

a) Custom training job and prediction using managed datasets

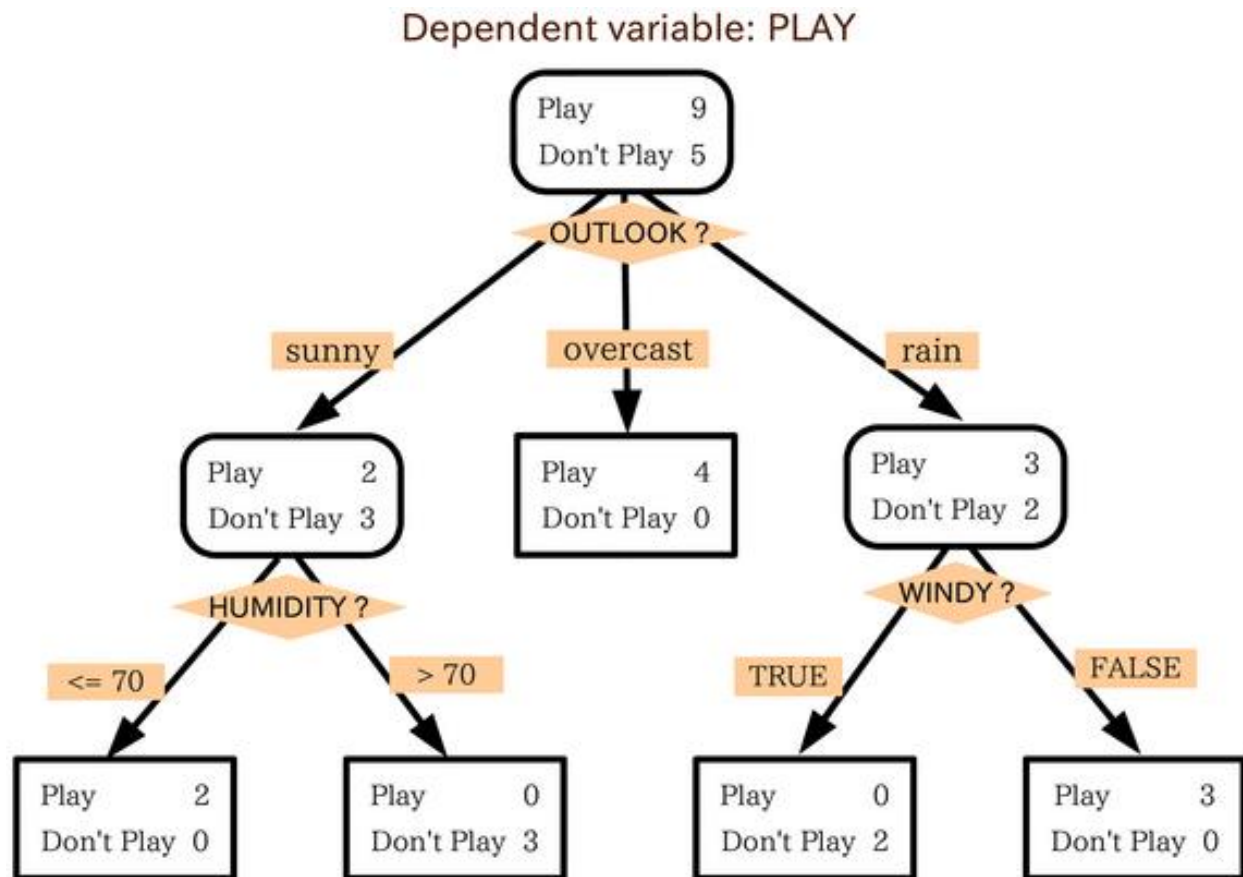
Reference: <https://codelabs.developers.google.com/vertex-xgb-wit#0>

