Download and install libraries

```
!pip install dalex shap lime pyDOE2 kaleido
import pandas as pd
import sklearn
import shap
import lime
from lime import lime_tabular
import kaleido

import xgboost as xgb
from sklearn.model_selection import train_test_split
import lightgbm as lgb
```

```
Requirement already satisfied: dalex in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: lime in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pyDOE2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: kaleido in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pandas>=1.5.3 in /usr/local/lib/python3.11/dist
Requirement already satisfied: numpy>=1.23.3 in /usr/local/lib/python3.11/dist
Requirement already satisfied: scipy>=1.6.3 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: plotly<6.0.0,>=5.1.0 in /usr/local/lib/python3.
Requirement already satisfied: tqdm>=4.61.2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dis
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.11
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/py
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dis
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.1
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.11/d
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-pack
```

Tutorial

Patch to match style consistency

```
import numpy as np
import pandas as pd
import plotly.express as px
```

```
import warnings
from copy import deepcopy

from dalex.predict_explanations._ceteris_paribus import checks, plot, utils
from dalex import _theme, _global_checks, _global_utils
from dalex._explanation import Explanation

import dalex.predict_explanations._ceteris_paribus.object

def patch_plot(self,
          objects=None,
          variable_type="numerical",
          variables=None,
          size=2,
          alpha=1,
          color="_label_",
          facet_ncol=2,
          show_observations=True,
          title="Ceteris Paribus Profiles",
          y_title='',
          horizontal_spacing=None,
          vertical_spacing=None,
          show=True):
    """Plot the Ceteris Paribus explanation

    Parameters
    ----------
    objects : CeterisParibus object or array_like of CeterisParibus objects
        Additional objects to plot in subplots (default is `None`).
    variable_type : {'numerical', 'categorical'}
        Plot the profiles for numerical or categorical variables
        (default is `'numerical'`).
    variables : str or array_like of str, optional
        Variables for which the profiles will be calculated
        (default is `None`, which means all of the variables).
    size : float, optional
        Width of lines in px (default is `2`).
    alpha : float <0, 1>, optional
        Opacity of lines (default is `1`).
    color : str, optional
        Variable name used for grouping
        (default is `'_label_'`, which groups by models).
    facet_ncol : int, optional
        Number of columns on the plot grid (default is `2`).
    show_observations : bool, optional
```

```
        Show observation points (default is `True`).
    title : str, optional
        Title of the plot (default is `"Ceteris Paribus Profiles"`).
    y_title : str, optional
        Title of the y/x axis (default is `"prediction"`).
    horizontal_spacing : float <0, 1>, optional
        Ratio of horizontal space between the plots
        (default depends on `variable_type`).
    vertical_spacing : float <0, 1>, optional
        Ratio of vertical space between the plots (default is `0.3/number of rows
    show : bool, optional
        `True` shows the plot; `False` returns the plotly Figure object that can
        be edited or saved using the `write_image()` method (default is `True`).

    Returns
    -----------
    None or plotly.graph_objects.Figure
        Return figure that can be edited or saved. See `show` parameter.
    """

    if variable_type not in ("numerical", "categorical"):
        raise TypeError("variable_type should be 'numerical' or 'categorical'")
    if isinstance(variables, str):
        variables = (variables,)

    # are there any other objects to plot?
    if objects is None:
        _result_df = self.result.assign(_original_yhat_=lambda x: self.new_observa
        _include = self.variable_splits_with_obs
    elif isinstance(objects, self.__class__):  # allow for objects to be a single
        _result_df = pd.concat([
            self.result.assign(_original_yhat_=lambda x: self.new_observation.loc
            objects.result.assign(_original_yhat_=lambda x: objects.new_observati
        _include = np.all([self.variable_splits_with_obs, objects.variable_splits_
    elif isinstance(objects, (list, tuple)):  # objects as tuple or array
        _result_df = self.result.assign(_original_yhat_=lambda x: self.new_observa
        _include = [self.variable_splits_with_obs]
        for ob in objects:
            _global_checks.global_check_object_class(ob, self.__class__)
            _result_df = pd.concat([
                _result_df, ob.result.assign(_original_yhat_=lambda x: ob.new_obs
            _include += [ob.variable_splits_with_obs]
        _include = np.all(_include)
    else:
        _global_checks.global_raise_objects_class(objects, self.__class__)
```

```
    if _include is False and show_observations:
            warnings.warn("'show_observations' parameter changed to False,"
                          "because the 'variable_splits_with_obs' attribute is Fa
                          "See `variable_splits_with_obs` parameter in `predict_p
            show_observations = False

    # variables to use
    all_variables = list(_result_df['_vname_'].dropna().unique())

    if variables is not None:
        all_variables = _global_utils.intersect_unsorted(variables, all_variables
        if len(all_variables) == 0:
            raise TypeError("variables do not overlap with " + ''.join(variables)

    # names of numeric variables
    numeric_variables = _result_df[all_variables].select_dtypes(include=np.number

    if variable_type == "numerical":
        variable_names = numeric_variables

        if len(variable_names) == 0:
            # change to categorical
            variable_type = "categorical"
            # send message
            warnings.warn("'variable_type' parameter changed to 'categorical' due
            # take all
            variable_names = all_variables
        elif variables is not None and len(variable_names) != len(variables):
            raise TypeError("There are no numerical variables")
    else:
        variable_names = np.setdiff1d(all_variables, numeric_variables).tolist()

        # there are variables selected
        if variables is not None:
            # take all
            variable_names = all_variables
        elif len(variable_names) == 0:
            # there were no variables selected and there are no categorical varia
            raise TypeError("There are no non-numerical variables.")

    # prepare profiles data
    _result_df = _result_df.loc[_result_df['_vname_'].isin(variable_names), ].res

    #  calculate y axis range to allow for fixedrange True
```

```
    dl = _result_df['_yhat_'].to_numpy()
    min_max_margin = np.ptp(dl) * 0.10
    min_max = [dl.min() - min_max_margin, dl.max() + min_max_margin]

    # create _x_
    if len(variable_names) == 1:
        _result_df.loc[:, '_x_'] = deepcopy(_result_df.loc[:, variable_names[0]])
    else:
        for variable in variable_names:
            where_variable = _result_df['_vname_'] == variable
            _result_df.loc[where_variable, '_x_'] = deepcopy(_result_df.loc[where_

    # change x column to proper character values
    if variable_type == 'categorical':
        _result_df.loc[:, '_x_'] = _result_df.apply(lambda row: str(row[row['_vna

    n = len(variable_names)
    facet_nrow = int(np.ceil(n / facet_ncol))
    if vertical_spacing is None:
        vertical_spacing = 0.3 / facet_nrow #if variable_type == 'numerical' else
    if horizontal_spacing is None:
        horizontal_spacing = 0.05 #if variable_type == 'numerical' else 0.1

    plot_height = 78 + 71 + facet_nrow * (280 + 60)

    _result_df = _result_df.assign(_text_=_result_df.apply(lambda obs: plot.toolt

    if variable_type == "numerical":
        m = len(_result_df[color].dropna().unique())
        _result_df[color] = _result_df[color].astype(object)  # prevent error whe

        fig = px.line(_result_df,
                      x="_x_", y="_yhat_", color=color, facet_col="_vname_", line
                      category_orders={"_vname_": list(variable_names)},
                      labels={'_yhat_': 'prediction', '_label_': 'label', '_ids_'
                      # hover_data={'_text_': True, '_yhat_': ':.3f', '_vname_':
                      custom_data=['_text_'],
                      facet_col_wrap=facet_ncol,
                      facet_row_spacing=vertical_spacing,
                      facet_col_spacing=horizontal_spacing,
                      template="none",
                      color_discrete_sequence=_theme.get_default_colors(m, 'line'
                .update_traces(dict(line_width=size, opacity=alpha,
                                    hovertemplate="%{customdata[0]}<extra></extra
                .update_xaxes({'matches': None, 'showticklabels': True,
```

```
                                'type': 'linear', 'gridwidth': 2, 'zeroline': Fal
                                'ticks': "outside", 'tickcolor': 'white', 'tickle
                    .update_yaxes({'type': 'linear', 'gridwidth': 2, 'zeroline': Fals
                                'ticks': 'outside', 'tickcolor': 'white', 'tickle
                                'range': [0,1]})#min_max})

            if show_observations:
                _points_df = _result_df.loc[_result_df['_original_'] == _result_df['_

                fig_points = px.scatter(_points_df,
                                        x='_original_', y='_yhat_', facet_col='_vname_
                                        category_orders={"_vname_": list(variable_nam
                                        labels={'_yhat_': 'prediction', '_label_': 'l
                                        custom_data=['_text_'],
                                        facet_col_wrap=facet_ncol,
                                        facet_row_spacing=vertical_spacing,
                                        facet_col_spacing=horizontal_spacing,
                                        color_discrete_sequence=["#371ea3"]) \
                                .update_traces(dict(marker_size=5*size, opacity=alpha
                                                    hovertemplate="%{customdata[0]}<extra></

                for _, value in enumerate(fig_points.data):
                    fig.add_trace(value)

            fig = _theme.fig_update_line_plot(fig, title, y_title, plot_height, 'clos

        else:
            m = len(_result_df[color].dropna().unique())
            _result_df[color] = _result_df[color].astype(object)  # prevent error whe

            _result_df = _result_df.assign(_diff_=lambda x: x['_yhat_'] - x['_origina

            fig = px.bar(_result_df,
                         x="_x_", y="_diff_", color=color, facet_col="_vname_",
                         category_orders={"_vname_": list(variable_names)},
                         labels={'_yhat_': 'prediction', '_label_': 'label', '_ids_'
                         # hover_data={'_text_': True, '_yhat_': ':.3f', '_vname_':
                         custom_data=['_text_'],
                         base="_original_yhat_",
                         facet_col_wrap=facet_ncol,
                         facet_row_spacing=vertical_spacing,
                         facet_col_spacing=horizontal_spacing,
                         template="none",
                         color_discrete_sequence=_theme.get_default_colors(m, 'line'
                         barmode='group',
```

```
                      orientation='v') \
                .update_traces(dict(opacity=alpha),
                               hovertemplate="%{customdata[0]}<extra></extra>") 
                .update_xaxes({'matches': None, 'showticklabels': True,
                               'type': 'linear', 'gridwidth': 2, 'zeroline': Fal
                               'ticks': "outside", 'tickcolor': 'white', 'tickle
                .update_yaxes({'type': 'linear', 'gridwidth': 2, 'zeroline': Fals
                               'ticks': 'outside', 'tickcolor': 'white', 'tickle
                               'range': [0,1]})#min_max})
                #.update_xaxes({'matches': None, 'showticklabels': True,
                #               'type': 'category', 'gridwidth': 2, 'automargin'
                #               'ticks': "outside", 'tickcolor': 'white', 'tickl
                #.update_yaxes({'type': 'linear', 'gridwidth': 2, 'zeroline': Fal
                #               'ticks': 'outside', 'tickcolor': 'white', 'tickl
                #               'range': min_max})

        for _, bar in enumerate(fig.data):
            fig.add_hline(y=bar.base[0], layer='below',
                          line={'color': "#371ea3", 'width': 1.5, 'dash': 'dot'})

        if show_observations:

            _points_df = _result_df.loc[_result_df['_original_'] == _result_df['_

            fig_points = px.scatter(_points_df,
                                    x='_original_', y='_original_yhat_', facet_co
                                    category_orders={"_vname_": list(variable_nam
                                    labels={'_yhat_': 'prediction', '_label_': 'l
                                    custom_data=['_text_'],
                                    facet_col_wrap=facet_ncol,
                                    facet_row_spacing=vertical_spacing,
                                    facet_col_spacing=horizontal_spacing,
                                    color_discrete_sequence=["#371ea3"]) \
                           .update_traces(dict(marker_size=5*size, opacity=alpha
                                               hovertemplate="%{customdata[0]}<extra><

            for _, value in enumerate(fig_points.data):
                fig.add_trace(value)

        fig = _theme.fig_update_bar_plot(fig, title, y_title, plot_height, 'close

    fig.update_layout(hoverlabel=dict(bgcolor='rgba(0,0,0,0.8)'))
    if show:
        fig.show(config=_theme.get_default_config())
    else:
```

```
        return fig

dalex.predict_explanations._ceteris_paribus.object.CeterisParibus.plot = patch_pl
```

