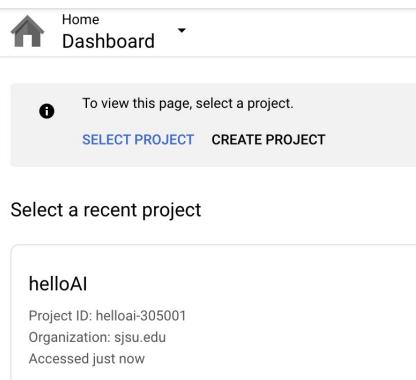
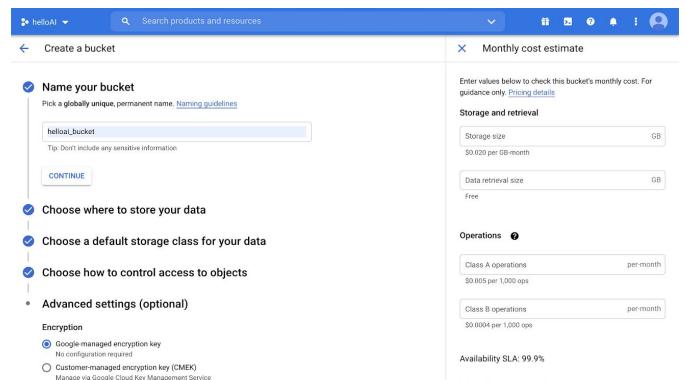


## HW #2.1 Hello AI Platform (Unified)

### Prerequisite:

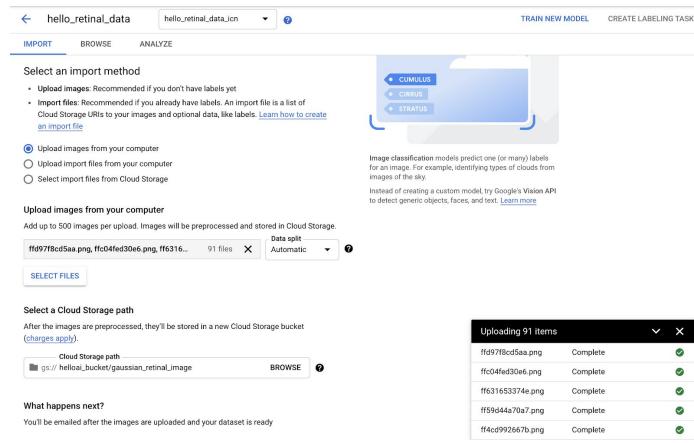
Created a project named <i>helloAI</i>	Created a cloud storage bucket named <i>helloai_bucket</i> to store datasets
	

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>)