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 console for the project 'cmpe260'. The 'Compute Engine' section is active, displaying the 'VM instances' page. The left sidebar shows various navigation options like 'Virtual machines', 'Storage', 'Instance groups', and 'VM Manager'. The main content area shows a table of VM instances. The first instance is 'tensorflow-2-9-20211209-130458' in the 'us-central1-a' zone, with an internal IP of '10.128.0.4' and an external IP of '35.223.119.126'. The second instance is 'tensorflow-2-7-20211208-091822' in the 'us-central1-a' zone, with an internal IP of '10.128.0.3' and no external IP. Below the table, there are several 'Related actions' such as '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 console for the project 'cmpe260', specifically the 'Vertex AI' section. The 'Dashboard' is active, displaying a 'Get started with Vertex AI' section with a tutorial link. Below this, there are three main sections: 'Prepare your training data', 'Recent models', and 'Recent notebook instances'. The 'Recent models' section shows two models: 'beans-model pipeline' and 'beans-model pipeline', both created 25 minutes ago. The 'Recent notebook instances' section shows two instances: 'tensorflow-2-9-20211209-130458' and 'tensorflow-2-7-20211208-091822'. The dashboard also includes a 'Recent endpoints' section with a table showing 'ONLINE TRAFFIC', 'REQUESTS', and 'ERROR RATE'.

## Step 3: Create a Notebooks instance

Google Cloud Platform Vertex AI

Notebooks

MANAGED NOTEBOOKS PREVIEW USER-MANAGED NOTEBOOKS EXECUTIONS PREVIEW SCHEDULES PREVIEW

As of the MBO DVM 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.

Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. [Learn more](#)

Filter: Enter property name or value

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

https://43d501b61e1b333a-dot-us-central1.notebooks.googleusercontent.com/authorer=2&username=Musketeers\_SJSU

## Step 4: Install XGBoost

43d501b61e1b333a-dot-us-central1.notebooks.googleusercontent.com/lab

File Edit View Run Kernel Git Tabs Settings Help

Filter files by name

Name Last Modified

src an hour ago

tutorials an hour ago

Untitled Folder a minute ago

Y1 beans\_deploy\_component.yaml an hour ago

Y1 beans\_model\_component.yaml an hour ago

Y1 create\_dataset.yaml an hour ago

(1) mlmd\_pipeline.json 43 minutes ago

Untitled.ipynb 38 minutes ago

XGboost.ipynb seconds ago

```
(base) jupyter@tensorflow-2-3-20211209-130458:~$ pip3 install xgboost==1.2
Collecting xgboost==1.2
  Downloading xgboost-1.2.0-py3-none-manylinux2010_x86_64.whl (148.9 MB)
    148.9 MB 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:~$
```

43d501b61e1b333a-dot-us-central1.notebooks.googleusercontent.com/lab/tree/XGboost.ipynb

