Using the ‘What-If Tool’ to investigate Machine Learning models.

An open source tool from Google to easily analyze ML models without the need to code.

Parul Pandey May 3, 2019 · 9 min read ★

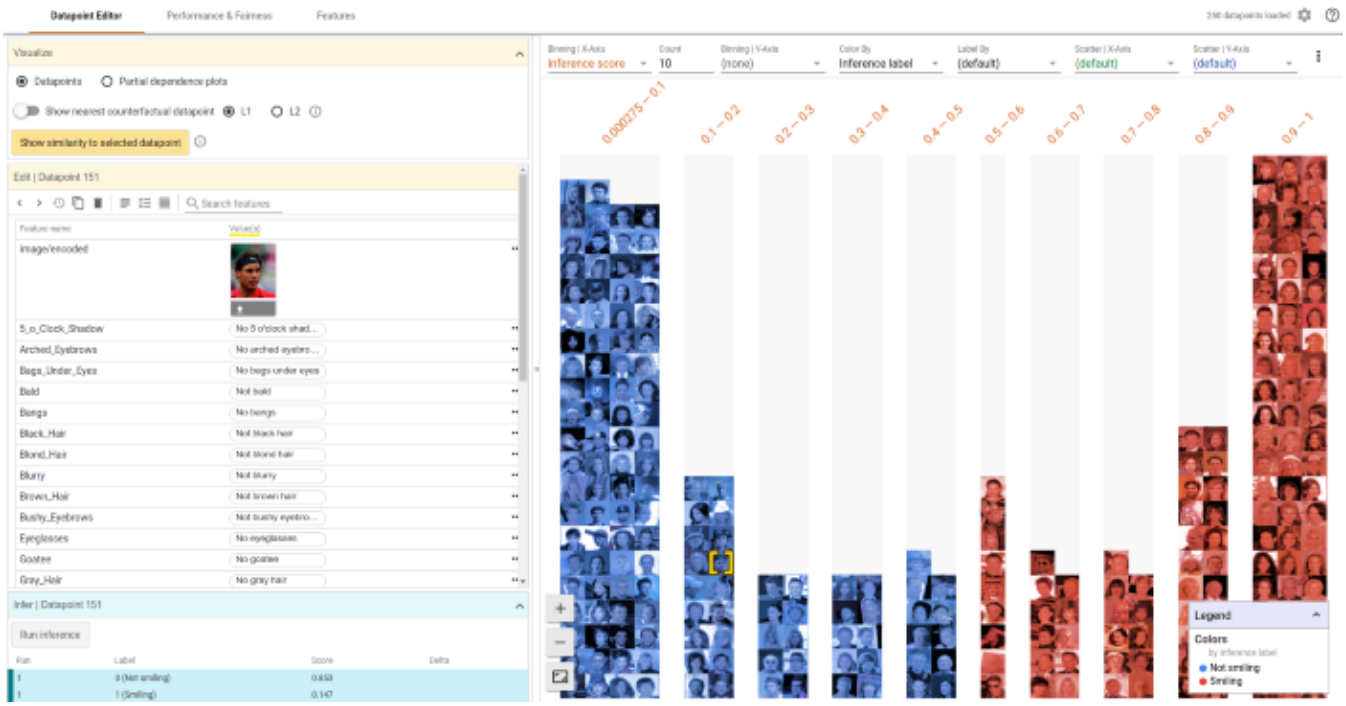
Photo by Pixabay from Pexels

understand their model better

In this era of explainable and interpretable Machine Learning, one merely cannot be content with simply training the model and obtaining predictions from it. To be able to really make an impact and obtain good results, we should also be able to probe and investigate our models. Apart from that, algorithmic fairness constraints and bias should also be clearly kept in mind before going ahead with the model.

Investigating a model requires asking a lot of questions and one needs to have an acumen of a detective to probe and look for issues and inconsistencies within the models. Also, such a task is usually complex requiring to write a lot of custom code. Fortunately, the **What-If Tool** has been created to address this issue making it easier for a broad set of people to examine, evaluate, and debug ML systems easily and accurately.

What-If Tool(WIT)