#### Step 1: Import image dataset

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

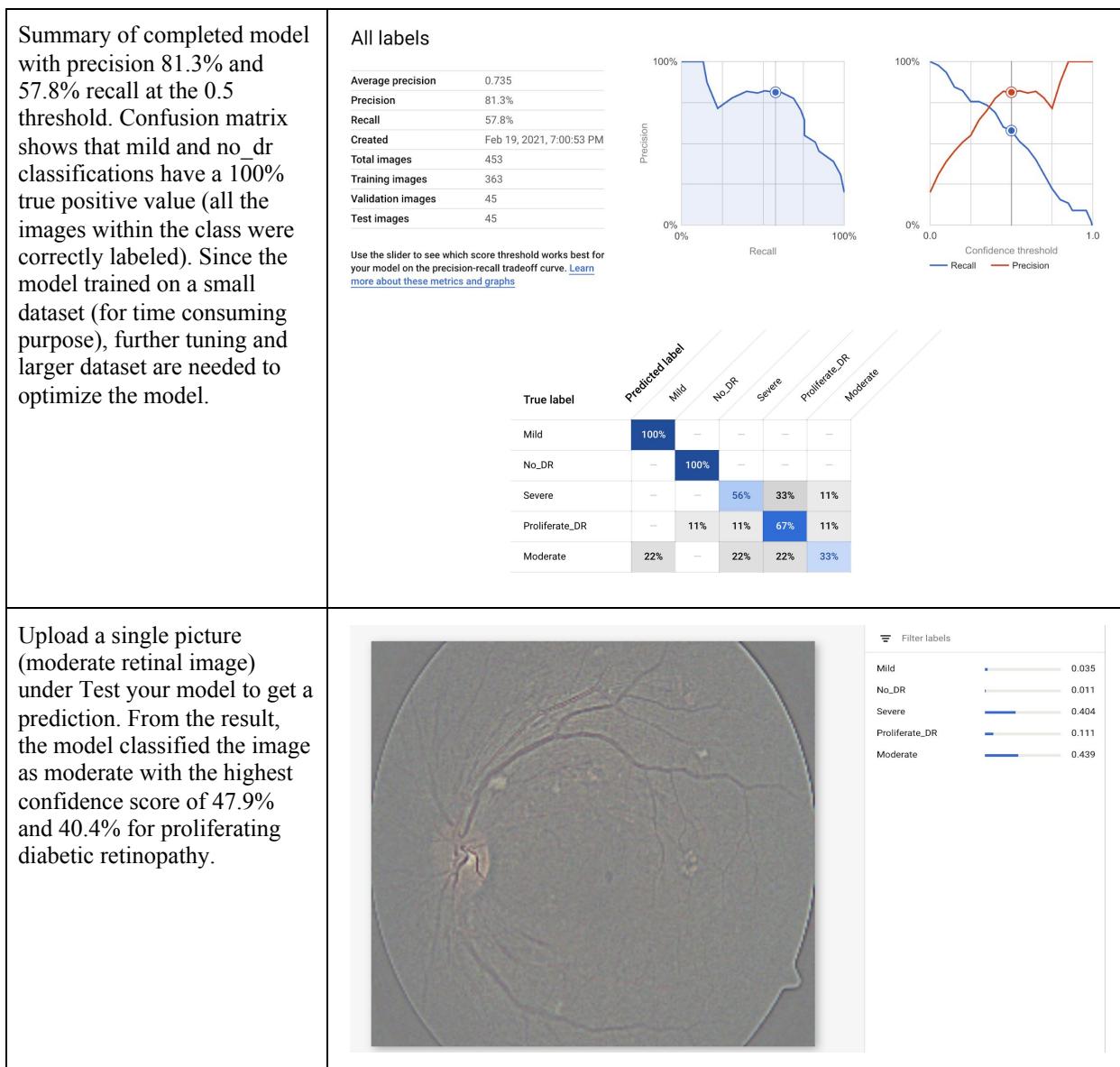
<p>After images finish uploading, select ADD NEW LABEL to assign labels to images. There are 453 total retinal images with 5 classes: Mild, Moderate, No_DR, Proliferate_DR, Severe.</p>	
--	--

### **Step2:** Training classification model

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

### **Step3:** Deploy model to an endpoint & make a prediction

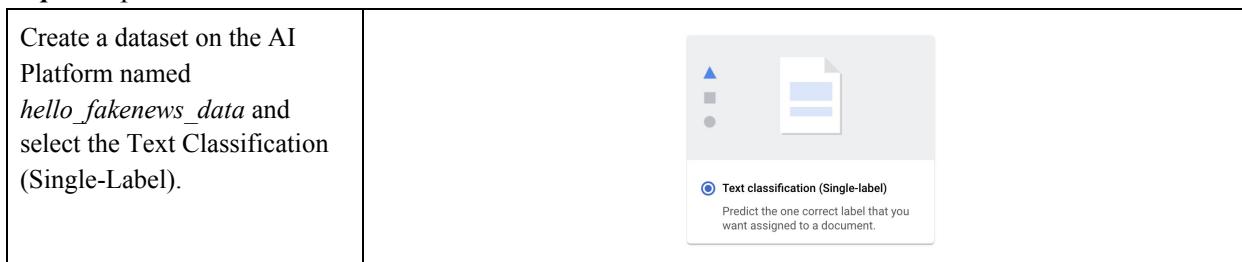
<p>On the Deploy &amp; Test tab under the trained model, define the endpoint and set 1 for the number of compute nodes.</p>	
---	--



## ➤ Hello Text

(<https://cloud.google.com/ai-platform-unified/docs/tutorials/text-classification-automl/dataset?authuser=2> )

### Step 1: Import text dataset



Using Fake and Real News dataset provided by user Clément Bisailon on Kaggle ([https://www.kaggle.com/clm\\_entbisailon/fake-and-real-news-dataset?select=True.csv](https://www.kaggle.com/clm_entbisailon/fake-and-real-news-dataset?select=True.csv)), upload true and false CSV files to the cloud storage to create the text dataset.