File Edit View Run Kernel Git Tabs Settings Help

Filter files by name

Name Last Modified

src an hour ago

tutorials an hour ago

Untitled Folder 2 minutes ago

Y1 beans\_deploy\_component.yaml an hour ago

Y1 beans\_model\_component.yaml an hour ago

Y1 create\_dataset.yaml an hour ago

(1) mlmd\_pipeline.json 44 minutes ago

Untitled.ipynb 38 minutes ago

XGboost.ipynb a minute ago

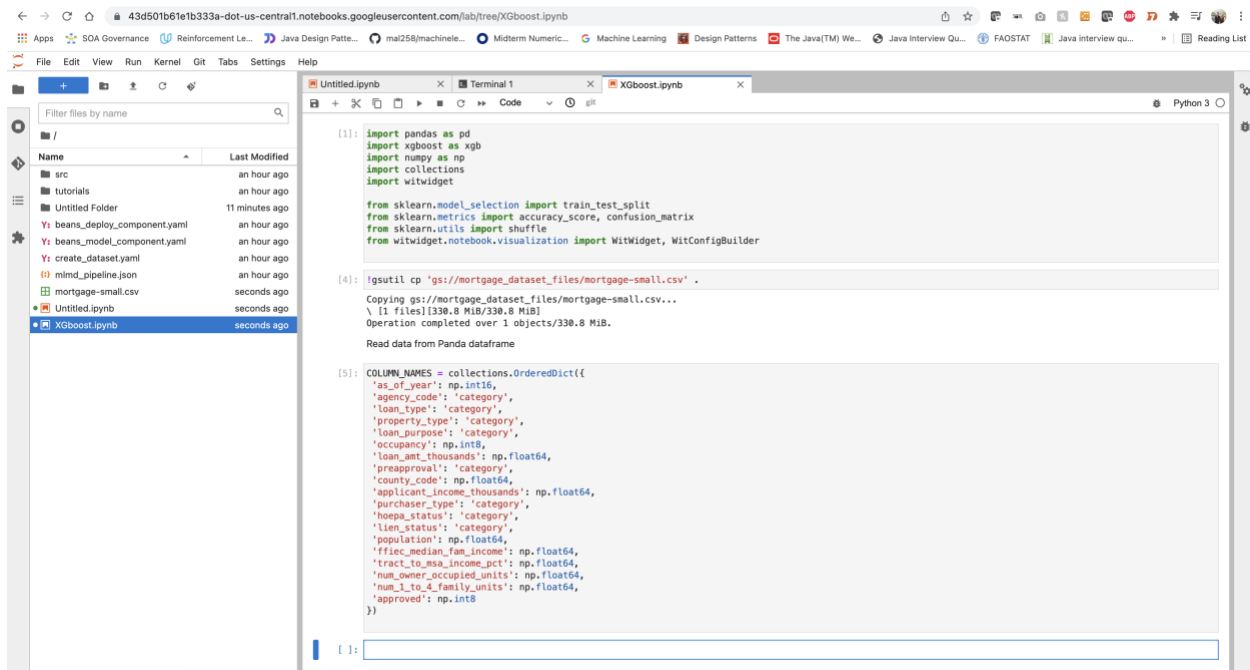
```
[1]: import pandas as pd
import xgboost as xgb
import numpy as np
import collections
import witwidget

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
```

## 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*.



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

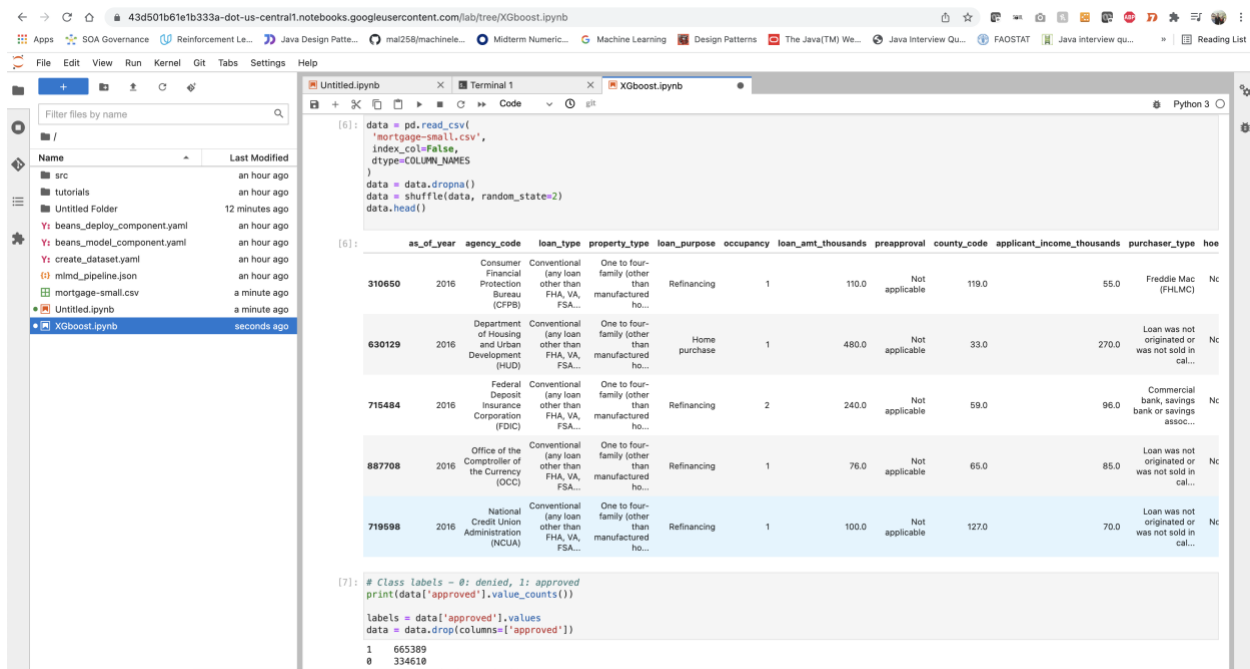
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder

[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 dataframe

[5]: COLUMN_NAMES = collections.OrderedDict({
    'as_of_year': np.int16,
    'agency_code': 'category',
    'loan_type': 'category',
    'property_type': 'category',
    'loan_purpose': 'category',
    'occupancy': np.int8,
    'loan_amt_thousands': np.float64,
    'preapproval': 'category',
    'county_code': np.float64,
    'applicant_income_thousands': np.float64,
    'purchaser_type': 'category',
    'hoepa_status': '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
})
```



```
[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	approved
310650	2016	Consumer Financial Protection Bureau (CFPB)	Conventional (any loan other than FHA, VA, FSA...)	One to four-family (other than manufactured ho...	Refinancing	1	110.0	Not applicable	119.0	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 ho...	Home purchase	1	480.0	Not applicable	33.0	270.0	Loan was not originated or was not sold in cal...	Nc
715484	2016	Federal Deposit Insurance Corporation (FDIC)	Conventional (any loan other than FHA, VA, FSA...)	One to four-family (other than manufactured ho...	Refinancing	2	240.0	Not applicable	59.0	96.0	Commercial bank, savings bank or savings assoc...	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 ho...	Refinancing	1	76.0	Not applicable	65.0	85.0	Loan was not originated or was not sold in cal...	Nc
719598	2016	National Credit Union Administration (NCUA)	Conventional (any loan other than FHA, VA, FSA...)	One to four-family (other than manufactured ho...	Refinancing	1	100.0	Not applicable	127.0	70.0	Loan was not originated or was not sold in cal...	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

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

data.head()
```

```
[9]:
```

	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
310660	2016	1	110.0	119.0	55.0	5930.0	64100.0	98.81	1305.1
630129	2016	1	480.0	33.0	270.0	4791.0	90300.0	144.06	1420.1
715484	2016	2	240.0	59.0	96.0	3439.0	105700.0	104.62	853.1
887708	2016	1	76.0	65.0	85.0	3952.0	61300.0	90.93	1272.1
719598	2016	1	100.0	127.0	70.0	2422.0	46400.0	88.37	650.1

5 rows x 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

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

```
[7]: Define and train the XGBoost model

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

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

[8]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bynode=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, missingnan, monotone_constraints=''),
n_estimators=100, n_jobs=0, num_parallel_tree=1,
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)
```

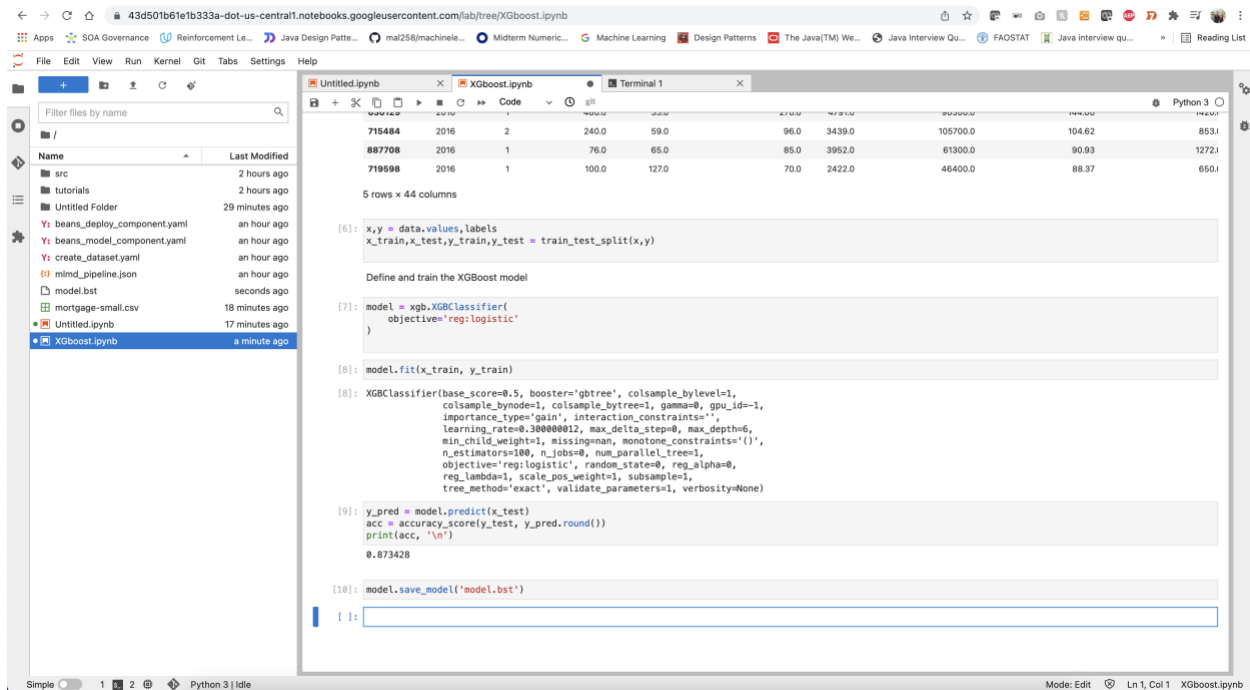
```
[9]: y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')

