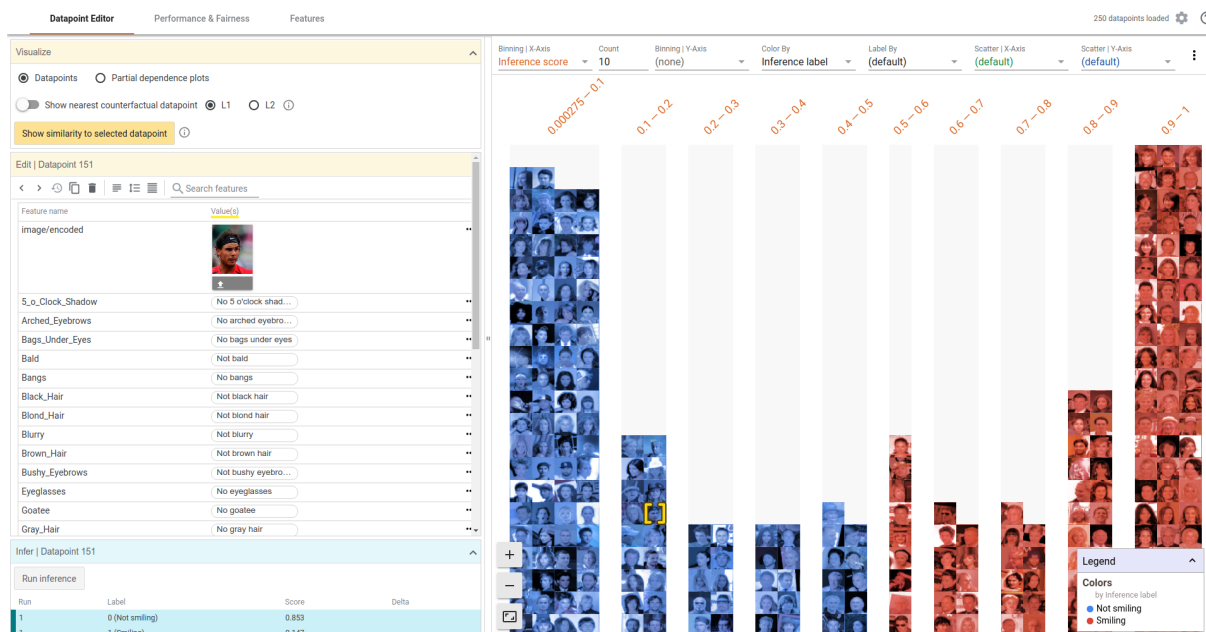


What-If Tool



The What-If Tool (WIT) provides an easy-to-use interface for expanding understanding of a black-box classification or regression ML model. With the plugin, you can perform inference on a large set of examples and immediately visualize the results in a variety of ways. Additionally, examples can be edited manually or programmatically and re-run through the model in order to see the results of the changes. It contains tooling for investigating model performance and fairness over subsets of a dataset.

The purpose of the tool is that give people a simple, intuitive, and powerful way to play with a trained ML model on a set of data through a visual interface with absolutely no code required.

The tool can be accessed through TensorBoard or as an extension in a Jupyter or Colab notebook.

I don't want to read this document. Can I just play with a demo?

Check out the large set of web and colab demos in the demo section of the What-If Tool website.

To build the web demos yourself: * Binary classifier for UCI Census dataset salary prediction * Dataset: UCI Census * Task: Predict whether a person earns more or less than \$50k based on their census information * To build and run the demo from code: `bazel run wit_dashboard/demo :demoserver` then navigate to `http://localhost:6006/wit-dashboard/demo.html` * Binary classifier for smile detection in images * Dataset: CelebA * Task: Predict whether the person in an image is smiling * To build and run the demo from code: `bazel run wit_dashboard/`

`demo:imagedemoserver` then navigate to http://localhost:6006/wit-dashboard/image_demo.html * Multiclass classifier for Iris dataset * Dataset: UCI Iris * Task: Predict which of three classes of iris flowers that a flower falls into based on 4 measurements of the flower * To build and run the demo from code: `bazel run wit_dashboard/demo:irisdemoserver` then navigate to http://localhost:6006/wit-dashboard/iris_demo.html * Regression model for UCI Census dataset age prediction * Dataset: UCI Census * Task: Predict the age of a person based on their census information * To build and run the demo from code: `bazel run wit_dashboard/demo:agedemoserver` then navigate to http://localhost:6006/wit-dashboard/age_demo.html * This demo model returns attribution values in addition to predictions (through the use of vanilla gradients) in order to demonstrate how the tool can display attribution values from predictions.

