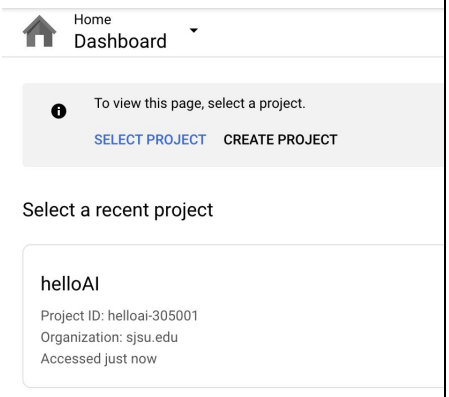
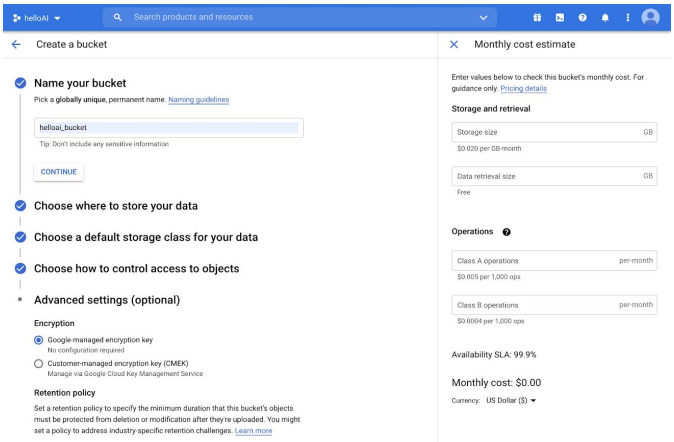
CMPE 258 Haley Feng
Professor Vijay Eranti

HW #2.1 Hello AI Platform (Unified)

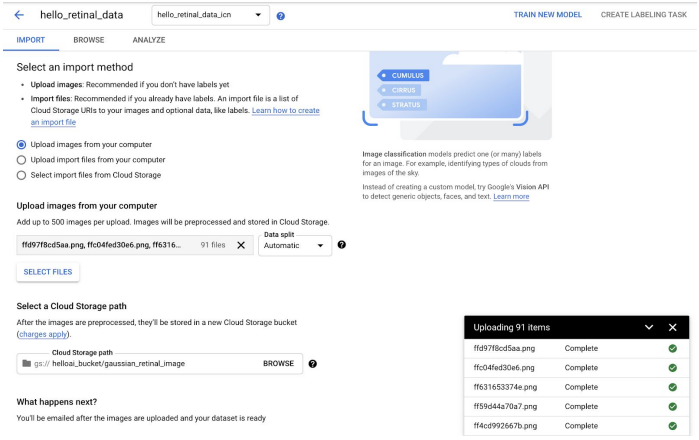
Prerequisite:

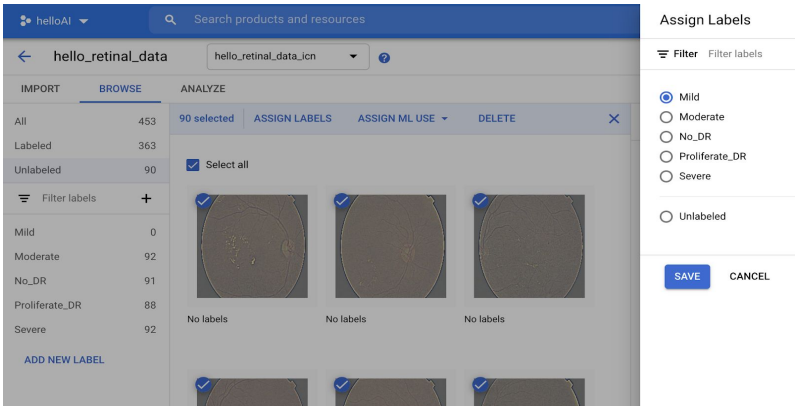
Created a project named *helloAI*	Created a cloud storage bucket named *helloai_bucket* to store datasets
	

After initial set-ups, proceed with the following for each different types of data:

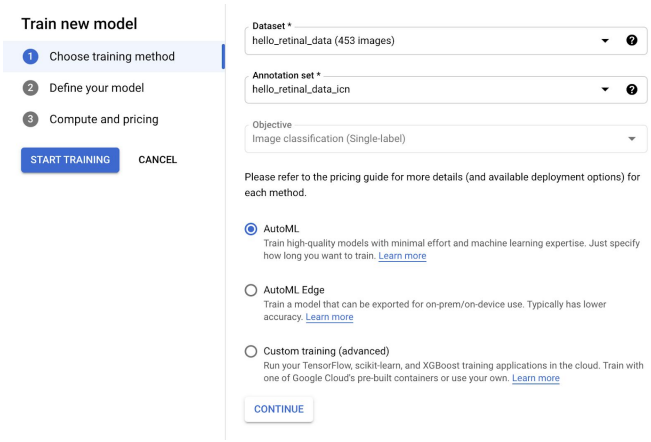
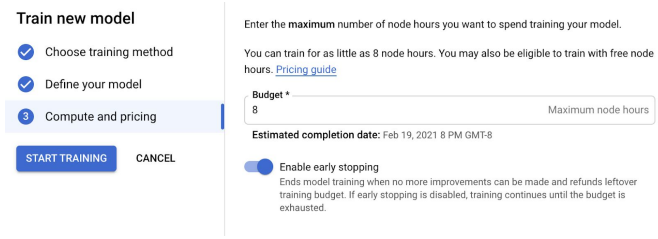
➢ Hello Image
(https://cloud.google.com/ai-platform-unified/docs/tutorials/image-recognition-automl?authuser=2)

