Import Libraries

Essential libraries for data manipulation, modeling, metrics calculation, fairness evaluation, explainability, and visualization are imported.

In [1]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score

from fairlearn.metrics import MetricFrame, selection_rate, true_positive_
from fairlearn.reductions import ExponentiatedGradient, DemographicParity

import lime
import lime.lime_tabular
from IPython.display import display, HTML

import matplotlib.pyplot as plt
```

Load Dataset

The UCI Adult Income dataset is loaded using Fairlearn and converted into a DataFrame for further processing.

In [2]:
```
from fairlearn.datasets import fetch_adult

# Load dataset
data = fetch_adult(as_frame=True)
df = data.frame.copy()
df.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationsh
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-ch
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husba
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husba
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husba
4	18	NaN	103497	Some-college	10	Never-married	NaN	Own-ch

Preprocessing and Data Splitting

Features and target are separated, the sensitive attribute is identified and removed from the features, categorical variables are label-encoded, the dataset is split into training and test sets, and numeric features are standardized.

In [3]:
```
# Features and target
X = df.drop(columns=["class"])
y = df["class"].apply(lambda x: 1 if x==">50K" else 0)  # Binary encoding

# Sensitive attribute
sensitive_feature = X["sex"]
X = X.drop(columns=["sex"])  # Remove sensitive attribute

# Label encode categorical variables
categorical_cols = X.select_dtypes(include='category').columns
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    label_encoders[col] = le  # Save encoder for mapping back if needed

# Split dataset
X_train, X_test, y_train, y_test, s_train, s_test = train_test_split(
    X, y, sensitive_feature, test_size=0.3, random_state=42, stratify=y
)

# Standardize numeric features
numeric_cols = X_train.select_dtypes(include=np.number).columns
scaler = StandardScaler()
X_train[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test[numeric_cols] = scaler.transform(X_test[numeric_cols])
```

Train Logistic Regression

A logistic regression classifier is trained on the training data and evaluated on the test data using accuracy, precision, recall, and AUC metrics.

In [4]:
```
clf = LogisticRegression(max_iter=500, solver="lbfgs")
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
y_proba = clf.predict_proba(X_test)[:,1]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)

accuracy, precision, recall, auc
```

Out[4]:
```
(0.8248822766668942,
 0.7214891611687088,
 0.43667997718197377,
 0.8496348209999329)
```

Fairness Metrics Evaluation

Fairness metrics including demographic parity, equal opportunity, and equalized odds are computed for each group based on the sensitive attribute using Fairlearn.

In [5]:
```
metrics = {
    "accuracy": accuracy_score,
    "selection_rate": selection_rate,
    "true_positive_rate": true_positive_rate,
    "false_positive_rate": false_positive_rate
}

metric_frame = MetricFrame(metrics=metrics, y_true=y_test, y_pred=y_pred,
metric_frame.by_group
```

Out[5]:

sex	accuracy	selection_rate	true_positive_rate	false_positive_rate
Female	0.891497	0.056002	0.269439	0.028578
Male	0.791854	0.188853	0.467999	0.068391

Transparency Analysis with LIME

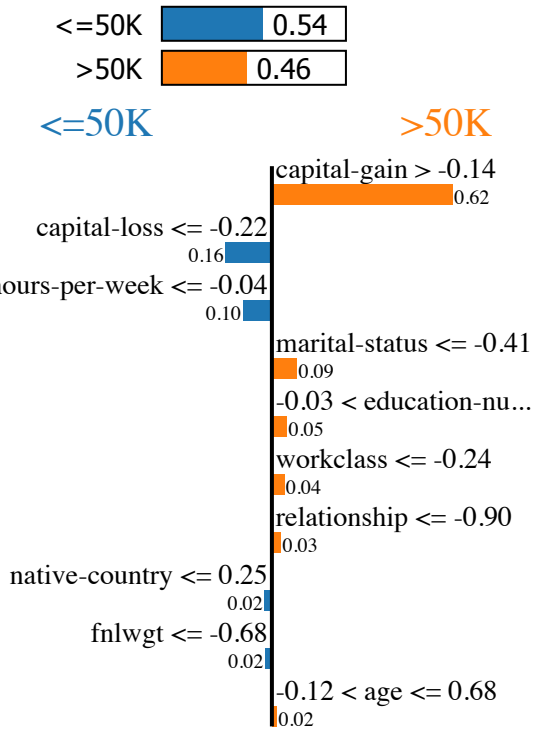
A LIME explainer is created to interpret model predictions. Explanations are generated and displayed for selected test instances to show feature contributions.

