```
!pip install shap pyDOE2
from IPython.core.display import display, HTML
import regex as re
import lightgbm
import pandas as pd
import shap
import sklearn

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


shap.initjs()
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pyDOE2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tqdm>=4.27.0 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: 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: 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: six>=1.5 in /usr/local/lib/python3.11/dist-pack
```

Patch to match style consistency

```
import numpy as np
from matplotlib import lines
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
```

```
plt.rcParams['figure.dpi'] = 300
import shap.plots._force_matplotlib

# PATCH draw_base_element
def patch_draw_base_element(base_value, ax):
    x, y = np.array([[base_value, base_value], [0.13, 0.25]])
    line = lines.Line2D(x, y, lw=2., color='#F2F2F2')
    line.set_clip_on(False)
    ax.add_line(line)

    font0 = FontProperties()
    font = font0.copy()
    font.set_weight('bold')

    text_out_val = plt.text(base_value, 0.25, f'{base_value:.2f}',
                            fontproperties=font,
                            fontsize=14,
                            horizontalalignment='center')
    text_out_val.set_bbox(dict(facecolor='white', edgecolor='white'))

    text_out_val = plt.text(base_value, 0.33, 'base value',
                            fontsize=12, alpha=0.5,
                            horizontalalignment='center')
    text_out_val.set_bbox(dict(facecolor='white', edgecolor='white'))
shap.plots._force_matplotlib.draw_base_element = patch_draw_base_element
# ENDPATCH draw_base_element
# PATCH update_axis_limits
def patch_update_axis_limits(ax, total_pos, pos_features, total_neg,
                       neg_features, base_value, out_value):

    print("patched")
    ax.set_ylim(-0.5, 0.15)
    pos_padding = np.max([np.abs(total_pos) * 1.5,
                      np.abs(total_neg) * 1.5])  #0.8 # 0.6 #1.1
    neg_padding = np.max([np.abs(total_pos) * 0.8,
                      np.abs(total_neg) * 0.8])
    padding = np.max([np.abs(total_pos) * 0.8,
                      np.abs(total_neg) * 0.8])



    print(f"pos {pos_padding}, neg {neg_padding}")

    if len(pos_features) > 0:
        min_x = min(np.min(pos_features[:, 0].astype(float)), base_value) - neg_pad
    else:
        min_x = out_value - padding
        min_x = out_value - padding
    if len(neg_features) > 0:
        max_x = max(np.max(neg_features[:, 0].astype(float)), base_value) + pos_pad
    else:
        max_x = out_value + padding



    ax.set_xlim(-0.04, 0.35)

    plt.tick_params(top=True, bottom=False, left=False, right=False, labelleft=Fals
                    labeltop=True, labelbottom=False)
    plt.locator_params(axis='x', nbins=12)

    for key, spine in zip(plt.gca().spines.keys(), plt.gca().spines.values()):
        if key != 'top':
            spine.set_visible(False)
shap.plots._force_matplotlib.update_axis_limits = patch_update_axis_limits
# ENDPATCH update_axis_limits
```

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 = lightgbm.Dataset(X_train, label=y_train)
d_test = lightgbm.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 = lightgbm.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-17-59731d8556ad>:17: FutureWarning: Downcasting behavior in `re
  X = X.replace({"Precipitation":PRECIPITATION})
```

▼ 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

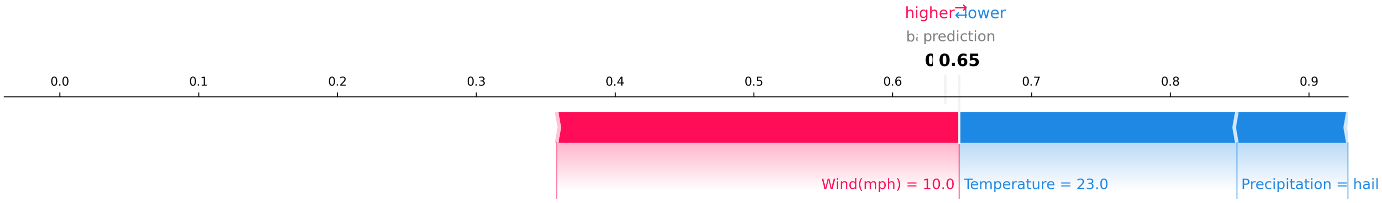
```
print(X.loc[(X['Precipitation'] == 5) & (X['Temperature'] == 23) & (X['Wind(mph)'
print(X.loc[(X['Precipitation'] == 0) & (X['Temperature'] == 70) & (X['Wind(mph)'
theloc = X.index.get_loc(330)
```

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

Generate a tutorial explanation

```
explainer = shap.Explainer(model, X, model_output="probability")
#shap_values = explainer(X)
shap.plots.force(0.638, shap_values = np.array([-0.08, -0.2, 0.29]), features = X_c
```

```
patched
pos 0.11999999999999994, neg 0.06399999999999997
```



Loan Instances

Edit and prepare 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 w

X = X.drop(['Capital Loss', 'Capital Gain', 'Race', 'Relationship', 'Country', 'Wor
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_sta
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 SHAP Explanation

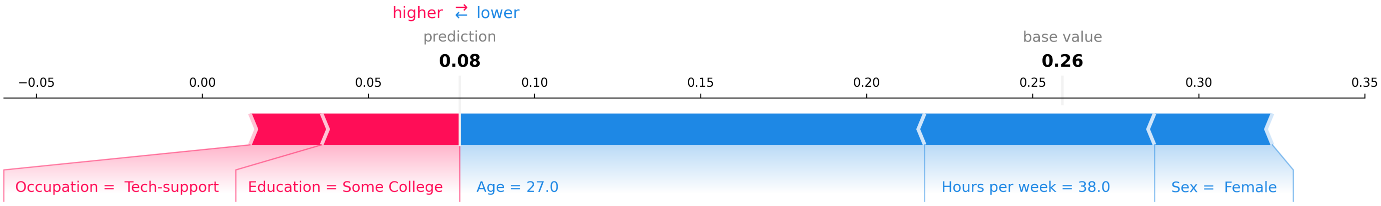
```
explainer = shap.Explainer(model, X, model_output="probability")
shap_values = explainer(X)
```

```
 99%|===================| 32293/32561 [01:44<00:00]
```

```
#shap_values_standin0 = pd.Series({'Age': 0.0307, 'Education': -0.0287, 'Occupati
shap_values_standin0 = pd.Series({'Age': -0.14, 'Education': 0.0416, 'Occupation'
#shap_values_standin0 = pd.Series({'Age': 0.1209, 'Education': 0.3008, 'Occupatio
#shap_values_standin0 = pd.Series({'Age': -0.2119, 'Education': 0.0011, 'Occupati
#shap_values_standin0 = pd.Series({'Age': 0.0565, 'Education': 0.1427, 'Occupatio
#shap_values_standin0 = pd.Series({'Age': -0.0012, 'Education': -0.189, 'Occupati
#shap_values_standin0 = pd.Series({'Age': 0.0774, 'Education': 0.1962, 'Occupatio
#shap_values_standin0 = pd.Series({'Age': 0.0668, 'Education': 0.1619, 'Occupatio

shap.plots.force(0.259, shap_values = np.array(shap_values_standin0), features =
```

```
patched
pos 0.15704999999999994, neg 0.08375999999999997
```


higher ⇄ lower
prediction
base value
Occupation = Tech-support
Education = Some College
Age = 27.0
Hours per week = 38.0
Sex = Female