Step1: Import image dataset

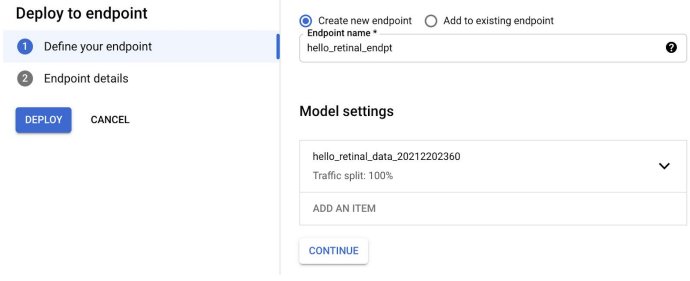
Create a dataset on the AI Platform named *hello_retinal_data* and select the Image Classification (Single-Label)	
Using Diabetic Retinopathy Retinal Images provided by user Sovit Ranjan Rath on Kaggle (https://www.kaggle.com/sovitrath/diabetic-retinopathy-224x224-gaussian-filtered), upload images to the cloud storage to create the retinal dataset.	

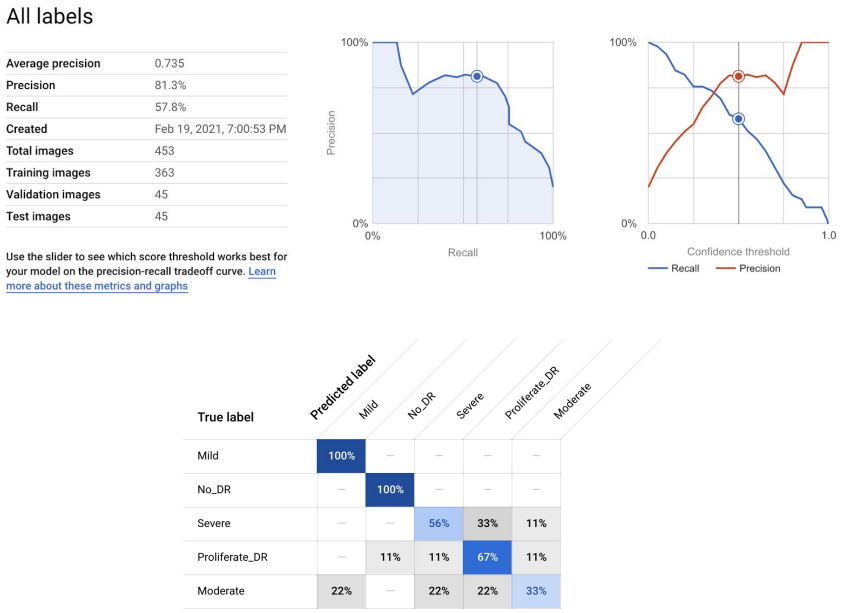
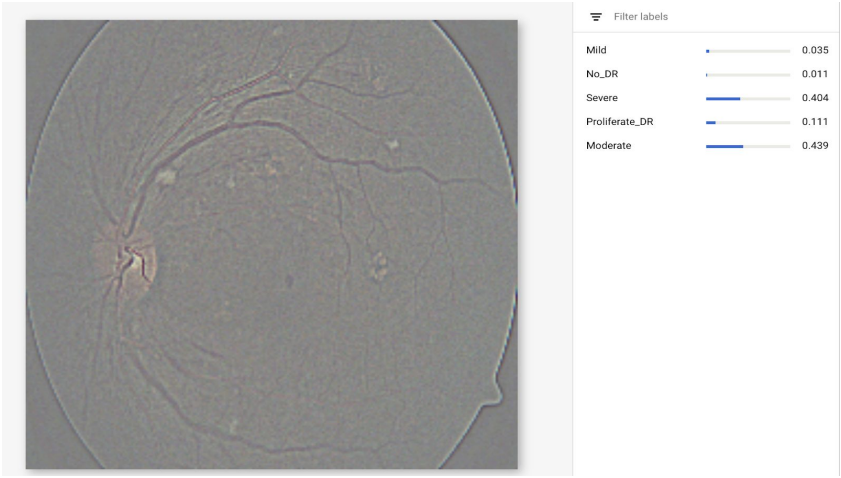
After images finish uploading, select ADD NEW LABEL to assign labels to images. There are 453 total retinal images with 5 classes: Mild, Moderate, No_DR, Proliferate_DR, Severe.	

