Assignment 14: Ethical AI Analysis and Explainability

(Afnan Madi)

```
# Step 1: Install Required Packages
!pip install fairlearn shap lime --quiet
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 6.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240.0/240.0 kB 16.6 MB/s eta 0:00:00
  Building wheel for lime (setup.py) ... done
```

```
# Step 2: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from fairlearn.metrics import MetricFrame, selection_rate, false_positive_rate, true_positive_rate
import shap
import lime
import lime.lime_tabular
import warnings
warnings.filterwarnings("ignore")
```

```
# Step 3: Load Dataset

adult = fetch_openml(data_id=1590, as_frame=True)  # 'adult' dataset (aka Census Income)
df = adult.frame
```

```
# Step 4: Data Preprocessing

df = df.dropna()  # Remove missing values
X = df.drop(columns=["class"])  # Features
y = df["class"]  # Target

#Convert income labels to binary (for Fairlearn compatibility)
y = y.apply(lambda x: 1 if x == '>50K' else 0)

# Encode categorical features
X_encoded = pd.get_dummies(X, drop_first=True)

# Define sensitive feature (e.g., 'sex')
sensitive_feature = X["sex"]
```

```
# Step 5: Split Data into Train and Test Sets

X_train, X_test, y_train, y_test, sens_train, sens_test = train_test_split(
    X_encoded, y, sensitive_feature, test_size=0.3, random_state=42
)
```

```
# Step 6: Train Logistic Regression Model

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

```
# Step 7: Evaluate Model Performance

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nConfusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", class_report)
```

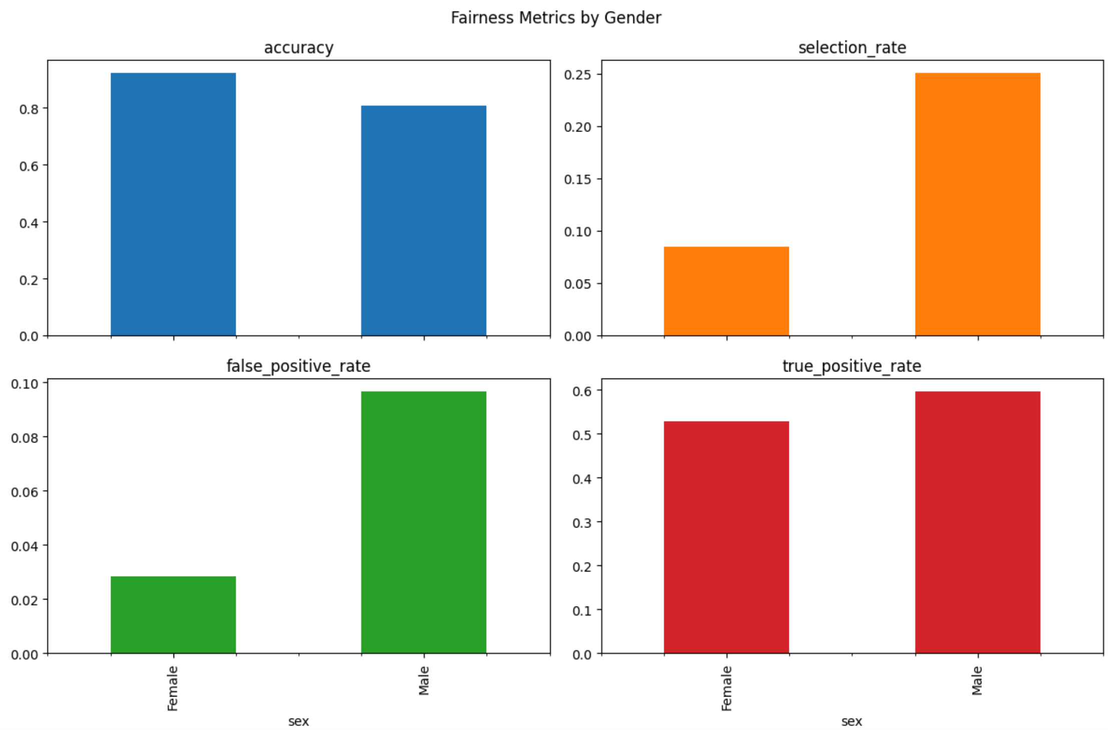
```
Accuracy: 0.8449915235497899