What do I need to use it in a jupyter or colab notebook?

You can use the What-If Tool to analyze a classification or regression TensorFlow Estimator that takes TensorFlow Example or SequenceExample protos (data points) as inputs directly in a jupyter or colab notebook.

Additionally, the What-If Tool can analyze AI Platform Prediction-hosted classification or regression models that take TensorFlow Example protos, SequenceExample protos, or raw JSON objects as inputs.

You can also use What-If Tool with a custom prediction function that takes Tensorflow examples and produces predictions. In this mode, you can load any model (including non-TensorFlow models that don't use Example protos as inputs) as long as your custom function's input and output specifications are correct.

With either AI Platform models or a custom prediction function, the What-If Tool can display and make use of attribution values for each input feature in relation to each prediction. See the below section on attribution values for more information.

If you want to train an ML model from a dataset and explore the dataset and model, check out the `What_If_Tool_Notebook_Usage.ipynb` notebook in colab, which starts from a CSV file, converts the data to tf.Example protos, trains a classifier, and then uses the What-If Tool to show the classifier performance on the data.

What do I need to use it in TensorBoard?

A walkthrough of using the tool in TensorBoard, including a pretrained model and test dataset, can be found on the What-If Tool page on the TensorBoard website.

To use the tool in TensorBoard, only the following information needs to be provided:

- The model server host and port, served using TensorFlow Serving. The model can use the TensorFlow Serving Classification, Regression, or Predict API.
 - Information on how to create a saved model with the [Estimator](#) API that will use these appropriate TensorFlow Serving Classification or Regression APIs can be found in the saved model documentation and in this helpful tutorial. Models that use these APIs are the simplest to use with the What-If Tool as they require no set-up in the tool beyond setting the model type.
 - If the model uses the Predict API, the input must be serialized `tf.Example` or `tf.SequenceExample` protos and the output must be following:
 - ★ For classification models, the output must include a 2D float tensor containing a list of class probabilities for all possible class indices for each inferred example.
 - ★ For regression models, the output must include a float tensor containing a single regression score for each inferred example.
 - The What-If Tool queries the served model using the gRPC API, not the RESTful API. See the TensorFlow Serving docker documentation for more information on the two APIs. The docker image uses port 8500 for the gRPC API, so if using the docker approach, the port to specify in the What-If Tool will be 8500.
 - Alternatively, instead of querying a model hosted by TensorFlow Serving, you can provide a python function for model prediction to the tool through the “-whatif-use-unsafe-custom-prediction” runtime argument as described in more detail below.
- A TFRecord file of `tf.Examples` or `tf.SequenceExamples` to perform inference on and the number of examples to load from the file.
 - Can handle up to tens of thousands of examples. The exact amount depends on the size of each example (how many features there are and how large the feature values are).
 - The file must be in the logdir provided to TensorBoard on startup. Alternatively, you can provide another directory to allow file loading from, through use of the -whatif-data-dir=PATH runtime parameter.
- An indication if the model is a regression, binary classification or multi-class classification model.
- An optional vocab file for the labels for a classification model. This file maps the predicted class indices returned from the model prediction into class labels. The text file contains one label per line, corresponding to the class indices returned by the model, starting with index 0.
 - If this file is provided, then the dashboard will show the predicted labels for a classification model. If not, it will show the predicted class indices.

Alternatively, the What-If Tool can be used to explore a dataset directly from a CSV file. See the next section for details.

The information can be provided in the settings dialog screen, which pops up automatically upon opening this tool and is accessible through the settings icon button in the top-right of the tool. The information can also be provided directly through URL parameters. Changing the settings through the controls automatically updates the URL so that it can be shared with others for them to view the same data in the What-If Tool.

All I have is a dataset. What can I do in TensorBoard? Where do I start?

