



# CDW Documentation

## Responsible AI Test

---

# Responsible AI Test

## Purpose

Evaluate the Responsible AI dashboard and see what it does.

## Test Process

Here's a structured list of **Responsible AI Dashboard Deployment Steps** using the corrected scripts. Each step includes:

- **Step Number & Action**
- **Purpose**
- **Expected Result**

---

### Step 1: Install Required Packages

```
bashCopyEditpip install --upgrade raiutils raiwidgets responsibleai ipywidgets
```

#### Purpose:

Install the Python packages required to run Responsible AI analysis and render the dashboard.

#### Expected Result:

Packages are installed without errors; dashboard widgets can render in the notebook (after kernel restart).

---

### Step 2: Load and Preprocess the Dataset

```
pythonCopyEditfrom sklearn.datasets import fetch_openml
import pandas as pd

data = fetch_openml(name='adult', version=2, as_frame=True)
df = data.frame.dropna()
```

#### Purpose:

Load a well-known classification dataset (Adult Census Income) and remove any missing values to avoid downstream errors.

#### Expected Result:

A clean DataFrame with no null values is loaded.

### □ Step 3: Split Dataset into Train and Test Sets

```
pythonCopyEditfrom sklearn.model_selection import train_test_split

target_column = 'class'
X = df.drop(columns=[target_column])
y = df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
random_state=42)
```

#### □ Purpose:

Separate features and target, then split into training/testing sets for model training and evaluation.

#### □ Expected Result:

X\_train, X\_test, y\_train, y\_test variables created and stratified properly.

### □ Step 4: Define Preprocessing and Train a Model

```
pythonCopyEditfrom sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier

categorical_cols = X_train.select_dtypes(include=['object',
'category']).columns.tolist()
numerical_cols = X_train.select_dtypes(include=['int64',
'float64']).columns.tolist()

preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols),
    ('num', StandardScaler(), numerical_cols)
])

clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=10, random_state=42))
])

clf.fit(X_train, y_train)
```

#### □ Purpose:

Build a model pipeline that encodes categorical features, scales numeric ones, and trains a classifier.

#### □ Expected Result:

Pipeline is trained successfully on the training data without conversion errors.

## □ Step 5: Prepare Data for RAIInsights

```
pythonCopyEdit# Ensure target column is a supported type
y_train_clean = y_train.astype(str)
y_test_clean = y_test.astype(str)

train_data = X_train.copy()
train_data[target_column] = y_train_clean

test_data = X_test.copy()
test_data[target_column] = y_test_clean
```

### □ Purpose:

Re-attach the target column (as string) to the feature DataFrames — required for RAIInsights.

### □ Expected Result:

train\_data and test\_data DataFrames contain all required columns including the target.

## □ Step 6: Initialize the Responsible AI Insights Object

```
pythonCopyEditfrom responsibleai import RAIInsights, FeatureMetadata

feature_metadata = FeatureMetadata(categorical_features=categorical_cols)

rai_insights = RAIInsights(
    model=clf,
    train=train_data,
    test=test_data,
    target_column=target_column,
    task_type="classification",
    feature_metadata=feature_metadata
)
```

### □ Purpose:

Create a RAIInsights object that acts as the core engine for the Responsible AI dashboard.

### □ Expected Result:

RAIInsights object is initialized successfully and ready for configuration.

## □ Step 7: Add Responsible AI Analysis Tools

```
pythonCopyEditrai_insights.explainer.add()
rai_insights.error_analysis.add()
```

```
rai_insights.counterfactual.add(total_CFs=5, desired_class='opposite')
rai_insights.causal.add(treatment_features=categorical_cols)
```

□ **Purpose:**

Attach various tools (explanation, error analysis, counterfactuals, causal inference) to the insights engine.

□ **Expected Result:**

No errors thrown; tools are queued for computation.

---

□ **Step 8: Compute Insights**

```
pythonCopyEditrai_insights.compute()
```

□ **Purpose:**

Run analysis for all selected tools. This step may take a minute or more.

□ **Expected Result:**

Tool outputs are generated for the first 5,000 rows of the test set.

---

□ **Step 9: Launch the Responsible AI Dashboard**

```
pythonCopyEditfrom raiwidgets import ResponsibleAIDashboard
ResponsibleAIDashboard(rai_insights)
```

□ **Purpose:**

Open an interactive dashboard to explore insights such as feature importance, what-if analysis, and error breakdowns.

□ **Expected Result:**

A dashboard is displayed inside the notebook. Interactive plots and controls are available for analysis.

[AI Knowledge](#)