

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

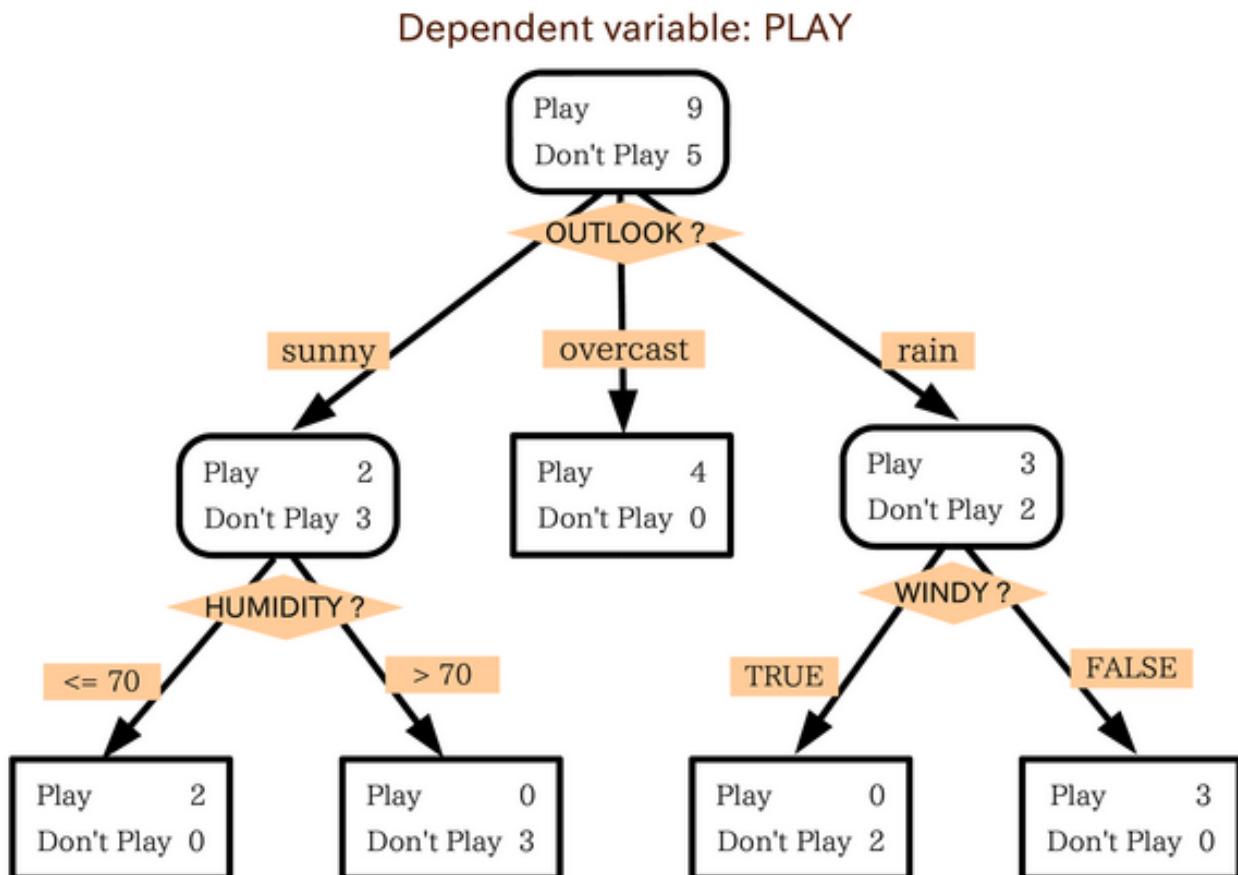
Reference: <https://codelabs.developers.google.com/vertex-xgb-wit#0>

Objective:

- Train an XGBoost model on a public mortgage dataset in a hosted notebook
- Analyze the model using the What-if Tool
- Deploy the XGBoost model to Vertex AI