Source

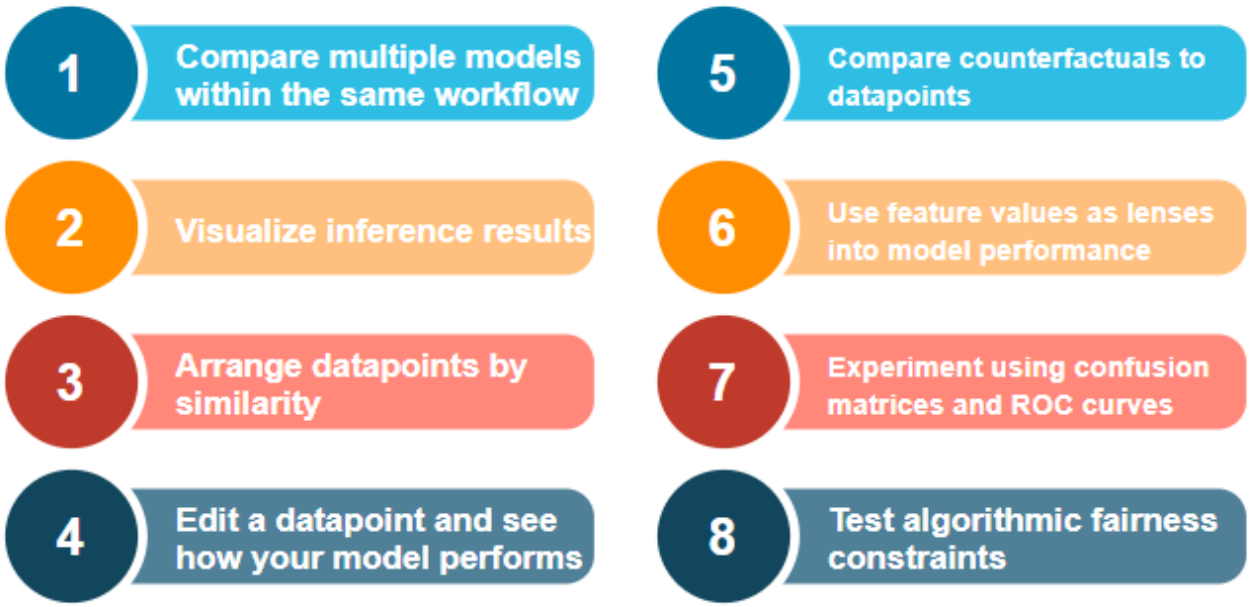
or Regression model by enabling people to examine, evaluate, and compare machine learning models. Due to its user-friendly interface and less dependency on complex coding, everyone from a developer, a product manager, a researcher or a student can use it for their purpose.

WIT is an open-source visualisation tool released by Google under the **PAIR(People + AI Research)** initiative. PAIR brings together researchers across Google to study and redesign the ways people interact with AI systems.

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

Advantages

The purpose of the tool is to give people a simple, intuitive, and a powerful way to play with a trained ML model on a set of data through a visual interface only. Here are the major advantages of WIT.

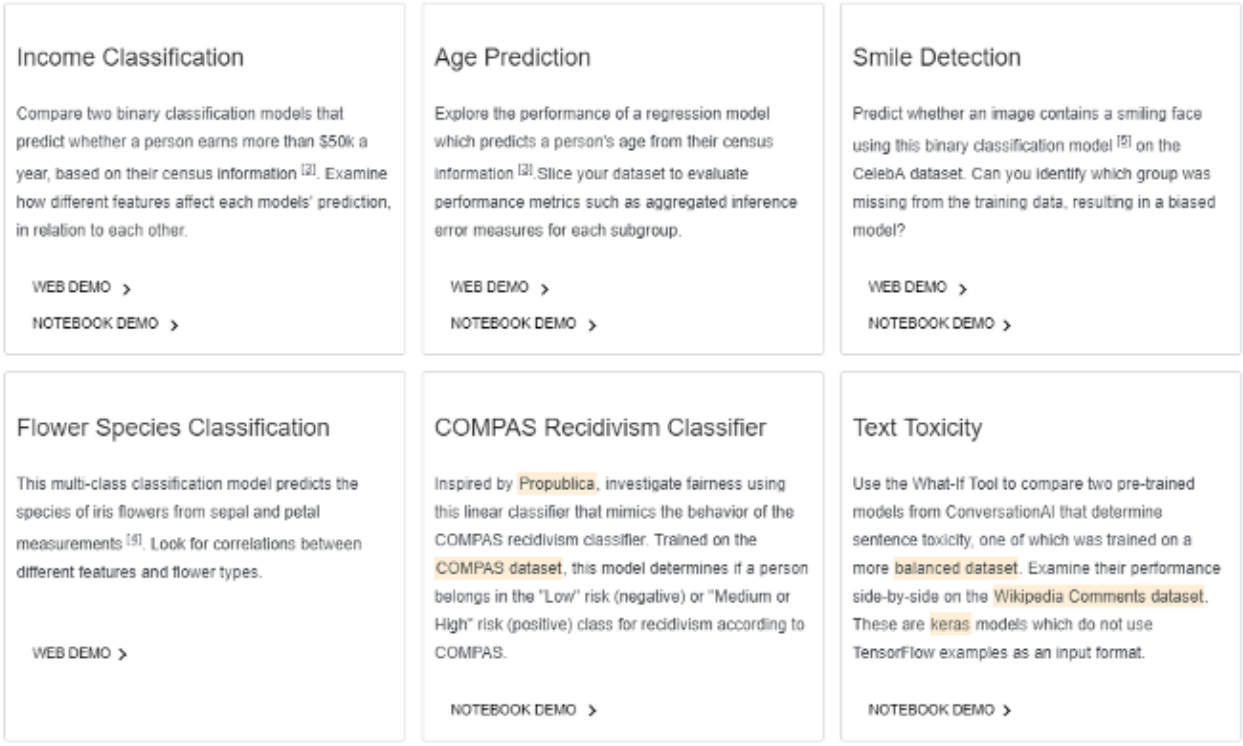


What can you do with the What-If Tool?