0.873428
```

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

## 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 Google Colab notebook with the following content:

```
[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_bylevel=1,
     colsample_bynode=1, colsample_bynode=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=nan, monotone_constraints='()',
     n_estimators=100, n_jobs=0, num_parallel_tree=1,
     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:

The screenshot shows a Google Colab notebook titled 'XGboost.ipynb'. The left sidebar displays a file explorer with various files and folders. The main area contains a Jupyter notebook interface with a code editor and a terminal. The code in the notebook includes:

```

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')

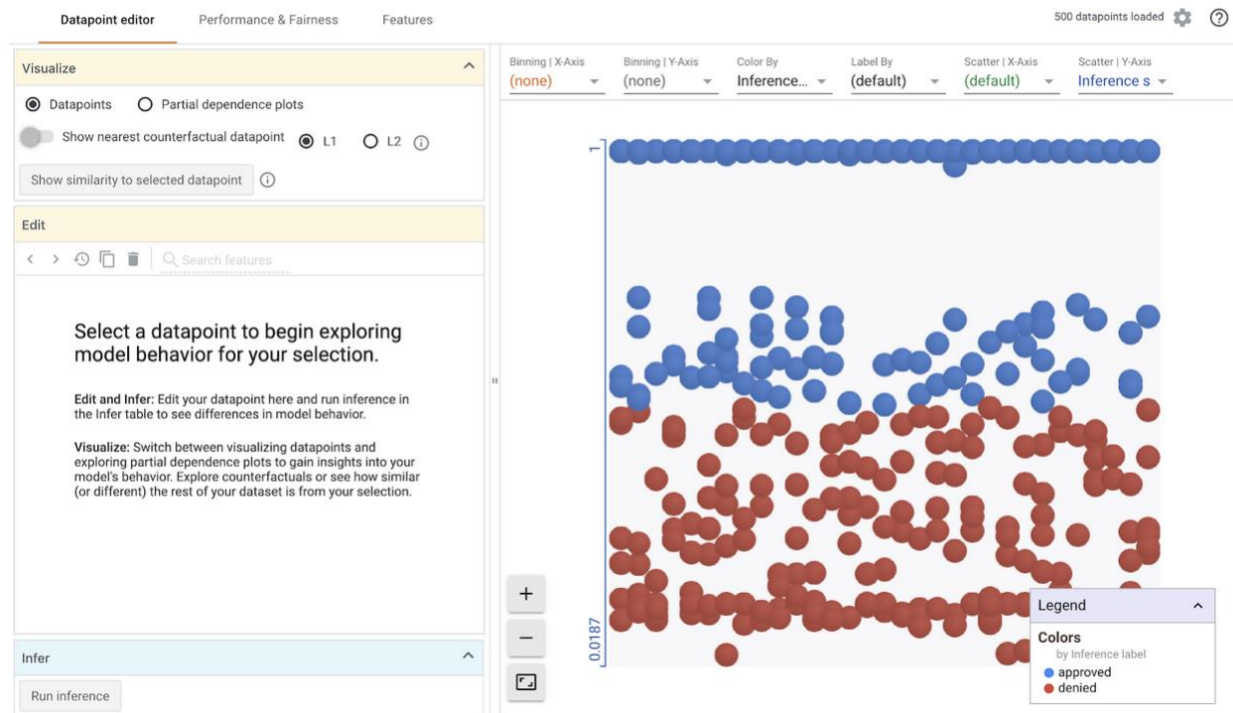
[14]: num_wit_examples = 500
test_examples = np.hstack((x_test[:num_wit_examples], y_test[:num_wit_examples].reshape(-1,1)))

[22]: test_examples
array([[2.016e+03, 1.000e+00, 5.850e+02, ..., 1.000e+00, 0.000e+00,
1.000e+00],
[2.016e+03, 1.000e+00, 6.840e+02, ..., 1.000e+00, 0.000e+00,
0.000e+00],
[2.016e+03, 1.000e+00, 1.760e+02, ..., 1.000e+00, 0.000e+00,
1.000e+00],
...,
[2.016e+03, 1.000e+00, 1.500e+02, ..., 1.000e+00, 0.000e+00,
1.000e+00],
[2.016e+03, 1.000e+00, 6.600e+01, ..., 1.000e+00, 0.000e+00,
1.000e+00],
[2.016e+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)

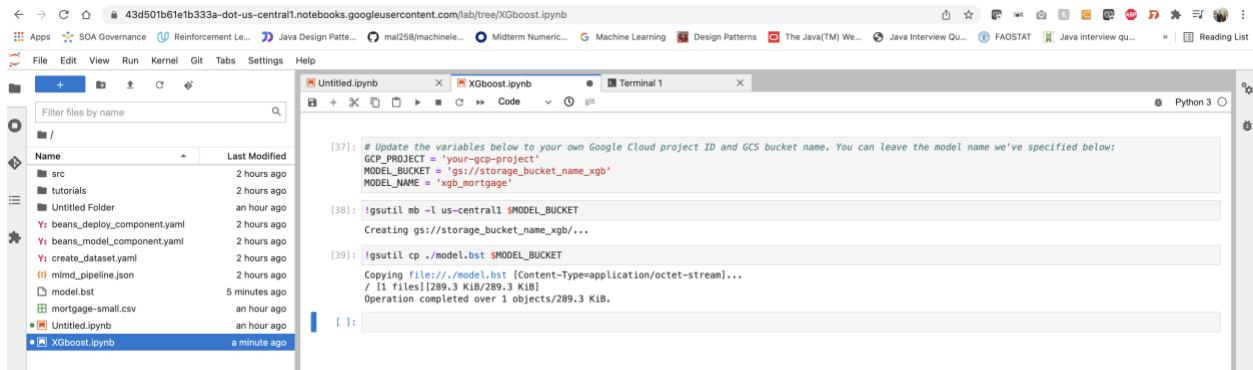
```