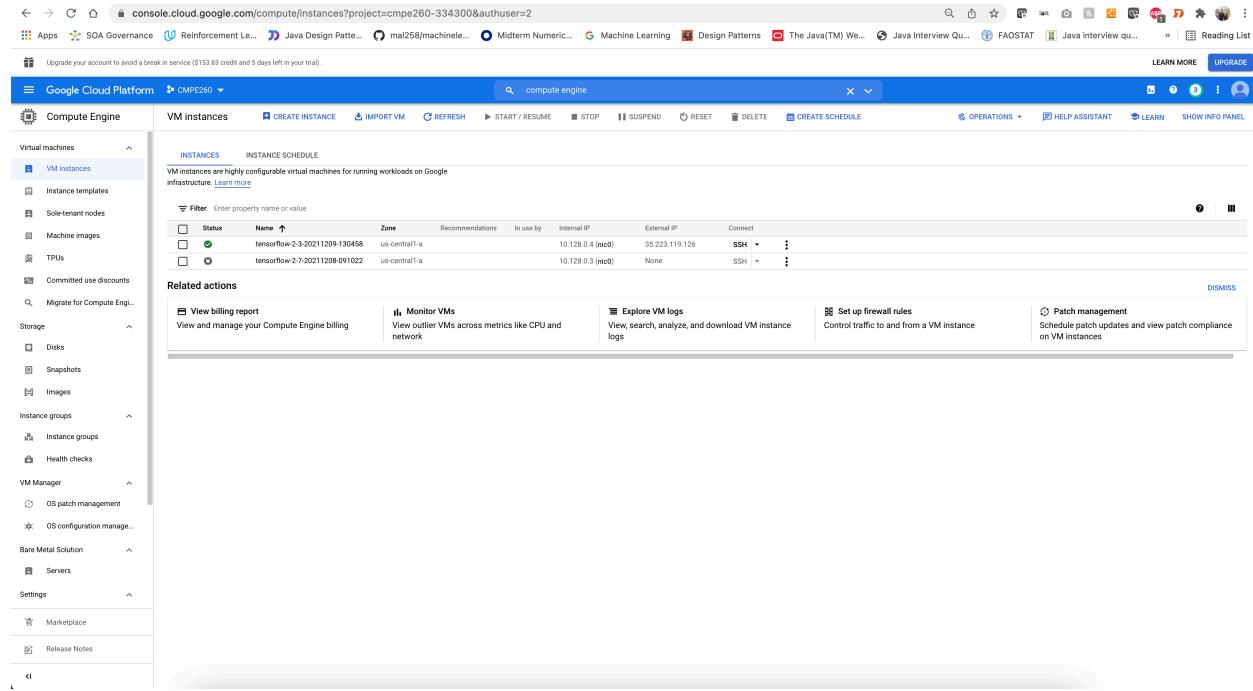
XGBoost is a machine learning framework that uses [decision trees](#) and [gradient boosting](#) to build predictive models. It works by ensembling multiple decision trees together based on the score associated with different leaf nodes in a tree.

The diagram below is a [visualization](#) of a simple decision tree model that evaluates whether a sports game should be played based on the weather forecast:



Why are we using XGBoost for this model? While traditional neural networks have been shown to perform best on unstructured data like images and text, decision trees often perform extremely well on structured data like the mortgage dataset we'll be using in this codelab.

Step 1: Setup Cloud environment

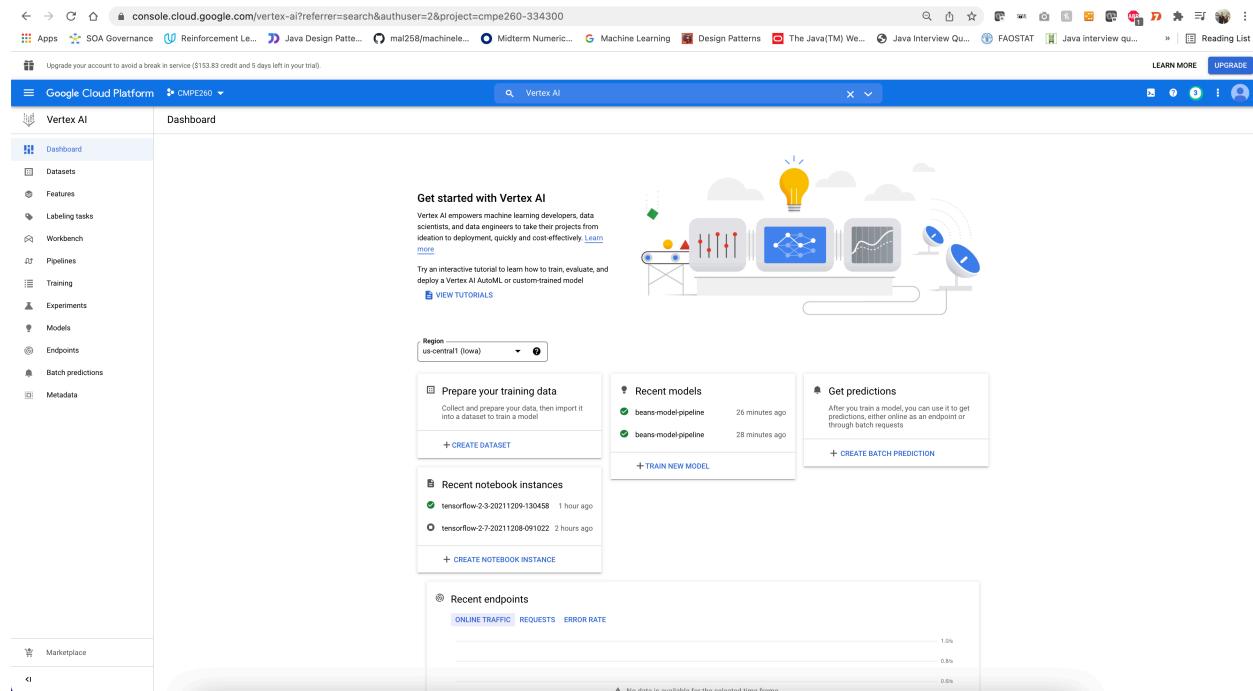


The screenshot shows the Google Cloud Platform Compute Engine VM Instances page. The left sidebar includes sections for Virtual machines, Storage, Instance groups, VM Manager, Bare Metal Solution, and Settings. The main content area displays two VM instances:

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
Green	tensorflow-2-3-20211209-130458	us-central1-a			10.128.0.4 (m2d)	35.233.119.126	SSH
Grey	tensorflow-2-7-20211208-091022	us-central1-a			10.128.0.3 (m2d)	None	SSH

Related actions include View billing report, Monitor VMs, Explore VM logs, Set up firewall rules, and Patch management.