Select an import method

- Upload text documents: Recommended if you don't have labels yet
- Import files: Recommended if you already have labels. An import file is a list of Cloud Storage URIs to your text documents and optional data, like labels. [Learn how to create an import file](#)

Upload text documents from your computer  
 Upload import files from your computer  
 Select import files from Cloud Storage

Upload import files from your computer

Text documents referenced in the import files will be preprocessed and stored in a new Cloud Storage bucket.

real\_text.csv and fake\_text.csv 2 files Data split Automatic

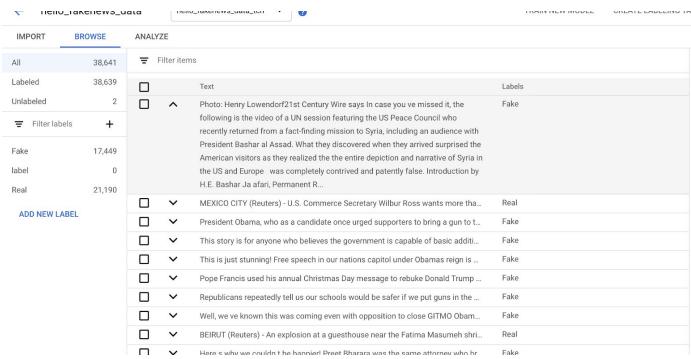
SELECT FILES

Select a Cloud Storage path

Choose where your import files will be stored ([charges apply](#))

Cloud Storage path gs:// helloai\_bucket/fakenews\_text BROWSE

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.

Model name \* hello\_fakenews\_data\_202121111836

Data split

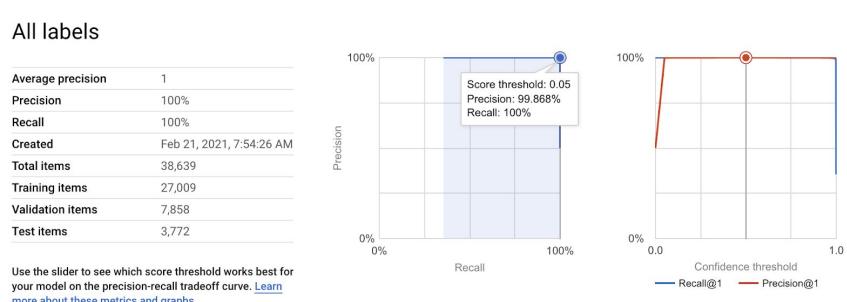
Randomly assigned  Manual (Advanced)

Your dataset will be automatically randomized and split into training, validation, and test sets using the following ratios.

Training 70 % Validation 20 % Test 10 %

### 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.



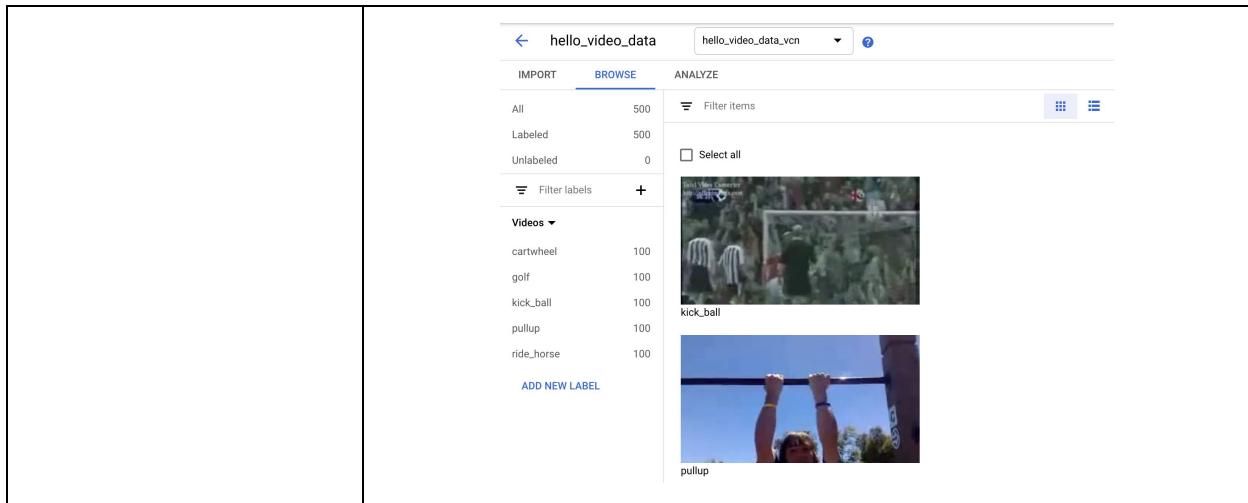
	<table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="2">Predicted label</th> </tr> <tr> <th colspan="2"></th> <th>False</th> <th>Real</th> </tr> <tr> <th rowspan="2">True label</th> <th>Fake</th> <td>100%</td> <td>-</td> </tr> </thead> <tbody> <tr> <th>Real</th> <td>-</td> <td>100%</td> <td></td> </tr> </tbody> </table>			Predicted label				False	Real	True label	Fake	100%	-	Real	-	100%			
		Predicted label																	
		False	Real																
True label	Fake	100%	-																
	Real	-	100%																
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.	<p>hello_fakenews_data_202121111836</p> <p>EVALUATE DEPLOY &amp; TEST BATCH PREDICTIONS MODEL PROPERTIES</p> <p>Deploy your model</p> <p>Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.</p> <p>DEPLOY TO ENDPOINT</p> <table border="1"> <thead> <tr> <th>Endpoint</th> <th>ID</th> <th>Models</th> <th>Region</th> <th>Last updated</th> <th>API</th> <th>Notification</th> <th>Metadata</th> <th>Encryption</th> </tr> </thead> <tbody> <tr> <td>hello_fakenews_endpt</td> <td>637579525058933552</td> <td>1</td> <td>us-central1</td> <td>Feb 22, 2021, 6:40:49 PM</td> <td>Sample request</td> <td></td> <td></td> <td>Google-managed key</td> </tr> </tbody> </table> <p>Test your model</p> <p>PREDICT</p> <p>How many more symptoms need to be uncovered before Hillary comes clean? Why would anyone want to take on the challenge of being in Clinton's shoes? Even with all the evidence, it's hard to believe in the turn to Clinton's side. In the debate, Secret Service used pointer flash lights pointing at the ground in front of Clinton's feet (watch carefully) in order to guide Hillary through the tunnel to/into her vehicle. It was the opposite of what you'd expect so that people could not see her entry. Pointer or Laser pointers are used for Parkinson's patients. Dr. Ted Noell discussed pointers used for Parkinson's patients in a recent video. No lasers were used to assist Trump into or around his vehicle by secret service via Gateway Pundit.</p> <p>Filter labels</p> <p>Real 0.000</p> <p>Fake 1.000</p>	Endpoint	ID	Models	Region	Last updated	API	Notification	Metadata	Encryption	hello_fakenews_endpt	637579525058933552	1	us-central1	Feb 22, 2021, 6:40:49 PM	Sample request			Google-managed key
Endpoint	ID	Models	Region	Last updated	API	Notification	Metadata	Encryption											
hello_fakenews_endpt	637579525058933552	1	us-central1	Feb 22, 2021, 6:40:49 PM	Sample request			Google-managed key											