We shall cover all the above points during an example walkthrough using the tool.

Demos

To illustrate the capabilities of the What-If Tool, the PAIR team has released a set of demos using pre-trained models. You can either run the demos in the notebook or directly through the web.



Take the What-If Tool for a spin!

Usage

WIT can be used inside a Jupyter or Colab notebook, or inside the TensorBoard web application. This has been nicely and clearly explained in the documentation and I highly encourage you to go through that since explaining the entire process wouldn't be possible through this short article.

> *The whole idea is to first train a model and then visualizes the results of the trained classifier on test data using the What-If Tool.*

Using WIT with Tensorboard

file. For more details, refer to the documentation for using WIT in TensorBoard.

Using WIT with Notebooks

To be able to access WIT within notebooks, you need a WitConfigBuilder object that specifies the data and model to be analyzed. This documentation provides a step-by-step outline for using WIT in a notebook.

```
from witwidget.notebook.visualization import WitConfigBuilder
from witwidget.notebook.visualization import WitWidget

config_builder = WitConfigBuilder(test_examples).set_estimator_and_feature_spec(
    classifier, feature_spec)
WitWidget(config_builder)
```

You can also use a demo notebook and edit the code to include your datasets to start working.

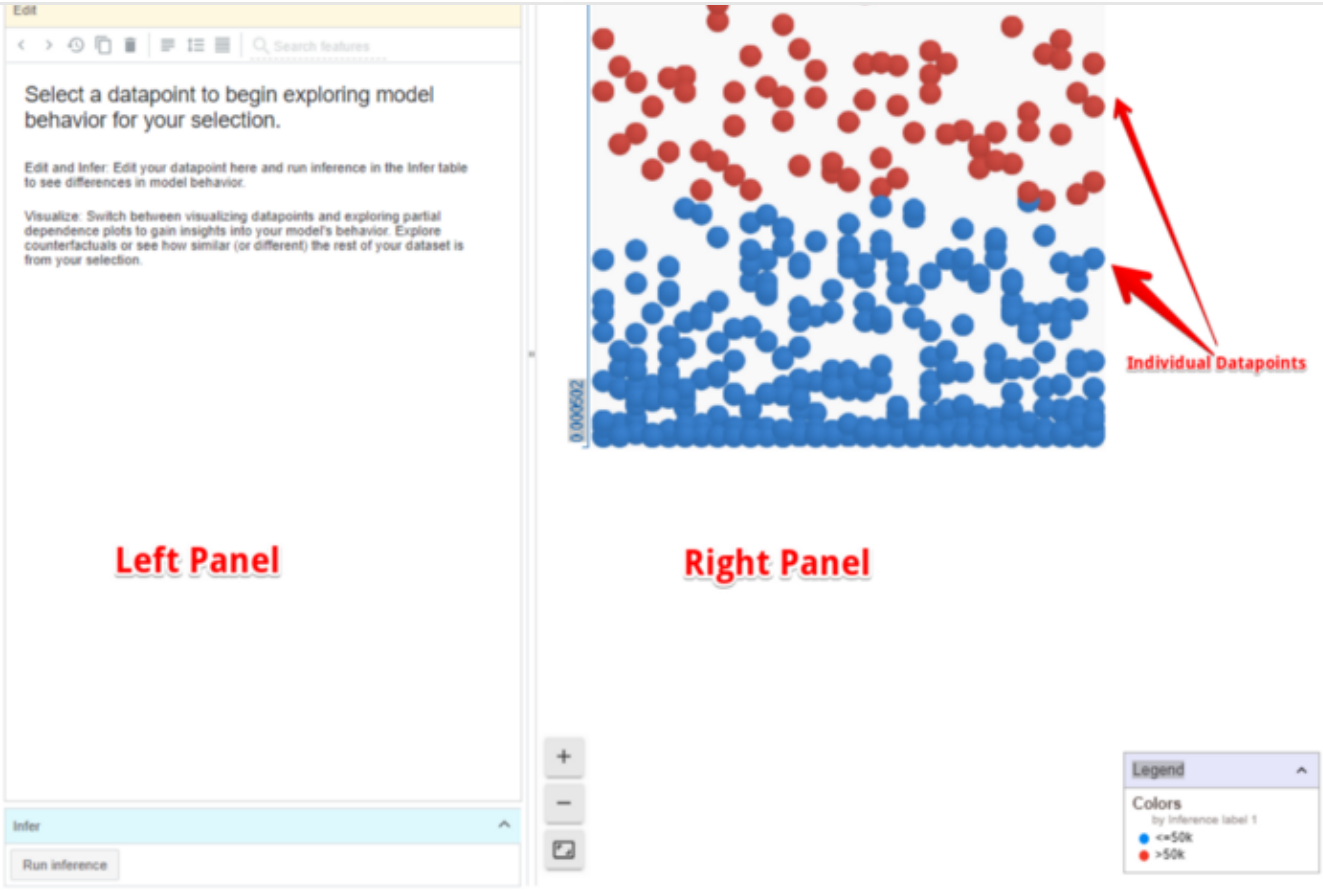
Walkthrough

Let's now explore the capabilities of the WIT tool with an example. The example has been taken from the demos provided on the website and is called **Income Classification** wherein we need to predict whether a person earns more than $50k a year, based on their census information. The Dataset belongs to the **UCI Census dataset** consisting of a number of attributes such as age, marital status and education level.

Overview

Let's begin by doing some Exploration of the dataset. Here is a link to the web Demo for following along.

What-if tool contains two main panels. The **right panel** contains a visualization of the individual data points in the data set you have loaded.



In this case, the **blue dots** are people for whom the model has inferred an income of **less than 50k** and the **red dots** are those that the model inferred earn **more than 50k.** By default, WIT uses a positive classification threshold of 0.5. This means that if the inference score is 0.5 or more, the data point is considered to be in a positive class, i.e. high income.

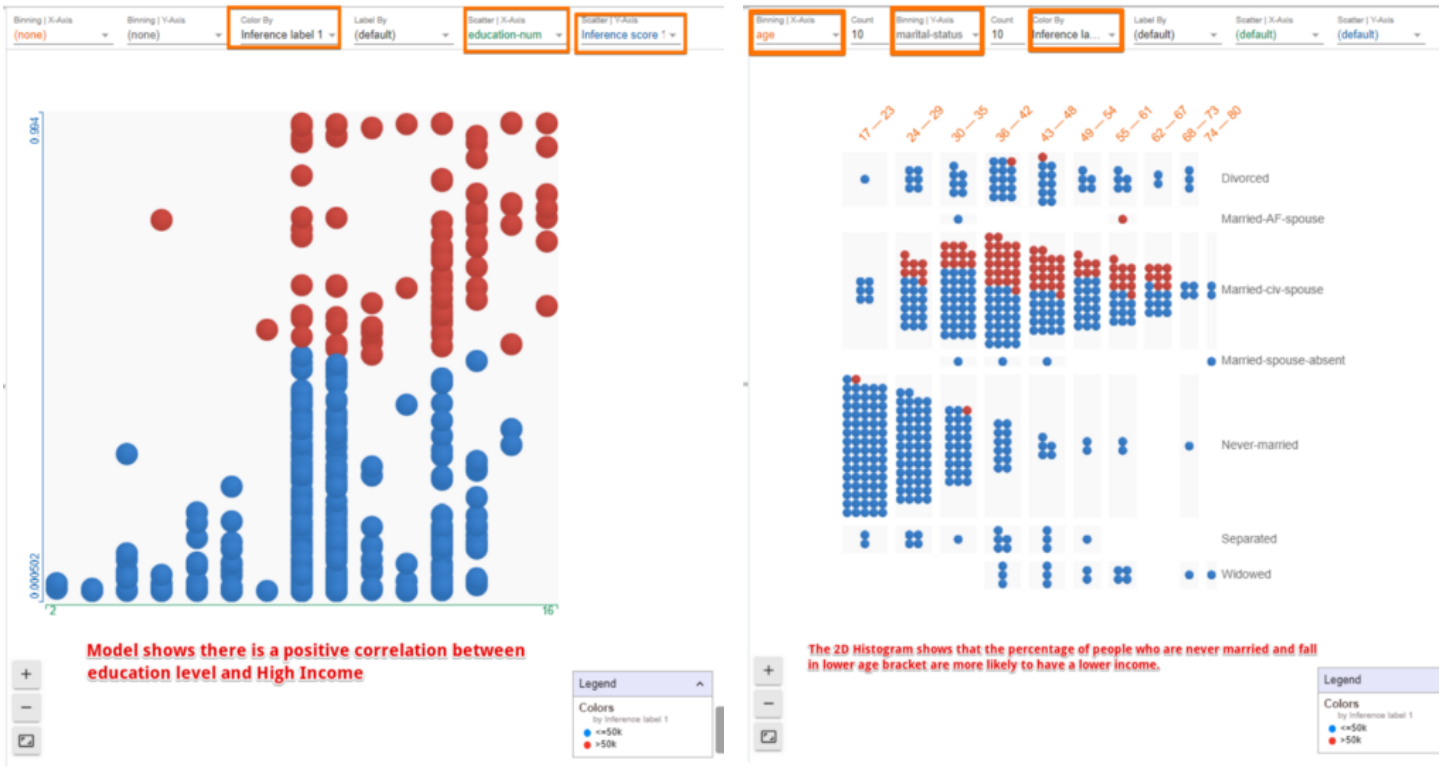
> *What is interesting to note here is that the dataset is visualized in* ***Facets Dive****. Facets Dive is a part of the* ***FACETS****' tool developed again by the PAIR team and helps us to understand the various features of data and explore them. In case you are not familiar with the tool, you may want to refer to this article on FACETS' capabilities, which I had written a while ago.*

Visualising Machine Learning Datasets with Google's FACETS.