If you just want to explore the information in a CSV file using the What-If Tool in TensorBoard, just set the path to the examples to the file (with a “.csv” extension) and leave the inference address and model name fields blank. The first line of the CSV file must contain column names. Each line after that contains one example from the dataset, with values for each of the columns defined on the first line. The pipe character (“|”) delimitates separate feature values in a list of feature values for a given feature.

In order to make use of the model understanding features of the tool, you can have columns in your dataset that contain the output from an ML model. If your file has a column named “predictions__probabilities” with a pipe-delimited (“|”) list of probability scores (between 0 and 1), then the tool will treat those as the output scores of a classification model. If your file has a numeric column named “predictions” then the tool will treat those as the output of a regression model. In this way, the tool can be used to analyze any dataset and the results of any model run offline against the dataset. Note that in this mode, the examples aren’t editable as there is no way to get new inference results when an example changes.

What can it do?

Details on the capabilities of the tool, including a guided walkthrough, can be found on the What-If Tool website. Here is a basic rundown of what it can do:

- Visualize a dataset of TensorFlow Example protos.
 - The main panel shows the dataset using Facets Dive, where the examples can be organized/sliced/positioned/colored by any of the dataset’s features.
 - ★ The examples can also be organized by the results of their inferences.
 - For classification models, this includes inferred label, confidence of inferred label, and inference correctness.

-
- For regression models, this includes inferred score and amount of error (including absolute or squared error) in the inference.
 - A selected example can be viewed in detail in the side panel, showing all feature values for all features of that example.
 - For examples that contain an encoded image in a bytes feature named “image/encoded”, Facets Dive will create a thumbnail of the image to display the point, and the full-size image is visible in the side panel for a selected example.
 - Aggregate statistics for all loaded examples can be viewed in the side panel using Facets Overview.
- Visualize the results of the inference
 - By default, examples in the main panel are colored by their inference results.
 - The examples in the main panel can be organized into confusion matrices and other custom layouts to show the inference results faceted by a number of different features, or faceted/positioned by inference result, allowing the creation of small multiples of 1D and 2D histograms and scatter plots.
 - For a selected example, detailed inference results (e.x. predicted classes and their confidence scores) are shown in the side panel.
 - If the model returns attribution values in addition to predictions, they are displayed for each selected example, and the attribution values can be used to control custom layouts and as dimensions to slice the dataset on for performance analysis.
 - Explore counterfactual examples
 - For classification models, for any selected example, with one click you can compare the example to the example most-similar to it but which is classified as a different.
 - Similarity is calculated based on the distribution of feature values across all loaded examples and similarity can be calculated using either L1 or L2 distance.
 - * Distance is normalized between features by:
 - For numeric features, use the distance between values divided by the standard deviation of the values across all examples.
 - For categorical features, the distance is 0 if the values are the same, otherwise the distance is the probability that any two examples have the same value for that feature across all examples.
 - In notebook mode, the tool also allows you to set a custom distance function using `set_custom_distance_fn` in `WitConfigBuilder`, where that function is used to compute closest counterfactuals instead. As in the case with `custom_predict_fn`, the custom distance function can be any python function.
-

-
- Edit a selected example in the browser and re-run inference and visualize the difference in the inference results.
 - See auto-generated partial dependence plots, which are plots that for every feature show the change in inference results as that feature has its value changed to different valid values for that feature.
 - Edit/add/remove any feature or feature value in the side panel and re-run inference on the edited datapoint. A history of the inference values of that point as it is edited and re-inferred is shown.
 - For examples that contain encoded images, upload your own image and re-run inference.
 - Clone an existing example for editing/comparison.
 - Revert edits to an edited example.
 - Compare the results of two models on the same input data.
 - If you provide two models to the tool during setup, it will run inference with the provided data on both models and you can compare the results between the two models using all the features defined above.
 - If using a binary classification model and your examples include a feature that describes the true label, you can do the following:
 - See the ROC curve and numeric confusion matrices in the side panel, including the point on the curve where your model lives, given the current positive classification threshold value.
 - See separate ROC curves and numeric confusion matrices split out for subsets of the dataset, sliced of any feature or features of your dataset (i.e. by gender).
 - Manually adjust the positive classification threshold (or thresholds, if slicing the dataset by a feature) and see the difference in inference results, ROC curve position and confusion matrices immediately.
 - Set the positive classification thresholds with one click based on concepts such as the cost of a false positive vs a false negative and satisfying fairness measures such as equality of opportunity or demographic parity.
 - If using a multi-class classification model and your examples include a feature that describes the true label, you can do the following:
 - See a confusion matrix in the side panel for all classifications and all classes.
 - See separate confusion matrices split out for subsets of the dataset, sliced on any feature or features of your dataset.
 - If using a regression model and your examples include a feature that describes the true label, you can do the following:
-