Confusion Matrix:
 [[9513  728]
 [1375 1951]]

Classification Report:
               precision    recall  f1-score   support

           0       0.87      0.93      0.90     10241
           1       0.73      0.59      0.65      3326

    accuracy                           0.84     13567
   macro avg       0.80      0.76      0.78     13567
weighted avg       0.84      0.84      0.84     13567
```

```
# Step 8: Fairness Analysis Using Fairlearn

metric_frame = MetricFrame(
    metrics={
        "accuracy": accuracy_score,
        "selection_rate": selection_rate,
        "false_positive_rate": false_positive_rate,
        "true_positive_rate": true_positive_rate
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sens_test
)

# Display metrics by group (e.g., Male vs. Female)
```

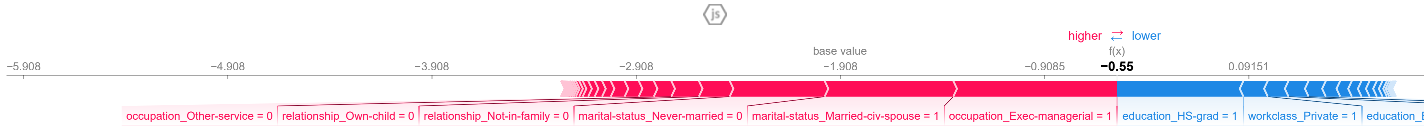
```
print("\nFairness Metrics by Gender:\n", metric_frame.by_group)
```

```
Fairness Metrics by Gender:
          accuracy  selection_rate  false_positive_rate  true_positive_rate
sex
Female  0.922012        0.084449             0.028571            0.528926
Male    0.808838        0.250514             0.096698            0.596411
```

```
# Step 9: Visualize Fairness Metrics
metric_frame.by_group.plot.bar(subplots=True, layout=(2, 2), legend=False, figsize=(12, 8))
plt.suptitle("Fairness Metrics by Gender")
plt.tight_layout()
plt.show()
```

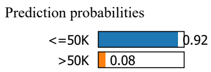


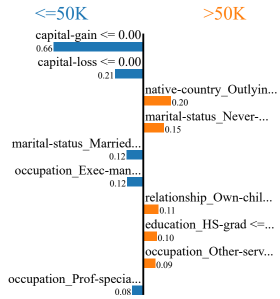

```
# Step 10: Explainability with SHAP (Fixed Version)
import shap

# Convert to numpy arrays (required by LinearExplainer)
X_sample = X_test.sample(100, random_state=42)
X_sample_np = X_sample.to_numpy()

# Re-train model using numpy input to match SHAP expectations
model.fit(X_train.to_numpy(), y_train)

# Use LinearExplainer for logistic regression
explainer = shap.LinearExplainer(model, X_train.to_numpy(), feature_perturbation="interventional")
shap_values = explainer.shap_values(X_sample_np)

# Local explanation - use SHAP force plot (not waterfall which fails on float models)
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[0], X_sample.iloc[0])
```




```
# Step 11: Explainability with LIME

import lime
import lime.lime_tabular

# Convert training and test sets to NumPy arrays
X_train_np = X_train.to_numpy()
X_test_np = X_test.to_numpy()

# Create a LIME explainer for tabular data
lime_explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train_np,
    feature_names=X_train.columns,
    class_names=["<=50K", ">50K"],
    mode='classification'
)

# Choose an instance to explain (e.g., first one from the test set)
instance_index = 0
instance_data = X_test_np[instance_index]

# Explain the prediction for that instance
```

```
# Explain the prediction for that instance
lime_exp = lime_explainer.explain_instance(
    data_row=instance_data,
    predict_fn=model.predict_proba  # use model's predict_proba for classification
)

# Show explanation in notebook
lime_exp.show_in_notebook(show_table=True, show_all=False)
```





Feature	Value
capital-gain	0.00
capital-loss	0.00
native-country_Outlying-US(Guam-USVI-etc)	0.00
marital-status_Never-married	0.00
marital-status_Married-civ-spouse	0.00
occupation_Exec-managerial	0.00
relationship_Own-child	0.00
education_HS-grad	0.00
occupation_Other-service	0.00
occupation_Prof-specialty	0.00