Set up tutorial examples

Start by training the "should you bring an umbrella?" model

```
preX = pd.read_csv("Umbrella.csv")
preX = preX.sample(frac=1)
X_display = preX.iloc[:,:-1]
y_display = preX.iloc[:,-1]

PRECIPITATION = {
    "none": 0,
    "drizzle": 1,
    "rain": 2,
    "snow": 3,
    "sleet": 4,
    "hail": 5
}

y = y_display
X = X_display
X = X.replace({"Precipitation":PRECIPITATION})

X_train = X.iloc[:300]
y_train = y.iloc[:300]

X_test = X.iloc[300:]
y_test = y.iloc[300:]

d_train = lgb.Dataset(X_train, label=y_train)
d_test = lgb.Dataset(X_test, label=y_test)

params = {
    "max_bin": 512,
    "learning_rate": 0.05,
    "boosting_type": "gbdt",
    "objective": "binary",
    "metric": "binary_logloss",
    "num_leaves": 10,
    "verbose": -1,
```

```
    "min_data": 100,
    "boost_from_average": True,
    "keep_training_booster": True
}

#model = lgb.train(params, d_train, 10000, valid_sets=[d_test]) #early_stopping_r
model = lgb.LGBMClassifier(max_bin= 512,
    learning_rate= 0.05,
    boosting_type= "gbdt",
    objective= "binary",
    metric= "binary_logloss",
    num_leaves= 10,
    verbose= -1,
    min_data= 100,
    boost_from_average= True)
model.fit(X_train, y_train)
```

```
<ipython-input-45-0ae06d0dfa92>:17: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a futur
```

LGBMClassifier

```
LGBMClassifier(boost_from_average=True, learning_rate=0.05, max_bin=512,
               metric='binary_logloss', min_data=100, num_leaves=10,
               objective='binary', verbose=-1)
```

Find the location of one of the two tutorial examples

```
print(X.loc[(X['Precipitation'] == 5) & (X['Temperature'] == 23) & (X['Wind(mph)'
print(X.loc[(X['Precipitation'] == 0) & (X['Temperature'] == 70) & (X['Wind(mph)'
theloc = X.index.get_loc(330)
```

```
     Precipitation  Temperature  Wind(mph)
330              5           23         10
    Precipitation  Temperature  Wind(mph)
96              0           70         30
```