-
- See the mean error, mean absolute error and mean squared error across the dataset.
 - See separate calculations of those mean error calculations split out for subsets of the dataset, sliced of any feature or features of your dataset.

Who is it for?

We imagine WIT to be useful for a wide variety of users. * ML researchers and model developers - Investigate your datasets and models and explore inference results. Poke at the data and model to gain insights, for tasks such as debugging strange results and looking into ML fairness. * Non-technical stakeholders - Gain an understanding of the performance of a model on a dataset. Try it out with your own data. * Lay users - Learn about machine learning by interactively playing with datasets and models.

Notebook mode details

As seen in the example notebook, creating the `WitWidget` object is what causes the What-If Tool to be displayed in an output cell. The `WitWidget` object takes a `WitConfigBuilder` object as a constructor argument. The `WitConfigBuilder` object specifies the data and model information that the What-If Tool will use.

The `WitConfigBuilder` object takes a list of `tf.Example` or `tf.SequenceExample` protos as a constructor argument. These protos will be shown in the tool and inferred in the specified model.

The model to be used for inference by the tool can be specified in many ways: - As a TensorFlow Estimator object that is provided through the `set_estimator_and_feature_spec` method. In this case the inference will be done inside the notebook using the provided estimator. - As a model hosted by AI Platform Prediction through the `set_ai_platform_model` method. - As a custom prediction function provided through `set_custom_predict_fn` method. In this case WIT will directly call the function for inference. - As an endpoint for a model being served by TensorFlow Serving, through the `set_inference_address` and `set_model_name` methods. In this case the inference will be done on the model server specified. To query a model served on host “localhost” on port 8888, named “my_model”, you would set on your builder `builder.set_inference_address('localhost:8888').set_model_name('my_model')`.

See the documentation of `WitConfigBuilder` for all options you can provide, including how to specify other model types (defaults to binary classification) and how to specify an optional second model to compare to the first model.

How can the What-If Tool use attribution values and other prediction-time information?

Feature attribution values are numeric values for each input feature to an ML model that indicate how much impact that feature value had on the model's prediction. There are a variety of approaches to get feature attribution values for a predicts from an ML model, including SHAP, Integrated Gradients, SmoothGrad, and more.

They can be a powerful way of analyzing how a model reacts to certain input values beyond just simply studying the effect that changing individual feature values has on model predictions as is done with partial dependence plots. Some attribution techniques require access to a model internals, such as the gradient-based methods, whereas others can be performed on black-box models. Regardless, the What-If Tool can visualize the results of attribution methods in addition to the standard model prediction results.

There are two ways to use the What-If Tool to visualize attribution values. If you have deployed a model to Cloud AI Platform with the explainability feature enabled, and provide this model to the tool through the standard `set_ai_platform_model` method, then attribution values will automatically be generated and visualized by the tool with no additional setup needed. If you wish to view attribution values for a different model setup, this can be accomplished through use of the custom prediction function.

As described in the `set_custom_predict_fn` documentation in `WitConfigBuilder`, this method must return a list of the same size as the number of examples provided to it, with each list entry representing the prediction-time information for that example. In the case of a standard model with no attribution information, the list entry is just a number (in the case of a regression model), or a list of class probabilities (in the case of a classification model).

However, if there is attribution or other prediction-time information, then the list entry can instead be a dictionary, with the standard model prediction output under the `predictions` key. Attribution information can be returned under the `attributions` key and any other supplemental information under its own descriptive key. The exact format of the attributions and other supplemental information can be found in the code documentation linked above.

