```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pickle
import os
# Load dataset
data = pd.read_csv("adult.csv")

# Drop rows with missing values (if any are present)
data = data.dropna()

# Encode target column
le = LabelEncoder()
data['income'] = le.fit_transform(data['income'])  # <=50K -> 0, >50K -> 1

# Separate features and target
X = data.drop(columns=['income'])
y = data['income']

# One-hot encode categorical variables
X = pd.get_dummies(X, drop_first=True)

# Standardize numeric features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, stratify=y, test_size=0.3, random_state=42
)
os.makedirs("models", exist_ok=True)
# Save the split dataset
pickle.dump((X_train, X_test, y_train, y_test), open("models/split.pkl", "wb"))



# 3. scripts/train.py
from sklearn.linear_model import LogisticRegression
import pickle
import os

# Load the dataset splits
X_train, X_test, y_train, y_test = pickle.load(open("models/split.pkl", "rb"))

# Train a binary classification model
model = LogisticRegression(solver='lbfgs', max_iter=1000)  # max_iter increased just in case

# Fit the model
model.fit(X_train, y_train)

# Ensure models directory exists
os.makedirs("models", exist_ok=True)

# Save the trained model
pickle.dump(model, open("models/model.pkl", "wb"))

print("Model training complete and saved to models/model.pkl")
```

```
Model training complete and saved to models/model.pkl
```

```
# 4. scripts/evaluate.py
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pickle

# Load test data and model
X_train, X_test, y_train, y_test = pickle.load(open("models/split.pkl", "rb"))
model = pickle.load(open("models/model.pkl", "rb"))

# Predict
y_pred = model.predict(X_test)

# Evaluate
acc = accuracy_score(y_test, y_pred)
print(f"✅ Test Accuracy: {acc:.2%}")
```

```
# Optional: Detailed report
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred))

# Optional: Confusion matrix
print("🟫 Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
✅ Test Accuracy: 85.57%

📊 Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     11147
           1       0.75      0.60      0.67      3506

    accuracy                           0.86     14653
   macro avg       0.81      0.77      0.79     14653
weighted avg       0.85      0.86      0.85     14653

🟫 Confusion Matrix:
[[10434   713]
 [ 1401  2105]]
```