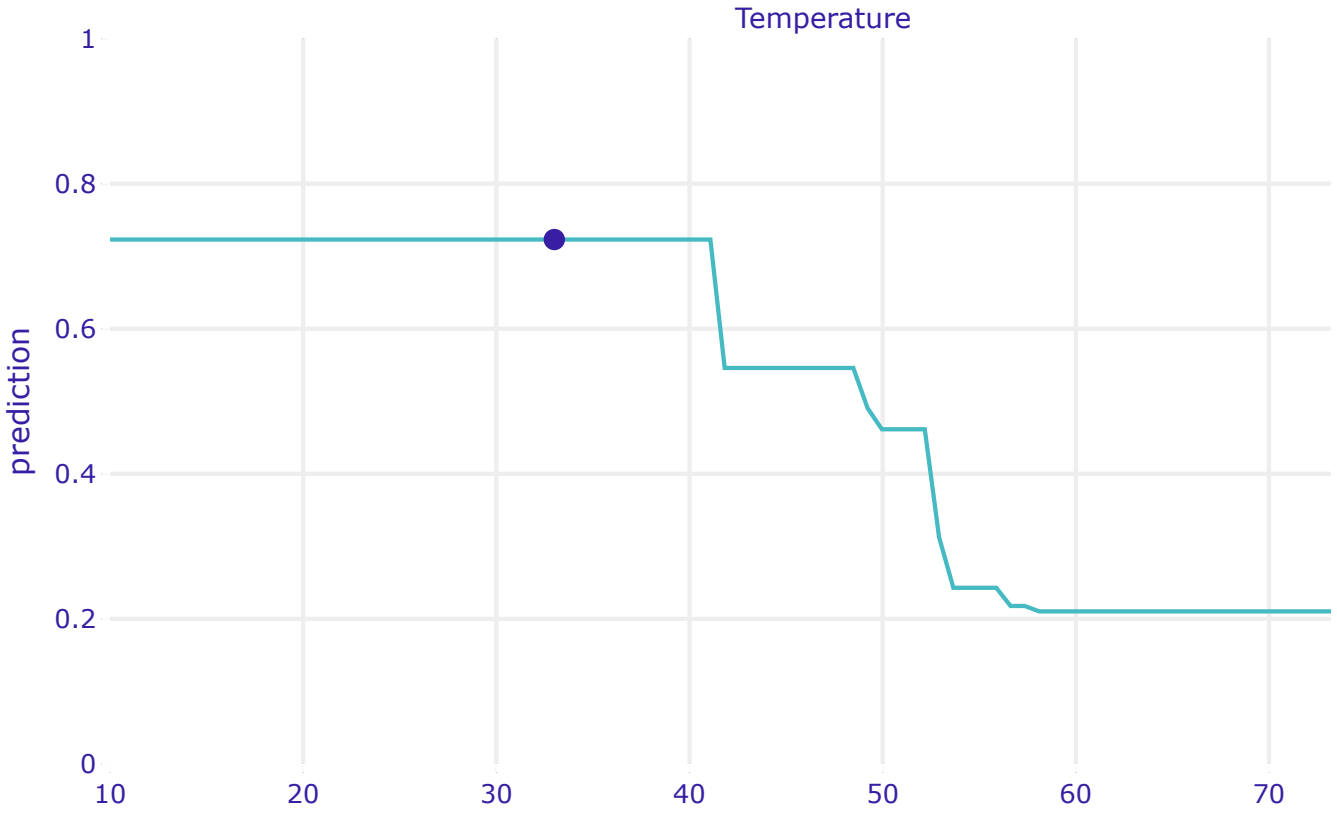
Generate a tutorial explanation

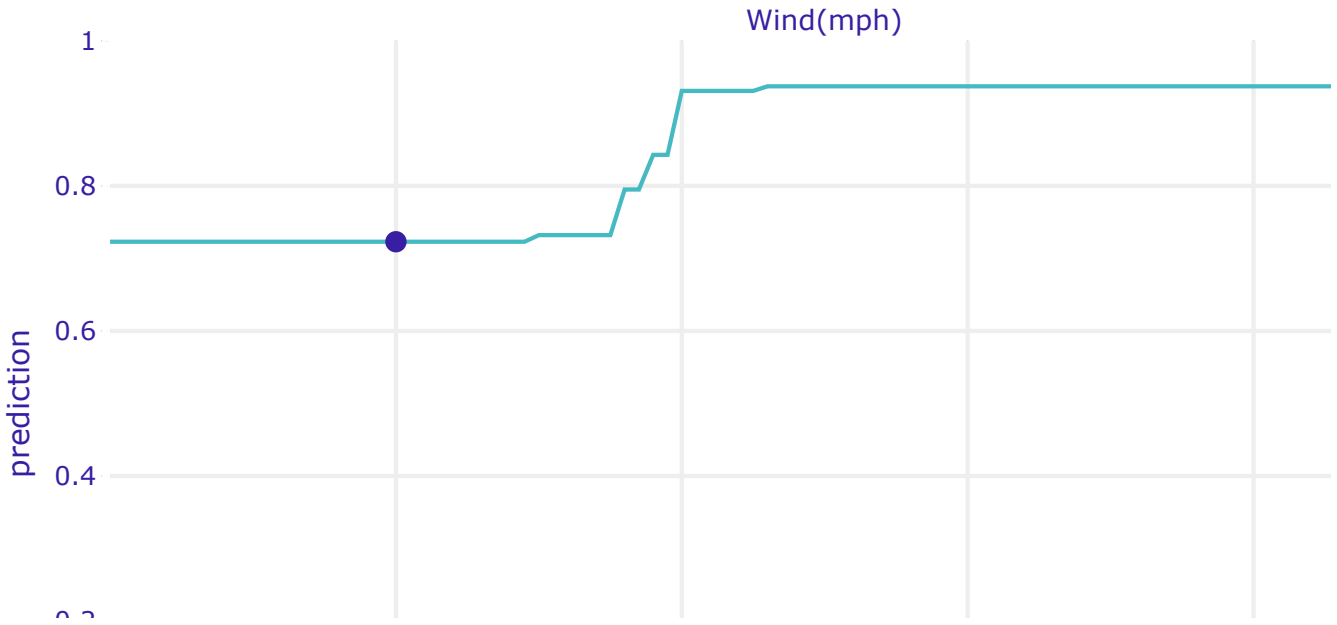
```
import dalex as dx
import kaleido
```