## ➤ Hello Video

(<https://cloud.google.com/ai-platform-unified/docs/tutorials/video-classification-automl/dataset?authuser=2>)

### Step 1: Import video dataset

Create a dataset on the AI Platform named <i>hello_video_data</i> and select the Video Classification	<p>Video classification</p> <p>Get label predictions for entire videos, shots, and frames.</p>
Load the quickstart video dataset from Google Cloud Storage with the file path: [gs://]automl-video-demo-data/hmdb_split1_5classes_all.csv	<p>hello_video_data</p> <p>hello_video_data.vcn</p> <p>IMPORT BROWSE ANALYZE</p> <p>Add videos to your dataset</p> <p>Select an import method</p> <ul style="list-style-type: none"> <li>Upload videos: Recommended if you don't have labels yet</li> <li>Import files: Recommended if you already have labels. An import file is a list of Cloud Storage URLs to your videos and optional data, like labels. <a href="#">Learn how to create an import file</a></li> </ul> <p>Select import files from Cloud Storage</p> <p>Import file path *</p> <p>gs://automl-video-demo-data/hmdb_split1_5cl</p> <p>BROWSE</p> <p>Data split: Automatic</p> <p>ADD ANOTHER FILE</p> <p>What happens next?</p> <p>You will be emailed once your videos are imported and your dataset is ready</p> <p>CONTINUE</p>



