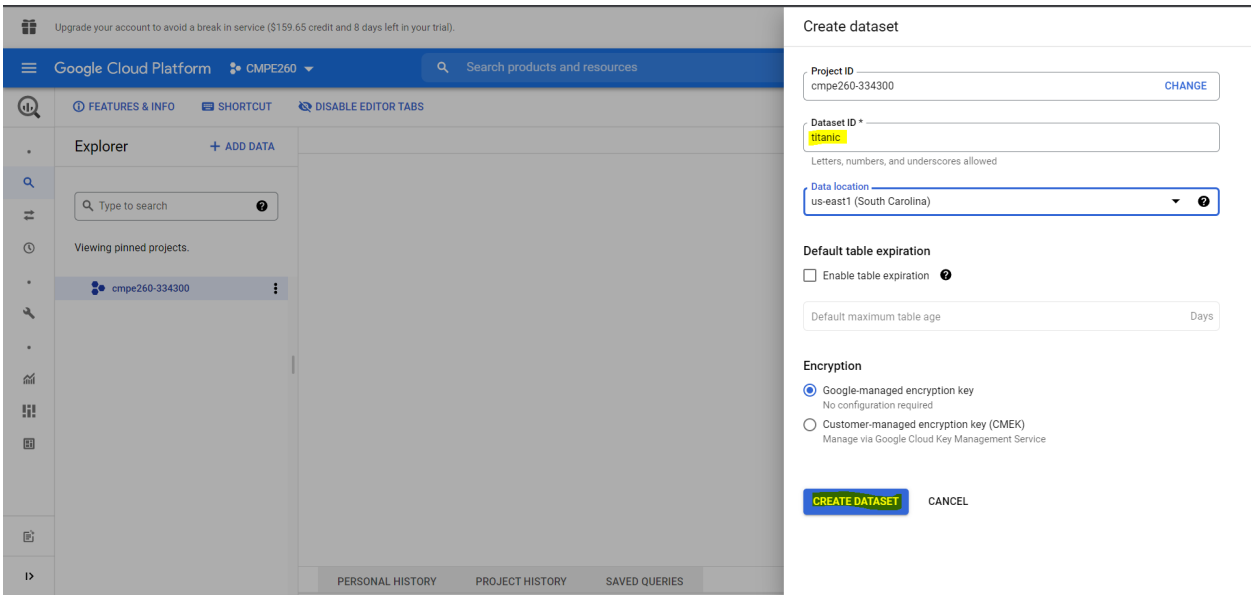
Load data in BigQuery

In order to train a Machine Learning model, you need access to data. BigQuery is a serverless, highly scalable, and cost-effective multi-cloud data warehouse and it is the perfect service for keeping your data.

Create dataset

1. Make sure that you select the right project from the top of console page
2. Navigate to Big Query
3. Select the project you want to create the Dataset in
4. Click Create Dataset

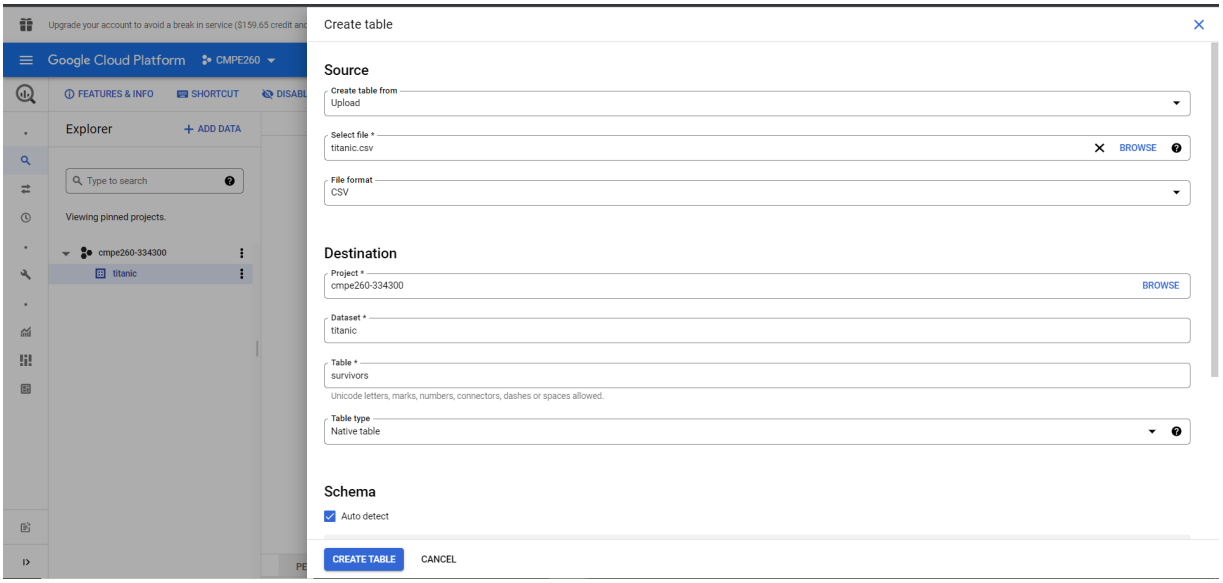




5. Select the ‘titanic’ dataset created above
6. Click ‘Create Table’

From the Sidebar select the following:

7. Create table from: Upload
8. Select file: Use the downloaded titanic dataset
9. File Format: CSV
10. Table Name: survivors
11. Auto-detect: Select auto-detect checkbox - Schema and input parameters
12. Click ‘Create Table’ button.

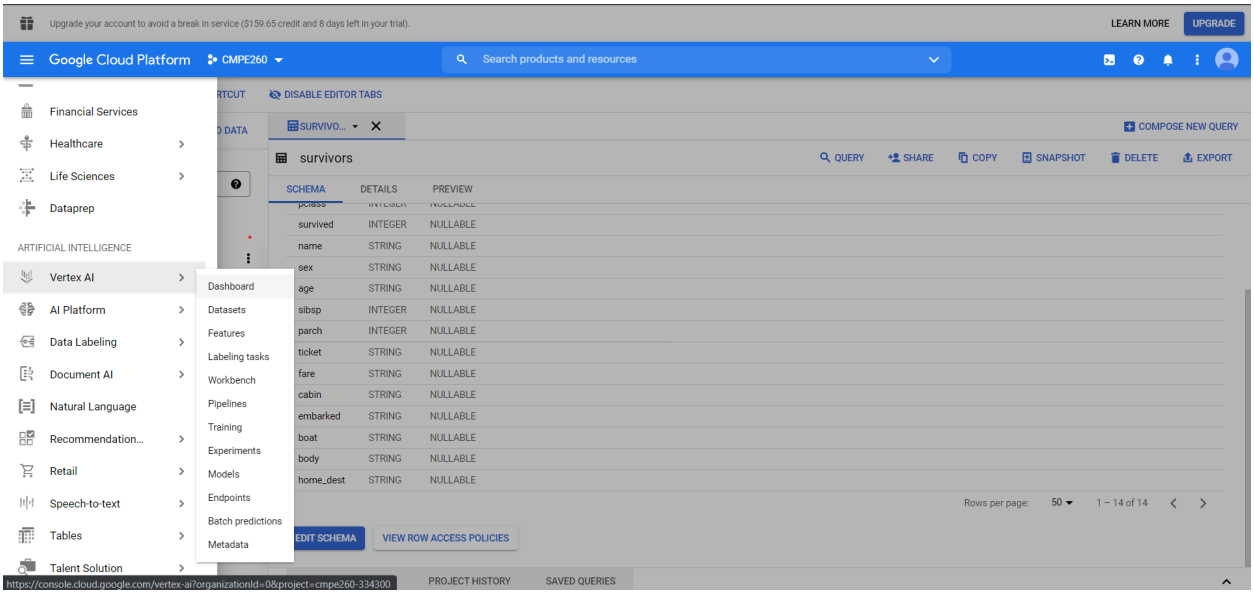


Create a dataset

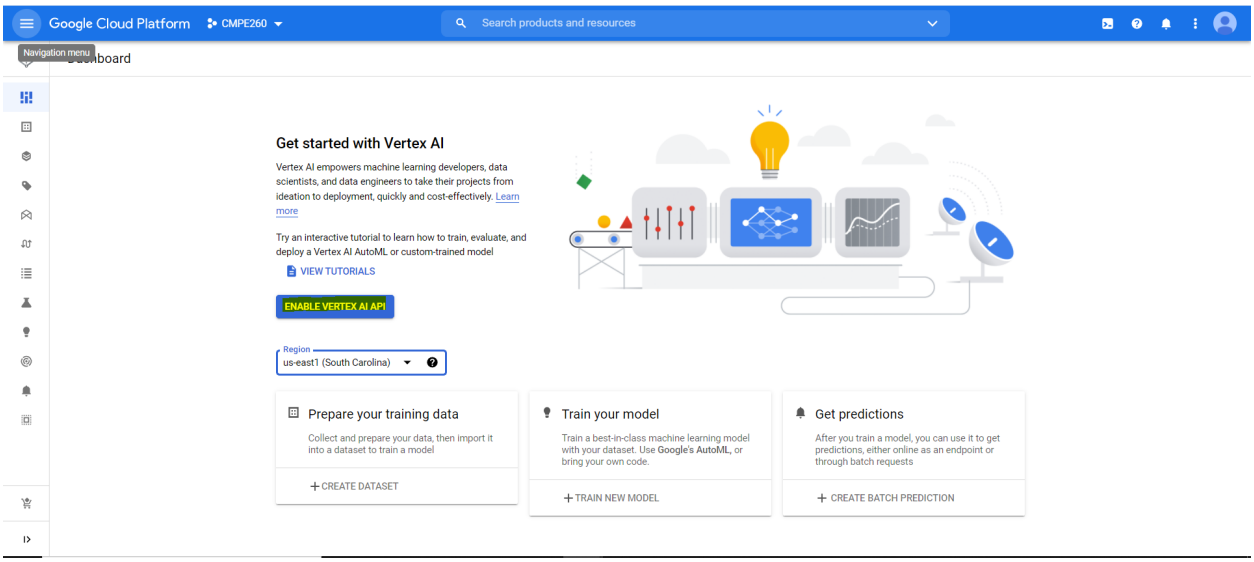
Datasets in Vertex AI allow you to create datasets for your Machine Learning workloads. You can create datasets for structured data (CSV files or BigQuery tables) or unstructured data such as Images and Text.

Create ML dataset

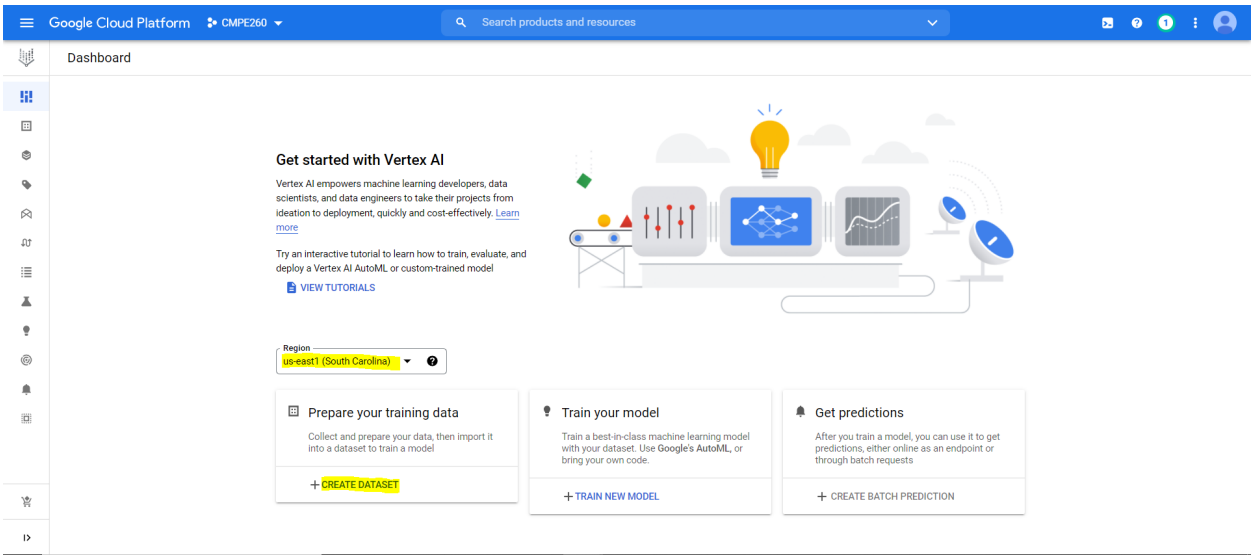
1. Find Vertex AI on the GCP side menu, under Artificial Intelligence



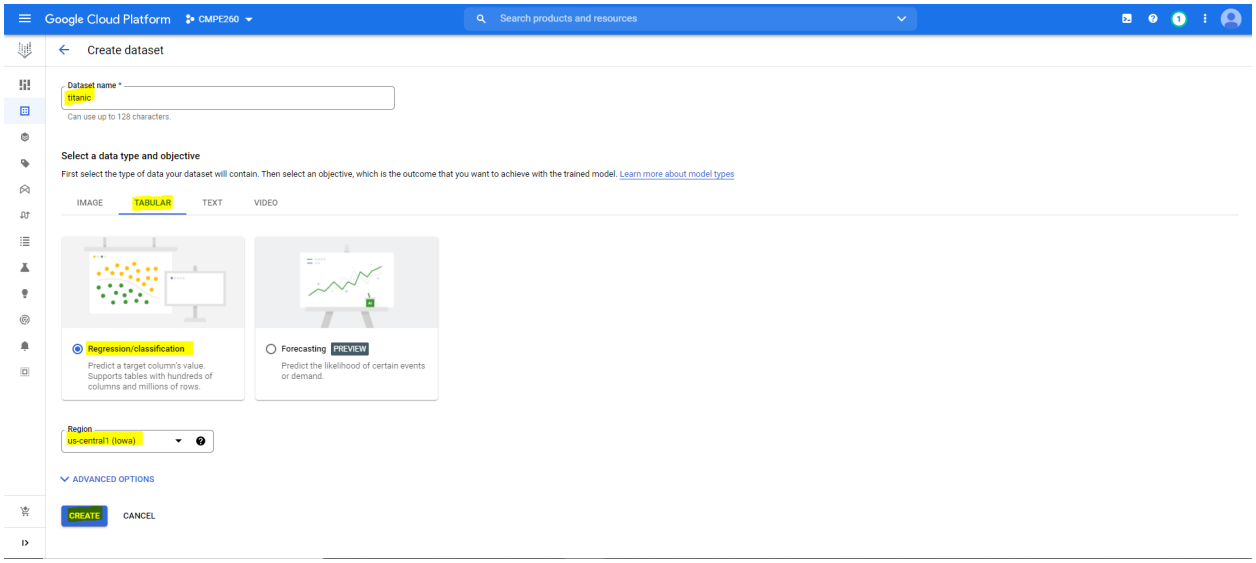
2. Select ‘Enable Vertex AI API’



3. Select the Region, and click ‘Create Dataset’ button



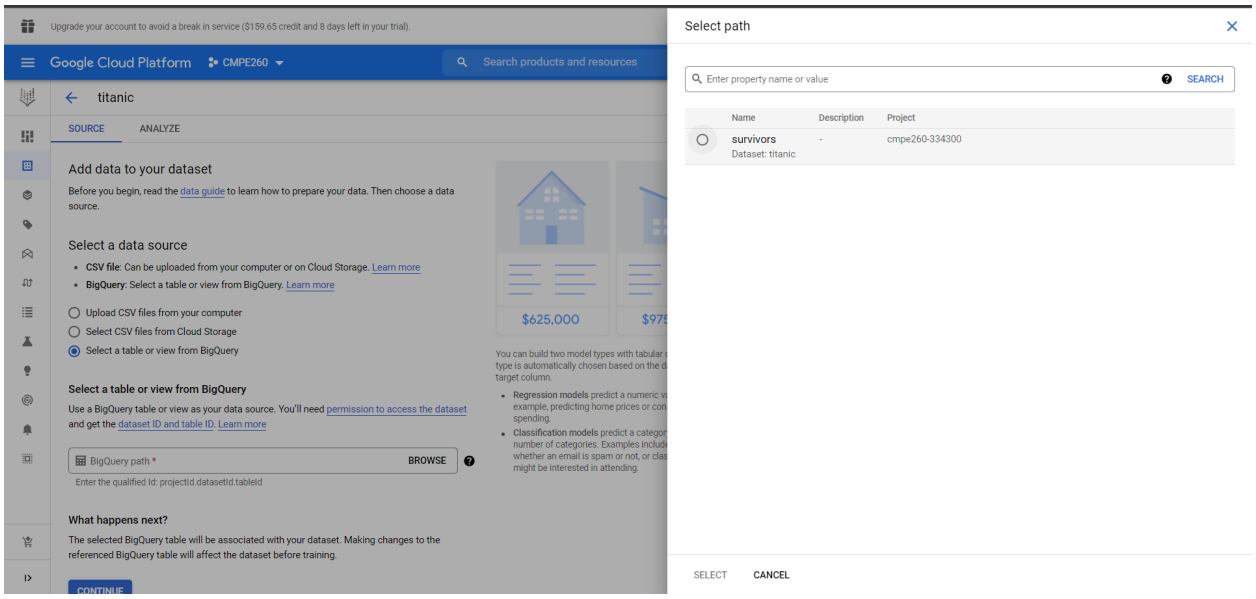
4. Enter dataset name as ‘titanic’
5. We can create datasets for images, text or videos as well as tabular data. The ‘titanic’ dataset is tabular so click the Tabular tab



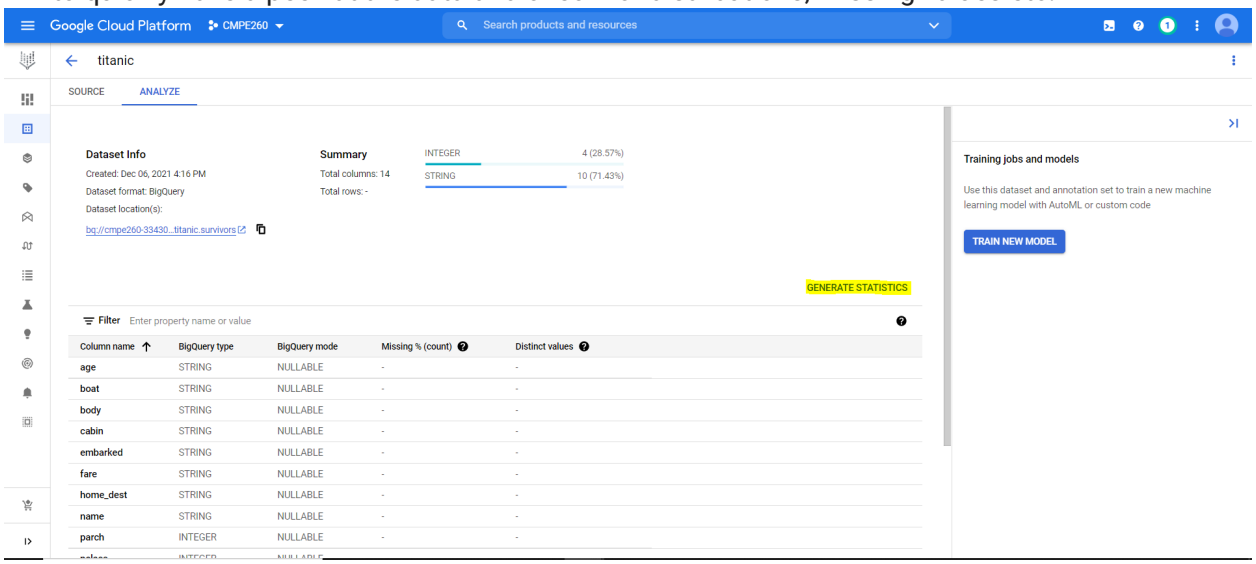
6. Select ‘Tabular’ and ‘Regression/Classification’
7. Click the ‘Create’ button

Select datasource

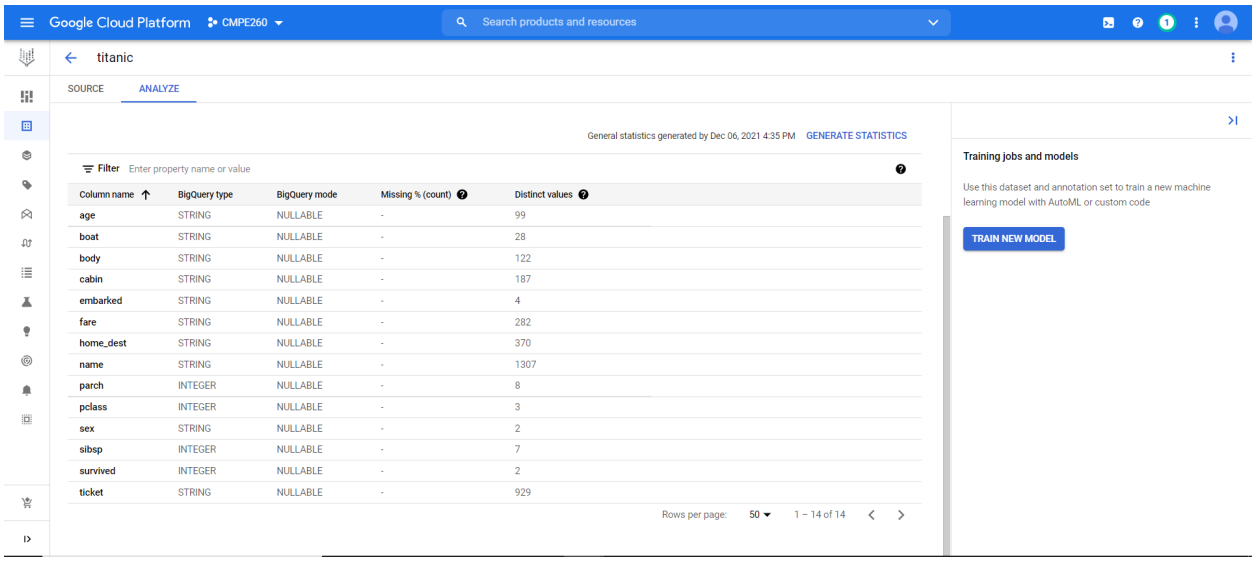
1. As we had already loaded the titanic dataset in BigQuery, we can connect our ML dataset to our BigQuery table.
2. Select ‘Select a table or view from BigQuery
3. Select the BigQuery Path
4. Select the ‘survivors’ table.



5. Click ‘Continue’
6. In the ‘Analyze’ tab we can generate statistics regarding your data. This gives you the ability to quickly have a peek at the data and check for distributions, missing values etc.



7. Click on ‘Generate Statistics’



8. Statistics are generated.

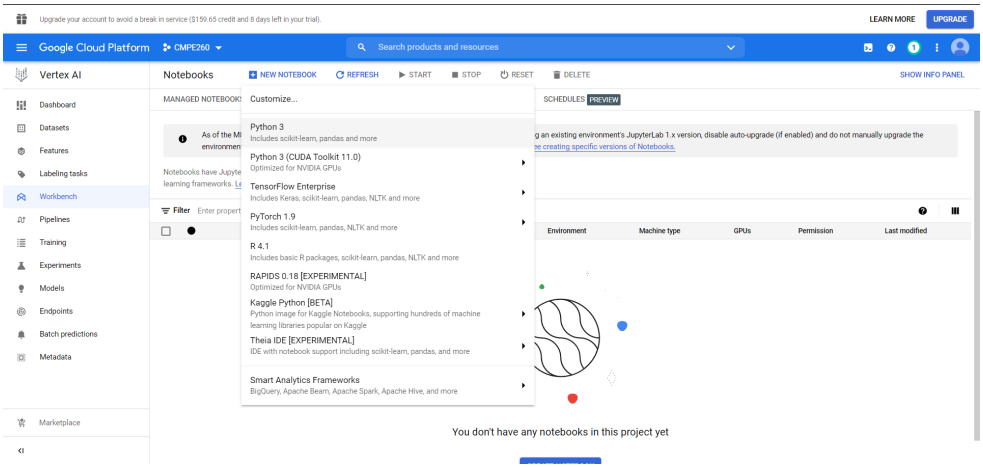
Custom training package using Notebooks

It is a good practice to package and parameterise your code so that it becomes a portable asset.

1. Create a training package with custom code using Notebooks

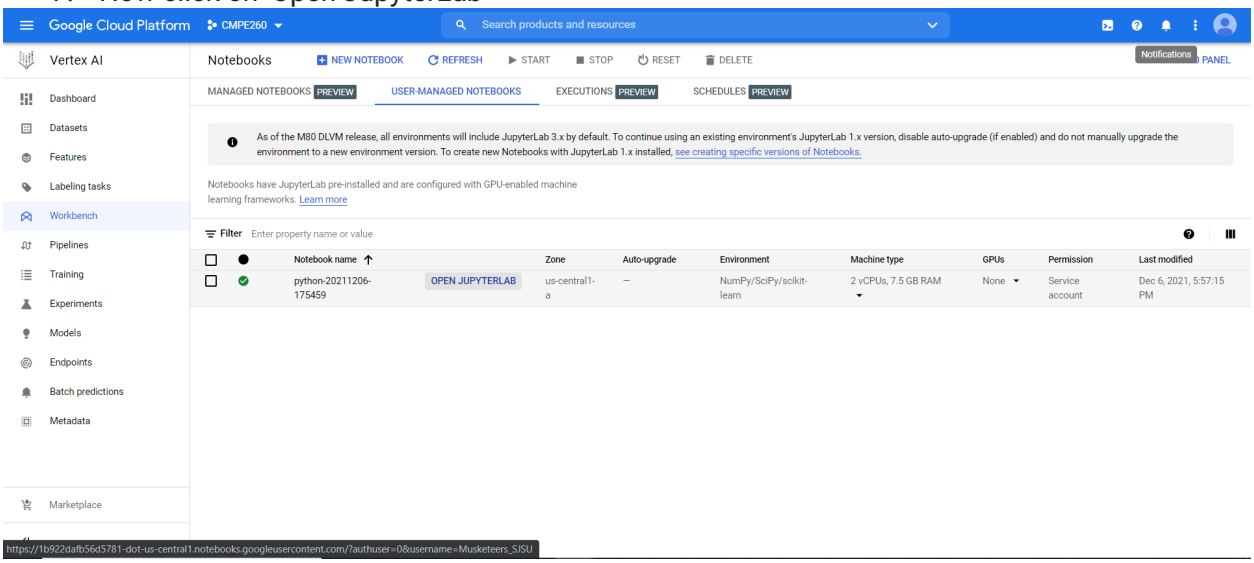
Create your notebook instance

1. From the Vertex AI navigate to notebooks and start an instance with **Python 3,** which includes scikit-learn
2. Select ‘Vertex AI’, Workbench
3. Enable Notebooks
4. On the top, select ‘New Notebook’

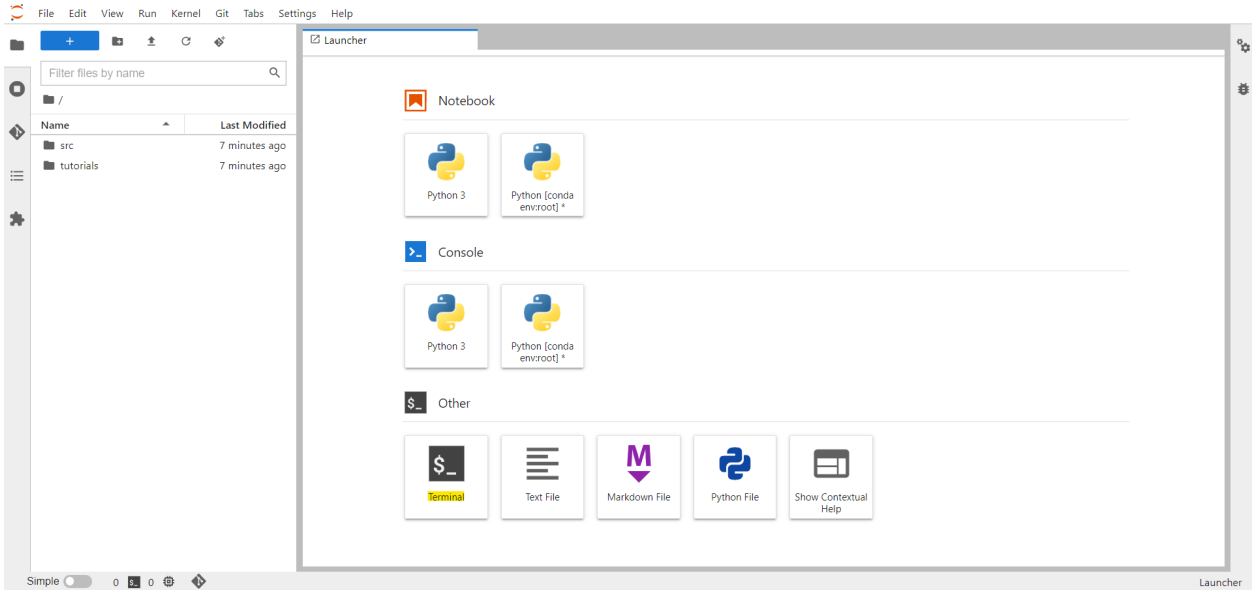


5. Select ‘Python3’, with scikit learn module.

6. The Notebook instance is created.
7. Now click on ‘Open JupyterLab’

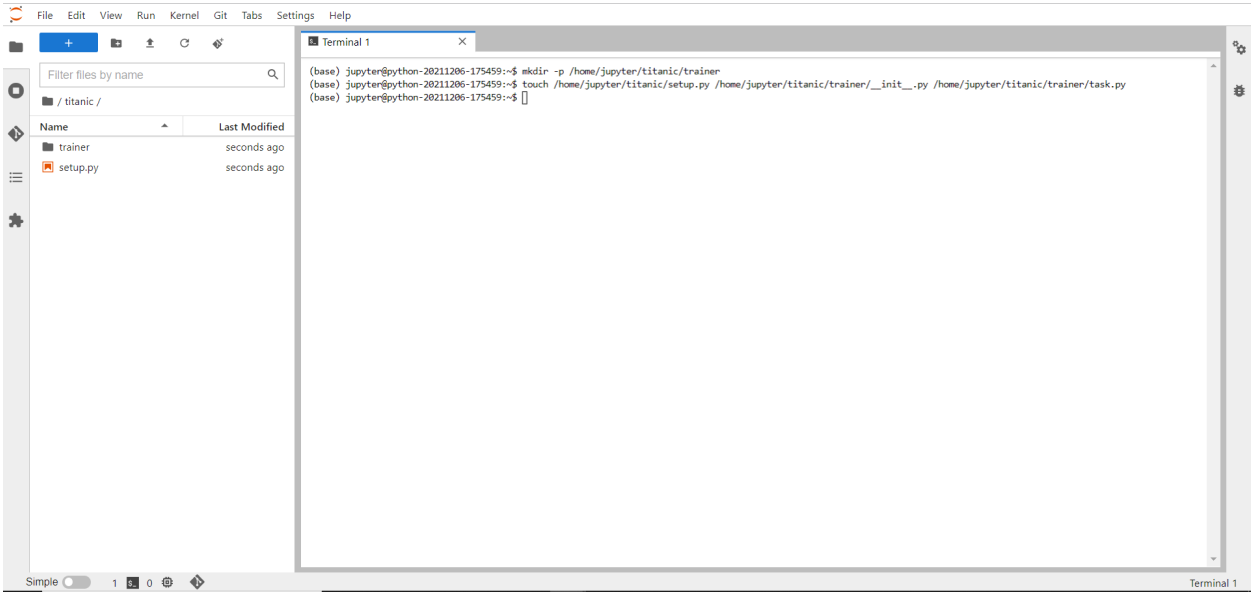


8. Select ‘Terminal’ in this window.

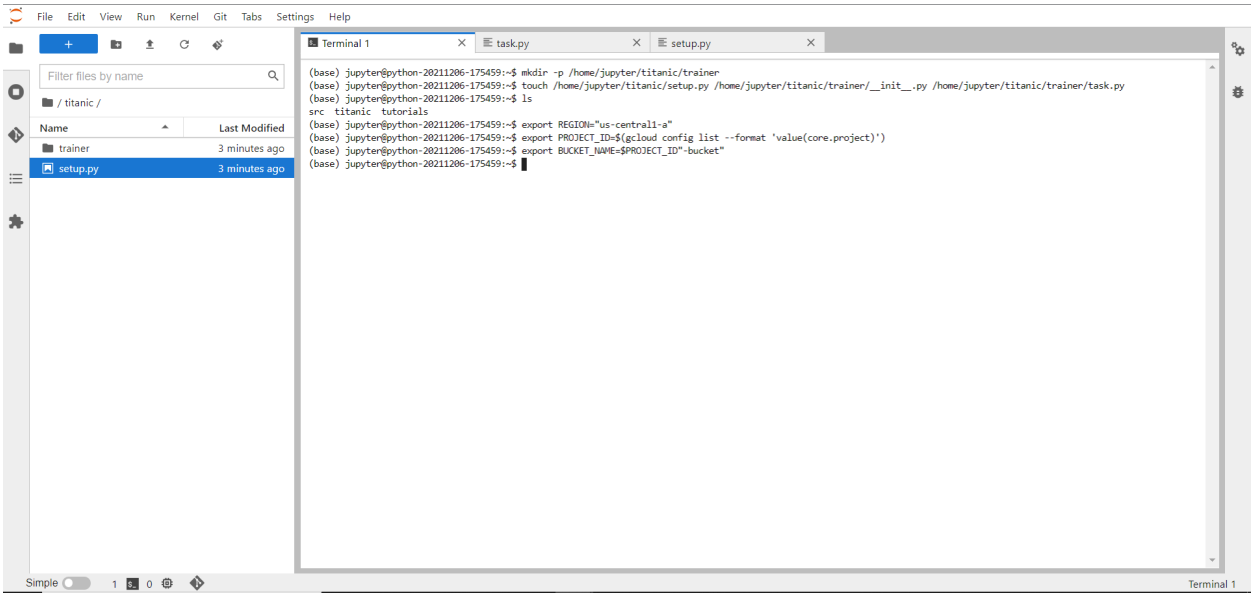


9. Run the following command to create the directory structure for our project.

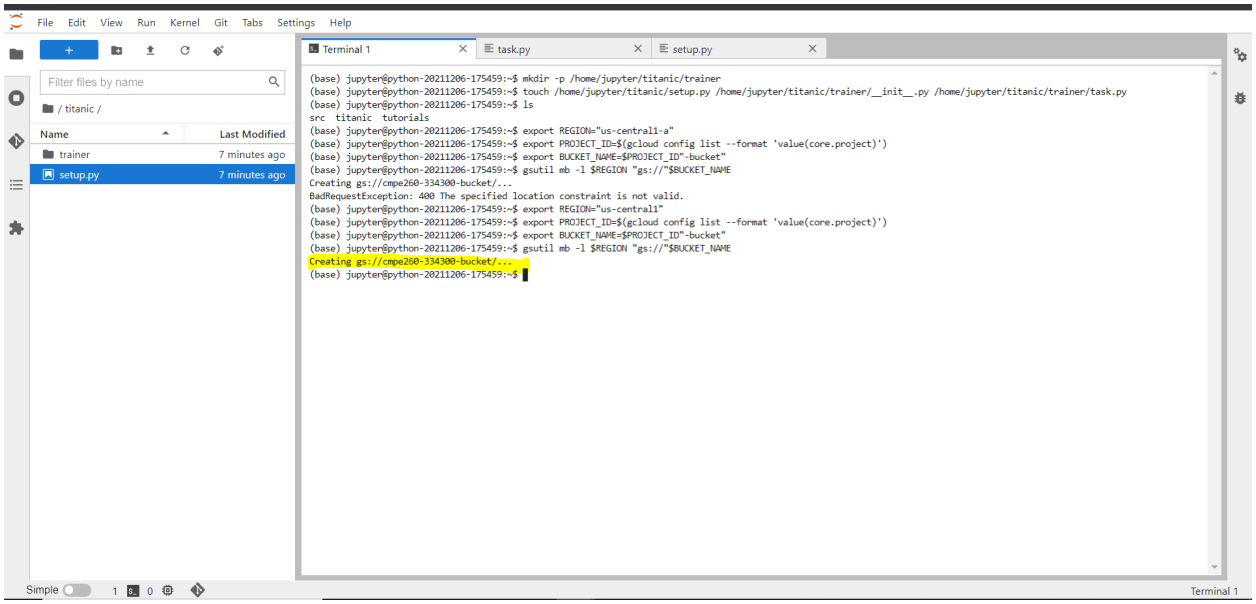
   mkdir -p /home/jupyter/titanic/trainer
   touch /home/jupyter/titanic/setup.py /home/jupyter/titanic/trainer/__init__.py
   /home/jupyter/titanic/trainer/task.py



