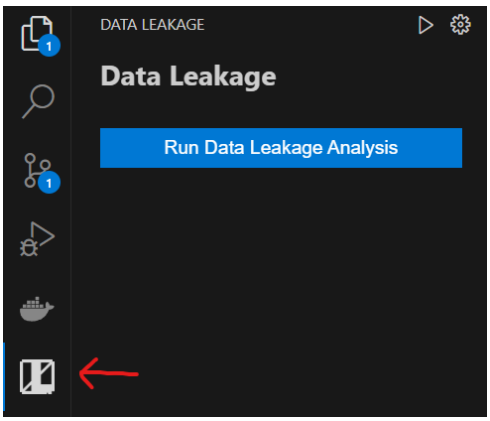
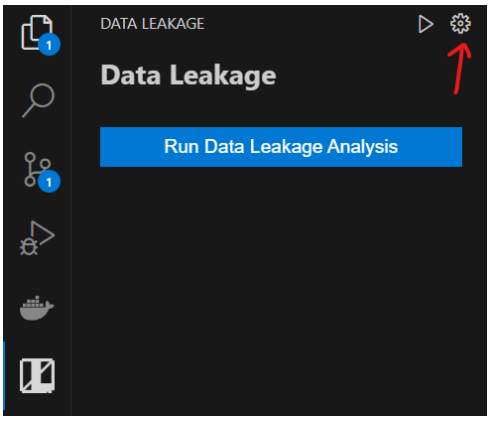
**Disclaimer:** This extension is compatible with both .venv virtual environments and Conda environments for analyzing Jupyter Notebooks for data leakage.

## Guide to Running the Data Leakage Extension in VS Code

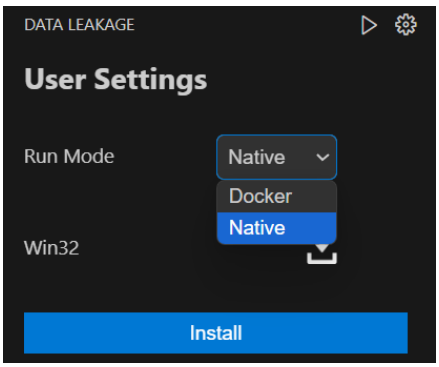
**Step 1:** Launch Visual Studio Code. From the activity bar on the left, choose the "Data Leakage" extension.



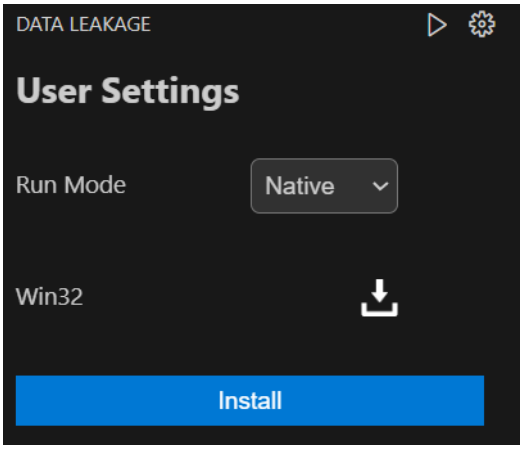

**Step 2:** Click on the settings icon to adjust the run settings.



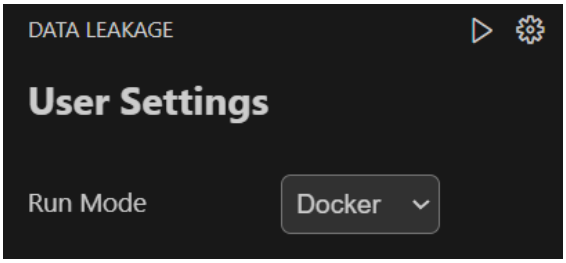
**Step 3:** Choose your preferred run mode from the dropdown menu, either "Native" or "Docker."




- **3.1 Native Mode:** This mode uses a downloaded binary specific to your operating system. Click the download icon beside your OS to get the binary. If you opt for Docker, skip this step.