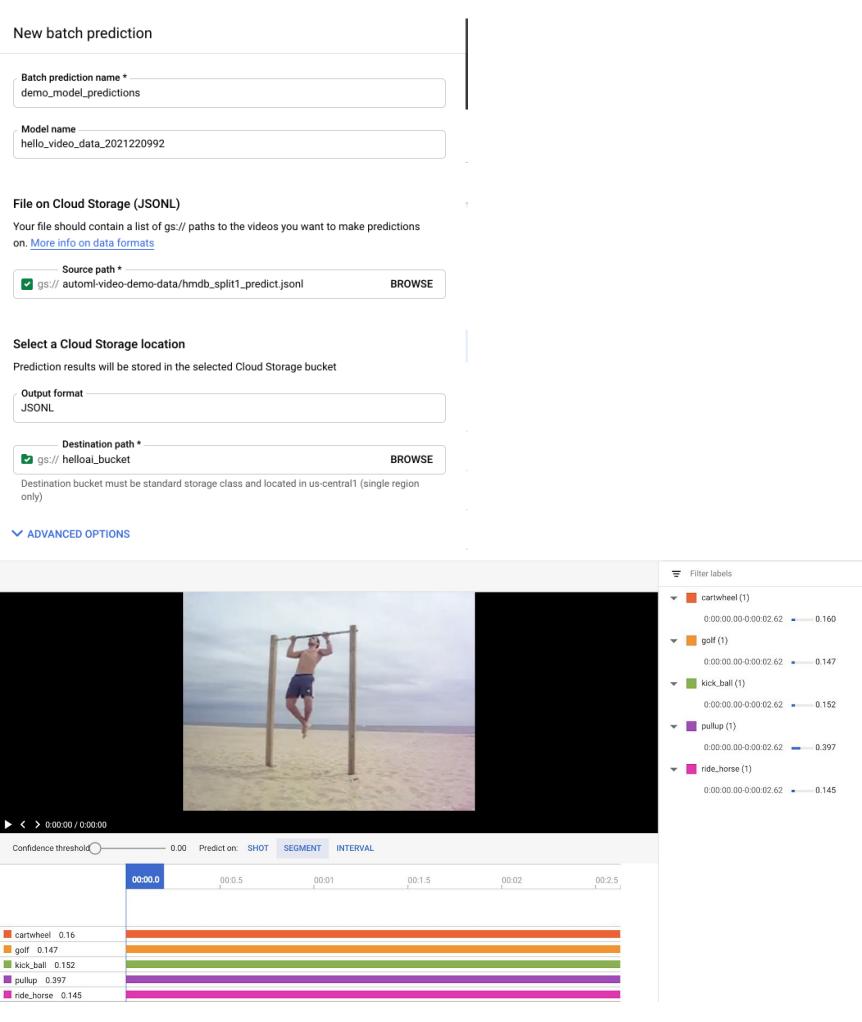
### **Step2:** Training classification model

<p>Create a model and use the default AutoML training method with 80% training and 20% test for splitting the data.</p>	<p><b>Train new model</b></p> <p><input checked="" type="checkbox"/> Choose training method</p> <p><b>2 Define your model</b></p> <p><b>START TRAINING</b>    CANCEL</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">Model name * hello_video_data_2021220992</div> <p><b>Data split</b></p> <p><input checked="" type="radio"/> Randomly assigned   <input type="radio"/> Manual (Advanced)</p> <p>Your dataset will be automatically randomized and split into training, validation, and test sets using the following ratios.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <span>Training</span> <input type="text" value="80"/> <span>%</span> <span>Test</span> <input type="text" value="20"/> <span>%</span> </div>
---	---

### **Step3:** Deploy model to an endpoint & make a prediction

<p>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.</p>	<p>Confidence threshold: 0.25</p> <p><b>All labels</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Average precision</th> <th>1</th> </tr> </thead> <tbody> <tr> <td>Precision</td> <td>99%</td> </tr> <tr> <td>Recall</td> <td>99%</td> </tr> <tr> <td>Created</td> <td>Feb 20, 2021, 3:08:31 AM</td> </tr> <tr> <td>Training videos</td> <td>400</td> </tr> <tr> <td>Test videos</td> <td>100</td> </tr> </tbody> </table> <p>Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve. <a href="#">Learn more about these metrics and graphs</a></p> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> <div style="text-align: center;"> <p>Precision</p> <p>Recall</p> </div> <div style="text-align: center;"> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2"></th> <th colspan="5">Predicted label</th> </tr> <tr> <th rowspan="2">True label</th> <th rowspan="2"></th> <th>ride_horse</th> <th>golf</th> <th>cartwheel</th> <th>pullup</th> <th>kick_ball</th> </tr> </thead> <tbody> <tr> <td>ride_horse</td> <td>100%</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> <tr> <td>golf</td> <td>—</td> <td>100%</td> <td>—</td> <td>—</td> <td>—</td> </tr> <tr> <td>cartwheel</td> <td>—</td> <td>—</td> <td>100%</td> <td>—</td> <td>—</td> </tr> <tr> <td>pullup</td> <td>—</td> <td>—</td> <td>—</td> <td>100%</td> <td>—</td> </tr> <tr> <td>kick_ball</td> <td>—</td> <td>—</td> <td>5%</td> <td>—</td> <td>95%</td> </tr> </tbody> </table> </div> </div>	Average precision	1	Precision	99%	Recall	99%	Created	Feb 20, 2021, 3:08:31 AM	Training videos	400	Test videos	100			Predicted label					True label		ride_horse	golf	cartwheel	pullup	kick_ball	ride_horse	100%	—	—	—	—	golf	—	100%	—	—	—	cartwheel	—	—	100%	—	—	pullup	—	—	—	100%	—	kick_ball	—	—	5%	—	95%
Average precision	1																																																								
Precision	99%																																																								
Recall	99%																																																								
Created	Feb 20, 2021, 3:08:31 AM																																																								
Training videos	400																																																								
Test videos	100																																																								
		Predicted label																																																							
True label		ride_horse	golf	cartwheel	pullup	kick_ball																																																			
		ride_horse	100%	—	—	—	—																																																		
golf	—	100%	—	—	—																																																				
cartwheel	—	—	100%	—	—																																																				
pullup	—	—	—	100%	—																																																				
kick_ball	—	—	5%	—	95%																																																				