Step 2: Enable the Vertex AI API



The screenshot shows the Google Cloud Platform Vertex AI Dashboard. The left sidebar includes sections for Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main content area features a "Get started with Vertex AI" section with a brief introduction and a "VIEW TUTORIALS" button. It also displays recent activity:

- Recent models:
 - beans-model-pipeline (26 minutes ago)
 - beans-model-pipeline (28 minutes ago)
- Recent notebook instances:
 - tensorflow-2-3-20211209-130458 (1 hour ago)
 - tensorflow-2-7-20211208-091022 (2 hours ago)
- Recent endpoints:
 - ONLINE TRAFFIC, REQUESTS, ERROR RATE (No data is available for the selected time frame)

Step 3: Create a Notebooks instance

The screenshot shows the Google Cloud Platform Vertex AI Notebooks interface. On the left, there's a sidebar with options like Dashboard, Datasets, Features, Labeling tasks, Workbench (selected), Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main area has tabs for Notebooks (selected), NEW NOTEBOOK, REFRESH, START, STOP, RESET, and DELETE. Below these tabs, there are buttons for PREVIEW, USER-MANAGED NOTEBOOKS, EXECUTIONS, and SCHEDULES. A message at the top states: "As of the M80 DLM release, all environments will include JupyterLab 3.x by default. To continue using an existing environment's JupyterLab 1.x version, disable auto-upgrade (if enabled) and do not manually upgrade the environment to a new environment version. To create new Notebooks with JupyterLab 1.x installed, see creating specific versions of Notebooks." Another message below says: "Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. Learn more". A table lists the two notebooks:

Notebook name	Zone	Auto-upgrade	Environment	Machine type	GPUs	Permission	Last modified
tensorflow-2-3-20211209-130458	OPEN JUPYTERLAB	us-central1-a	-	TensorFlow 2.3	4 vCPUs, 15 GB RAM	None	Service account Dec 9, 2021, 1:08:12 PM
tensorflow-2-7-20211208-091022	OPEN JUPYTERLAB	us-central1-a	-	TensorFlow 2.7	4 vCPUs, 15 GB RAM	None	Service account Dec 9, 2021, 11:57:12 AM

Step 4: Install XGBoost

The screenshot shows a Jupyter Notebook interface. On the left, there's a file browser with files like Untitled.ipynb, Terminal 1, and XGboost.ipynb. The XGboost.ipynb notebook is selected. In the terminal tab, the command `pip3 install xgboost==1.2` is run, and the output shows the package being downloaded and successfully installed.

```
(base) jupyter@tensorflow-2-3-20211209-130458:~$ pip3 install xgboost==1.2
Collecting xgboost==1.2
  Downloading xgboost-1.2.0-py3-none-any.whl (148.9 kB)
    148.9 kB / 24 kB/s
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from xgboost==1.2) (1.19.5)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from xgboost==1.2) (1.7.3)
Installing collected packages: xgboost
  Successfully installed xgboost-1.2.0
(base) jupyter@tensorflow-2-3-20211209-130458:~$
```

The XGboost.ipynb notebook contains the following code:

```
[*]: import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import witwidget

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder
```

Step 5: Download and process data

Download the pre-processed dataset

We'll use a [mortgage dataset from ffiec.gov](#) to train an XGBoost model. We've done some preprocessing on the original dataset and created a smaller version for you to use to train the model. The model will predict *whether or not a particular mortgage application will get approved*.

The screenshot shows a Jupyter Notebook interface with two tabs: 'Untitled.ipynb' and 'XGboost.ipynb'. The 'XGboost.ipynb' tab is active, displaying Python code. The code imports pandas, xgboost, numpy, collections, and witwidget. It then uses gsutil to copy a CSV file from a Google Cloud bucket. Finally, it reads the data into a Pandas DataFrame.

```
[1]: import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import witwidget

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder

[4]: gsutil cp 'gs://mortgage_dataset_files/mortgage-small.csv' .
Copying gs://mortgage_dataset_files/mortgage-small.csv...
[1 files][330.8 MiB/330.8 MiB]
Operation completed over 1 objects/330.8 MiB.

Read data from Panda datafram

[5]: COLUMN_NAMES = collections.OrderedDict({
    'as_of_year': np.int16,
    'agency_code': 'category',
    'loan_type': 'category',
    'property_type': 'category',
    'occupancy': 'category',
    'loan_amt_thousands': np.float64,
    'preapproval': 'category',
    'county_code': np.float64,
    'applicant_income_thousands': np.float64,
    'purchaser_type': 'category',
    'num_1_to_4_family_units': 'category',
    'lien_status': 'category',
    'population': np.float64,
    'ffiec_median_fam_income': np.float64,
    'tract_to_msa_income_pct': np.float64,
    'num_owner_occupied_units': np.float64,
    'num_1_to_4_family_units': np.float64,
    'approved': np.int8
})
```