Below the code editor, there are tabs for 'Datapoint editor', 'Performance & Fairness', and 'Features'. The 'Datapoint editor' tab is active, showing a 'Visualize' section with options for 'Datapoints' and 'Partial dependence plots'. The 'Performance & Fairness' tab shows 'No datapoints loaded yet'.



## Step 7: Deploy model to Vertex AI

# Create a Cloud Storage bucket for our model



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `src`, `tutorials`, `beans_deploy_component.yaml`, `beans_model_component.yaml`, `create_dataset.yaml`, `miml_pipeline.json`, `model.bst`, `mortgage-small.csv`, `Untitled.ipynb`, and `XGboost.ipynb`. The code editor shows the following code:

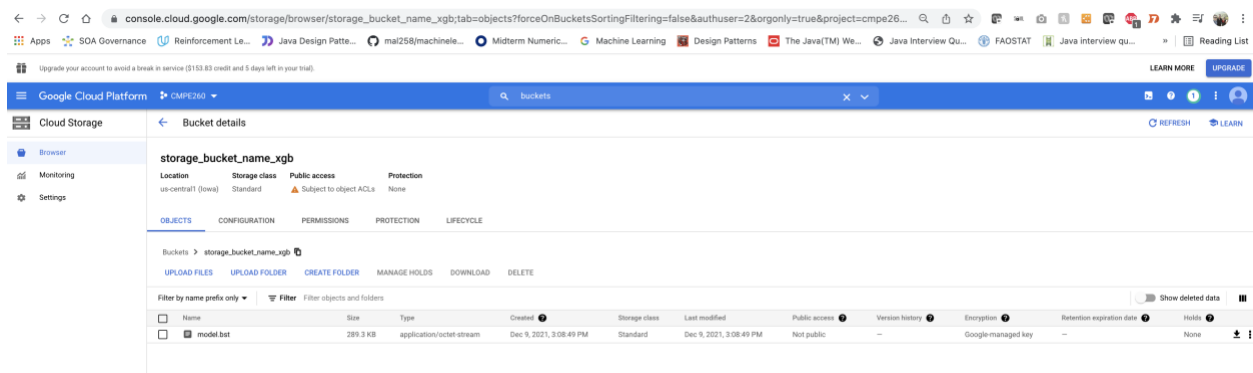
```
[37]: # Update the variables below to your own Google Cloud project ID and GCS bucket name. You can leave the model name we've specified below:
      GCP_PROJECT = 'your-gcp-project'
      MODEL_BUCKET = 'gs://storage_bucket_name_xgb'
      MODEL_NAME = 'xgb_mortgage'

[38]: !gsutil mb -l us-central1 $MODEL_BUCKET
      Creating gs://storage_bucket_name_xgb/...

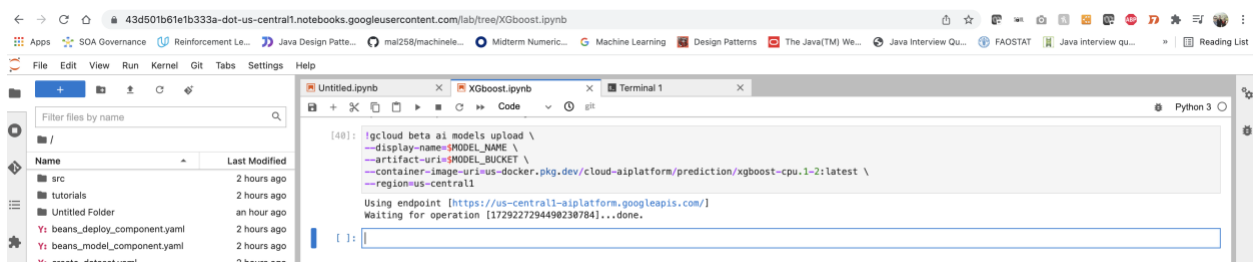
[39]: !gsutil cp ./model.bst $MODEL_BUCKET
      Copying file://./model.bst [Content-Type=application/octet-stream]...
      / [1 files] [289.3 KiB/289.3 KiB]
      Operation completed over 1 objects/289.3 KiB.

[ ]:
```

# Copy the model file to Cloud Storage



# Create the model and deploy to an endpoint



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `src`, `tutorials`, `beans_deploy_component.yaml`, `beans_model_component.yaml`, `create_dataset.yaml`, `miml_pipeline.json`, `model.bst`, `mortgage-small.csv`, `Untitled.ipynb`, and `XGboost.ipynb`. The code editor shows the following code:

```