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

```
haiyi_feng@cloudshell:~ (helloai-305001)$ gsutil cp gs://cloud-samples-data/ai-platform/hello-custom/hello-custom-sample-v1beta1.tar.gz - | tar -xvz
hello-custom-sample/
hello-custom-sample/webapp/
hello-custom-sample/webapp/function/
hello-custom-sample/setup.py
hello-custom-sample/trainer/
hello-custom-sample/trainer/task.py
hello-custom-sample/trainer/trainer.py
hello-custom-sample/webapp/function/requirements.txt
hello-custom-sample/function/main.py
hello-custom-sample/webapp/_index.html
hello-custom-sample/webapp/index.html
hello-custom-sample/webapp/function/list.txt
hello-custom-sample/webapp/index.txt
hello-custom-sample/webapp/main.js
hello-custom-sample/webapp/function-url.js
haiyi_feng@cloudshell:~ (helloai-305001)$ cd hello-custom-sample
haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ ls -lpR
:
total 16
```

**Step2:** Upload training application to Cloud Storage

Inside the task.py include the training application	<pre>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.  trainer/task.py</pre>
Create a source distribution (package) of the demo-code and upload it to the helloai_bucket	<pre>haiyi.feng@cloudshell:~/helloai-sample (helloai-305001)\$ gsutil cp dist/hello-custom-training-2.0.tar.gz \ &gt; 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.</pre>

**Step3:** Run custom training pipeline and create endpoint

Create model name hello_custom	
In the training container app, do the following: <ul style="list-style-type: none"> <li>Model framework: TensorFlow 2.1</li> <li>Package location: the storage helloai_bucket contains the source distribution previously uploaded</li> <li>Python module: trainer.task is the name of the module the AI Platform needs to run</li> <li>Create a new folder inside storage name output</li> </ul>	<p><b>Pre-built container settings</b></p> <p>Before you begin, you need to package and upload your application code and dependencies to a Cloud Storage bucket. <a href="#">Learn more</a></p> <p>In order to run in a pre-built container, your code needs to be in Python 3.7</p> <p>Model framework * TensorFlow</p> <p>Model framework version * 2.1</p> <p>Package location (Cloud Storage path) * <input checked="" type="checkbox"/> gs:// helloai_bucket/training/hello-custom-training-2.0.tar.gz <a href="#">BROWSE</a></p> <p>Learn how to <a href="#">package and upload</a> your application code and dependencies</p> <p>+ ADD PACKAGE</p> <p>Python module * trainer.task</p> <p>Model output directory <input checked="" type="checkbox"/> gs:// helloai_bucket/output/ <a href="#">BROWSE</a></p> <p>Your model artifacts and other data needed for training will be stored on Cloud Storage. You should specify a path here if you do not set an output directory in your application code or arguments.</p>

<p>In the Compute and pricing section, select n1-standard-4 as machine type.</p> <p>In the Prediction container, select Pre-built container and ensure the model framework is Tensorflow 2.1 with the correct model director.</p>	<p><b>Compute settings</b></p> <p>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 <a href="#">documentation</a></p> <p><b>Worker pool 0</b></p> <p><b>Machine type *</b> n1-standard-4, 4 vCPUs, 15 GiB memory</p>
<p>Create an endpoint to serve online predictions (same procedure as previous model)</p>	<p><b>hello_custom</b></p> <p>Traffic split * 100 % <a href="#">?</a></p> <p><b>Compute resources</b></p> <p>Choose how compute resources will serve prediction traffic to your model</p> <ul style="list-style-type: none"> <li>Autoscaling: If you set a minimum and maximum, compute nodes will scale to meet traffic demand within those boundaries</li> <li>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)</li> </ul> <p>Once scaling settings are set, they can't be changed unless you redeploy the model. <a href="#">Pricing guide</a></p> <p><b>Minimum number of compute nodes *</b> 1</p> <p>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.</p> <p><b>Maximum number of compute nodes (optional)</b></p> <p>Enter a number equal to or greater than the minimum nodes. Can reduce costs but may cause reliability issues for high traffic.</p> <p><b>Machine type *</b> n1-standard-2, 2 vCPUs, 7.5 GiB memory</p>