If attribution values are provided to the What-If Tool, they can be used in a number of ways. First, when selecting a datapoint in the Datapoint Editor tab, the attribution values are displayed next to each feature value and the features can be ordered by their attribution strength instead of alphabetically. Additionally, the feature values are colored by their attribution values for quick interpretation of attribution strengths.

Beyond displaying the attribution values for the selected datapoint, the attribution values for each feature can be used in the tool in the same ways as any other feature of the datapoints. They can be used selected in the datapoints visualization controls to use those values to create custom scatter

plots and histograms. For example, you can create a scatterplot showing the relationship between the attribution value of two different features, with the datapoints colored by the predicted result from the model. They can also be used in the Performance tab as a way to slice a dataset for comparing performance statistics of different slices. For example, you can quickly compare the aggregate performance of a model on datapoints with low attribution of a specified feature, against the datapoints with high attribution of that feature.

Any other supplemental information returned from a custom prediction function will appear in the tool as a feature named after its key in the dictionary. They can also be used in the same way, driving custom visualizations and as a dimension to slice when analyzing aggregate model performance.

When a datapoint is edited and the re-inferred through the model with the “Run inference” button, the attributions and other supplemental information is recalculated and updated in the tool.

For an example of returning attribution values from a custom prediction function (in this case using the SHAP library to get attributions), see the WIT COMPAS with SHAP notebook.

How do I enable it for use in a Jupyter notebook?

First, install and enable WIT for Jupyter through the following commands:

```
1 pip install witwidget
2 jupyter nbextension install --py --symlink --sys-prefix witwidget
3 jupyter nbextension enable --py --sys-prefix witwidget
```

Then, use it as seen at the bottom of the `What_If_Tool_Notebook_Usage.ipynb` notebook.

How do I enable it for use in a Colab notebook?

Install the widget into the runtime of the notebook kernel by running a cell containing:

```
1 !pip install witwidget
```

Then, use it as seen at the bottom of the `What_If_Tool_Notebook_Usage.ipynb` notebook.

How do I enable it for use in a JupyterLab or Cloud AI Platform notebook?

WIT has been tested in JupyterLab versions 1.x, 2.x, and 3.x.

Install and enable WIT for JupyterLab 3.x by running a cell containing:

```
1 !pip install witwidget
2 !jupyter labextension install wit-widget
```

```
3 !jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

Note that you may need to specify the correct version of `jupyterlab-manager` for your JupyterLab version as per <https://www.npmjs.com/package/@jupyter-widgets/jupyterlab-manager>.

Note that you may need to run `!sudo jupyter labextension ...` commands depending on your notebook setup.

Use of WIT after installation is the same as with the other notebook installations.

Can I use a custom prediction function in TensorBoard?

Yes. You can do this by defining a python function named `custom_predict_fn` which takes two arguments: a list of examples to preform inference on, and the serving bundle object which contains information about the model to query. The function should return a list of results, one entry per example provided. For regression models, the result is just a number. For classification models, the result is a list of numbers, representing the class scores for each possible class. Here is a minimal example that just returns random results:

```
1 import random
2
3 # The function name "custom_predict_fn" must be exact.
4 def custom_predict_fn(examples, serving_bundle):
5     # Examples are a list of TFRecord objects, each object contains the
6     # features of each point.
7     # serving_bundle is a dictionary that contains the setup information
8     # provided to the tool,
9     # such as server address, model name, model version, etc.
10
11     number_of_examples = len(examples)
12     results = []
13     for _ in range(number_of_examples):
14         score = random.random()
15         results.append([score, 1 - score]) # For binary classification
16         # results.append(score) # For regression
17     return results
```

Define this function in a file you save to disk. For this example, let's assume the file is saved as `/tmp/my_custom_predict_function.py`. Then the TensorBoard server with `tensorboard --whatif-use-unsafe-custom-prediction /tmp/my_custom_predict_function.py` and the function should be invoked once you have set up your data and model in the What-If Tool setup dialog. The `unsafe` means that the function is not sandboxed, so make sure that your function doesn't do anything destructive, such as accidentally delete your experiment data.

How can I help develop it?

Check out the development guide.

What's new in WIT?

Check out the release notes.