The screenshot shows the same Jupyter Notebook interface. The 'XGboost.ipynb' tab is active, displaying Python code to read the CSV file, drop columns, shuffle the data, and print the head of the DataFrame. Below the code, the first five rows of the DataFrame are displayed. The final cell shows the count of approved loans.

```
[6]: data = pd.read_csv(
    'mortgage-small.csv',
    index_col=False,
    dtype=COLUMN_NAMES
)
data = data.dropna()
data = shuffle(data, random_state=2)
data.head()

[6]:
```

	as_of_year	agency_code	loan_type	property_type	loan_purpose	occupancy	loan_amt_thousands	preapproval	county_code	applicant_income_thousands	purchaser_type	hoe
310650	2016	Consumer Financial Protection Bureau (CFPB)	Conventional	Any loan other than FHA, VA, FSA...	One to four-family (other than manufactured home)	Refinancing	1	110.0	Not applicable	110.0	55.0	Freddie Mac (FHLMC) Nc
630129	2016	Department of Housing and Urban Development (HUD)	Conventional	Any loan other than FHA, VA, FSA...	One to four-family (other than manufactured home)	Home purchase	1	480.0	Not applicable	33.0	270.0	Loan was not originated or was not sold in calendar year Nc
715484	2016	Federal Deposit Insurance Corporation (FDIC)	Conventional	Any loan other than FHA, VA, FSA...	One to four-family (other than manufactured home)	Refinancing	2	240.0	Not applicable	59.0	96.0	Commercial bank, savings bank or savings associate Nc
887708	2016	Office of the Comptroller of the Currency (OCC)	Conventional	Any loan other than FHA, VA, FSA...	One to four-family (other than manufactured home)	Refinancing	1	76.0	Not applicable	65.0	85.0	Loan was not originated or was not sold in calendar year Nc
7199508	2016	National Credit Union Administration (NCUA)	Conventional	Any loan other than FHA, VA, FSA...	One to four-family (other than manufactured home)	Refinancing	1	100.0	Not applicable	127.0	70.0	Loan was not originated or was not sold in calendar year Nc

```
[7]: # Class labels - 0: denied, 1: approved
print(data['approved'].value_counts())

labels = data['approved'].values
data = data.drop(columns=['approved'])

1 665389
0 334610
Name: approved, dtype: int64
```

Creating dummy column for categorical values

The screenshot shows a Jupyter Notebook interface with several files listed on the left: beans_deploy_component.yaml, beans_model_component.yaml, create_dataset.yaml, mlmd_pipeline.json, mortgage-small.csv, Untitled.ipynb, and XGboost.ipynb. The XGboost.ipynb file is open in the main area. In cell [8], the following Python code is run:

```
[8]: dummy_columns = list(data.dtypes[data.dtypes == 'category'].index)
data = pd.get_dummies(data, columns=dummy_columns)

data.head()
```

The output shows the first few rows of a DataFrame with 10 columns, including 'as_of_year', 'occupancy', 'loan_amt_thousands', 'county_code', 'applicant_income_thousands', 'population', 'ffiec_median_fam_income', 'tract_to_msa_income_pct', 'num_owner_occupied_unit', and 'id'. The data consists of approximately 5 rows by 44 columns.

Splitting data into train and test sets

```
[9]: x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)
```

Define and train the XGBoost model

Creating a model in XGBoost is simple. We'll use the XGBClassifier class to create the model, and just need to pass the right objective parameter for our specific classification task. In this case we use reg:logistic since we've got a binary classification problem and we want the model to output a single value in the range of (0,1): 0 for not approved and 1 for approved

The screenshot shows the same Jupyter Notebook environment. The XGboost.ipynb file is open. In cell [6], the following code is run:

```
[6]: x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)
```

In cell [7], the model is defined and trained:

```
[7]: model = xgb.XGBClassifier(
    objective='reg:logistic'
)
```

```
[8]: model.fit(x_train, y_train)
```

In cell [9], the model's prediction is made on the test set and accuracy is calculated:

```
[9]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
```

