In [1]:
```
# predicting whether a person earns more than $50K from their census information

# Define useful functions
import pandas as pd
import numpy as np
import tensorflow as tf
import functools

# Creates a tf feature spec from the dataframe and columns specified.
def create_feature_spec(df, columns=None):
    feature_spec = {}
    if columns == None:
        columns = df.columns.values.tolist()
    for f in columns:
        if df[f].dtype is np.dtype(np.int64):
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.int64)
        elif df[f].dtype is np.dtype(np.float64):
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.float32)
        else:
            feature_spec[f] = tf.io.FixedLenFeature(shape=(), dtype=tf.string)
    return feature_spec

# Creates simple numeric and categorical feature columns from a feature spec and a
# list of columns from that spec to use.
#
# NOTE: Models might perform better with some feature engineering such as bucketed
# numeric columns and hash-bucket/embedding columns for categorical features.
def create_feature_columns(columns, feature_spec):
    ret = []
    for col in columns:
        if feature_spec[col].dtype is tf.int64 or feature_spec[col].dtype is tf.float32:
            ret.append(tf.feature_column.numeric_column(col))
        else:
            ret.append(tf.feature_column.indicator_column(
                tf.feature_column.categorical_column_with_vocabulary_list(col, list(df[col].unique()))))
    return ret

# An input function for providing input to a model from tf.Examples
def tfexamples_input_fn(examples, feature_spec, label, mode=tf.estimator.ModeKeys.EVAL,
                        num_epochs=None,
                        batch_size=64):
    def ex_generator():
        for i in range(len(examples)):
            yield examples[i].SerializeToString()
    dataset = tf.data.Dataset.from_generator(
      ex_generator, tf.dtypes.string, tf.TensorShape([]))
    if mode == tf.estimator.ModeKeys.TRAIN:
        dataset = dataset.shuffle(buffer_size=2 * batch_size + 1)
    dataset = dataset.batch(batch_size)
    dataset = dataset.map(lambda tf_example: parse_tf_example(tf_example, label, feature_spec))
    dataset = dataset.repeat(num_epochs)
    return dataset

# Parses Tf.Example protos into features for the input function.
def parse_tf_example(example_proto, label, feature_spec):
    parsed_features = tf.io.parse_example(serialized=example_proto, features=feature_spec)
    target = parsed_features.pop(label)
    return parsed_features, target

# Converts a dataframe into a list of tf.Example protos.
def df_to_examples(df, columns=None):
    examples = []
    if columns == None:
        columns = df.columns.values.tolist()
    for index, row in df.iterrows():
        example = tf.train.Example()
        for col in columns:
            if df[col].dtype is np.dtype(np.int64):
                example.features.feature[col].int64_list.value.append(int(row[col]))
            elif df[col].dtype is np.dtype(np.float64):
                example.features.feature[col].float_list.value.append(row[col])
            elif row[col] == row[col]:
                example.features.feature[col].bytes_list.value.append(str(row[col]).encode('utf-8'))
        examples.append(example)
    return examples

# Converts a dataframe column into a column of 0's and 1's based on the provided test.
# Used to force label columns to be numeric for binary classification using a TF estimator.
def make_label_column_numeric(df, label_column, test):
  df[label_column] = np.where(test(df[label_column]), 1, 0)
```

In [2]:
```
# Read training dataset from CSV
import pandas as pd

# Set the path to the CSV containing the dataset to train on.
# Source: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
csv_path = './adultdata/adult.train.txt'
# csv_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'

# Set the column names for the columns in the CSV. If the CSV's first line is a header line containing
# the column names, then set this to None.
csv_columns = [
  "Age", "Workclass", "fnlwgt", "Education", "Education-Num", "Marital-Status",
  "Occupation", "Relationship", "Race", "Sex", "Capital-Gain", "Capital-Loss",
  "Hours-per-week", "Country", "Over-50K"]

# Read the dataset from the provided CSV and print out information about it.
df = pd.read_csv(csv_path, names=csv_columns, skipinitialspace=True)

# df
```