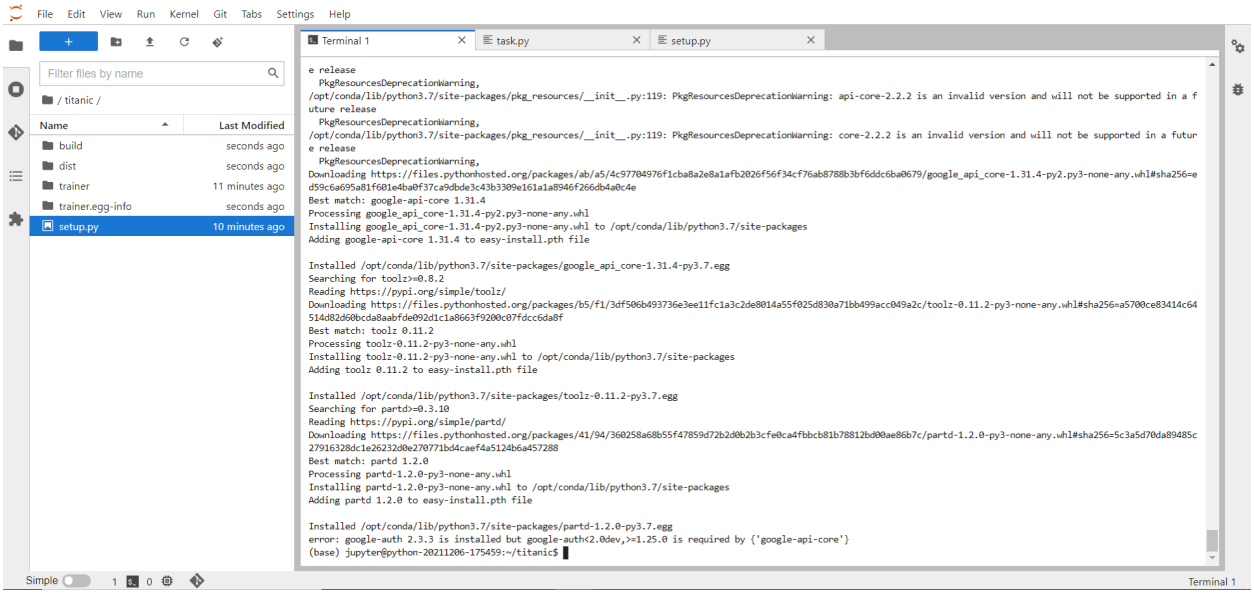
10. Create the code for ‘task.py’ and ‘setup.py’, to train and run our model
11. Setup the environment variables, by running the below commands:
    export REGION="us-central1"
    export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
    export BUCKET_NAME=$PROJECT_ID"-bucket"



12. Create a bucket where you want to export your trained model
Run the command: gsutil mb -l $REGION "gs://"$BUCKET_NAME

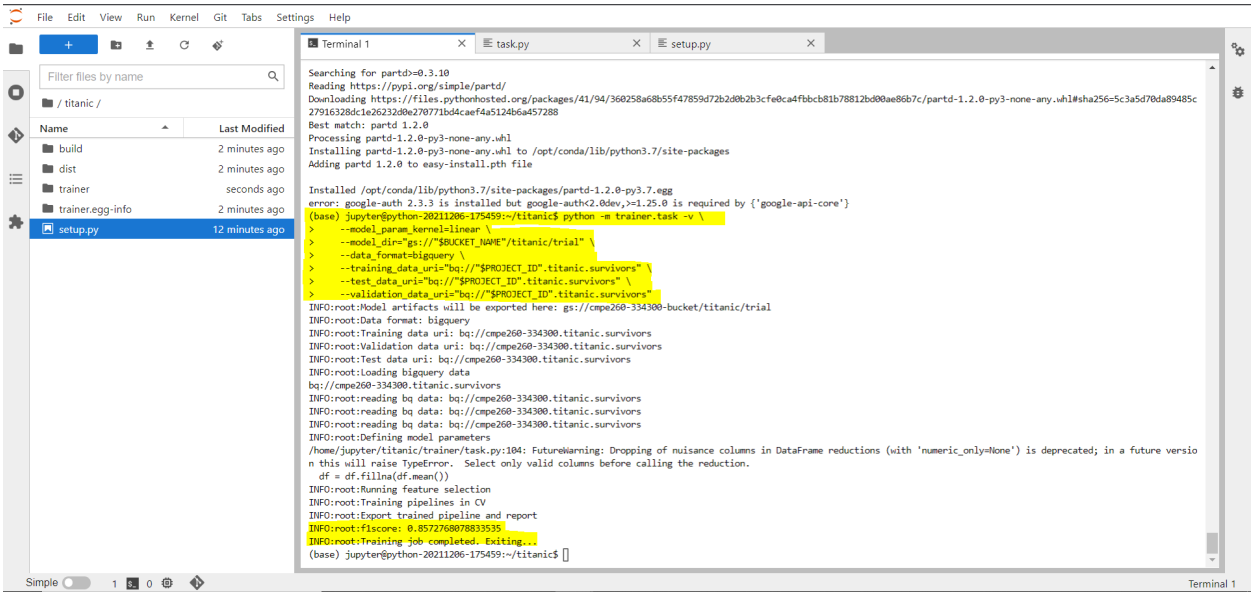


13. Install the required libraries
cd /home/jupyter/titanic
pip install setuptools
python setup.py install



14. Run the training code to verify that it executes without issues

```
python -m trainer.task -v \
  --model_param_kernel=linear \
  --model_dir="gs://"$BUCKET_NAME"/titanic/trial" \
  --data_format=bigquery \
  --training_data_uri="bq://"$PROJECT_ID".titanic.survivors" \
  --test_data_uri="bq://"$PROJECT_ID".titanic.survivors" \
  --validation_data_uri="bq://"$PROJECT_ID".titanic.survivors"
```

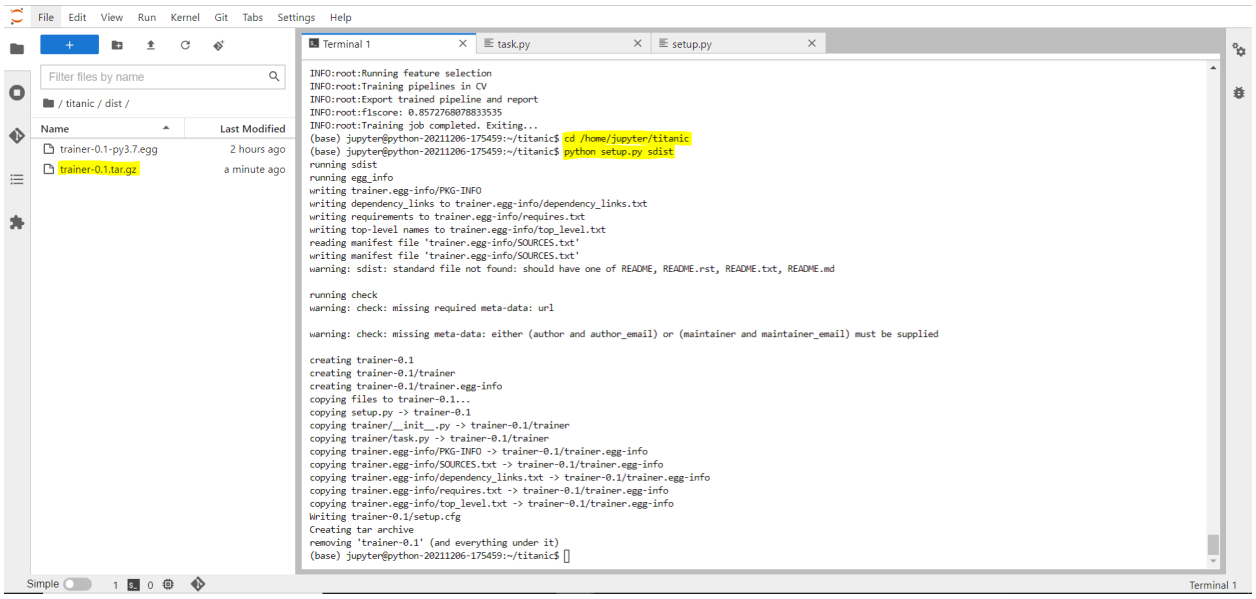


15. The two lines indicate the f1 score which is around 0.85 and the last line indicating that the training job completed successfully

16. Run the below commands to create the distributable file.

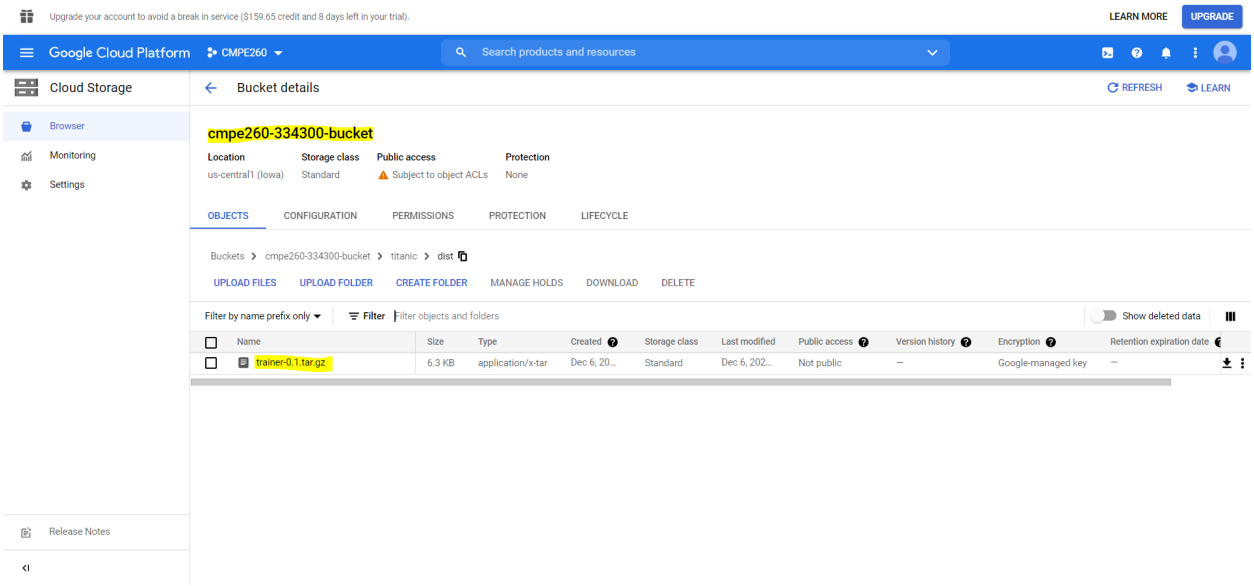
    ```
    cd /home/jupyter/titanic
    python setup.py sdist
    ```

17. Check for the tar.gz file created.



18. Copy the tar.gz file to GCS so that the training service can use it to train a new model

```
gsutil cp dist/trainer-0.1.tar.gz "gs://"$BUCKET_NAME"/titanic/dist/trainer-0.1.tar.gz"
```