Step2: Training classification model

Create a model and use the default AutoML training method with 80% training, 10% validation, and 10% test for splitting the data.	
Insert 8 node hours for the node hour budget and start training.	

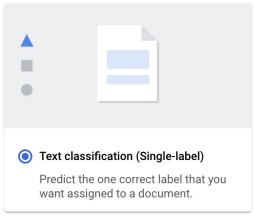
Step3: Deploy model to an endpoint & make a prediction

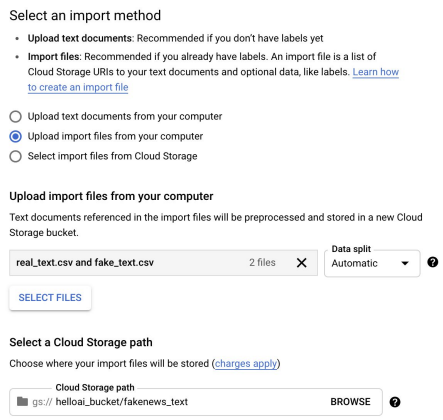
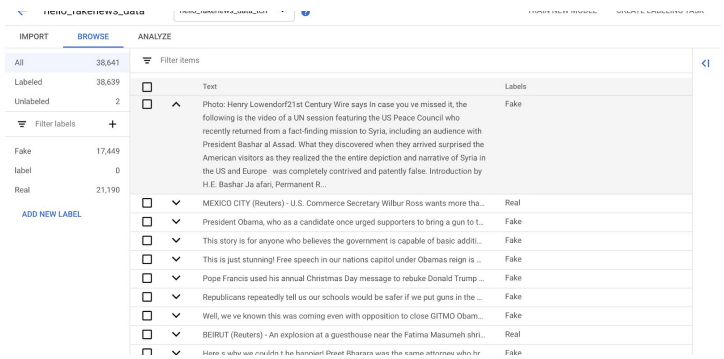
On the Deploy & Test tab under the trained model, define the endpoint and set 1 for the number of compute nodes.	

Summary of completed model with precision 81.3% and 57.8% recall at the 0.5 threshold. Confusion matrix shows that mild and no_dr classifications have a 100% true positive value (all the images within the class were correctly labeled). Since the model trained on a small dataset (for time consuming purpose), further tuning and larger dataset are needed to optimize the model.	
Upload a single picture (moderate retinal image) under Test your model to get a prediction. From the result, the model classified the image as moderate with the highest confidence score of 47.9% and 40.4% for proliferating diabetic retinopathy.	

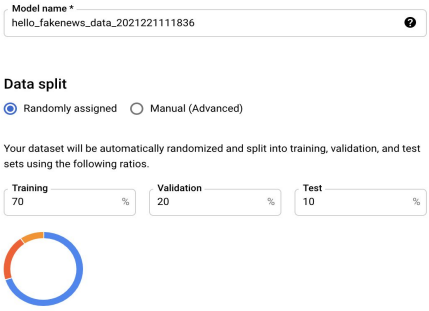
➢ **Hello Text**
(https://cloud.google.com/ai-platform-unified/docs/tutorials/text-classification-automl/dataset?authuser=2)

Step1: Import text dataset

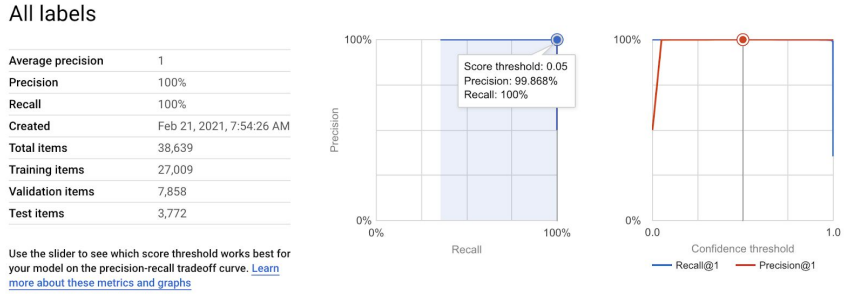
Create a dataset on the AI Platform named *hello_fakenews_data* and select the Text Classification (Single-Label).	