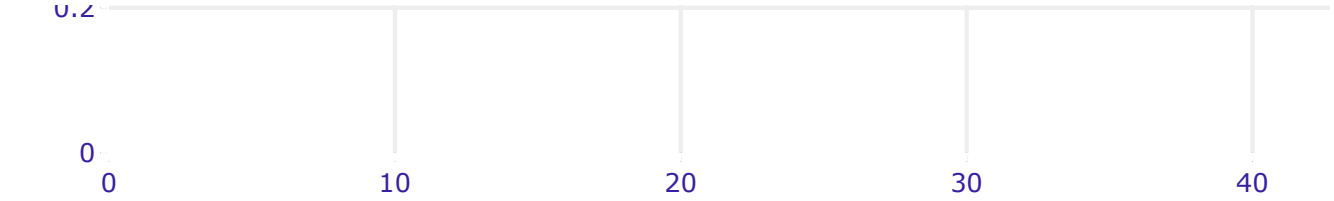
```
exp_dalex = dx.Explainer(model, X, y, label='')
#theloc = 21
theloc = 16
print(X.loc[theloc])
#0l21
cp = exp_dalex.predict_profile(X.iloc[theloc], type='ceteris_paribus',variables=[
cp3 = exp_dalex.predict_profile(X.iloc[theloc], type='ceteris_paribus',variables=
cp2 = exp_dalex.predict_profile(X.iloc[theloc], type='ceteris_paribus',variables=
                              variable_splits = {'Precipitation': [0,1,2,3,4,5]}

plotobj = cp.plot(show=False)
plotobj['layout']['title'] = None
plotobj['layout']['yaxis1']['title'] = "prediction"
plotobj['layout']['xaxis1']['title'] = ""
print("hi")
print(plotobj.__dir__())

plotobj3 = cp3.plot(show=False)
plotobj3['layout']['title'] = None
plotobj3['layout']['yaxis1']['title'] = "prediction"
plotobj3['layout']['xaxis1']['title'] = ""

plotobj2 = cp2.plot(variables=['Precipitation'],variable_type='categorical',show=

plotobj2['layout']['title'] = None
plotobj2['layout']['yaxis1']['range'] = [0,1.0]
plotobj2['layout']['xaxis1']['tickmode'] = 'array'
plotobj2['layout']['yaxis1']['title'] = "prediction"
plotobj2['layout']['xaxis1']['tickvals'] = [0,1,2,3,4,5]
plotobj2['layout']['xaxis1']['ticktext'] = ["none", "drizzle", "rain","snow","sle
plotobj2['layout']['xaxis1']['title'] = ""
#plotobj2.add_scatter(x=[0.483],
#                y=[0],
#                marker=dict(
#                    color='blue',
#                    size=10
#                ), showlegend=False)

plotobj2.update_layout(
    autosize=False,
    width=780,
)

plotobj.update_layout(
```

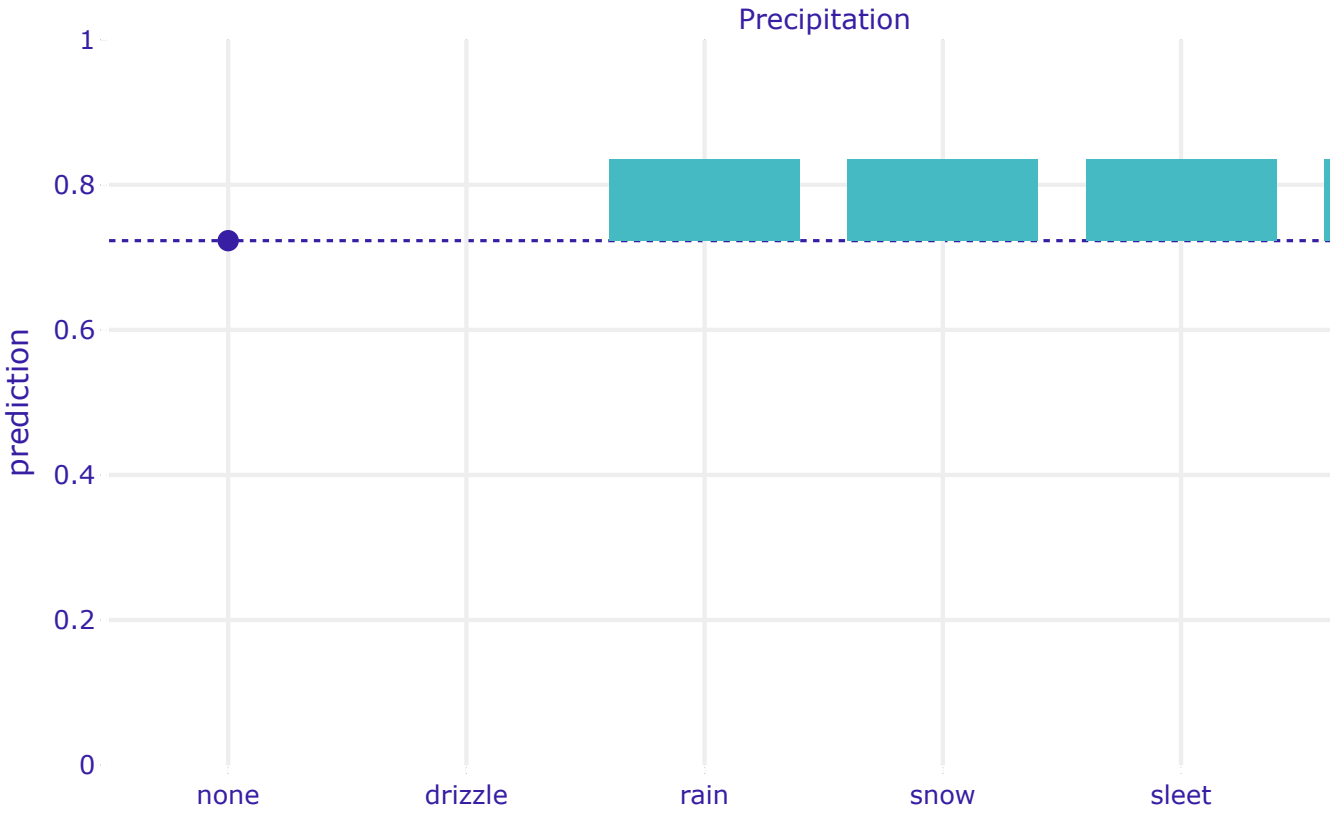
```
    autosize=False,
    width=780,
)

plotobj3.update_layout(
    autosize=False,
    width=780,
)

plotobj.show()
#plotobj.write_image(file='cp-fig.pdf', format='pdf')
plotobj3.show()
plotobj2.show()
```

```
Calculating ceteris paribus:   0%|          | 0/1 [00:00<?, ?it/s]/usr/local/l

Setting an item of incompatible dtype is deprecated and will raise in a future
 18.88 19.62 20.36 21.1  21.84 22.58 23.32 24.06 24.8  25.54 26.28 27.02
 27.76 28.5  29.24 29.98 30.72 31.46 32.2  32.94 33.   33.68 34.42 35.16
 35.9  36.64 37.38 38.12 38.86 39.6  40.34 41.08 41.82 42.56 43.3  44.04
 44.78 45.52 46.26 47.   47.74 48.48 49.22 49.96 50.7  51.44 52.18 52.92
 53.66 54.4  55.14 55.88 56.62 57.36 58.1  58.84 59.58 60.32 61.06 61.8
 62.54 63.28 64.02 64.76 65.5  66.24 66.98 67.72 68.46 69.2  69.94 70.68
 71.42 72.16 72.9  73.64 74.38 75.12 75.86 76.6  77.34 78.08 78.82 79.56
 80.3  81.04 81.78 82.52 83.26 84.  ]' has dtype incompatible with int64, plea

Calculating ceteris paribus: 100%|██████████| 1/1 [00:00<00:00, 103.70it/s]
Calculating ceteris paribus:   0%|          | 0/1 [00:00<?, ?it/s]/usr/local/l

Setting an item of incompatible dtype is deprecated and will raise in a future
  7.   7.5  8.   8.5  9.   9.5 10.  10.5 11.  11.5 12.  12.5 13.  13.5
 14.  14.5 15.  15.5 16.  16.5 17.  17.5 18.  18.5 19.  19.5 20.  20.5
 21.  21.5 22.  22.5 23.  23.5 24.  24.5 25.  25.5 26.  26.5 27.  27.5
 28.  28.5 29.  29.5 30.  30.5 31.  31.5 32.  32.5 33.  33.5 34.  34.5
 35.  35.5 36.  36.5 37.  37.5 38.  38.5 39.  39.5 40.  40.5 41.  41.5
 42.  42.5 43.  43.5 44.  44.5 45.  45.5 46.  46.5 47.  47.5 48.  48.5
 49.  49.5 50. ]' has dtype incompatible with int64, please explicitly cast to

Calculating ceteris paribus: 100%|██████████| 1/1 [00:00<00:00, 80.54it/s]
/usr/local/lib/python3.11/dist-packages/dalex/predict_explanations/_ceteris_pa

Parameter `variable_splits` overrides `variables`. Variables taken from `varia

Calculating ceteris paribus: 100%|██████████| 1/1 [00:00<00:00, 192.10it/s]
hi
['_validate', '_grid_str', '_grid_ref', '_data_validator', '_data_objs', '_dat
<ipython-input-44-3621b823fe02>:153: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise in a future
```