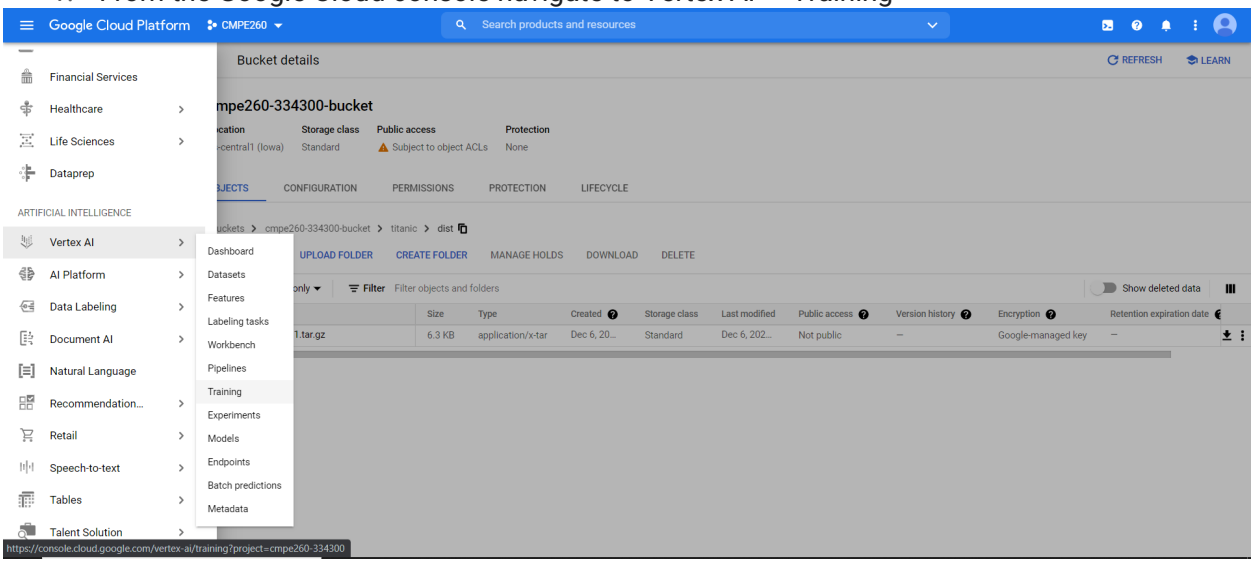


19. Check if the dist file is copied into the bucket.

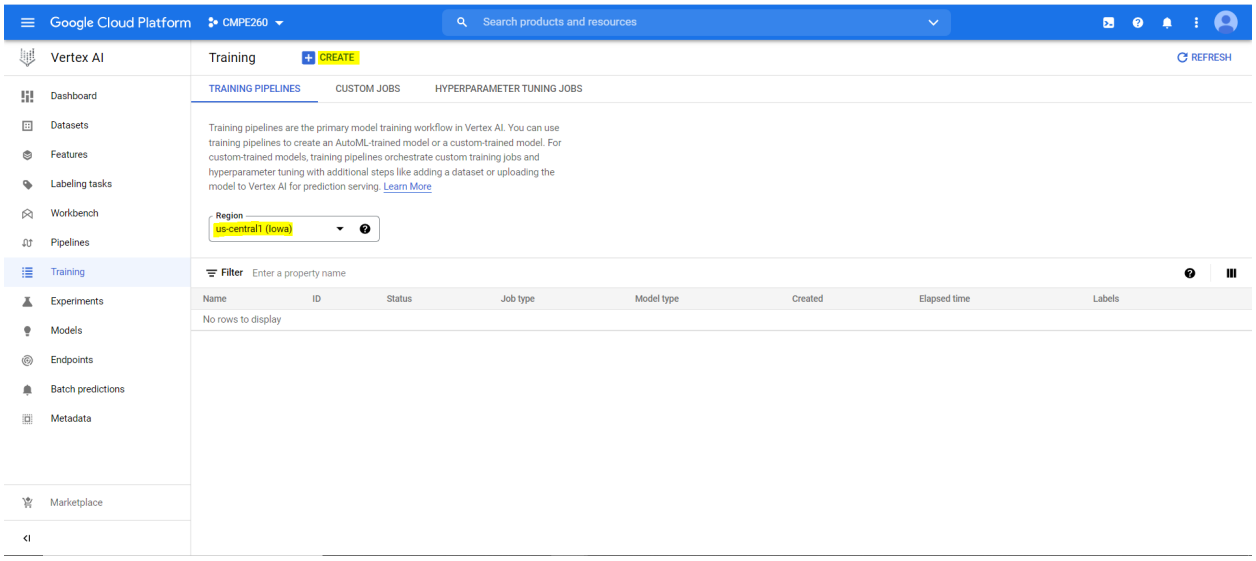
Model Training

In this, we will train a model on Vertex AI

1. From the Google Cloud console navigate to Vertex AI -> Training

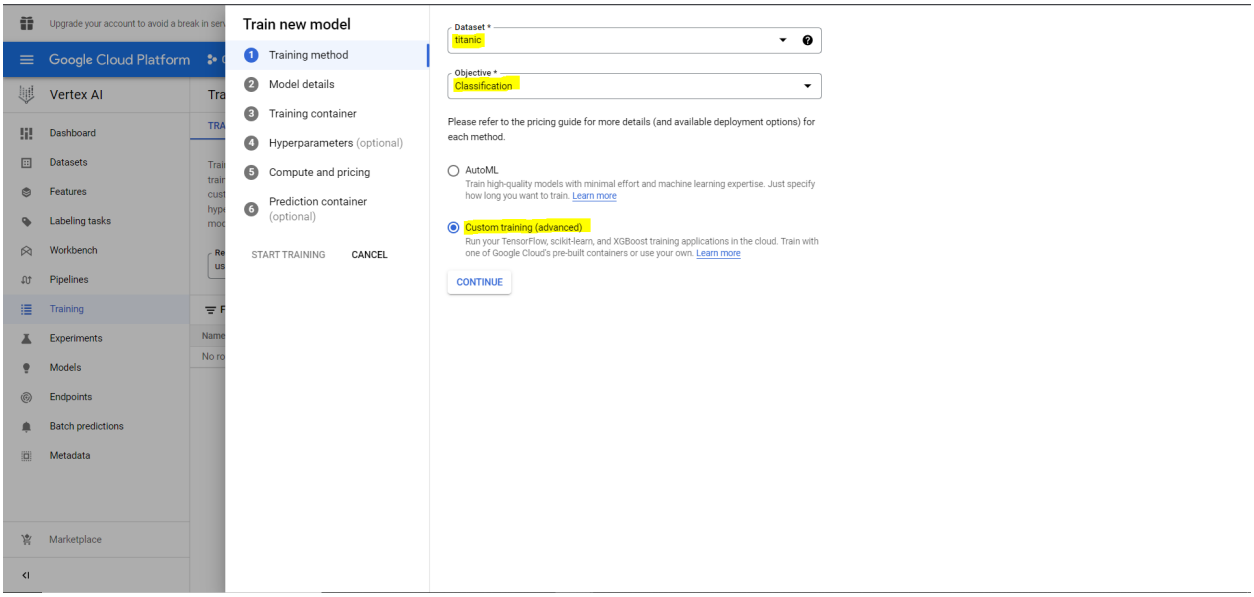


2. Select the Region and select ‘Create’



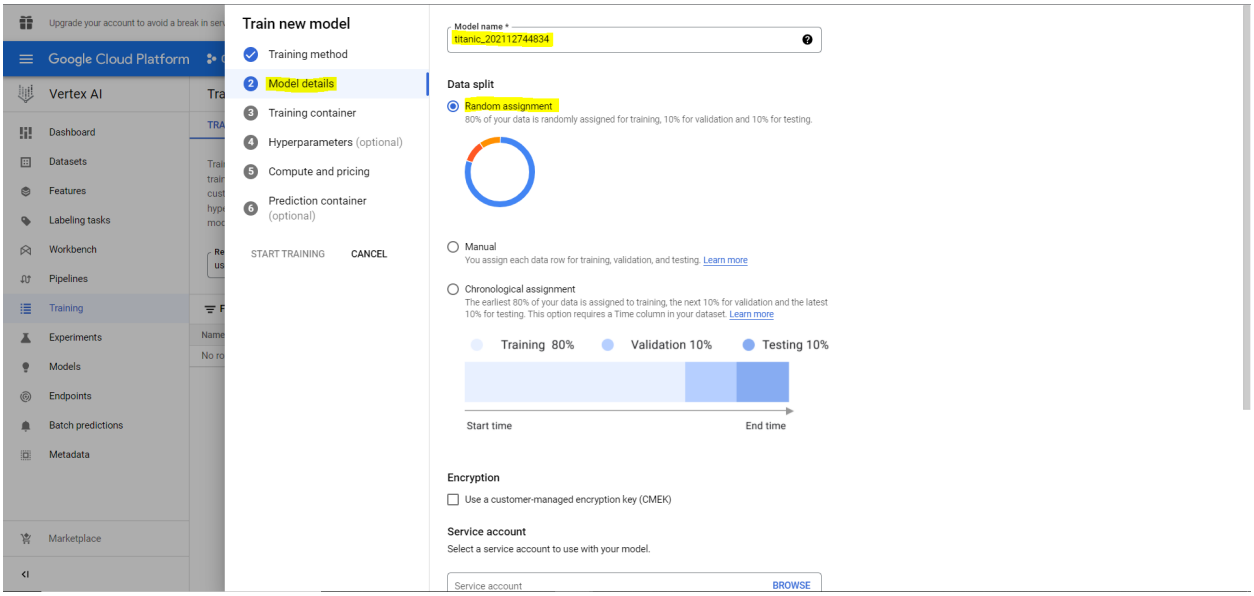
Step 1: Training method

Select the DataSet as ‘titanic’, Objective as ‘Classification’



Step 2: Model details

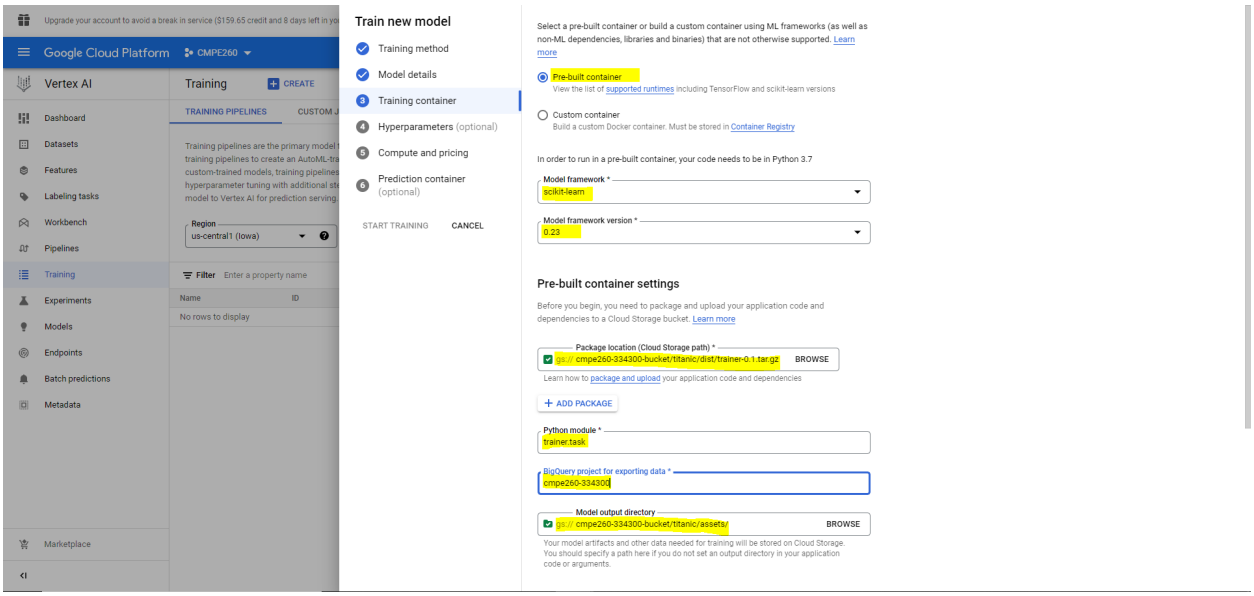
Enter a Model name, and Data split to ‘Random Assignment’