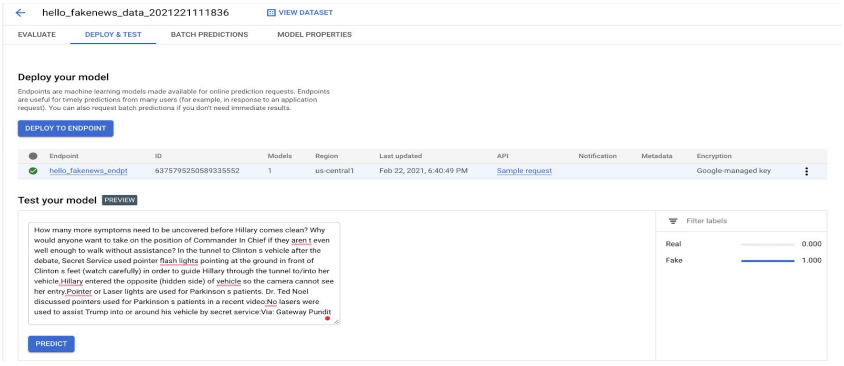
Using Fake and Real News dataset provided by user Clément Bisaillon on Kaggle (https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset?select=True.csv), upload true and false CSV files to the cloud storage to create the text dataset.	
The dataset consists of 38,639 unique texts from political/world news articles with 17,449 of them being fake news and 21,190 being real news.	

Step2: Training classification model

Create a model and use the default AutoML training method. Split the dataset to 70% training, 20% validation, and 10% test.	

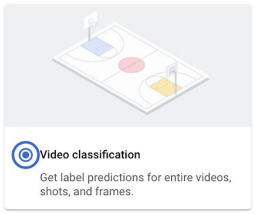
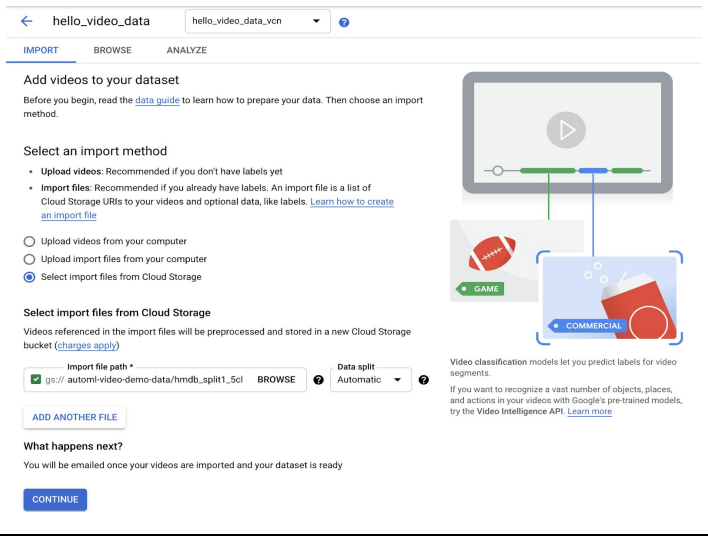
Step3: Deploy model to an endpoint & make a prediction

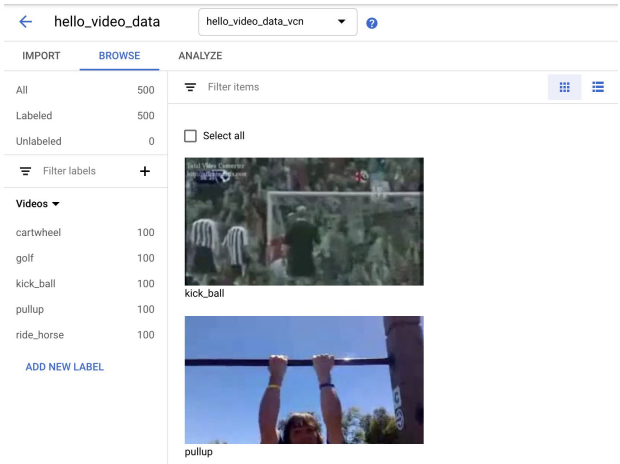
After training for 4 hours, the model has a precision of 99.86 and recall of 100% at the 0.05 score threshold. The confusion matrix shows that the two classification has a 100% true positive score. With a larger dataset, the model trained is optimal.	

	
Deploy to the endpoint (similar to previous model) and test the model with an example of fake text. The model predicted the text to be 100% fake.	

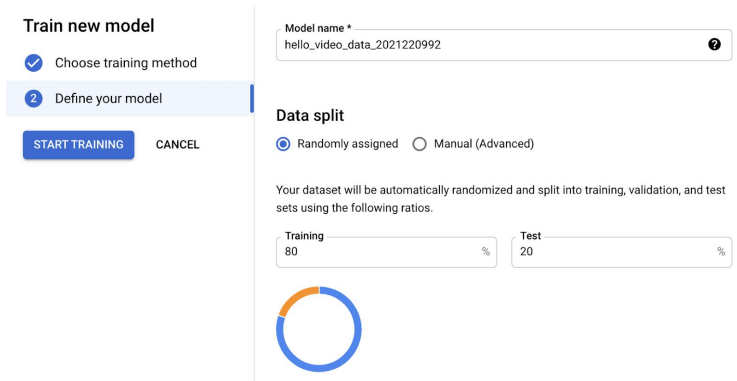
➢ **Hello Video**
(https://cloud.google.com/ai-platform-unified/docs/tutorials/video-classification-automl/dataset?authuser=2)