An open source tool from Google to easily learn patterns from large amounts of data

towardsdatascience.com

fields from the drop-down menu. A few examples have been presented below.



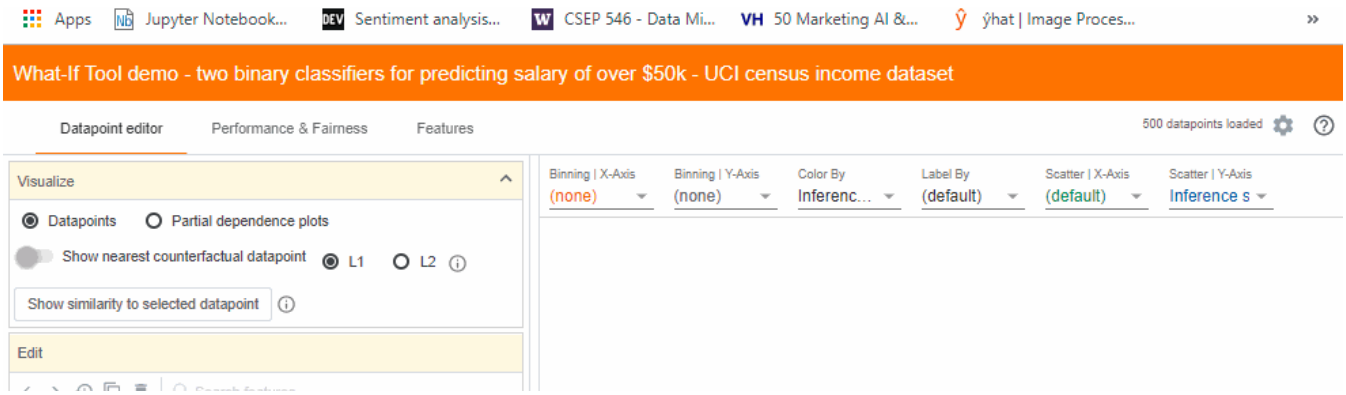
The **left panel** contains three tabs called `Datapoint Editor, Performance & Fairness` ; and `Features.`

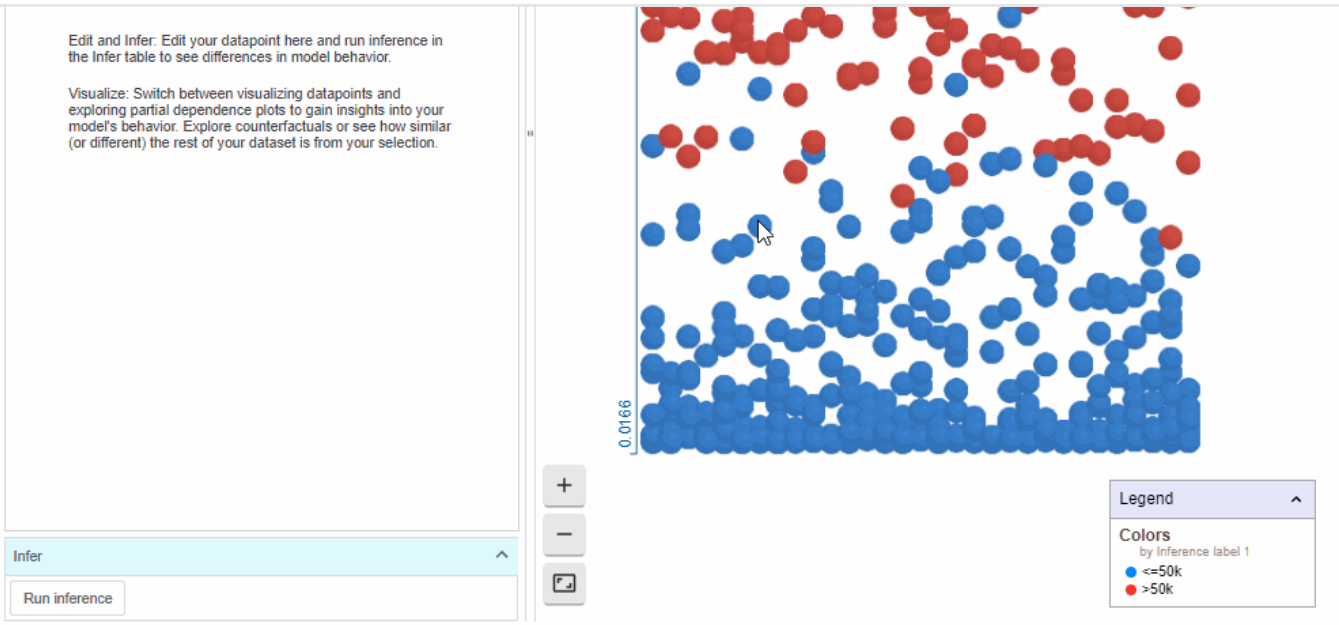
1. Datapoint Editor Tab

The Datapoint Editor helps to perform data analysis through:

- **Viewing and Editing details of Datapoints**

It allows diving into a selected data point which gets highlighted in yellow on the right panel. Let's try changing the age from 53 to 58 and clicking the "Run inference" button to see what effect it has on the model's performance.