Temperature
prediction
1
0.8
0.6
0.4
0.2
0
10
20
30
40
50
60
70


Wind(mph)
prediction
1
0.8
0.6
0.4


0.2
0
0
10
20
30
40


Precipitation
1
0.8
0.6
0.4
0.2
0
prediction
none
drizzle
rain
snow
sleet

Loan Instances

Edit and prepare the dataset

```
# load dataset
X,y = shap.datasets.adult()
X_display,y_display = shap.datasets.adult(display=True)

EDUCATION_NUM = {
    16.0: "Doctorate",
    15.0: "Prof. School",
    14.0: "Masters",
    13.0: "Bachelors",
    12.0: "Some College",
    11.0: "Associate", #Assoc-acdm
    10.0: "Vocational", #Assoc-voc
    9.0: "HS grad",
    8.0: "12th",
    7.0: "11th",
    6.0: "10th",
    5.0: "9th",
    4.0: "7th-8th",
    3.0: "5th-6th",
    2.0: "1st-4th",
    1.0: "Preschool"
}

OCCUPATION_NUM = {
    "Tech-support": "Tech Support",
    "Craft-repair": "Craft/Repair",
    "Other-service": "Other Service",
    "Sales": "Sales",
    "Exec-managerial": "Exec. Managerial",
    "Prof-specialty": "Prof. Specialty",
```

```
    "Handlers-cleaners": "Handler/Cleaner",
    "Machine-op-inspct": "Machine Op. Inspector",
    "Adm-clerical": "Admin. Clerical",
    "Farming-fishing": "Farming/Fishing",
    "Transport-moving": "Transport/Moving",
    "Priv-house-serv": "Private House Service",
    "Protective-serv": "Protective Service",
    "Armed-Forces": "Armed Forces"

}
X_display = X_display.replace({"Education-Num":EDUCATION_NUM})
X_display = X_display.replace({"Occupation":OCCUPATION_NUM})
X = X.rename(columns={"Education-Num": "Education"})
X_display = X_display.rename(columns={"Education-Num": "Education"})#, "Hours per

X = X.drop(['Capital Loss', 'Capital Gain', 'Race', 'Relationship', 'Country', 'W
X_display = X_display.drop(['Capital Loss', 'Capital Gain', 'Race', 'Relationship

# create a train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
d_train = lgb.Dataset(X_train, label=y_train)
d_test = lgb.Dataset(X_test, label=y_test)
```

Train the model

```
params = {
    "max_bin": 512,
    "learning_rate": 0.05,
    "boosting_type": "gbdt",
    "objective": "binary",
    "metric": "binary_logloss",
    "num_leaves": 10,
    "verbose": -1,
    "min_data": 100,
    'objective':'multi:softprob',
    "boost_from_average": True
}

params_xgb={
    'base_score':0.5,
    'learning_rate':0.05,
    'max_depth':5,
    'min_child_weight':100,
    'n_estimators':200,
    'num_class': 2,
    'nthread':-1,
    'objective':'multi:softprob',
    'seed':2018,
    'eval_metric':'auc'
}

model = lgb.LGBMClassifier(max_bin= 512,
    learning_rate= 0.05,
    boosting_type= "gbdt",
    objective= "binary",
    metric= "binary_logloss",
    num_leaves= 10,
    verbose= -1,
    min_data= 100,
    boost_from_average= True)
model.fit(X_train, y_train)
```

LGBMClassifier

```
LGBMClassifier(boost_from_average=True, learning_rate=0.05, max_bin=512,
               metric='binary_logloss', min_data=100, num_leaves=10,
               objective='binary', verbose=-1)
```

Our 7 loan application instances

```
#val = 610 # Woman Side-by-side
#val = 11116 # Man Side-by-side
#val = 32353 # Man 3
#val = 217 # Man 2
#val = 15040 # Man 1
#val = 32429 # Woman 3
val = 32556 # Woman 2
#val = 91#91 # Woman 1

theloc = val
```

Generate Ceteris-Paribus Explanation

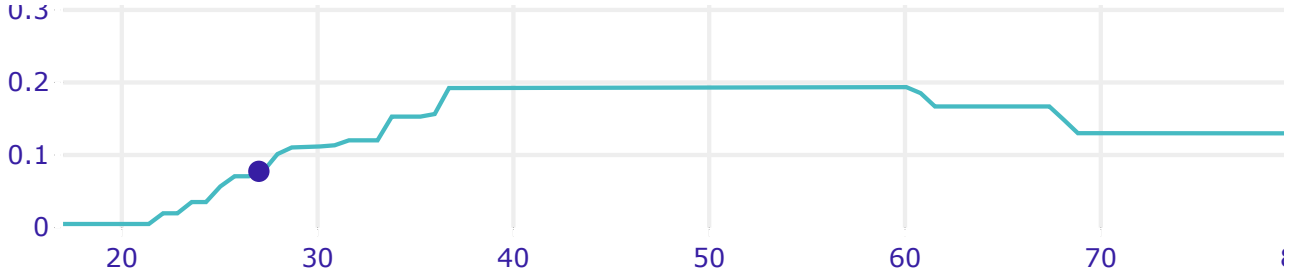
```
import dalex as dx

categorical_names={1:["None","Preschool", "1st-4th", "5th-6th", "7th-8th", "9th", "
                   2: ["None", "Admin. Clerical", "Armed Forces", "Craft Repair", "
                   3:["Female","Male"]},

exp_dalex = dx.Explainer(model, X, np.append(y_train,y_test), label="")

cp = exp_dalex.predict_profile(X.iloc[val], type='ceteris_paribus',variables=['Educ
                               variable_splits = {'Education': [int(x) for x in [1.

cp4 = exp_dalex.predict_profile(X.iloc[val], type='ceteris_paribus',variables=['Occ
                               variable_splits = {'Occupation': [1.0,2.0,3.0,4.0,5.

cp5 = exp_dalex.predict_profile(X.iloc[val], type='ceteris_paribus',variables=['Sex
                               variable_splits = {'Sex':[0,1]}, grid_points=20)

cp2 = exp_dalex.predict_profile(X.iloc[val], type='ceteris_paribus',variables=['Age

cp3 = exp_dalex.predict_profile(X.iloc[val], type='ceteris_paribus',variables=['Hou

#cp.plot(variables=['Age','Hours worked per week'])
plotobj = cp2.plot(show=False)
plotobj3 = cp3.plot(show=False)
plotobj['layout']['title'] = None
plotobj['layout']['xaxis']['title'] = None
plotobj['layout']['yaxis1']['title'] = "prediction"
```