Step1: Import video dataset

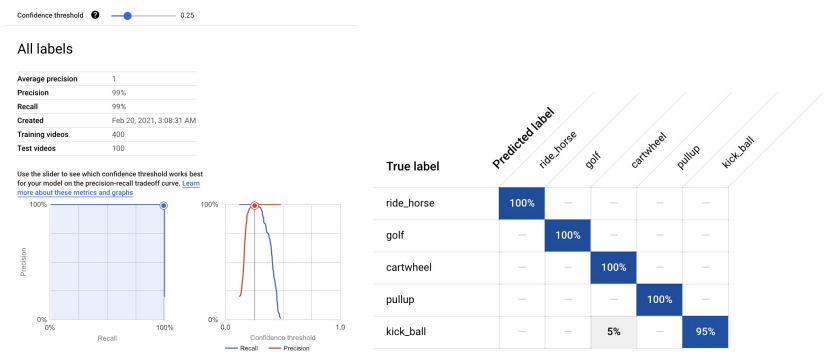
Create a dataset on the AI Platform named *hello_video_data* and select the Video Classification	
Load the quickstart video dataset from Google Cloud Storage with the file path: `[gs://]automl-video-demo-data/hmdb_split1_5classes_all.csv`	

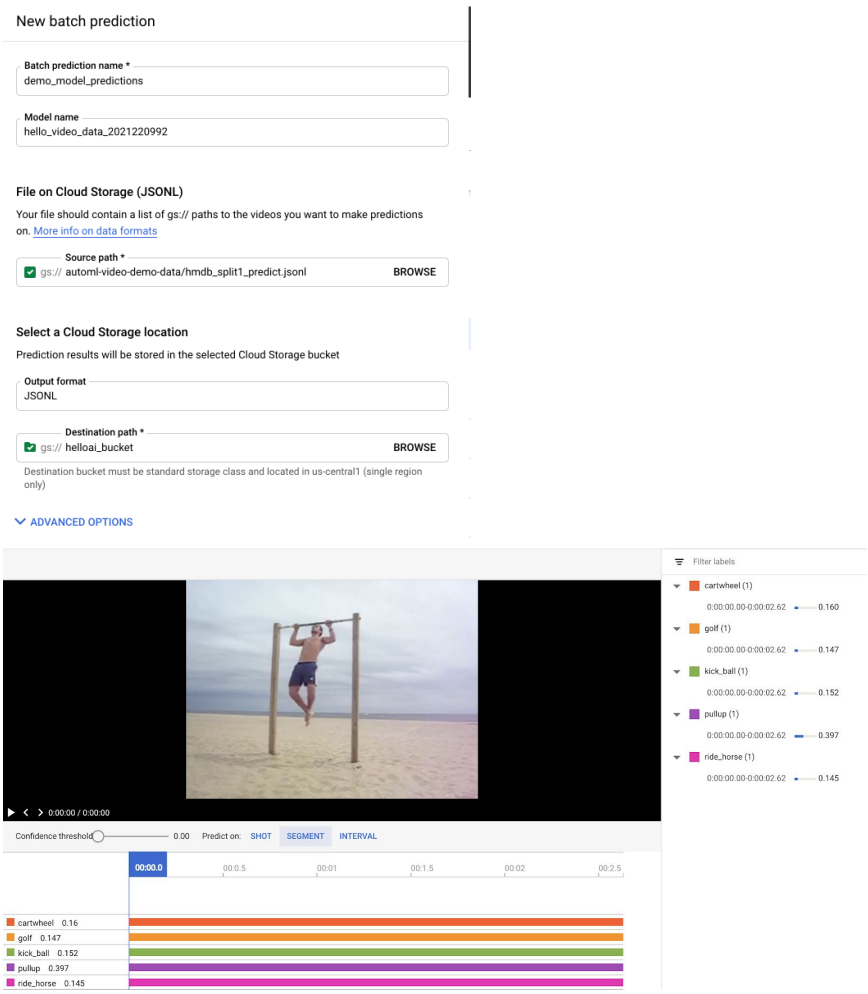
	

Step2: Training classification model

Create a model and use the default AutoML training method with 80% training and 20% test for splitting the data.	

Step3: Deploy model to an endpoint & make a prediction

After the model finishes training, the evaluation page shows that 99% precision and 99% recall is reached when the confidence threshold is at 0.25. The confusion matrix shows that all five classification has 100% true positive results with a 5% false positive on mistaking kick_ball video as cartwheel.	

Using the provided video-demo-data, create a batch prediction. The results shows that for the video of a man doing pull up, the model classified it as pull up with a 39.7% confidence with the second highest confidence being cartwheel with 16%.	