In [6]:
```
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=np.array(X_train),
    feature_names=X_train.columns,
    class_names=["<=50K", ">50K"],
    mode="classification"
)

idxs_to_explain = [0, 1, 2]
for idx in idxs_to_explain:
    exp = explainer.explain_instance(
        data_row=X_test.iloc[idx],
        predict_fn=clf.predict_proba
    )
    print(f"Explanation for test instance {idx}:")
    display(HTML(exp.as_html()))
```

```
Explanation for test instance 0:
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:544: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  binary_column = (inverse_column == first_row[column]).astype(int)
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/skl
earn/utils/validation.py:2749: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:427: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  discretized_instance[f])]
```

Prediction probabilities

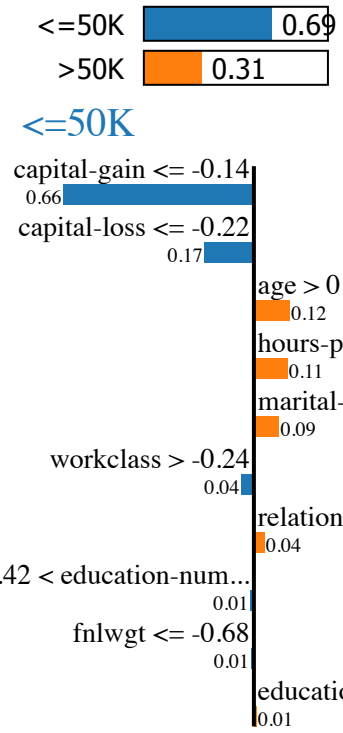


Feature	Value
capital-gain	0.24
capital-loss	-0.22
hours-per-week	-0.04
marital-status	-0.41
education-num	0.36
workclass	-0.24
relationship	-0.90
native-country	0.25
fnlwgt	-1.54
age	0.09

Explanation for test instance 1:

```
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:544: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  binary_column = (inverse_column == first_row[column]).astype(int)
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/skl
earn/utils/validation.py:2749: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:427: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  discretized_instance[f])]
```

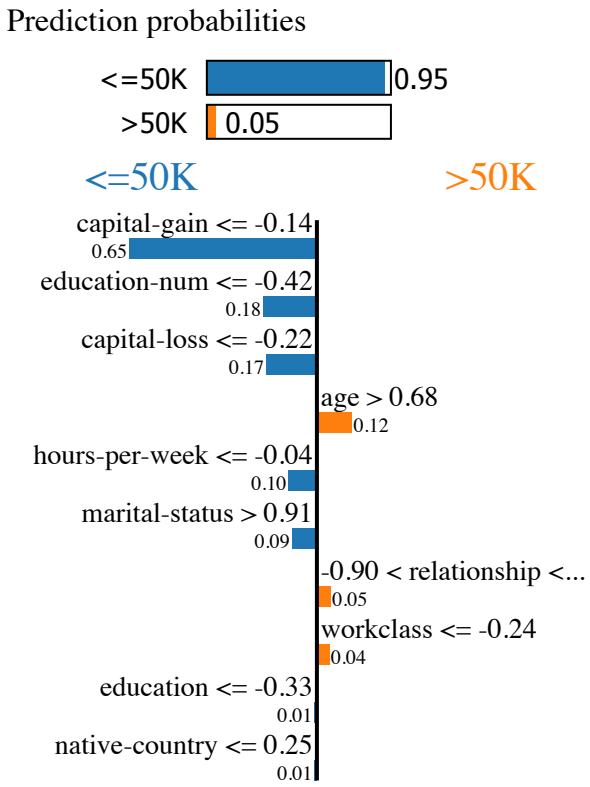
Prediction probabilities



Feature	Value
capital-gain	-0.14
capital-loss	-0.22
age	0.97
hours-per-week	0.77
marital-status	-0.41
workclass	1.02
relationship	-0.90
education-num	-0.03
fnlwgt	-1.12
education	1.22

Explanation for test instance 2:

```
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:544: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  binary_column = (inverse_column == first_row[column]).astype(int)
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/skl
earn/utils/validation.py:2749: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__getitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/discretize.py:110: FutureWarning: Series.__setitem__ treating keys as po
sitions is deprecated. In a future version, integer keys will always be tr
eated as labels (consistent with DataFrame behavior). To set a value by po
sition, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/Users/afan/miniconda3/envs/ResponsibleAI/lib/python3.11/site-packages/lim
e/lime_tabular.py:427: FutureWarning: Series.__getitem__ treating keys as
positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value
by position, use `ser.iloc[pos]`
  discretized_instance[f])]
```



Feature	Value
capital-gain	-0.14
education-num	-1.20
capital-loss	-0.22
age	0.75
hours-per-week	-0.04
marital-status	1.57
relationship	-0.28
workclass	-0.24
education	-2.40
native-country	-0.74

Fairness Intervention with ExponentiatedGradient

The ExponentiatedGradient algorithm with a demographic parity constraint is applied to mitigate bias. Metrics are recomputed for the adjusted model predictions.

In [7]:
```
constraint = DemographicParity()
exp_grad = ExponentiatedGradient(LogisticRegression(max_iter=500, solver=
exp_grad.fit(X_train, y_train, sensitive_features=s_train)

y_pred_fg = exp_grad.predict(X_test)

metric_frame_fg = MetricFrame(metrics=metrics, y_true=y_test, y_pred=y_pr
metric_frame_fg.by_group
```

Out[7]:

sex	accuracy	selection_rate	true_positive_rate	false_positive_rate
Female	0.879761	0.116327	0.482821	0.069238
Male	0.768681	0.125664	0.324754	0.039749

Visualize Fairness Improvement

Selection rates before and after the fairness intervention are visualized to show changes in demographic parity.

In [8]:
```
df_metrics = pd.DataFrame({
    "Baseline": metric_frame.by_group["selection_rate"],
    "After Intervention": metric_frame_fg.by_group["selection_rate"]
})

df_metrics.plot(kind="bar", figsize=(6,4))
plt.ylabel("Selection Rate (>50K prediction)")
plt.title("Demographic Parity Improvement")
plt.show()
```