#### Step4: Deploy a Cloud Function

<p>Click on Sample Request under API to get the endpoint and project IDs</p>	<table border="1"> <thead> <tr> <th>Endpoint</th><th>ID</th><th>Models</th><th>Region</th><th>Last updated</th><th>API</th><th>Notification</th><th>Metadata</th><th>Encryption</th></tr> </thead> <tbody> <tr> <td>hello_custom</td><td>5020211762750816256</td><td>1</td><td>us-central1</td><td>Feb 22, 2021, 11:36:12 PM</td><td><a href="#">Sample request</a></td><td></td><td></td><td>Google-managed key</td></tr> </tbody> </table> <p>4. Create environment variables to hold your endpoint and project IDs, as well as your JSON object.</p> <pre>\$ ENDPOINT_ID="5020211762750816256 PROJECT_ID="helloai-305001" INPUT_DATA_FILE="INPUT-JSON"</pre>	Endpoint	ID	Models	Region	Last updated	API	Notification	Metadata	Encryption	hello_custom	5020211762750816256	1	us-central1	Feb 22, 2021, 11:36:12 PM	<a href="#">Sample request</a>			Google-managed key
Endpoint	ID	Models	Region	Last updated	API	Notification	Metadata	Encryption											
hello_custom	5020211762750816256	1	us-central1	Feb 22, 2021, 11:36:12 PM	<a href="#">Sample request</a>			Google-managed key											
<p>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.)</p>	<pre>Welcome to Cloud Shell! Type "help" to get started. Your Cloud Platform project in this session is set to <b>helloai-305001</b>. Use "gcloud config set project [PROJECT_ID]" to change to a different project. haiyi_feng@cloudshell:~/hello-custom-sample (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 \ &gt; --region us-central1 \ &gt; --source=funciton \ &gt; --runtime=python37 \ &gt; --memory=2048MB \ &gt; --trigger=http \ &gt; --allow-unauthenticated \ &gt; --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) ?</pre>																		

```
haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ PROJECT_ID=helloai-305001
haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ gsutil -m cp -r webapp gs://${BUCKET_NAME}/
> > webapp/function-url.js
haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ gsutil -m acl ch -u AllUsers:R gs://${BUCKET_NAME}/webapp/*
Copying file://webapp/_index.html [Content-Type=text/html]...
Copying file://webapp/index.html [Content-Type=text/html]...
Copying file://webapp/function-url.js [Content-Type=application/javascript]...
Copying file://webapp/main.js [Content-Type=application/javascript]...
Copying file://webapp/image-list.txt [Content-Type=text/plain]...
Copying file://webapp/index.css [Content-Type=text/css]...
Copied 6 files|133.4 KB| 0:00 0:00
Operation completed over 6 objects/133.4 KB.
haiyi_feng@cloudshell:~/hello-custom-sample (helloai-305001)$ gsutil -m acl ch -u AllUsers:R gs://${BUCKET_NAME}/webapp/*
Updated ACL on gs://helloai_bucket/webapp/_index.html
Updated ACL on gs://helloai_bucket/webapp/index.html
Updated ACL on gs://helloai_bucket/webapp/main.js
Updated ACL on gs://helloai_bucket/webapp/image-list.txt
Updated ACL on gs://helloai_bucket/webapp/index.css
Updated ACL on gs://helloai_bucket/webapp/index.css
Updated ACL on gs://helloai_bucket/webapp/index.htm
```

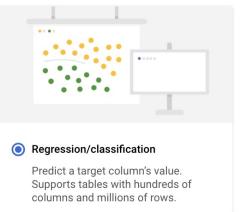
### Step5: Navigate to Web App

Open <a href="https://storage.googleapis.com/helloai_bucket/webapp/index.html">https://storage.googleapis.com/helloai_bucket/webapp/index.html</a>	<p>Hello custom training</p> <p>Click on any of the following images to request a prediction from your image classification model.</p> <p><a href="#">GET SIX NEW IMAGES</a></p> <table border="1"><tbody><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></tbody></table>						
							
							
							

### ➤ 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 <i>hello_baankrupt_data</i> and select the Regression/classification under the	 <p>Regression/classification Predict a target column's value. Supports tables with hundreds of columns and millions of rows.</p>
Using Taiwan Bankruptcy dataset provided by user fedesoriano on Kaggle ( <a href="https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction">https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction</a> ), 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.)

The screenshot shows the 'hello\_bankruptcy' dataset in the Google Cloud AutoML interface. It includes options to upload CSV files from the computer or select from Cloud Storage, and a preview of two values: \$625,000 and \$975,000. A sidebar provides information about model types: Regression models predict a numeric value (e.g., home prices), and Classification models predict a category from a fixed number of categories (e.g., spam or not).

### **Step2:** Training classification model

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

The screenshot shows the 'Train new model' interface in the Google Cloud AutoML interface. It displays the 'Define your model' step. The right panel shows fields for 'Model name' (hello\_bankruptcy\_20212233421), 'Target column' (Bankrupt?), 'Budget' (1), and 'Estimated completion date' (Feb 22, 2021 9 PM GMT-8). Advanced options like 'Enable early stopping' are also shown.