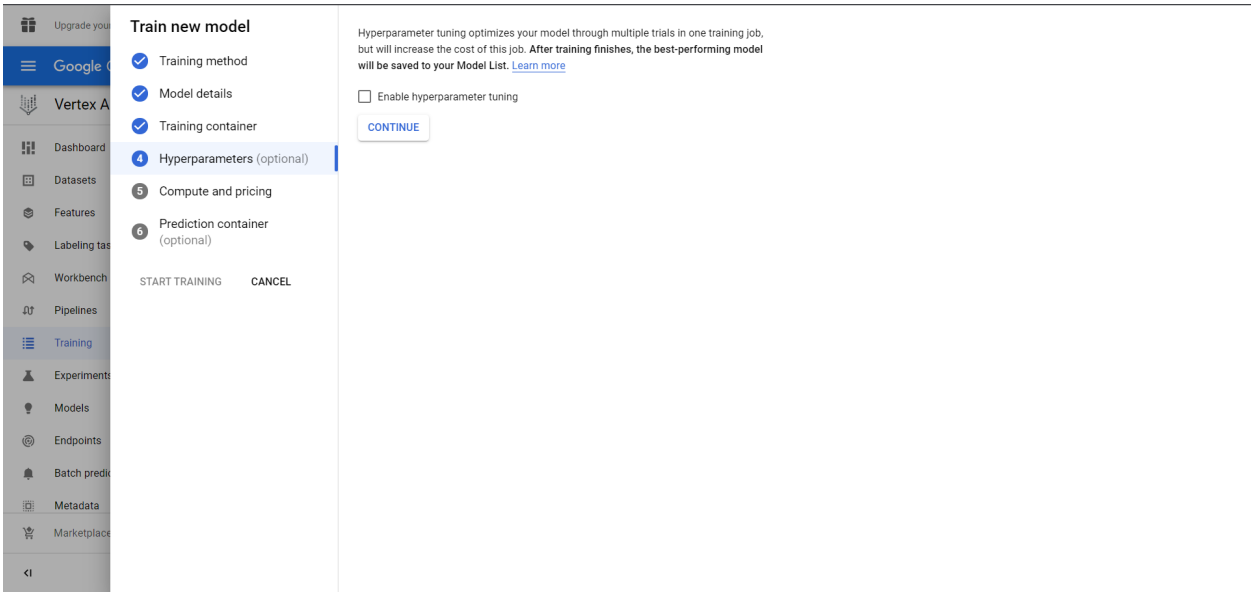
Step 3: Training container

Define your training environment

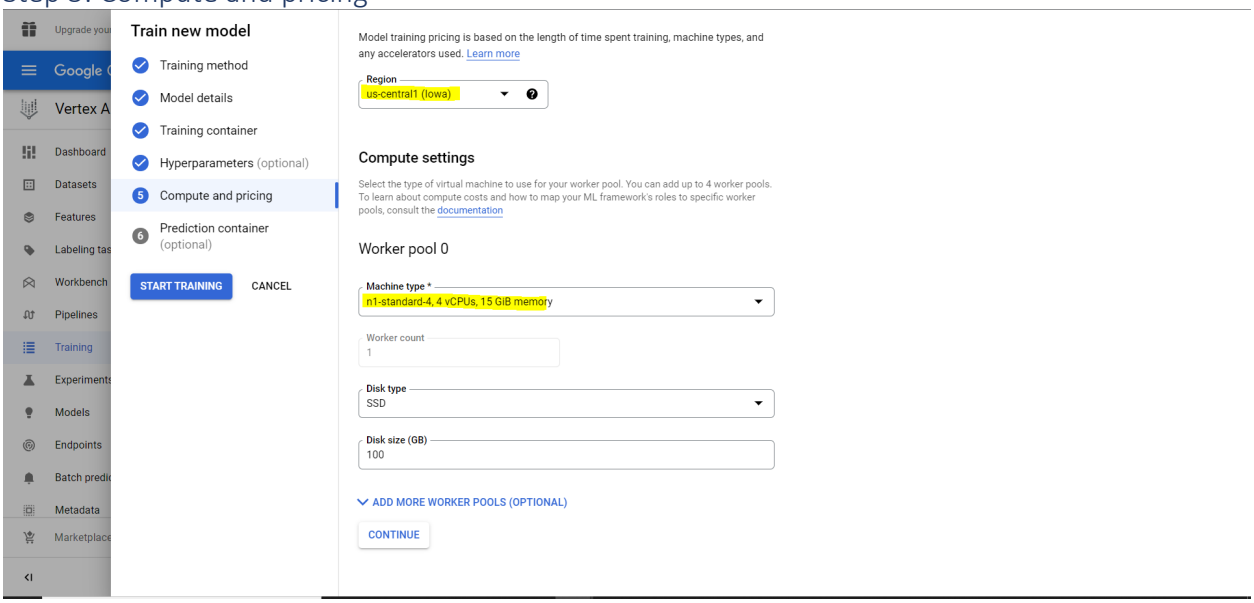
1. **Pre-built container**: Google cloud offers a set of prebuilt containers that make it easy to train your models. Those containers support frameworks such as Scikit-Learn, Tensorflow and XGBoost. Your model is based on scikit-learn and prebuilt container already exists.
2. Model framework: **Scikit-learn.** This is the library you used for model training.
3. Model framework version: Your code is compatible with **0.23**.
4. Package location: You can browse to the location of your training package. This is the bucket location where the training-0.1.tar.gz got uploaded.
5. Python Module: The python module you created in Notebooks. It will correspond to the folder that has your training code/module and the name of the entry file. This should be trainer.task
6. BigQuery project for exporting data:



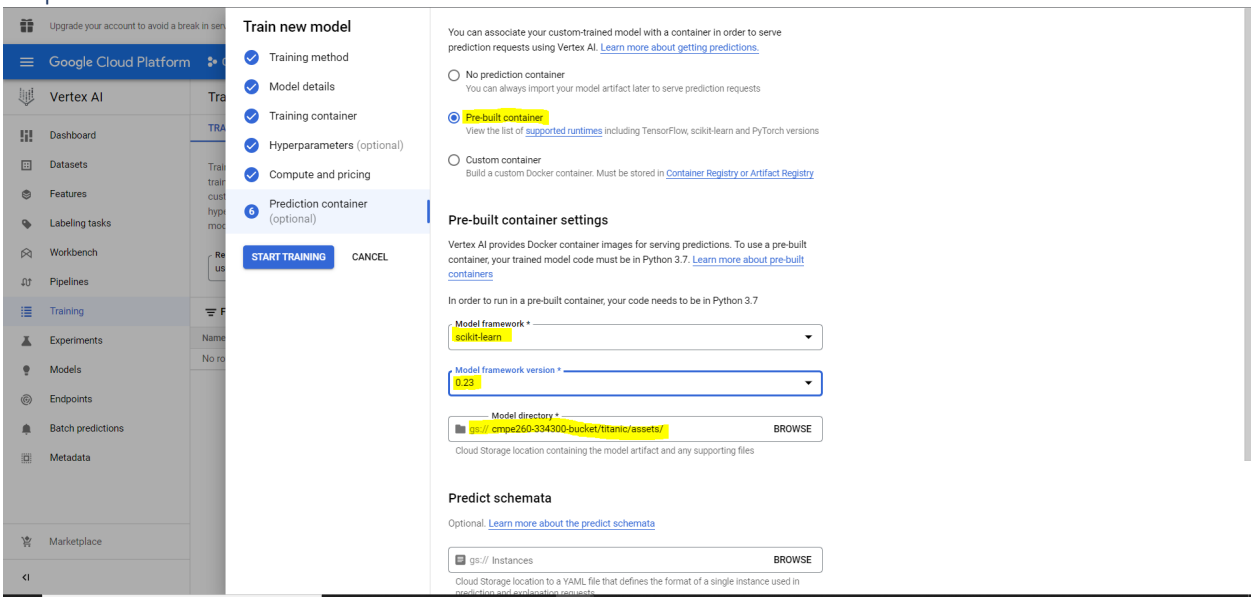
Step 4: Hyperparameter tuning



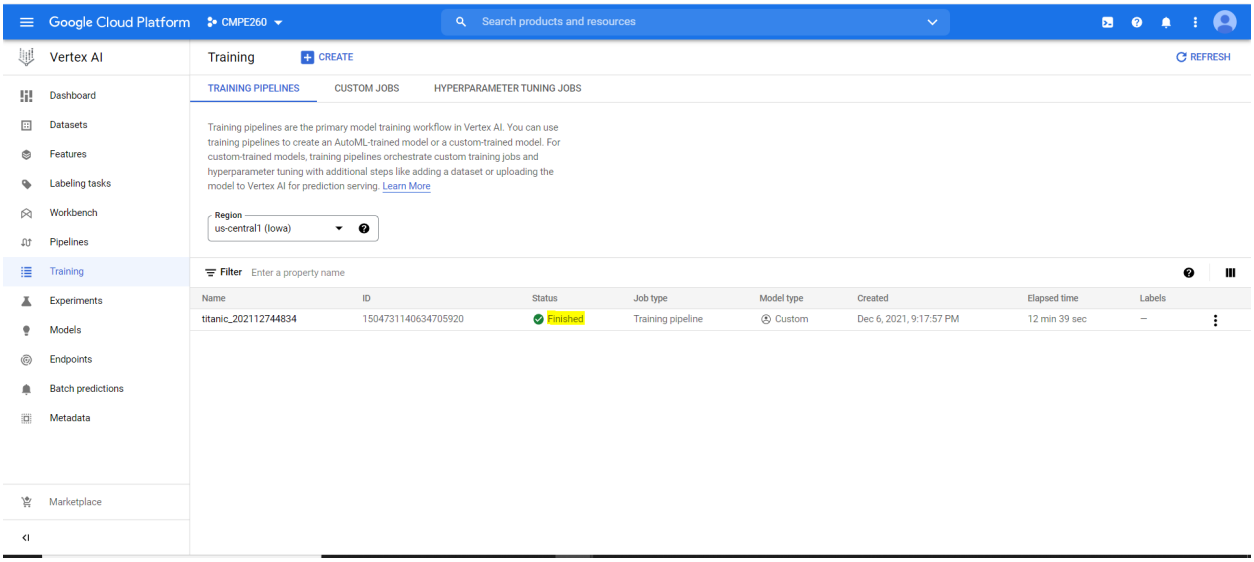
Step 5: Compute and pricing



Step 6: Prediction container



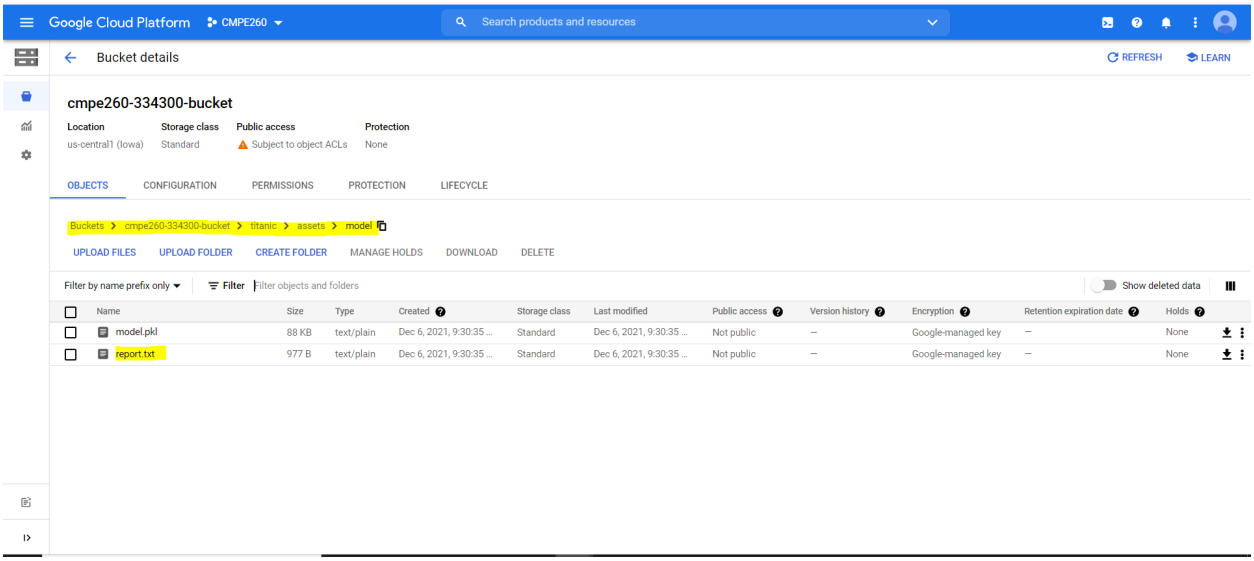
Click on ‘Start Training’.



Check for the training to be ‘Finished’

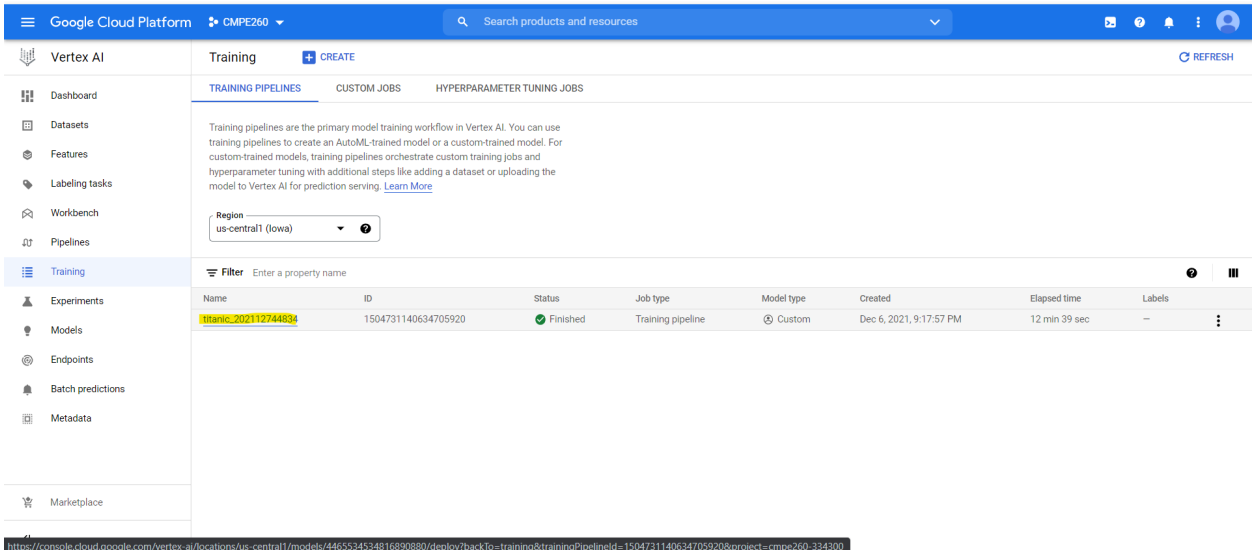
Model Evaluation

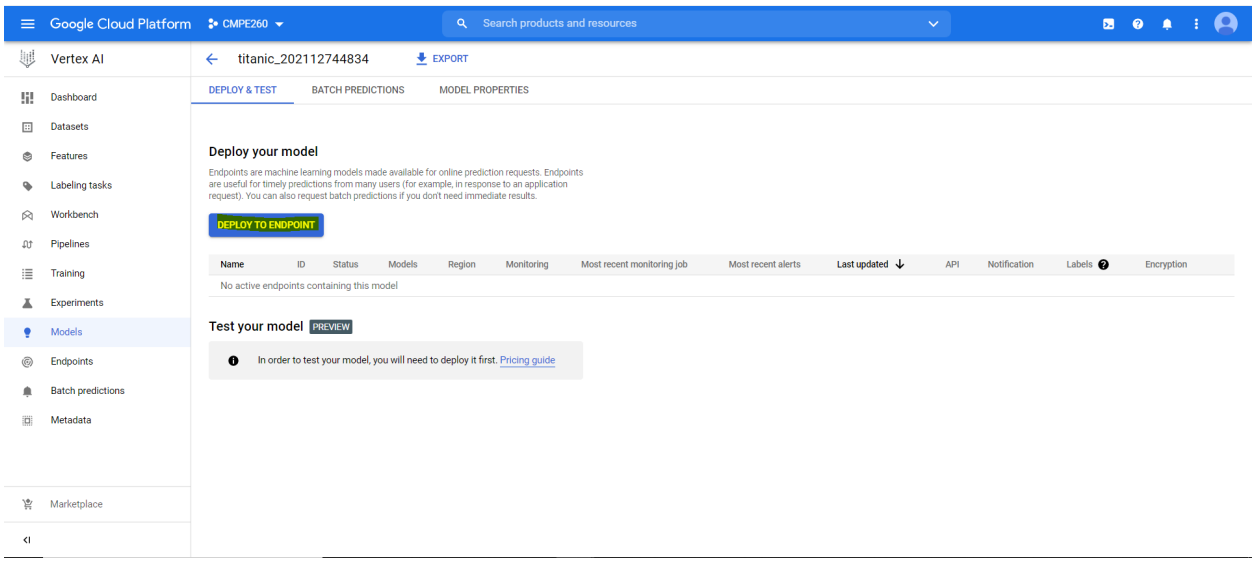
1. After the training job completion artifacts will be exported under gs://YOUR-BUCKET-NAME/training/assets
2. Check for the report.txt, under this folder.