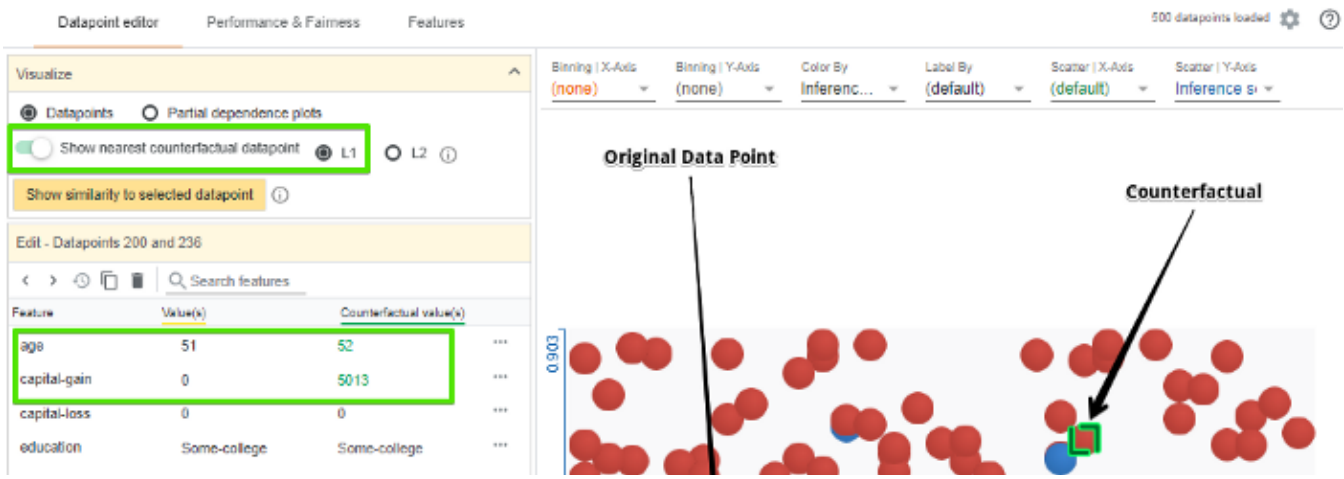




By simply changing the age of this person, the model now predicts that the person belongs to high-income category. For this data point, earlier the inference score for the positive (high income) class was 0.473, and the score for negative (low income) class was 0.529. However, by changing the age, the positive class score became 0.503.

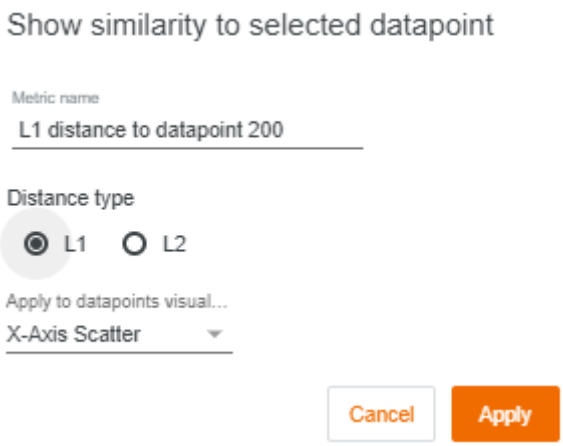
- **Finding Nearest Counterfactuals**

Another way to understand the model's behaviour is to look at what small set of changes can cause the model to flip its decision which is called **counterfactuals**. With one click we can see the most similar counterfactual, which is highlighted in green, to our selected data point. In the data point editor tab we now also see the feature values for the counterfactual next to the feature values for our original data point. The green text represents features where the two data points differ. WIT uses L1 and L2 distances to calculate the similarity between the data points.



In this case, the nearest counterfactual is slightly older and has a different occupation and capital gain, but is otherwise identical.

We can also see the similarity between the selected points and others using the "**show similarity to selected datapoint**" button. WIT measures the distance from the selected point to every other datapoint. Let's change our X-axis scatter to show the L1 distance to the selected datapoint.



- **Analysing partial dependence plots**

The partial dependence plot (short PDP or PD plot) shows the marginal effect one or two features have on the predicted outcome of a machine learning model(J. H. Friedman 2001).