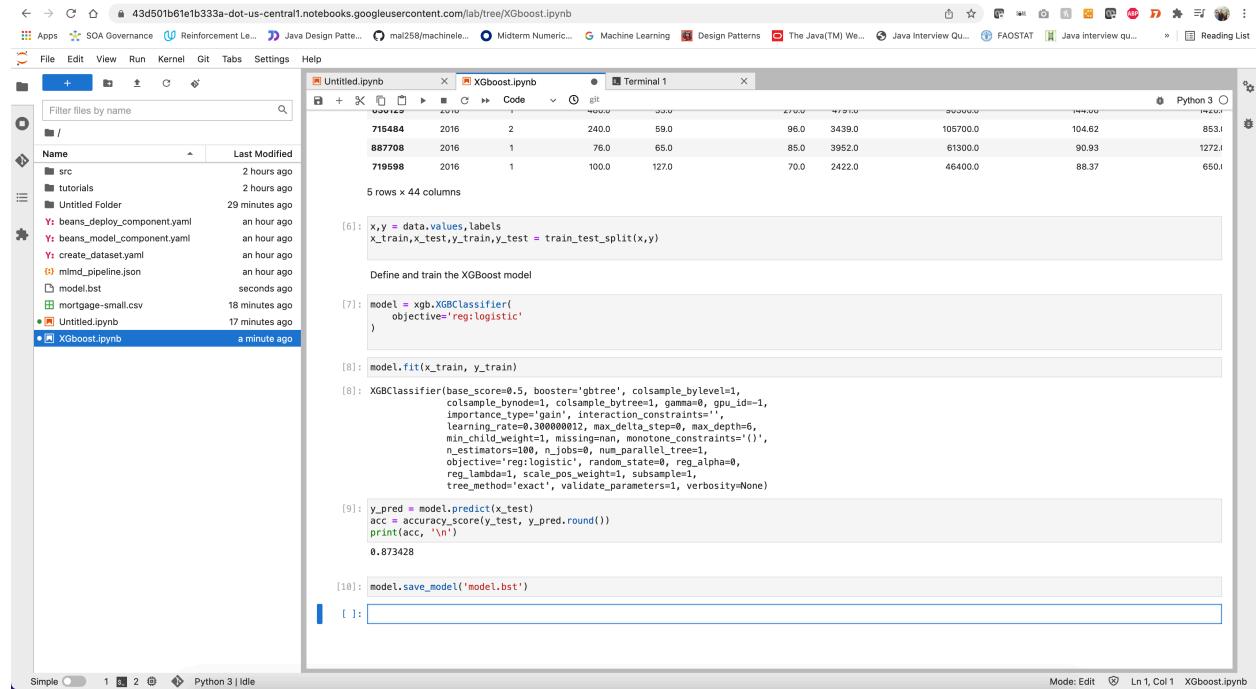
The output shows an accuracy of 0.873428.

In cell [10], the model is saved:

```
[10]: model.save_model('model.bst')
```

Evaluate the accuracy of your model and save the model

We can now use our trained model to generate predictions on our test data with the predict() function. Then we'll use Scikit-learn's accuracy_score() function to calculate the accuracy of our model based on how it performs on our test data. We'll pass it the ground truth values along with the model's predicted values for each example in our test set:



The screenshot shows a Jupyter Notebook interface with a sidebar containing files like beans_deploy_component.yaml, beans_model_component.yaml, and a local file Untitled.ipynb. The main area displays Python code for training an XGBoost classifier and saving the model. The code includes imports, loading data, defining the classifier, fitting it, making predictions, calculating accuracy, and saving the model.

```
[6]: x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)

Define and train the XGBoost model

[7]: model = xgb.XGBClassifier(
    objective='reg:logistic'
)

[8]: model.fit(x_train, y_train)

[8]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bytree=1,
       colsample_bynode=1, colsample_bylevel=1, gamma=0, gpu_id=-1,
       importance_type='gain', interaction_constraints='',
       learning_rate=0.300000012, max_delta_step=0, max_depth=6,
       min_child_weight=1, missing=None, n_estimators=100, n_jobs=-1,
       nthread=None, objective='reg:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, subsample=1,
       tree_method='exact', validate_parameters=1, verbosity=None)

[9]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
0.873428

[10]: model.save_model('model.bst')

[ ]:
```

Step 6: Use the What-if Tool to interpret your model

Create the What-if Tool visualization

To connect the [What-if Tool](#) to your local model, you need to pass it a subset of your test examples along with the ground truth values for those examples. Let's create a Numpy array of 500 of our test examples along with their ground truth labels:

File Edit View Run Kernel Git Tabs Settings Help

Untitled.ipynb XGboost.ipynb Terminal 1 Python 3

```

[9]: reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact', validate_parameters=1, verbosity=None)

[9]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
0.873428

[10]: model.save_model('model.bst')

[14]: num_wit_examples = 500
test_examples = np.hstack((x_test[:num_wit_examples],y_test[:num_wit_examples].reshape(-1,1)))

[22]: test_examples

[22]: array([[2.015e+03, 1.000e+00, 5.050e+02, ..., 1.000e+00, 0.000e+00,
   1.000e+00],
          [2.015e+03, 1.000e+00, 6.040e+02, ..., 1.000e+00, 0.000e+00,
   1.000e+00],
          [2.015e+03, 1.000e+00, 1.760e+02, ..., 1.000e+00, 0.000e+00,
   1.000e+00],
          ...,
          [2.015e+03, 1.000e+00, 1.500e+02, ..., 1.000e+00, 0.000e+00,
   1.000e+00],
          [2.015e+03, 1.000e+00, 6.600e+01, ..., 1.000e+00, 0.000e+00,
   1.000e+00],
          [2.015e+03, 1.000e+00, 2.720e+02, ..., 1.000e+00, 0.000e+00,
   1.000e+00]])

[15]: config_builder = (WitConfigBuilder(test_examples.tolist(), data.columns.tolist() + ['mortgage_status'])
.set_custom_predict_fn(model.predict_proba)
.set_target_feature('mortgage_status')
.set_label_vocab(['denied', 'approved']))
WitWidget(config_builder, height=600)
```