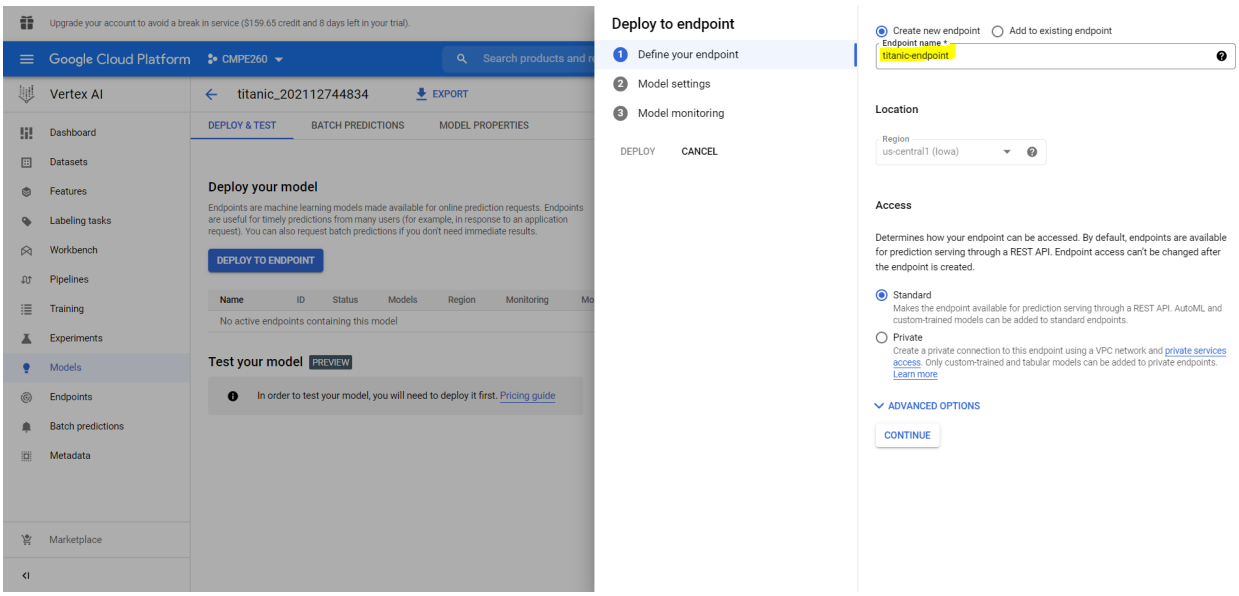
Model Deployment

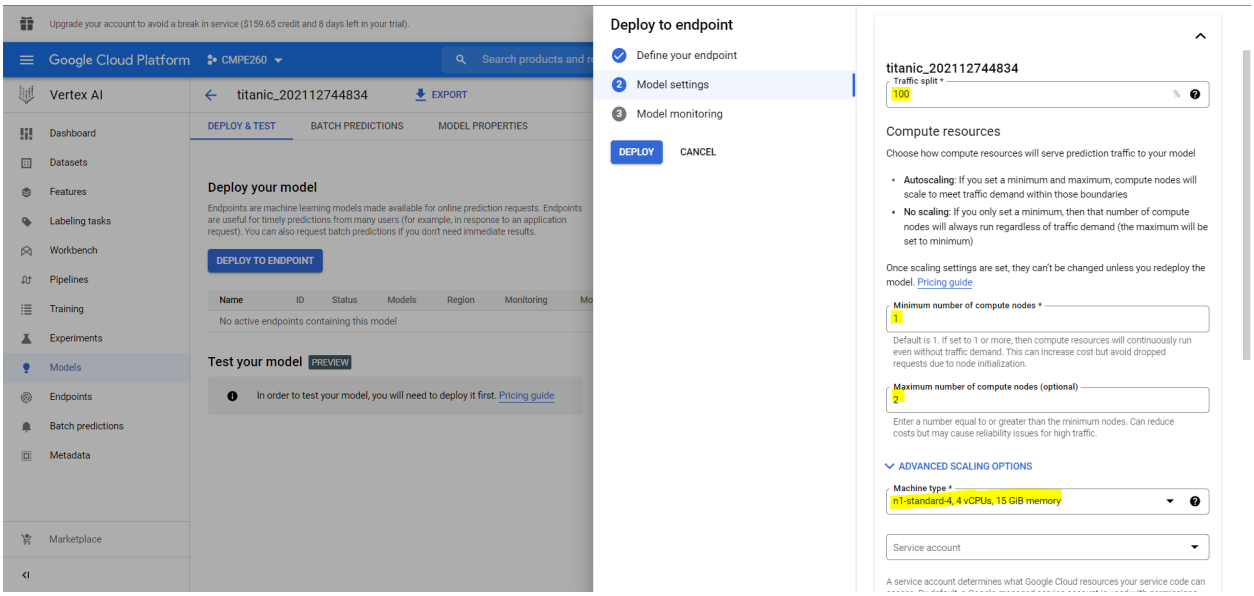
1. Last step is model deployment
2. Click on the trained model and **DEPLOY TO ENDPOINT**



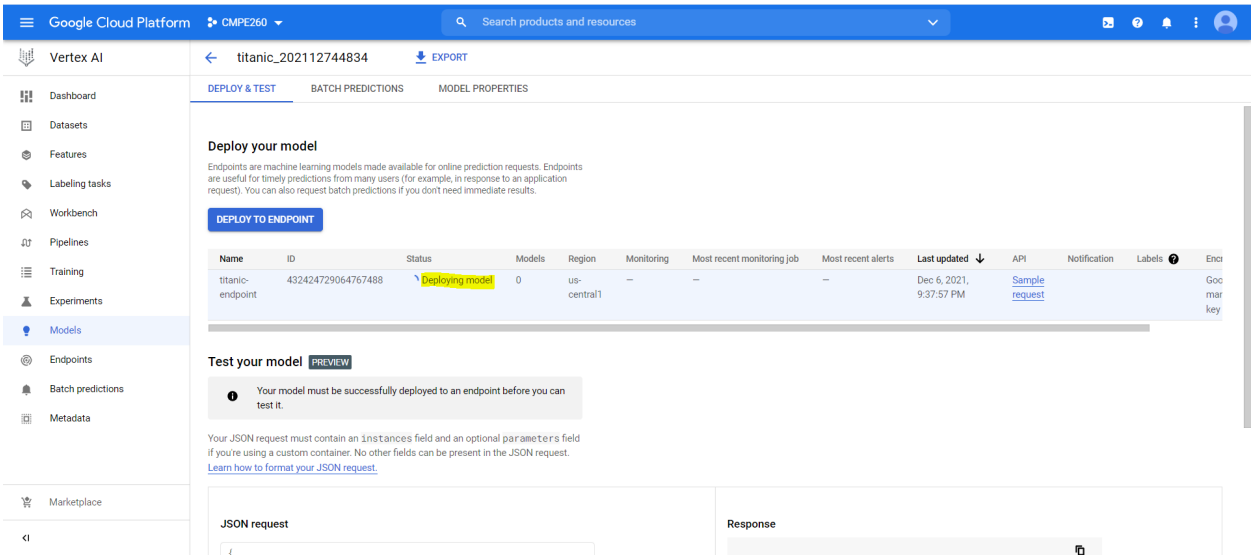


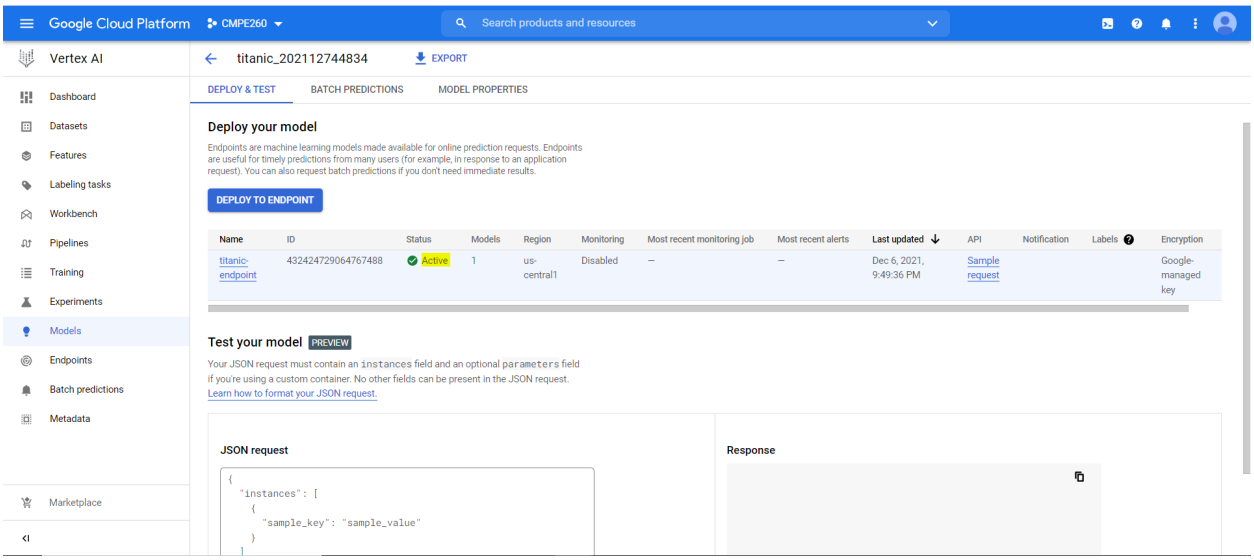
- **Endpoint name:** titanic-endpoint Endpoint URL where the model is served.
- **Traffic split:** Defines the percentage of traffic that you want to direct to this model. An endpoint can have multiple models and you can despite how to split the traffic among them. In this case you are deploying a single model so the traffic has to be 100 percent.
- **Minimum number of compute nodes:** The minimum number of nodes required to serve model predictions. Start with 1. Additionally the prediction service will autoscale in case there is traffic
- **Maximum number of compute nodes:** In case of autoscaling, this variable defines the upper limit of nodes. It helps protecting against unwanted costs that autoscaling might result in. Set this variable to 2
- **Machine type:** Google cloud offers a set of machine types you can deploy your model to. Each machine has its own memory and vcpu specs. Your model is simple so serving on an n1-standard-4 instance will do the job





3. Click **CONTINUE** and **DEPLOY**



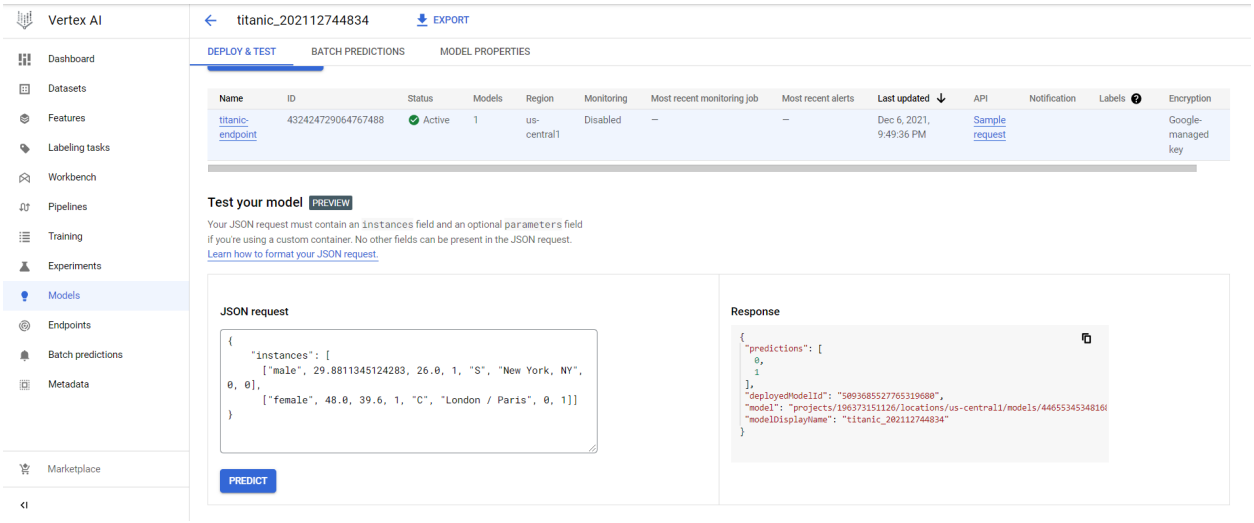


Model Prediction

Under **Models** test the model prediction endpoint.