The plot above shows that:

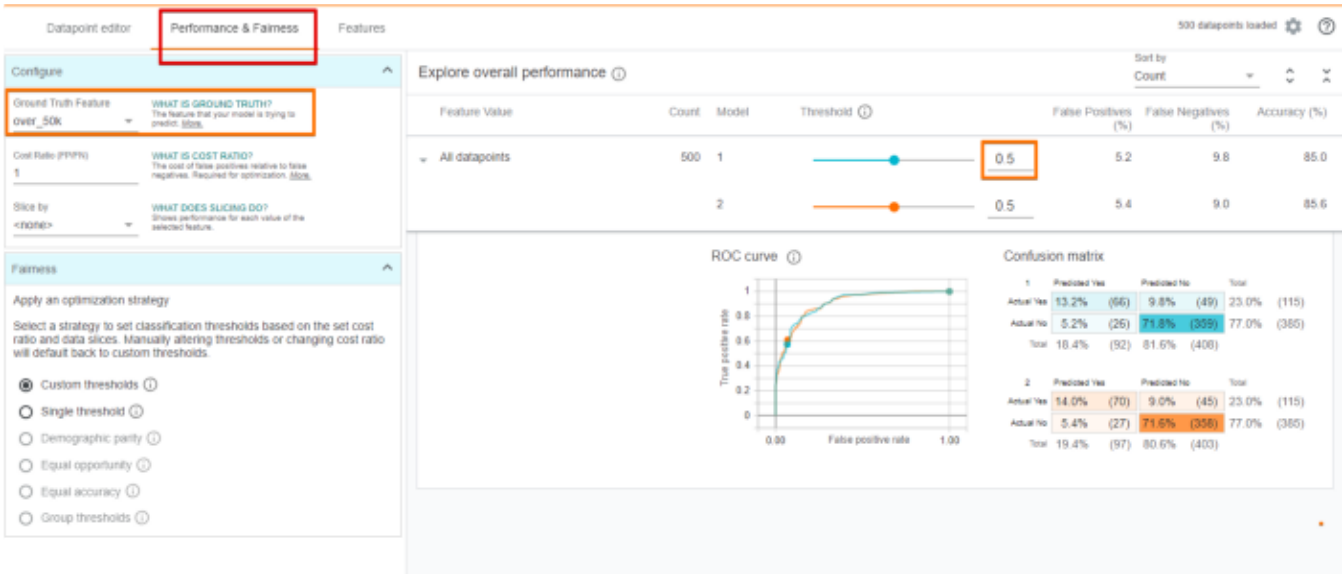
- The model has learned a positive correlation between age and income
- More advanced degrees give the model more confidence in higher income.
- High capital gains is a very strong indicator of high income, much more than any other single feature.

2. Performance & Fairness Tab

This tab allows us to look at the overall model performance using confusion matrices and ROC curves.

- **Model Performance Analysis**

50K".

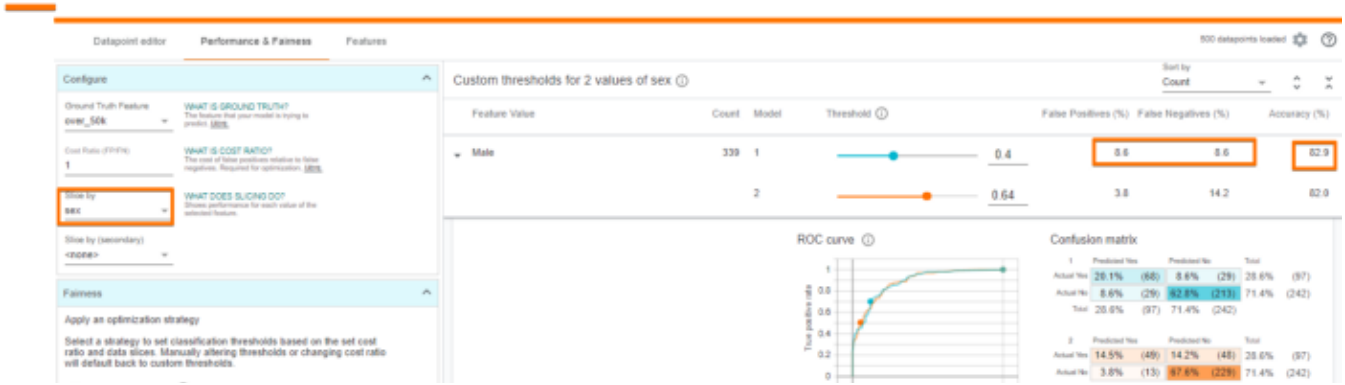


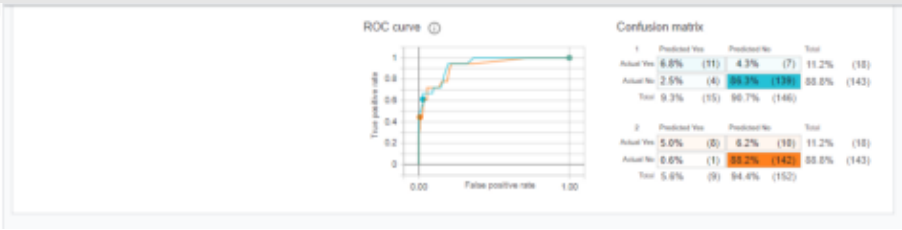
We can see that at the default threshold level of 0.5, our model is incorrect about 15% of the time, with about 5% of the time being false positives and 10% of the time being false negatives. Change the threshold values to see its impact on the model's accuracy.