[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 [1729227294490230784]...done.

[ ]:
```



console.cloud.google.com/vertex-ai/models?authuser=2&orgonly=true&project=cme260-334300&supportedpurview=organizationid

Google Cloud Platform CMPE260 vertex

Vertex AI Models CREATE IMPORT REFRESH

Models are built from your datasets or unmanaged data sources. There are many different types of machine learning models available on Vertex AI, depending on your use case and level of experience with machine learning. [Learn more](#)

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	34417925819465728	Ready	--	1	us-central1	Imported Custom training	Dec 9, 2021, 1:42:22 PM		
beans-model-pipeline	699261687525160448	Ready	--	1	us-central1	Imported Custom training	Dec 9, 2021, 1:40:14 PM		

To deploy your endpoint, run the gcloud command below:

43d501b61e1b333a-dot-us-central1.notebooks.googleusercontent.com/lab/tree/XGboost.ipynb

File Edit View Run Kernel GIT Tabs Settings Help

Filter files by name

Terminal 1

```
[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 [1729227294498230784]...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 [7309609445267341312]...done.
Created Vertex AI endpoint: projects/196373151126/locations/us-central1/endpoints/9065825214734008328.

[43]: ENDPOINT_ID = "9065825214734008328"

[4]: 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 [5826284434190499840]...#
```

console.cloud.google.com/vertex-ai/locations/us-central1/models/3457291168239321088/deploy?authuser=2&orgonly=true&project=cme260-334300&supportedpurview=organizationid

Google Cloud Platform CMPE260 vertex

Vertex AI xgb\_mortgage EXPORT

DEPLOY & TEST BATCH PREDICTIONS MODEL PROPERTIES

Deploy your model

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.