➢ **Hello Custom Training**
(https://cloud.google.com/ai-platform-unified/docs/tutorials/image-recognition-custom/training?authuser=2)

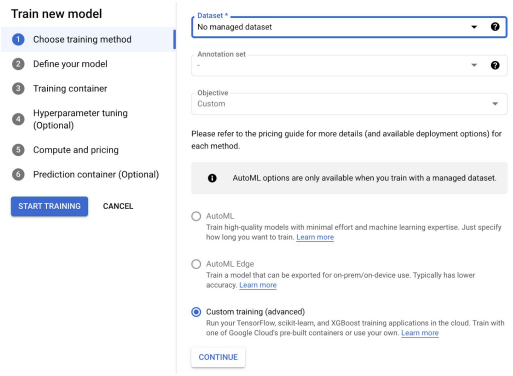
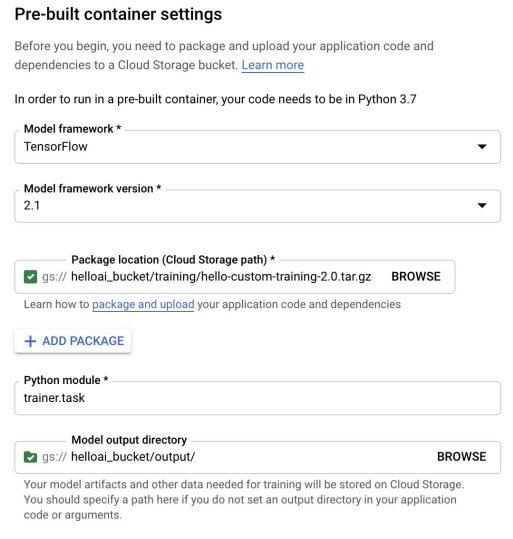
Step1: Setup

Download hello-custom-sample code through the cloud shell	

Step2: Upload training application to Cloud Storage

Inside the task.py include the training application	`trainer/task.py` ``` import logging import os import tensorflow as tf import tensorflow_datasets as tfds IMG_WIDTH = 128 def normalize_img(image): """Normalizes image. * Resizes image to IMG_WIDTH x IMG_WIDTH pixels * Casts values from `uint8` to `float32` * Scales values from [0, 255] to [0, 1] Returns: A tensor with shape (IMG_WIDTH, IMG_WIDTH, 3). (3 color channels) """ image = tf.image.resize_with_pad(image, IMG_WIDTH, IMG_WIDTH) return image / 255. ```
Create a source distribution (package) of the demo-code and upload it to the helloai_bucket	``` haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ gsutil cp dist/hello-custom-training-2.0.tar.gz \ > gs://helloai_bucket/training/ Copying file://dist/hello-custom-training-2.0.tar.gz [Content-Type=application/x-tar]... / [1 files][1.9 KiB/ 1.9 KiB] Operation completed over 1 objects/1.9 KiB. ```

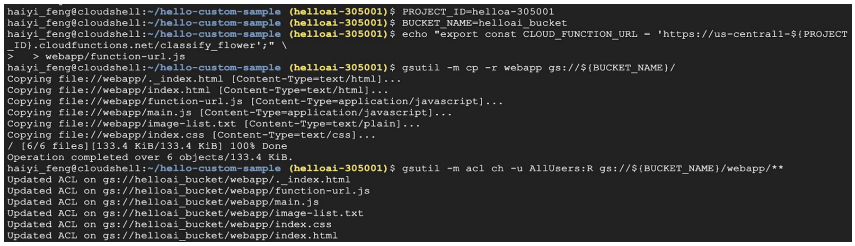
Step3: Run custom training pipeline and create endpoint

Create model name hello_custom	
In the training container app, do the following: • Model framework: TensorFlow 2.1 • Package location: the storage helloai_bucket contains the source distribution previously uploaded • Python module: trainer.task is the name of the module the AI Platform needs to run • Create a new folder inside storage name output	