There is also a setting for "**cost ratio**" and an "**optimize threshold**" button which can also be tweaked.

- **ML Fairness**

Fairness in Machine Learning is as important as model building and predicting an outcome. Any bias in the training data will be reflected in the trained model and if such a model is deployed, the resultant outputs will also be biased. The WIT can help investigate fairness concerns in a few different ways. We can set an input feature (or set of features) with which to slice the data. For example, let's see the effect of gender on model performance.





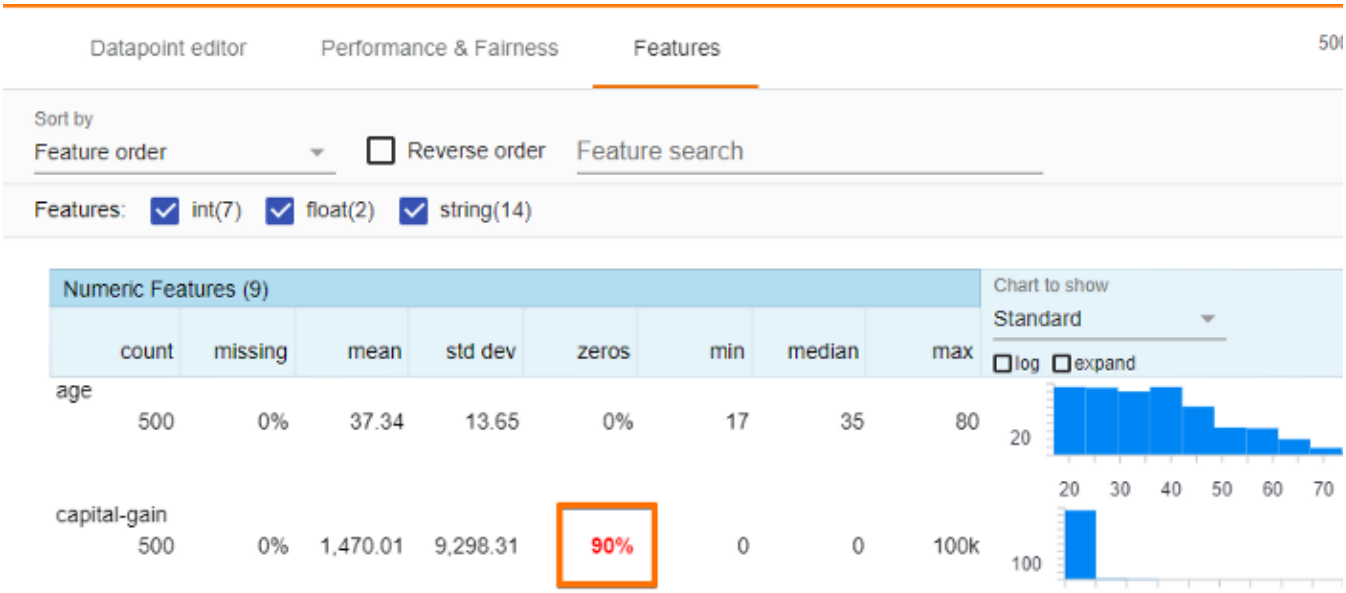
Effect of gender on Model's performance

We can see that the model is more accurate on females than males. Also, the model predicts high income for females much less than it does for males (9.3% of the time for females vs 28.6% of the time for males). One probable reason might be due to the under-representation of females in the dataset which we shall explore in the next section.

Additionally, the tool can optimally set the decision threshold for the two subsets while taking into account any of a number of constraints related to algorithmic fairness such as demographic parity or equal opportunity.

3. Features Tab

The features tab gives the summary statistics of each of the features in the dataset including histograms, quantile charts, bar charts etc. The tab also enables to look into the distribution of values for each feature in the dataset. For instance, let us explore the sex, capital gain and race features.



Native Country DIstribution || Sex distribution

Similarly, most datapoints belong to the United States while females are not well represented in the dataset. Since the data is biased, it is but natural that its predictions are targeted towards one group only. Afterall a model learns from the data it is provided and if the source is skewed so will be the results. Machine learning has proved its mettle in a lot of applications and areas. However, one of the key hurdles for industrial applications of machine learning models is to determine whether the raw input data used to train the model contains discriminatory bias or not.

Conclusion

This was just a quick run-through of some of the what if tools features. WIT is a pretty handy tool which gives the ability to probe the models, into the hands of the people to whom it matters the most. Simply creating and training a model isn't the purpose of Machine Learning but understanding why and how that model was created is Machine Learning in true sense.

References:

1. The What-If Tool: Code-Free Probing of Machine Learning Models
2. https://pair-code.github.io/what-if-tool/walkthrough.html
3. https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/interactive_inference