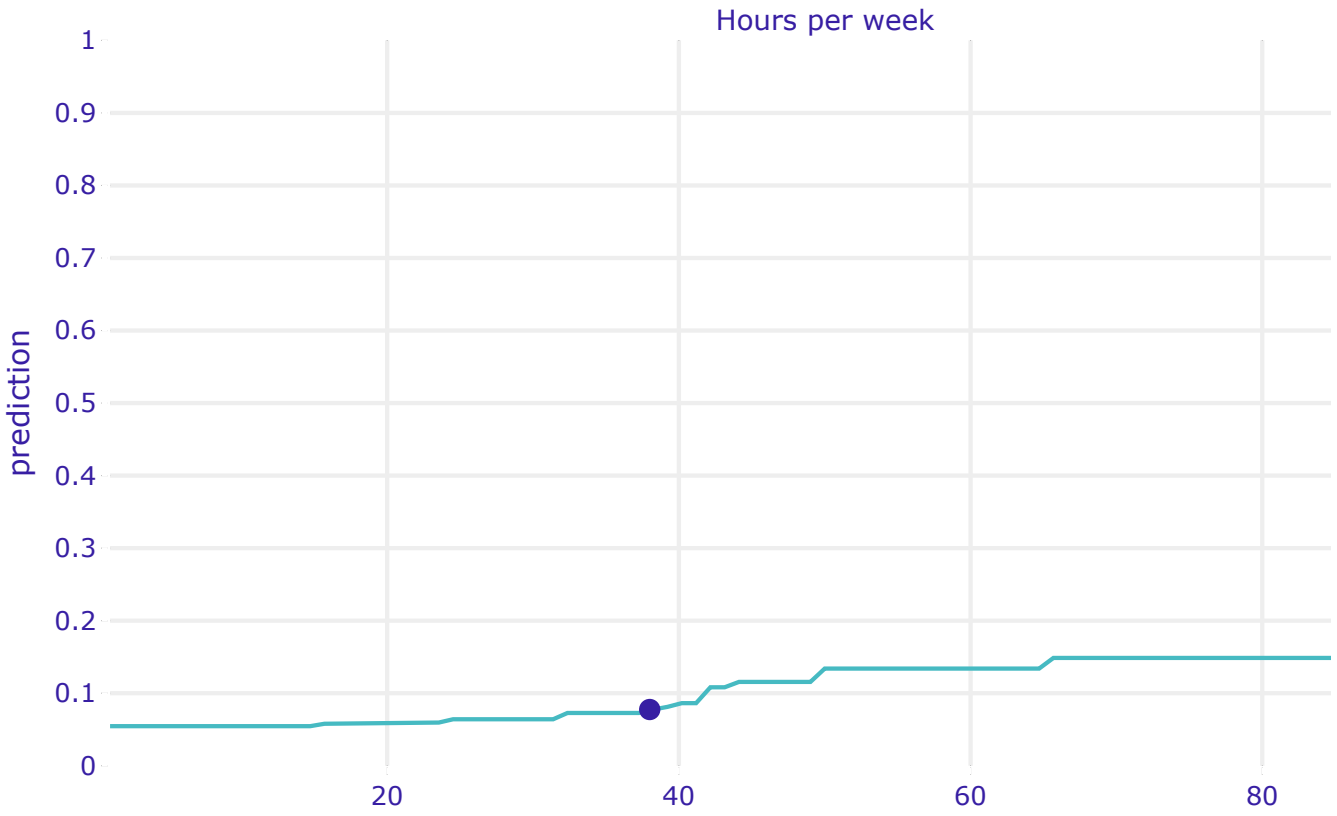
```
plotobj['layout']['yaxis1']['tickmode'] = 'array'
plotobj['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,

plotobj3['layout']['title'] = None
plotobj3['layout']['xaxis']['title'] = None
plotobj3['layout']['yaxis1']['title'] = "prediction"
plotobj3['layout']['yaxis1']['tickmode'] = 'array'
plotobj3['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9


#cp2.plot()

plotobj2 = cp.plot(variables=['Education'],variable_type='categorical',show=False)
plotobj4 = cp4.plot(variables=['Occupation'],variable_type='categorical',show=False
plotobj5 = cp5.plot(variables=['Sex'],variable_type='categorical',show=False)


plotobj['layout']['title'] = None
plotobj2['layout']['title'] = None
plotobj3['layout']['title'] = None
plotobj4['layout']['title'] = None
plotobj5['layout']['title'] = None
plotobj2['layout']['xaxis1']['title'] = None
plotobj4['layout']['xaxis1']['title'] = None
plotobj5['layout']['xaxis1']['title'] = None
plotobj['layout']['yaxis1']['title'] = "prediction"
plotobj2['layout']['yaxis1']['title'] = "prediction"
plotobj3['layout']['yaxis1']['title'] = "prediction"
plotobj4['layout']['yaxis1']['title'] = "prediction"
plotobj4['layout']['yaxis1']['title'] = "prediction"
plotobj4['layout']['xaxis1']['tickmode'] = 'array'
plotobj4['layout']['xaxis1']['tickvals'] = [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.
plotobj4['layout']['xaxis1']['ticktext'] = ["Admin. Clerical", "Armed Forces", "Cra

plotobj5['layout']['xaxis1']['tickmode'] = 'array'
plotobj5['layout']['xaxis1']['tickvals'] = [0.0,1.0]
plotobj5['layout']['xaxis1']['ticktext'] = ["Female","Male"]
plotobj['layout']['yaxis1']['tickmode'] = 'array'
plotobj['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,
plotobj2['layout']['yaxis1']['tickmode'] = 'array'
plotobj2['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
plotobj3['layout']['yaxis1']['tickmode'] = 'array'
plotobj3['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
plotobj4['layout']['yaxis1']['tickmode'] = 'array'
plotobj4['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
plotobj5['layout']['yaxis1']['tickmode'] = 'array'
```