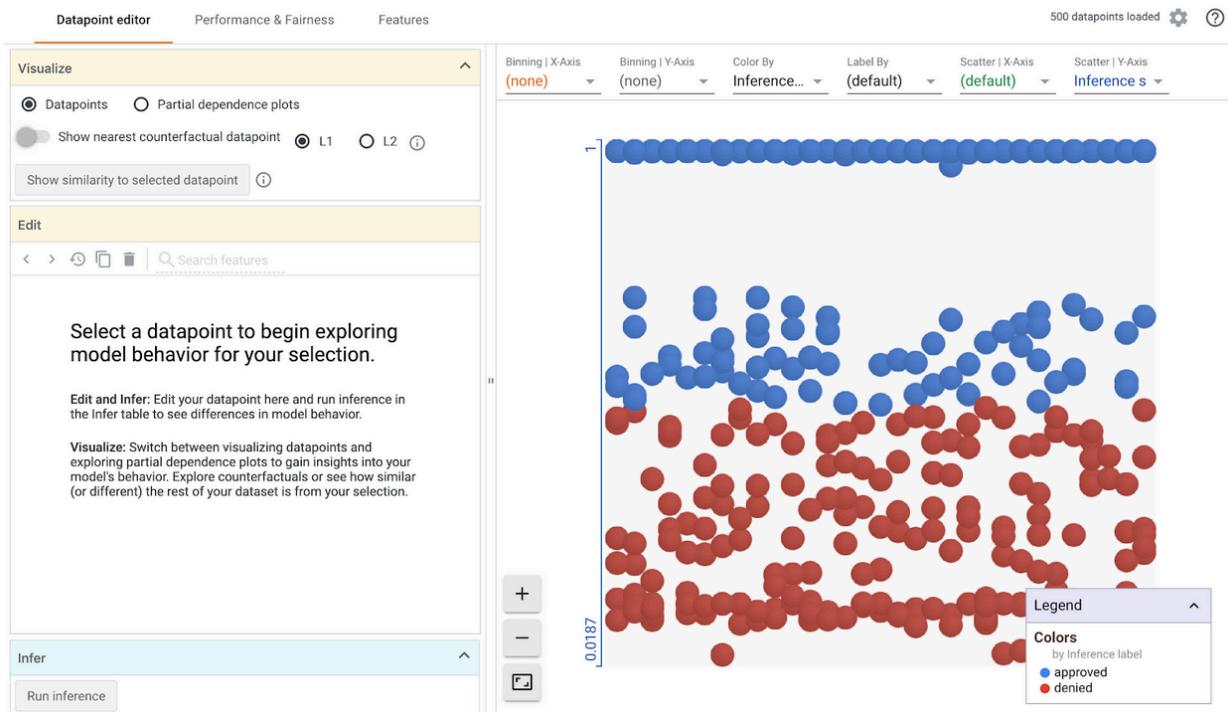
Datapoint editor Performance & Fairness Features No datapoints loaded yet

Visualize Datapoints Partial dependence plots Nearest counterfactual L1 L2

Binning | X-Axis (none) Binning | Y-Axis (none) Color By Inference... Label By (default) Scatter | X-Axis (default) Scatter | Y-Axis Inference s

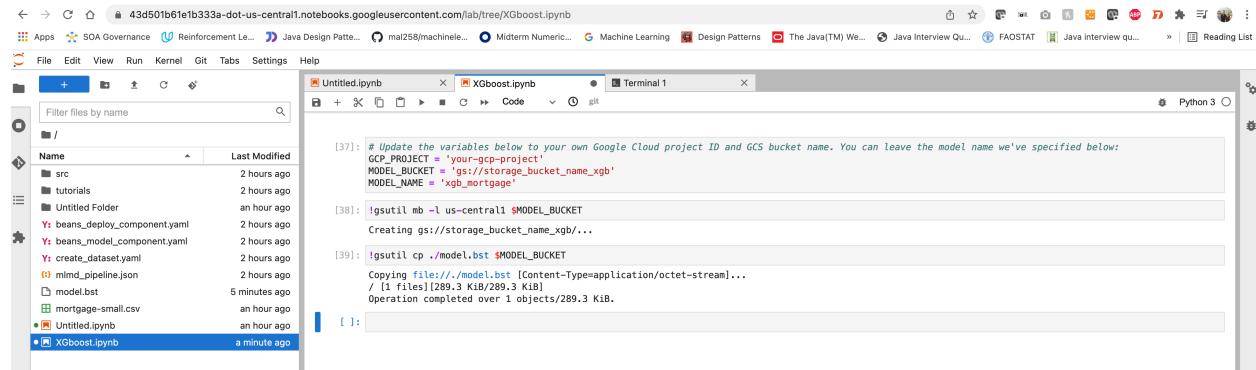
Datapoints and their inference results will be displayed here.