**Step3:** Deploy model to an endpoint & make a prediction

<p>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%.</p>	<p>EVALUATE DEPLOY &amp; TEST BATCH PREDICTIONS MODEL PROPERTIES</p> <p>All labels: 0 0: 0.99322 1: 0.27676</p> <p>Confidence threshold: 0.5</p> <p>All labels</p> <p>PR AUC: 0.988 ROC AUC: 0.988 Log loss: 0.128 F1 score: 0.9616519 Precision: 96.2% Recall: 96.2% Created: Feb 22, 2021, 10:36:30 PM</p> <p>Precision vs Recall: A step function starting at 100% precision for 0 recall, dropping to ~72% at 100% recall.</p> <p>True positive rate vs False positive rate: A step function starting at 100% true positive rate for 0 false positive rate, dropping to ~72% at 100% false positive rate.</p> <p>Confidence threshold vs Recall@1, Precision@1: A plot showing Recall@1 (blue line) and Precision@1 (red line) as the confidence threshold varies from 0.0 to 1.0. Both curves are nearly horizontal at 100% until a threshold of approximately 0.7, where they drop sharply to about 72%.</p> <p>Feature importance bar chart:</p> <table border="1"> <thead> <tr> <th>Feature</th> <th>Importance</th> </tr> </thead> <tbody> <tr><td>c</td><td>~0.24</td></tr> <tr><td>j</td><td>~0.18</td></tr> <tr><td>d</td><td>~0.15</td></tr> <tr><td>e</td><td>~0.12</td></tr> <tr><td>i</td><td>~0.08</td></tr> <tr><td>b</td><td>~0.05</td></tr> <tr><td>h</td><td>~0.04</td></tr> <tr><td>f</td><td>~0.03</td></tr> <tr><td>g</td><td>~0.02</td></tr> </tbody> </table>	Feature	Importance	c	~0.24	j	~0.18	d	~0.15	e	~0.12	i	~0.08	b	~0.05	h	~0.04	f	~0.03	g	~0.02															
Feature	Importance																																			
c	~0.24																																			
j	~0.18																																			
d	~0.15																																			
e	~0.12																																			
i	~0.08																																			
b	~0.05																																			
h	~0.04																																			
f	~0.03																																			
g	~0.02																																			
<p>Under the Deploy &amp; 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.</p>	<p>Test your model <span style="background-color: #f0f0f0; border: 1px solid #ccc; padding: 2px 5px;">PREVIEW</span></p> <table border="1"> <thead> <tr> <th>Feature column name</th> <th>Type</th> <th>Required or optional</th> <th>Value</th> <th>Local feature importance</th> </tr> </thead> <tbody> <tr><td>b</td><td>Text</td><td>Required</td><td>0.450787305611076</td><td>-0.01468221843242645</td></tr> <tr><td>c</td><td>Text</td><td>Required</td><td>0.510030527692979</td><td>0.07726727798581123</td></tr> <tr><td>d</td><td>Text</td><td>Required</td><td>0.508592537073719</td><td>-0.02790813520550728</td></tr> <tr><td>e</td><td>Text</td><td>Required</td><td>0.596808832643884</td><td>-0.02500594034790993</td></tr> <tr><td>f</td><td>Text</td><td>Required</td><td>0.597097104311103</td><td>-0.006588339805603027</td></tr> <tr><td>g</td><td>Text</td><td>Required</td><td>0.998974518370276</td><td>0.001693006604909897</td></tr> </tbody> </table> <p>Predict label: 0</p> <p>Prediction result: 0 Confidence score: 0.9358543157577515</p>	Feature column name	Type	Required or optional	Value	Local feature importance	b	Text	Required	0.450787305611076	-0.01468221843242645	c	Text	Required	0.510030527692979	0.07726727798581123	d	Text	Required	0.508592537073719	-0.02790813520550728	e	Text	Required	0.596808832643884	-0.02500594034790993	f	Text	Required	0.597097104311103	-0.006588339805603027	g	Text	Required	0.998974518370276	0.001693006604909897
Feature column name	Type	Required or optional	Value	Local feature importance																																
b	Text	Required	0.450787305611076	-0.01468221843242645																																
c	Text	Required	0.510030527692979	0.07726727798581123																																
d	Text	Required	0.508592537073719	-0.02790813520550728																																
e	Text	Required	0.596808832643884	-0.02500594034790993																																
f	Text	Required	0.597097104311103	-0.006588339805603027																																
g	Text	Required	0.998974518370276	0.001693006604909897																																