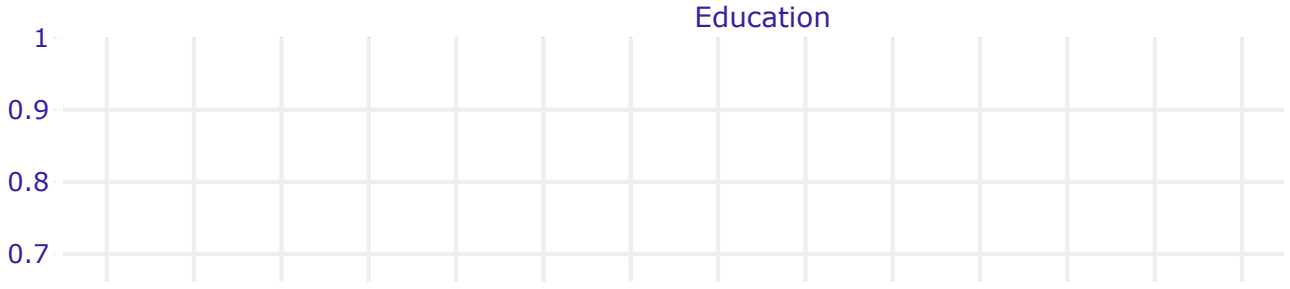
```
plotobj5['layout']['yaxis1']['tickmode'] = 'array'
plotobj5['layout']['yaxis1']['tickvals'] = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9



plotobj2['layout']['xaxis1']['tickmode'] = 'array'
plotobj2['layout']['xaxis1']['tickvals'] = [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.
plotobj2['layout']['xaxis1']['ticktext'] = ["Preschool", "1st-4th", "5th-6th", "7th

plotobj.update_layout(
    autosize=False,
    width=780,
)

plotobj2.update_layout(
    autosize=False,
    width=780,
)

plotobj3.update_layout(
    autosize=False,
    width=780,
)

plotobj4.update_layout(
    autosize=False,
    width=780,
)

plotobj5.update_layout(
    autosize=False,
    width=780,
)

plotobj.show()
plotobj3.show()
plotobj2.show()
plotobj4.show()
plotobj5.show()
```

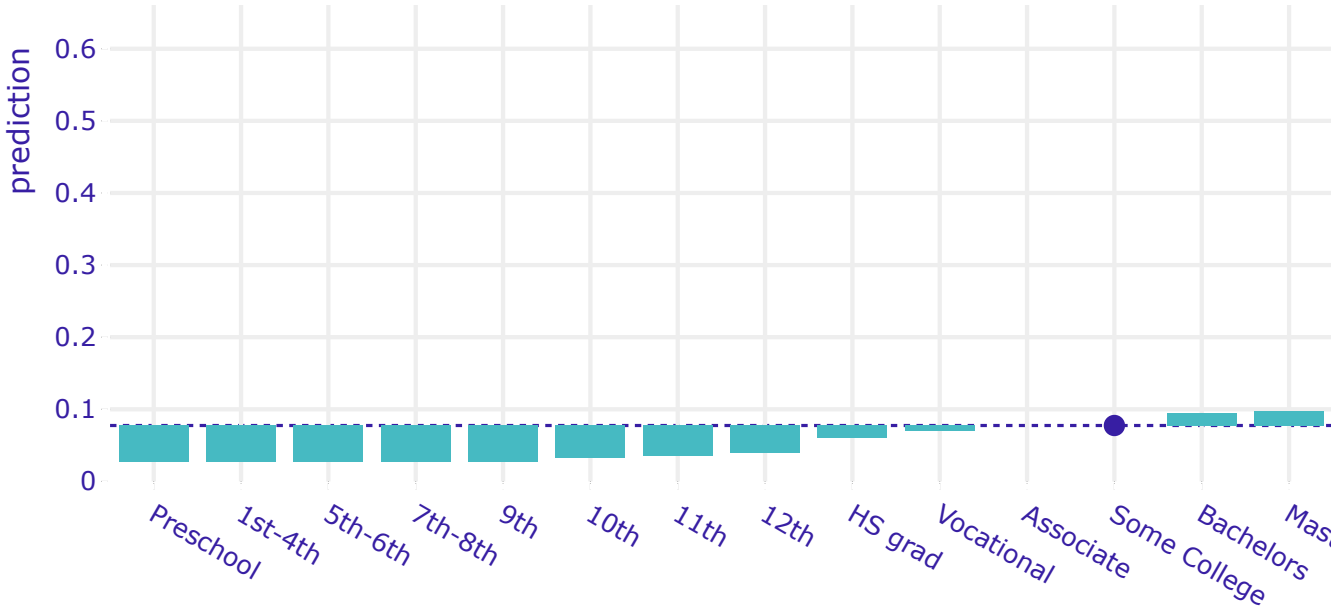
```
Calculating ceteris paribus:   0%|          | 0/1 [00:00<?, ?it/s]/usr/local/l

Setting an item of incompatible dtype is deprecated and will raise in a future
 25.76 26.49 27.   27.22 27.95 28.68 29.41 30.14 30.87 31.6  32.33 33.06
 33.79 34.52 35.25 35.98 36.71 37.44 38.17 38.9  39.63 40.36 41.09 41.82
 42.55 43.28 44.01 44.74 45.47 46.2  46.93 47.66 48.39 49.12 49.85 50.58
 51.31 52.04 52.77 53.5  54.23 54.96 55.69 56.42 57.15 57.88 58.61 59.34
 60.07 60.8  61.53 62.26 62.99 63.72 64.45 65.18 65.91 66.64 67.37 68.1
```