DEPLOY TO ENDPOINT

Name	ID	Status	Models	Region	Monitoring	Most recent monitoring job	Most recent alerts	Last updated	API	Notification	Labels	Encryption
xgb_mortgage_v1	9065825214734008328	Deploying model	0	us-central1	Disabled	--	--	Dec 9, 2021, 3:12:54 PM	<a href="#">Sample request</a>			Google-managed key

Test your model [GETVIEW](#)

Your model must be successfully deployed to an endpoint before you can test it.

Your JSON request must contain an `instances` field and an optional `parameters` field if you're using a custom container. No other fields can be present in the JSON request. [Learn how to format your JSON request](#)

JSON request

```
{
  "instances": [
    {
      "sample_key": "sample_value"
    }
  ]
}
```

PREDICT

Response

# Test the deployed model

The screenshot shows the Google Cloud Platform Vertex AI console. The left sidebar contains navigation options: Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models (selected), Endpoints, Batch predictions, and Metadata. The main content area is titled 'xgb\_mortgage' and has tabs for 'DEPLOY & TEST', 'BATCH PREDICTIONS', and 'MODEL PROPERTIES'. The 'DEPLOY & TEST' tab is active, showing a 'Deploy your model' section with a 'DEPLOY TO ENDPOINT' button. Below this is a table of endpoints:

Name	ID	Status	Models	Region	Monitoring	Most recent monitoring job	Most recent alerts	Last updated	API	Notification	Labels	Encryption
xgb_mortgage_v1	9065823214734008320	Active	0	us-central1	Disabled	--	--	Dec 9, 2021, 3:12:54 PM	<a href="#">Sample request</a>			Google managed key

Below the table is the 'Test your model' section, which includes a 'JSON request' input field and a 'Response' output field. The JSON request is a 2D array of floats, and the response is a JSON object containing 'predictions'.

```
{
  "predictions": [
    0.99995124191284
  ]
}
```

Clean Up remove endpoint , delete Model and delete bucket.

The screenshot shows the Google Cloud Platform Vertex AI console with the 'Endpoints' tab selected. A 'Remove endpoint' dialog box is open, asking for confirmation to remove the endpoint 'xgb\_mortgage\_v1'. The dialog box contains the text: 'Your endpoint will no longer be available for online prediction requests. Remove endpoint "xgb\_mortgage\_v1"?'. The 'CONFIRM' button is highlighted.

The background shows the 'Endpoints' table with the following data:

Name	ID	Status	Models	Region	Monitoring	Most recent alerts	Last updated	API	Notification	Labels
xgb_mortgage_v1	9065823214734008320	Ready	0	us-central1	Disabled		Dec 9, 2021, 3:23:02 PM	Sample request		
beans-model-pipeline_endpoint	1416461247645220864	Active					Dec 9, 2021, 1:50:03 PM	Sample request		
beans-model-pipeline_endpoint	328299218695758464	Active					Dec 9, 2021, 1:49:59 PM	Sample request		

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

Google Cloud Platform CMPE260

Vertex AI Models

Models are built from your datasets or unmanaged data sources. There are many different types of machine learning models available on Vertex AI, depending on your use case and level of experience with machine learning. [Learn more](#)

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	us-central1	Imported Custom training	Dec 9, 2021, 1:42:22 PM		
beans-model-pipeline	6992616875725160448	Ready	--	1	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=

Google Cloud Platform CMPE260

Cloud Storage Browser

Filter: 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-65473780-6280-46f...	Dec 9, 2021, 4:17:06 PM	Region	us-central1 (Iowa)	Regional	Dec 9, 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 (Iowa)	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 (Iowa)	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