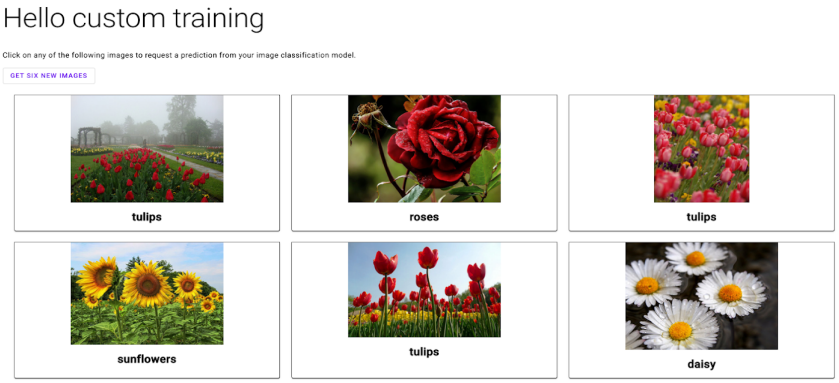
In the Compute and pricing section, select n1-standard-4 as machine type. In the Prediction container, select Pre-built container and ensure the model framework is Tensorflow 2.1 with the correct model director.	**Compute settings** Select the type of virtual machine to use for your worker pool. You can add up to 4 worker pools. To learn about compute costs and how to map your ML framework's roles to specific worker pools, consult the documentation Worker pool 0 Machine type * n1-standard-4, 4 vCPUs, 15 GiB memory
Create an endpoint to serve online predictions (same procedure as previous model)	hello_custom Traffic split * 100 % Compute resources Choose how compute resources will serve prediction traffic to your model • **Autoscaling**: If you set a minimum and maximum, compute nodes will scale to meet traffic demand within those boundaries • **No scaling**: If you only set a minimum, then that number of compute nodes will always run regardless of traffic demand (the maximum will be set to minimum) Once scaling settings are set, they can't be changed unless you redeploy the model. Pricing guide Minimum number of compute nodes * 1 Default is 1. If set to 1 or more, then compute resources will continuously run even without traffic demand. This can increase cost but avoid dropped requests due to node initialization. Maximum number of compute nodes (optional) Enter a number equal to or greater than the minimum nodes. Can reduce costs but may cause reliability issues for high traffic. Machine type * n1-standard-2, 2 vCPUs, 7.5 GiB memory

Step4: Deploy a Cloud Function

Click on Sample Request under API to get the endpoint and project IDs	Endpoint: hello_custom ID: 5020211762750816256 Models: 1 Region: us-central1 Last updated: Feb 22, 2021, 11:36:12 PM API: Sample request Notification: Metadata: Encryption: Google-managed key 4. Create environment variables to hold your endpoint and project IDs, as well as your JSON object. `$ ENDPOINT_ID="5020211762750816256"` `PROJECT_ID="helloai-305001"` `INPUT_DATA_FILE="INPUT-JSON"`
Run the block of code above in the cloud shell and the command needed to deploy the cloud function. (Make sure to enable Cloud Build API for the latter to work.)	Welcome to Cloud Shell! Type "help" to get started. Your Cloud Platform project in this session is set to helloai-305001. Use "gcloud config set project [PROJECT_ID]" to change to a different project. haiyi_feng@cloudshell:~ (helloai-305001)$ cd hello-custom-sample haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ less function/main.py haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ ENDPOINT_ID="5020211762750816256" haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ PROJECT_ID="helloai-305001" haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ INPUT_DATA_FILE="INPUT-JSON" haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ gcloud functions deploy classify_flower \ > --region us-central1 \ > --source=function \ > --runtime=python37 \ > --memory=2048MB \ > --trigger-http \ > --allow-unauthenticated \ > --set-env-vars ENDPOINT_ID=${ENDPOINT_ID} API [cloudfunctions.googleapis.com] not enabled on project [206166460160]. Would you like to enable and retry (this will take a few minutes)? (y/N)?