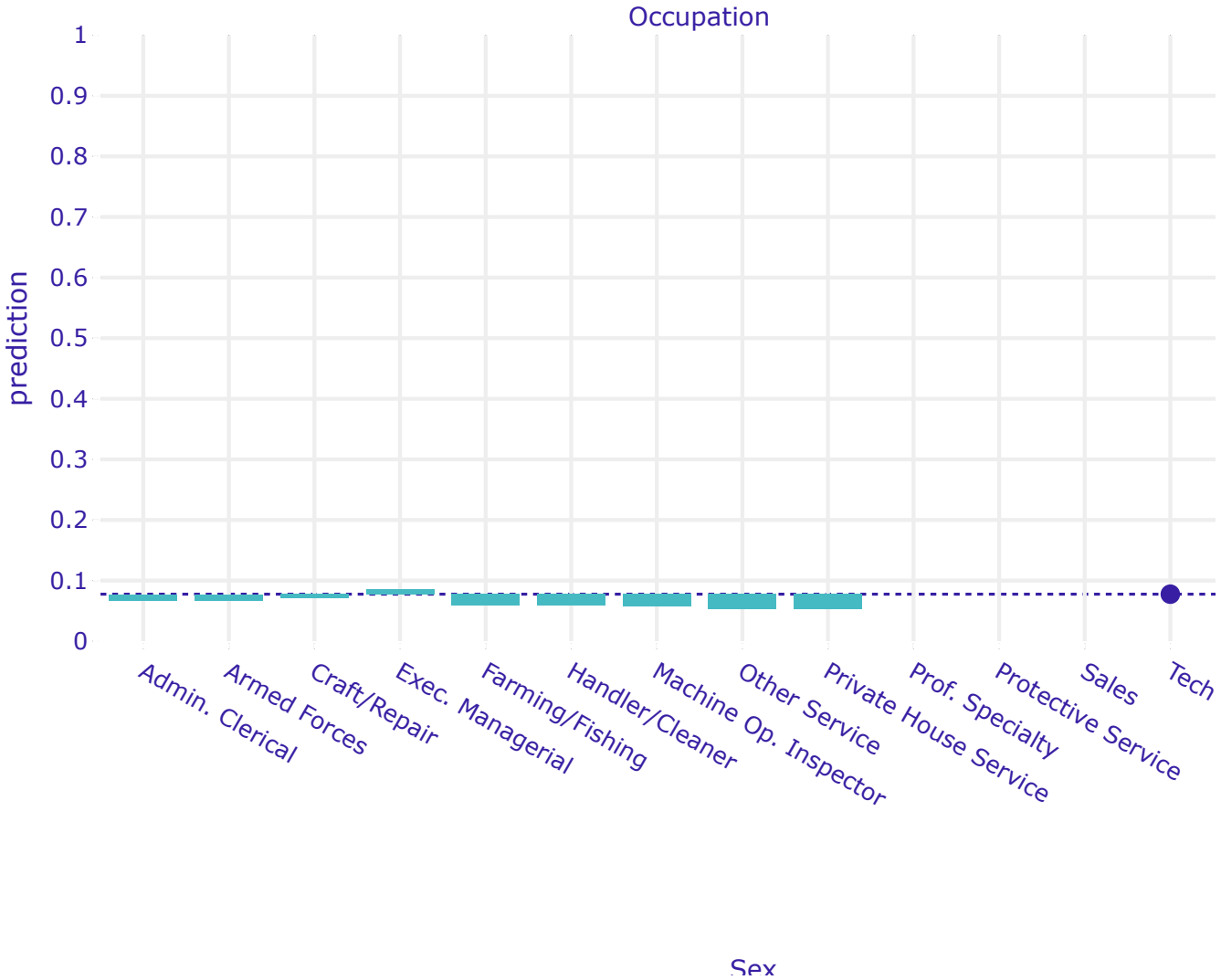
```
 68.83 69.56 70.29 71.02 71.75 72.48 73.21 73.94 74.67 75.4  76.13 76.86
 77.59 78.32 79.05 79.78 80.51 81.24 81.97 82.7  83.43 84.16 84.89 85.62
 86.35 87.08 87.81 88.54 89.27 90.  ]' has dtype incompatible with float32, pl

Calculating ceteris paribus: 100%|██████████| 1/1 [00:00<00:00, 172.98it/s]
Calculating ceteris paribus:   0%|          | 0/1 [00:00<?, ?it/s]/usr/local/l

Setting an item of incompatible dtype is deprecated and will raise in a future
 12.76 13.74 14.72 15.7  16.68 17.66 18.64 19.62 20.6  21.58 22.56 23.54
 24.52 25.5  26.48 27.46 28.44 29.42 30.4  31.38 32.36 33.34 34.32 35.3
 36.28 37.26 38.   38.24 39.22 40.2  41.18 42.16 43.14 44.12 45.1  46.08
 47.06 48.04 49.02 50.   50.98 51.96 52.94 53.92 54.9  55.88 56.86 57.84
 58.82 59.8  60.78 61.76 62.74 63.72 64.7  65.68 66.66 67.64 68.62 69.6
 70.58 71.56 72.54 73.52 74.5  75.48 76.46 77.44 78.42 79.4  80.38 81.36
 82.34 83.32 84.3  85.28 86.26 87.24 88.22 89.2  90.18 91.16 92.14 93.12
 94.1  95.08 96.06 97.04 98.02 99.  ]' has dtype incompatible with float32, pl

Calculating ceteris paribus: 100%|██████████| 1/1 [00:00<00:00, 180.19it/s]
<ipython-input-79-531c791eb7f6>:153: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise in a future
 '12.0' '13.0' '14.0' '15.0' '16.0']' has dtype incompatible with float32, ple

<ipython-input-79-531c791eb7f6>:153: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise in a future
 '12.0' '13.0' '14.0']' has dtype incompatible with float32, please explicitly

<ipython-input-79-531c791eb7f6>:153: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise in a future
```

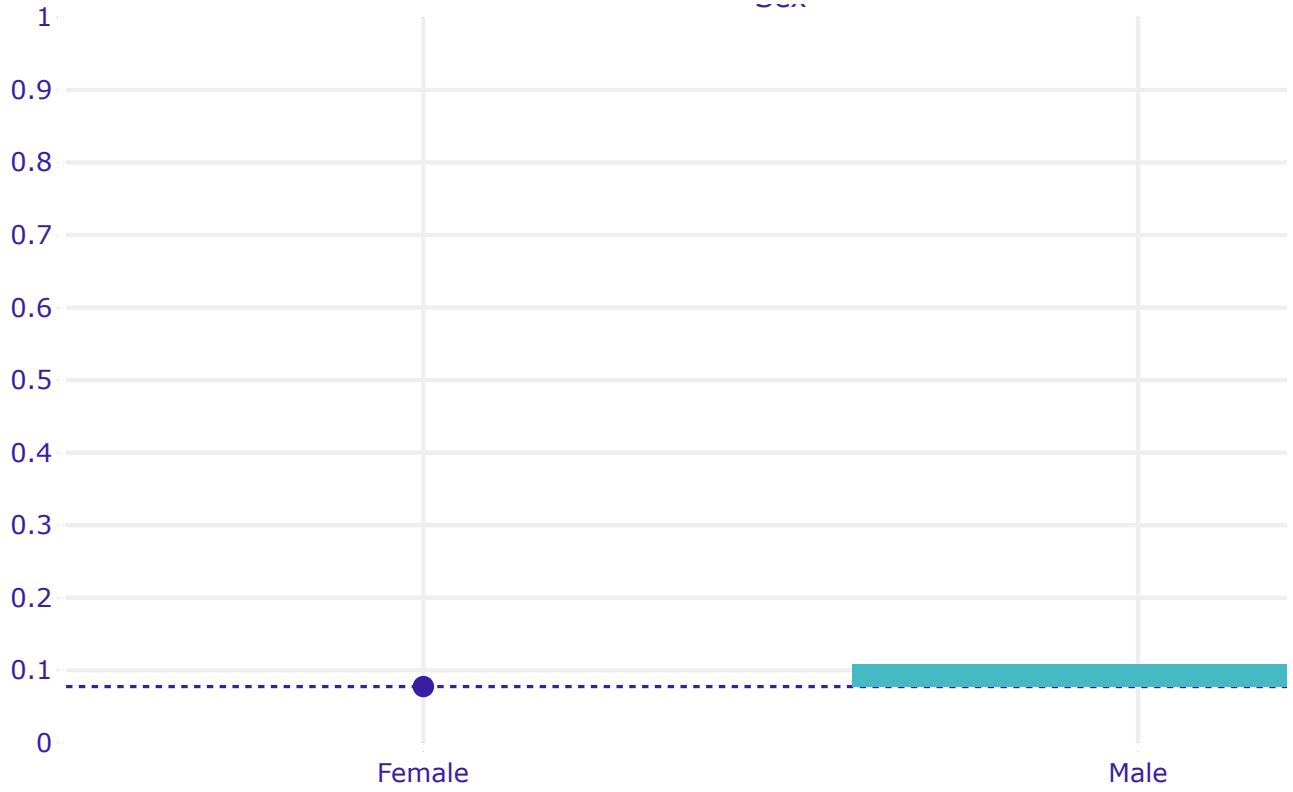

0.3
0.2
0.1
0
20
30
40
50
60
70


Hours per week
prediction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80


Education
1
0.9
0.8
0.7


prediction
0.6
0.5
0.4
0.3
0.2
0.1
0
Preschool
1st-4th
5th-6th
7th-8th
9th
10th
11th
12th
HS grad
Vocational
Associate
Some College
Bachelors


Occupation
prediction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Admin. Clerical
Armed Forces
Craft/Repair
Exec. Managerial
Farming/Fishing
Handler/Cleaner
Machine Op. Inspector
Other Service
Private House Service
Prof. Specialty
Protective Service
Sales
Tech
Sex


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Female
Male