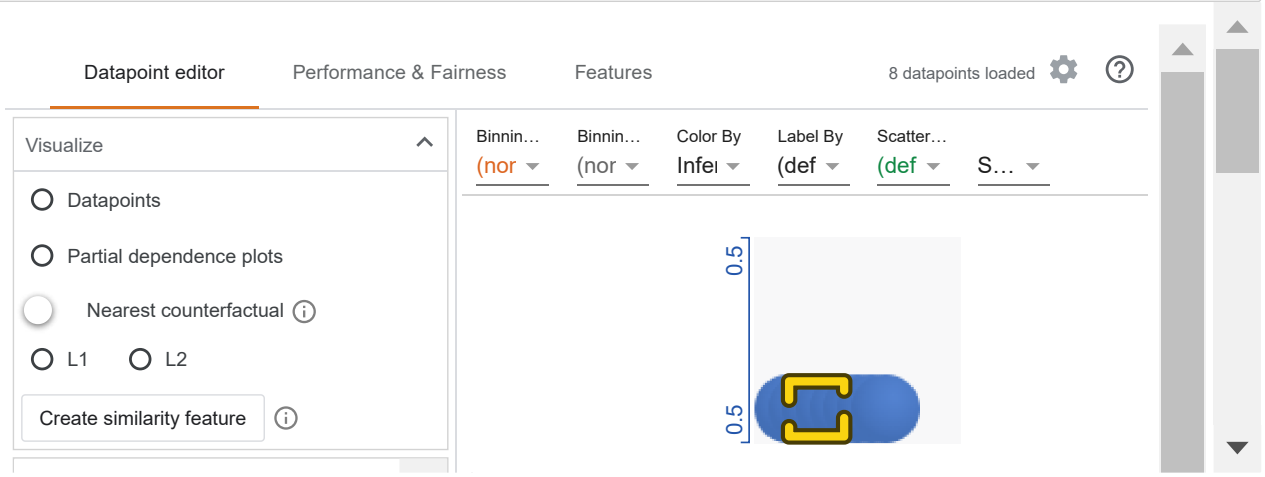
In [3]:
```
# Specify input columns and column to predict

import numpy as np

# Set the column in the dataset you wish for the model to predict
label_column = 'Over-50K'

# Make the label column numeric (0 and 1), for use in our model.
# In this case, examples with a target value of '>50K' are considered to be in
# the '1' (positive) class and all other examples are considered to be in the
# '0' (negative) class.
make_label_column_numeric(df, label_column, lambda val: val == '>50K')

# Set list of all columns from the dataset we will use for model input.
input_features = [
  'Age', 'Workclass', 'Education', 'Marital-Status', 'Occupation',
  'Relationship', 'Race', 'Sex', 'Capital-Gain', 'Capital-Loss',
  'Hours-per-week', 'Country']


# Create a list containing all input features and the label column
features_and_labels = input_features + [label_column]
```

In [4]:
```
# Convert dataset to tf.Example protos

examples = df_to_examples(df)
```

In [5]:
```
# Create and train the classifier

num_steps = 50 #5000

# Create a feature spec for the classifier
feature_spec = create_feature_spec(df, features_and_labels)

# Define and train the classifier
try:
    train_inpf = functools.partial(tfexamples_input_fn, examples, feature_spec, label_column)
    classifier = tf.estimator.LinearClassifier(
        feature_columns=create_feature_columns(input_features, feature_spec))
    classifier.train(train_inpf, steps=num_steps)
except:
    print("An exception occurred")
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\cjtan\AppData\Local\Temp\tmpzlz02xlq
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\cjtan\\AppData\\Local\\Temp\\tmpzlz02xlq', '_tf_random_see
d': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_conf
ig': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribu
te': None, '_device_fn': None, '_protocol': None, '_eval_distribute': None, '_experimental_distribute': None, '_expe
rimental_max_worker_delay_secs': None, '_session_creation_timeout_secs': 7200, '_checkpoint_save_graph_def': True,
'_service': None, '_cluster_spec': ClusterSpec({}), '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster':
0, '_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
WARNING:tensorflow:From C:\Users\cjtan\AppData\Local\Programs\Python\Python310\lib\site-packages\tensorflow\python\t
raining\training_util.py:396: Variable.initialized_value (from tensorflow.python.ops.variables) is deprecated and wi
ll be removed in a future version.
Instructions for updating:
Use Variable.read_value. Variables in 2.X are initialized automatically both in eager and graph (inside tf.defun) co
ntexts.
WARNING:tensorflow:From C:\Users\cjtan\AppData\Local\Temp\ipykernel_3216\2391053652.py:45: calling DatasetV2.from_ge
nerator (from tensorflow.python.data.ops.dataset_ops) with output_types is deprecated and will be removed in a futur
e version.
Instructions for updating:
Use output_signature instead
WARNING:tensorflow:From C:\Users\cjtan\AppData\Local\Temp\ipykernel_3216\2391053652.py:45: calling DatasetV2.from_ge
nerator (from tensorflow.python.data.ops.dataset_ops) with output_shapes is deprecated and will be removed in a futu
re version.
Instructions for updating:
Use output_signature instead
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From C:\Users\cjtan\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\optimizers\op
timizer_v2\ftrl.py:153: calling Constant.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and
will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 0...
INFO:tensorflow:Saving checkpoints for 0 into C:\Users\cjtan\AppData\Local\Temp\tmpzlz02xlq\model.ckpt.
INFO:tensorflow:C:\Users\cjtan\AppData\Local\Temp\tmpzlz02xlq\model.ckpt-0.data-00000-of-00001
INFO:tensorflow:0
INFO:tensorflow:C:\Users\cjtan\AppData\Local\Temp\tmpzlz02xlq\model.ckpt-0.index
INFO:tensorflow:0
INFO:tensorflow:C:\Users\cjtan\AppData\Local\Temp\tmpzlz02xlq\model.ckpt-0.meta
INFO:tensorflow:400
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 0...
An exception occurred
```

In [7]:
```
# Invoke What-If Tool for test data and the trained model

num_datapoints = 2000
tool_height_in_px = 1000

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

# Load up the test dataset
# Source: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test
test_csv_path = './adultdata/adult.test.txt'
#test_csv_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test'

test_df = pd.read_csv(test_csv_path, names=csv_columns, skipinitialspace=True,
  skiprows=1)
make_label_column_numeric(test_df, label_column, lambda val: val == '>50K.')
test_examples = df_to_examples(test_df[0:num_datapoints])

# Setup the tool with the test examples and the trained classifier
config_builder = WitConfigBuilder(test_examples).set_estimator_and_feature_spec(
    classifier, feature_spec).set_label_vocab(['Under 50K', 'Over 50K'])
WitWidget(config_builder, height=tool_height_in_px)
```



In [ ]: