```
!pip install fairlearn
```

```
Collecting fairlearn
  Downloading fairlearn-0.12.0-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: numpy>=1.24.4 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (2.0.2)
Requirement already satisfied: pandas>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (2.2.2)
Requirement already satisfied: scikit-learn>=1.2.1 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (1.6.1)
Requirement already satisfied: scipy>=1.9.3 in /usr/local/lib/python3.11/dist-packages (from fairlearn) (1.16.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.0.3->fairlearn) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2.1->fairlearn) (1.5.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2.1->fairlearn
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0.3->fair
Downloading fairlearn-0.12.0-py3-none-any.whl (240 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240.0/240.0 kB 15.7 MB/s eta 0:00:00
Installing collected packages: fairlearn
Successfully installed fairlearn-0.12.0
```

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from fairlearn.metrics import MetricFrame, selection_rate
import matplotlib.pyplot as plt
from fairlearn.reductions import ExponentiatedGradient, DemographicParity
```

```
data= pd.read_csv("/content/HeartDiseaseTrain-Test.csv")
print(data.head())
```

```
   age     sex chest_pain_type  resting_blood_pressure  cholestoral  \
0   52    Male  Typical angina                     125          212   
1   53    Male  Typical angina                     140          203   
2   70    Male  Typical angina                     145          174   
3   61    Male  Typical angina                     148          203   
4   62  Female  Typical angina                     138          294   

      fasting_blood_sugar               rest_ecg  Max_heart_rate  \
0    Lower than 120 mg/ml  ST-T wave abnormality             168   
1  Greater than 120 mg/ml                 Normal             155   
2    Lower than 120 mg/ml  ST-T wave abnormality             125   
3    Lower than 120 mg/ml  ST-T wave abnormality             161   
4  Greater than 120 mg/ml  ST-T wave abnormality             106   

  exercise_induced_angina  oldpeak        slope vessels_colored_by_flourosopy  \
0                      No      1.0  Downsloping                           Two   
1                     Yes      3.1    Upsloping                          Zero   
2                     Yes      2.6    Upsloping                          Zero   
3                      No      0.0  Downsloping                           One   
4                      No      1.9         Flat                         Three   

         thalassemia  target  
0  Reversable Defect       0  
1  Reversable Defect       0  
2  Reversable Defect       0  
3  Reversable Defect       0  
4      Fixed Defect       0  
```

```
data = data[['age', 'sex', 'chest_pain_type', 'resting_blood_pressure', 'cholestoral', 'Max_heart_rate', 'oldpeak', 'target']]
```

```
print(type(data))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
print(data.dtypes)
```

```
age                         int64
sex                        object
chest_pain_type            object
resting_blood_pressure      int64
cholestoral                 int64
Max_heart_rate              int64
oldpeak                   float64
target                      int64
dtype: object
```

```
data = pd.get_dummies(data, columns=['chest_pain_type'], drop_first=True)
```

```
X = data.drop('target', axis=1)
y = data['target']
sensitive_feature = X['sex']


# Encode the 'sex' column before splitting
X = pd.get_dummies(X, columns=['sex'], drop_first=True)

X_train, X_test, y_train, y_test, sf_train, sf_test = train_test_split(
    X, y, sensitive_feature, test_size=0.3, random_state=42
)


model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)



#  Evaluate fairness
metric_frame = MetricFrame(
    metrics={
        "Selection Rate": selection_rate,
        "Accuracy": accuracy_score
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sf_test
)


print("\nBaseline Model Metrics by Sex:")
print(metric_frame.by_group)
print("\nOverall Accuracy:", metric_frame.overall['Accuracy'])
```

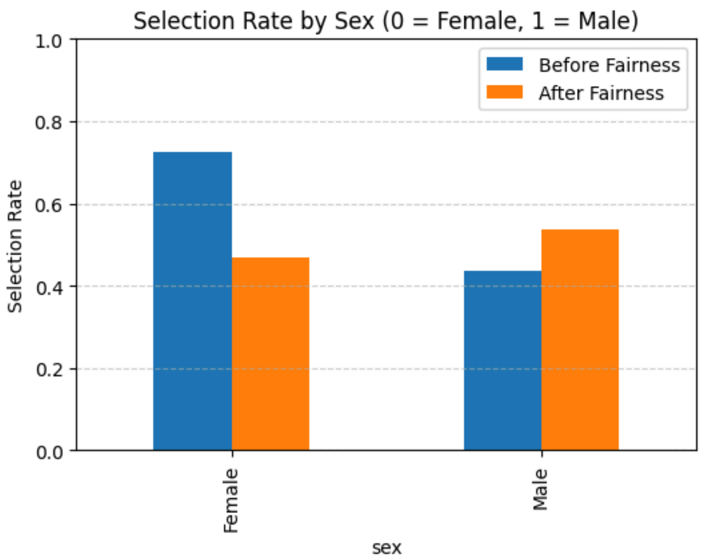
```
Baseline Model Metrics by Sex:
        Selection Rate  Accuracy
sex
Female        0.725490  0.823529
Male          0.436893  0.752427

Overall Accuracy: 0.775974025974026
```

```
fair_model = ExponentiatedGradient(
    LogisticRegression(max_iter=1000),
    constraints=DemographicParity(),
    eps=0.01
)

fair_model.fit(X_train, y_train, sensitive_features=sf_train)
y_pred_fair = fair_model.predict(X_test)



# Evaluate fairness after mitigation
metric_frame_fair = MetricFrame(
    metrics={
        "Selection Rate": selection_rate,
        "Accuracy": accuracy_score
    },
    y_true=y_test,
    y_pred=y_pred_fair,
    sensitive_features=sf_test
)

print("\nFair Model Metrics by Sex:")
print(metric_frame_fair.by_group)
print("\nOverall Accuracy (Fair Model):", metric_frame_fair.overall['Accuracy'])
```

```
Fair Model Metrics by Sex:
        Selection Rate  Accuracy
sex
Female        0.470588  0.725490
Male          0.538835  0.728155

Overall Accuracy (Fair Model): 0.7272727272727273
```

```
# 6. Visualize selection rates before & after fairness constraint
```

```
before = metric_frame.by_group["Selection Rate"]
after = metric_frame_fair.by_group["Selection Rate"]

df_plot = pd.DataFrame({
    'Before Fairness': before,
    'After Fairness': after
})


df_plot.plot(kind='bar', figsize=(6,4))
plt.title("Selection Rate by Sex (0 = Female, 1 = Male)")
plt.ylabel("Selection Rate")
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```



Start coding or generate with AI.

1. Before Fairness (Blue Bars)
Female (~0.73): The model predicted a positive outcome (heart disease) for ~73% of females.

Male (~0.44): The model predicted a positive outcome for ~44% of males.
→ This is a big gap (~29%) — the model was much more likely to predict heart disease for women than for men.

2. After Fairness (Orange Bars)
Female (~0.44): The positive prediction rate for women dropped significantly.

Male (~0.54): The positive prediction rate for men increased.
→ The gap shrank to ~10%, making the model’s predictions more balanced across genders.

Before fairness: The model was biased toward predicting heart disease more often for women.

After fairness: The model’s prediction rates are closer between genders, reducing gender bias.

This comes at the cost of changing predictions, which can sometimes reduce accuracy slightly — a common trade-off in fairness int