Try for different payloads:
Request:

```
{
   "instances": [
     ["male", 29.8811345124283, 26.0, 1, "S", "New York, NY", 0, 0],
     ["female", 48.0, 39.6, 1, "C", "London / Paris", 0, 1]]
}
```

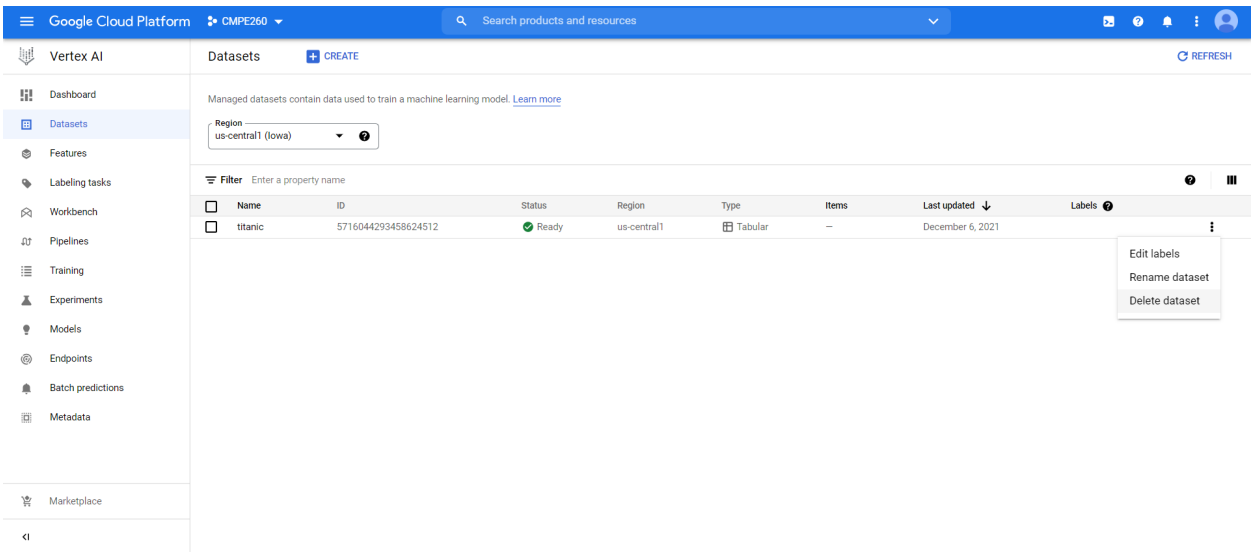


Response:

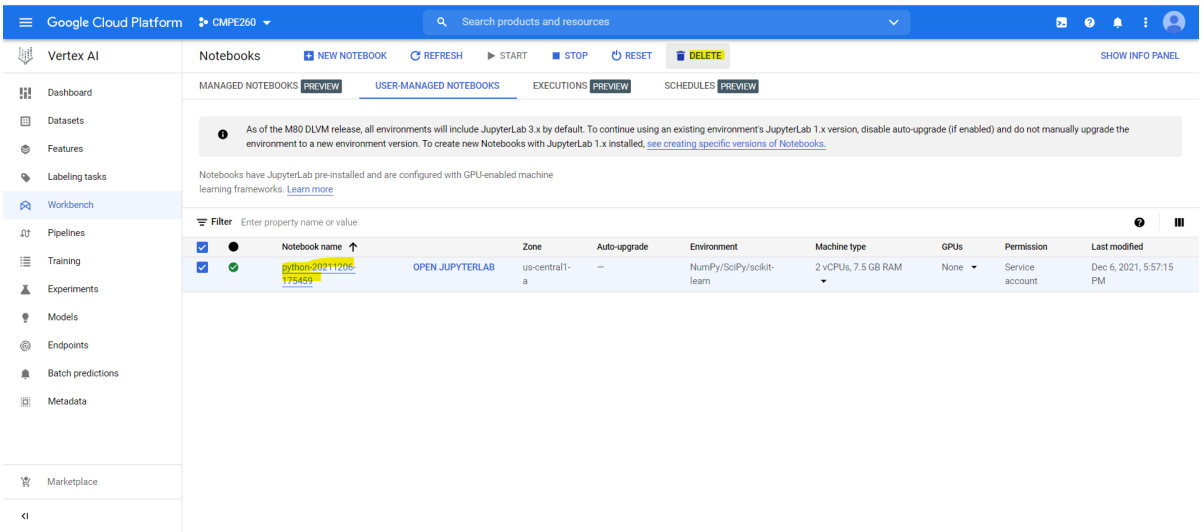
```
{
 "predictions": [
   0,
   1
 ],
 "deployedModelId": "5093685527765319680",
 "model": "projects/196373151126/locations/us-central1/models/4465534534816890880",
 "modelDisplayName": "titanic_202112744834"
}
```


Cleaning up

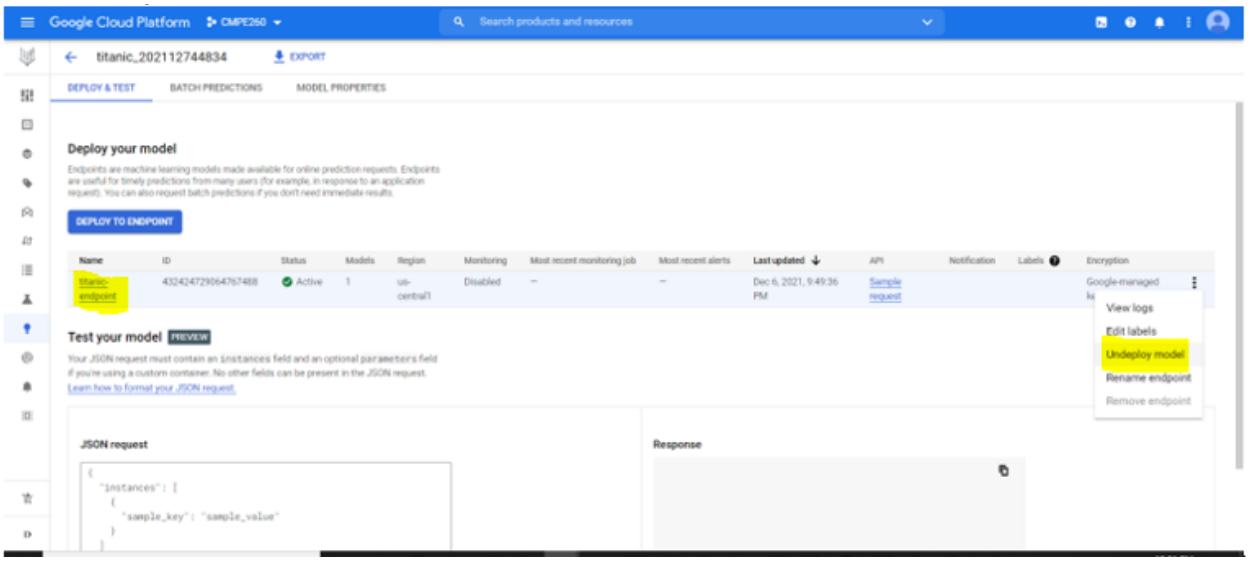
Delete ML Dataset



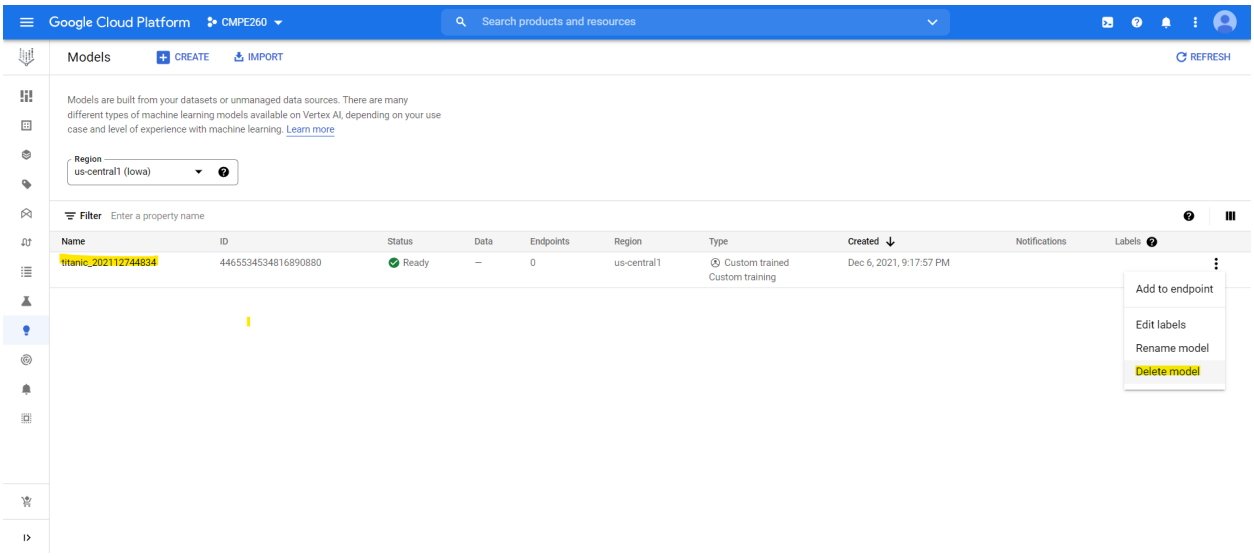
Delete Notebook



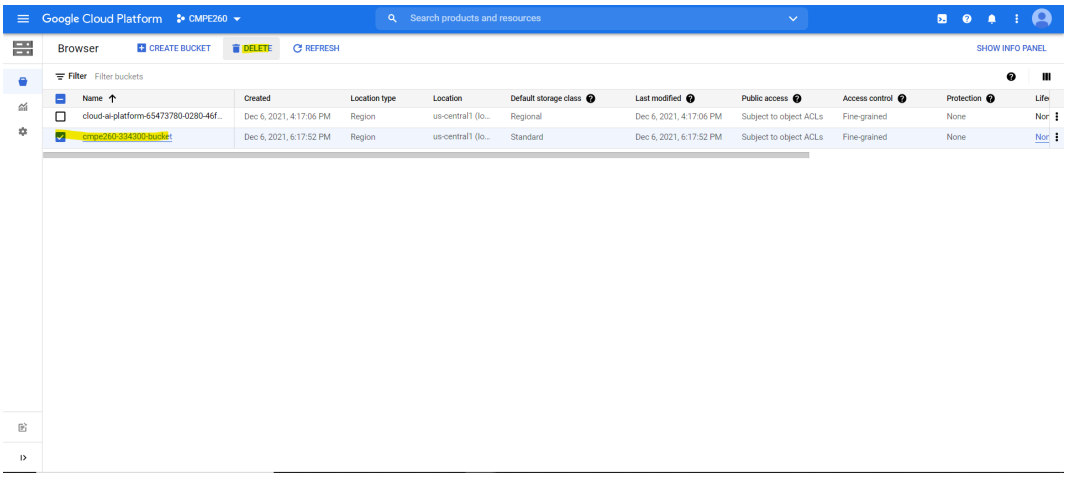
Delete Endpoint



Delete Model



Delete Objects, GCP Bucket



Delete BigQuery dataset