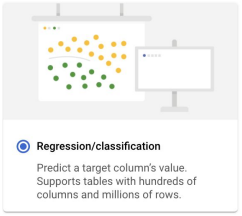
	

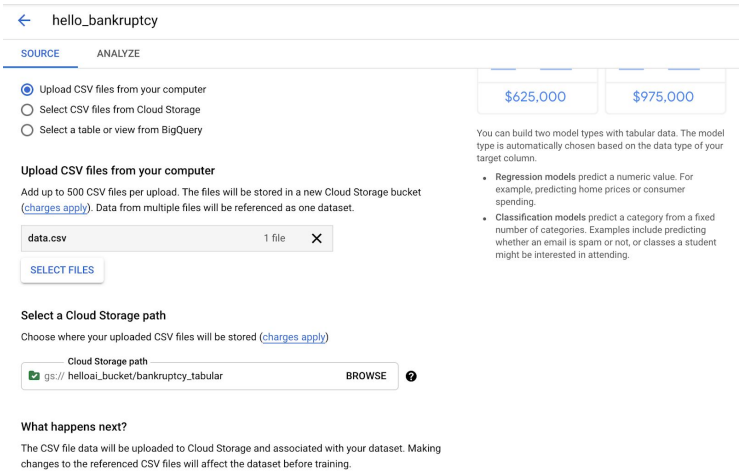
***Step5*: Navigate to Web App**

Open https://storage.googleapis.com/helloai_bucket/webapp/index.html	

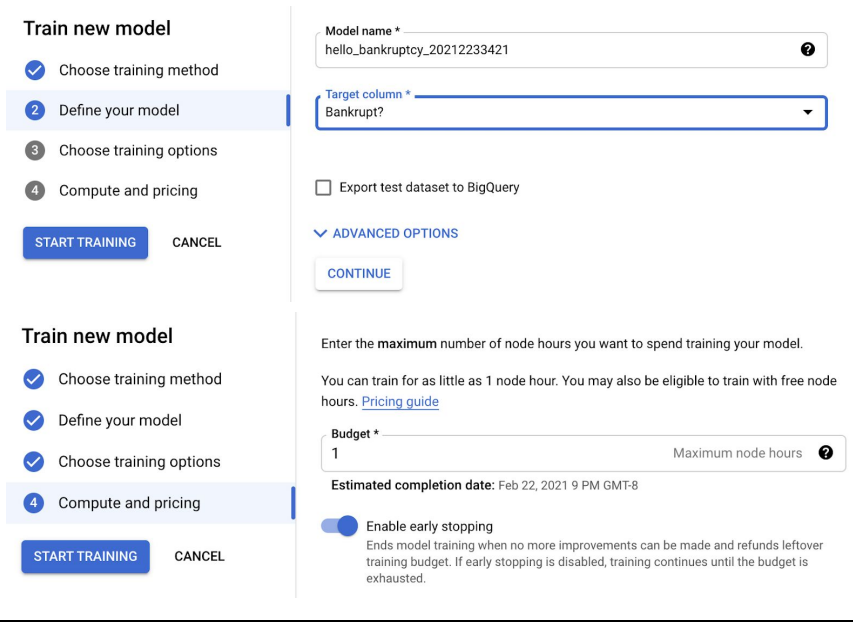
➢ **Hello Structured**
(https://cloud.google.com/ai-platform-unified/docs/tutorials/tabular-automl/dataset-train?authuser=2)

***Step1*: Import tabular dataset**

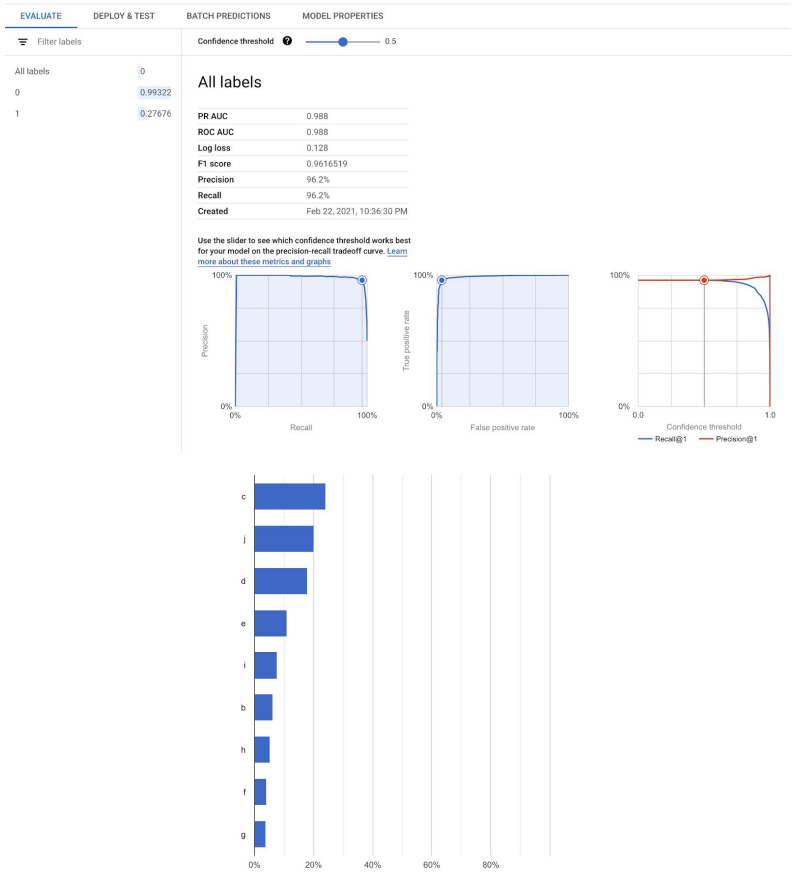
Create a dataset on the AI Platform named *hello_baankrupt_data* and select the Regression/classification under the	
Using Taiwan Bankruptcy dataset provided by user fedesoriano on Kaggle (https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction), Upload the CSV files to the cloud storage to create the tabular dataset. (For time purposes, only the first 10 (out	

of 96) columns were used and the column names were replaced with alphabet due to errors arising at the training stage.)	

Step2: Training classification model

Create a model and use the default AutoML training method with 80% training and 20% test for splitting the data.	

Step3: Deploy model to an endpoint & make a prediction

Summary of the model after training will show that 99.3% of the samples for 0 (not bankrupt) is predicted correctly while the class for 1 (bankrupt) has a 27.6% accurate prediction. A feature importance graph is generated to show that c which corresponds to "ROA(A) before interest and % after tax" has the most influence in training the model with 24%.	
Under the Deploy & Test section, create an endpoint and request a prediction after it finishes deploying. After entering information for one of the companies (not bankrupt), the model predicted this sample to not go bankrupt with a 93% confidence score. Further investigation is needed to understand why the model did not do so well with classifying bankrupt companies.