Mode: Edit Ln 4, Col 44 XGboost.ipynb



Step 7: Deploy model to Vertex AI

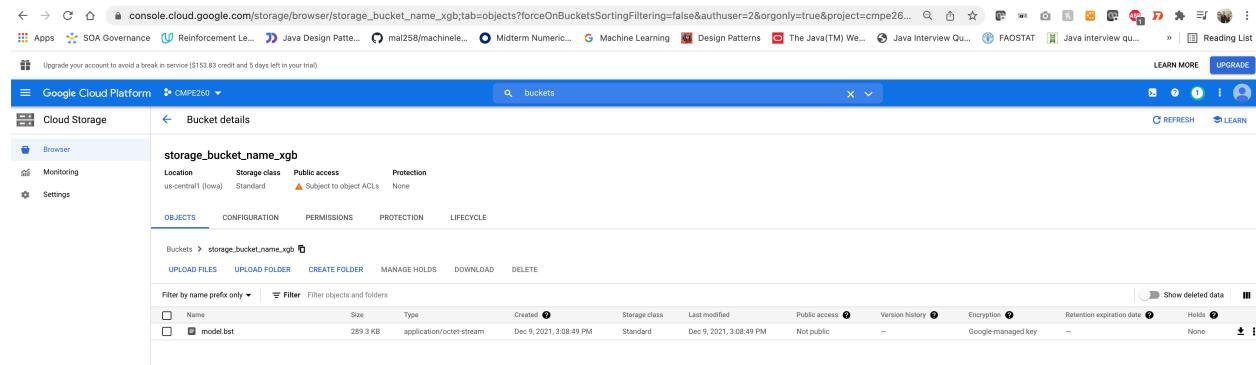
Create a Cloud Storage bucket for our model



A screenshot of a Jupyter Notebook interface. On the left, there's a file browser showing various files like 'beans_deploy_component.yaml', 'model.bst', and 'XGboost.ipynb'. The main area has a terminal window and a code editor. The code in the editor is:

```
[37]: # Update the variables below to your own Google Cloud project ID and GCS bucket name. You can leave the model name we've specified below:  
GCP_PROJECT = 'your-gcp-project'  
MODEL_BUCKET = 'gs://storage_bucket_name_xgb'  
MODEL_NAME = 'xgb_mortgage'  
  
[38]: !gsutil mb -l us-central1 $MODEL_BUCKET  
Creating gs://storage_bucket_name_xgb...  
  
[39]: !gsutil cp ./model.bst $MODEL_BUCKET  
Copying file://./model.bst [Content-Type=application/octet-stream]...  
/ (1 files)[289.3 KiB/289.3 KiB]  
Operation completed over 1 objects/289.3 KiB.
```

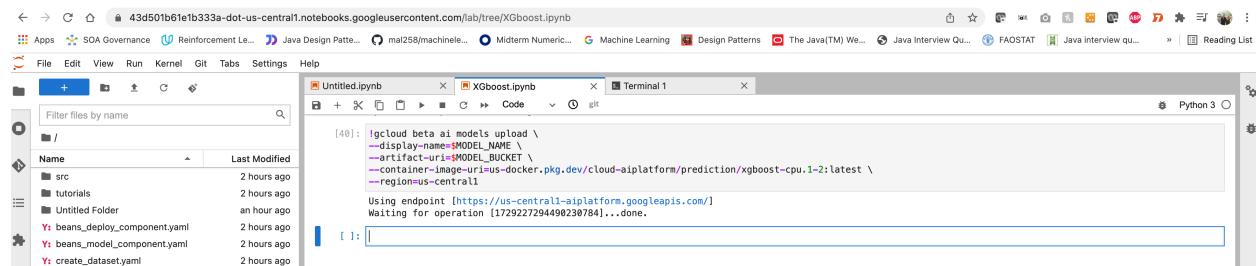
Copy the model file to Cloud Storage



A screenshot of the Google Cloud Platform Storage browser. It shows a bucket named 'storage_bucket_name_xgb' located in 'us-central1 (Iowa)'. The 'OBJECTS' tab is selected, showing a single file 'model.bst' which was uploaded from the Jupyter notebook. The file details are:

Name	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date	Holds
model.bst	application/octet-stream	Dec 9, 2021, 3:08:49 PM	Standard	Dec 9, 2021, 3:08:49 PM	Not public	—	Google-managed key	—	None

Create the model and deploy to an endpoint



A screenshot of a Jupyter Notebook interface. The file browser on the left shows 'XGboost.ipynb'. The code in the editor is:

```
[40]: !gcloud beta ai models upload \  
--display-name=$MODEL_NAME \  
--artifact-uri=$MODEL_BUCKET \  
--container-image-uri=us-docker.pkg.dev/cloud-aiplatform/prediction/xgboost-cpu.1-2:latest \  
--region=us-central1  
Using endpoint [https://us-central1-aiplatform.googleapis.com]  
Waiting for operation [17922794496230784]...done.
```

The screenshot shows the Google Cloud Platform Vertex AI Models page. The left sidebar includes options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models (selected), Endpoints, Marketplace, and a search bar for 'vertex'. The main area displays a table of models with columns: Name, ID, Status, Data, Endpoints, Region, Type, Created, Notifications, and Labels. One row for 'xgb_mortgage' is selected, showing its ID as 3457291168239321088.

To deploy your endpoint, run the gcloud command below:

```
[40]: !gcloud beta ai models upload \
--display-name=MODEL_NAME \
--artifact-uri=MODEL_BUCKET \
--container-image-uri=us-docker.pkg.dev/cloud-aiplatform/prediction/xgboost-cpu.1-2:latest \
--region=us-central1

Using endpoint [https://us-central1-aiplatform.googleapis.com]
Waiting for operation [179227294490230784]...done.

[41]: MODEL_ID = "3457291168239321088"

[42]: !gcloud beta ai endpoints create \
--display-name=xgb_mortgage_v1 \
--region=us-central1

Using endpoint [https://us-central1-aiplatform.googleapis.com]
Waiting for operation [179227294490230784]...done.
Created Vertex AI endpoint: projects/196373151126/locations/us-central1/endpoints/9065825214734008320.

[43]: ENDPOINT_ID = "9065825214734008320"

[44]: !gcloud beta ai endpoints deploy-model $ENDPOINT_ID \
--region=us-central1 \
--model=$MODEL_ID \
--display-name=xgb_mortgage_v1 \
--machine-type=n1-standard-2 \
--traffic-split=0:100

Using endpoint [https://us-central1-aiplatform.googleapis.com]
Waiting for operation [15026284434198499840]...done.
```

The screenshot shows the Google Cloud Platform Vertex AI Models page. The left sidebar includes options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models (selected), Endpoints, Batch predictions, and Metadata. The main area shows a table for the 'xgb_mortgage' endpoint with columns: Name, ID, Status, Models, Region, Monitoring, Most recent monitoring job, Most recent alerts, Last updated, API, Notification, Labels, Encryption, and a more button. The status for the endpoint is currently 'Deploying model'.

Test the deployed model

The screenshot shows the Google Cloud Platform Vertex AI interface. On the left, there's a sidebar with options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main area is titled "Deploy & Test" for a model named "xgb_mortgage". It shows a table with one row for "xgb_mortgage_v1". The "Status" column shows "Active", "Models" shows "0", and "Region" shows "us-central1". A "DEPLOY TO ENDPOINT" button is visible. Below this, there's a "Test your model" section with a "PREVIEW" tab selected. It contains a JSON request template and a "PREDICT" button. To the right, there's a "Responses" section showing a JSON response with fields like "predictions", "deployedModelId", "model", and "modelDisplayName".

Clean Up remove endpoint , delete Model and delete bucket.

The screenshot shows the Google Cloud Platform Vertex AI interface, specifically the "Endpoints" section. The sidebar is identical to the previous screenshot. The main area shows a list of endpoints. One endpoint, "xgb_mortgage_v1", is selected. A modal dialog box is open over the list, titled "Remove endpoint". It contains the message "Your endpoint will no longer be available for online prediction requests." and a question "Remove endpoint 'xgb_mortgage_v1'?" with "CANCEL" and "CONFIRM" buttons.

console.cloud.google.com/vertex-ai/models?authuser=2&orgonly=true&project=cmpe260-334300&supportedpurview=organizationId

LEARN MORE UPGRADE

Google Cloud Platform CMPE260 ▾

vertex AI Models + CREATE IMPORT

Dashboard datasets Features Labeling tasks Workbench Pipelines Training Experiments Models Endpoints Batch predictions Metadata

Region: us-central1 (Iowa)

Filter: Enter a property name

Name	ID	Status	Data	Endpoints	Region	Type	Created	Notifications	Labels
xgb_mortgage	3457291168239321088	Ready	—	0	us-central1	Imported Custom training	Dec 9, 2021, 3:10:54 PM		
beans-model-pipeline	344177925819465728	Ready	—	1 SHOW ENDPOINTS	us-central1	Imported Custom training	Dec 9, 2021, 1:42:22 PM		
beans-model-pipeline	6992516875725160448	Ready	—	1 SHOW ENDPOINTS	us-central1	Imported Custom training	Dec 9, 2021, 1:40:14 PM		

Delete model

CANCEL DELETE

console.cloud.google.com/storage/browser?authuser=2&orgonly=true&project=cmpe260-334300&supportedpurview=organizationId&prefix=

LEARN MORE UPGRADE

Google Cloud Platform CMPE260 ▾

Cloud Storage Browser + CREATE BUCKET ⌂ DELETE ⌂ REFRESH SHOW INFO PANEL

Monitoring Settings

Browser Filter buckets

Name	Created	Location type	Location	Default storage class	Last modified	Public access	Access control	Protection	Lifecycle rules	Labels	Requester Pays
cloud-ai-platform-65473780-0280-4fd...	Dec 6, 2021, 4:17:06 PM	Region	us-central1 (Io...	Regional	Dec 6, 2021, 4:17:06 PM	Subject to object ACLs	Fine-grained	None	None	OFF	
cmpe260-334300-bucket	Dec 9, 2021, 12:43:00 PM	Region	us-central1 (Io...	Standard	Dec 9, 2021, 12:43:00 PM	Subject to object ACLs	Fine-grained	None	None	OFF	
storage_bucket_name_xgb	Dec 9, 2021, 3:08:40 PM	Region	us-central1 (Io...	Standard	Dec 9, 2021, 3:08:40 PM	Subject to object ACLs	Fine-grained	None	None	OFF	

Delete bucket 'storage_bucket_name_xgb'?

This action will permanently delete the bucket and the objects it contains (including versions) from Google Cloud. Data will not be recoverable.

You are deleting the bucket storage_bucket_name_xgb.

Deletion is performed in the background and may take some time depending on the size of the objects being deleted.

Confirm deletion by typing 'DELETE' below:

DELETE

CANCEL DELETE