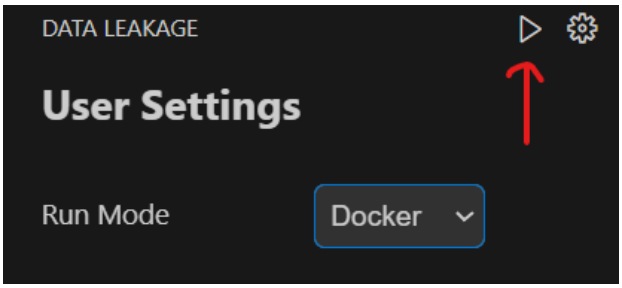



- **3.2 Docker Mode:** This mode automatically installs the Docker image and sets up the container. Ensure Docker Desktop is running in the background.

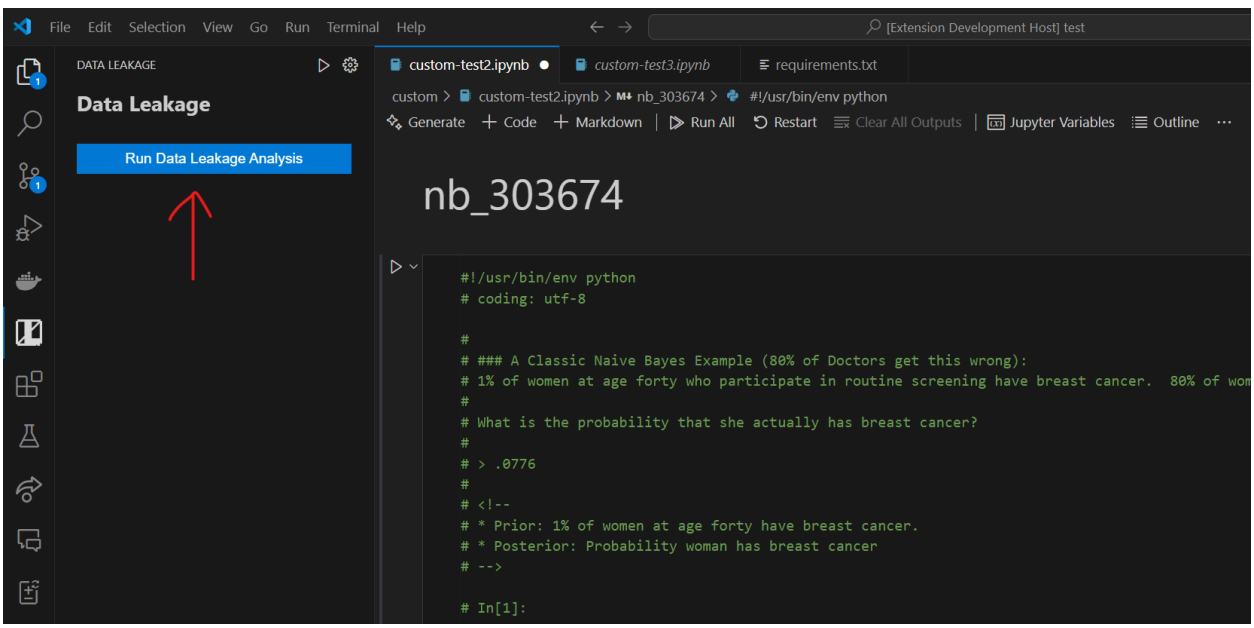


**Step 4:** If you have downloaded the binary, extract this zipped binary to a directory of your choice. Click "Install" and select the extracted binary from your file directory.

**Step 5:** Return to the main extension page by clicking the run icon.

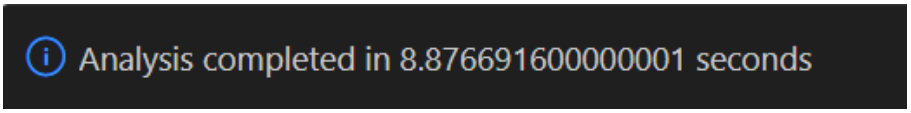



**Step 6:** Open a Jupyter Notebook file in the active tab of VS Code. In the extension window, click "Run Data Leakage Analysis" to start the process.

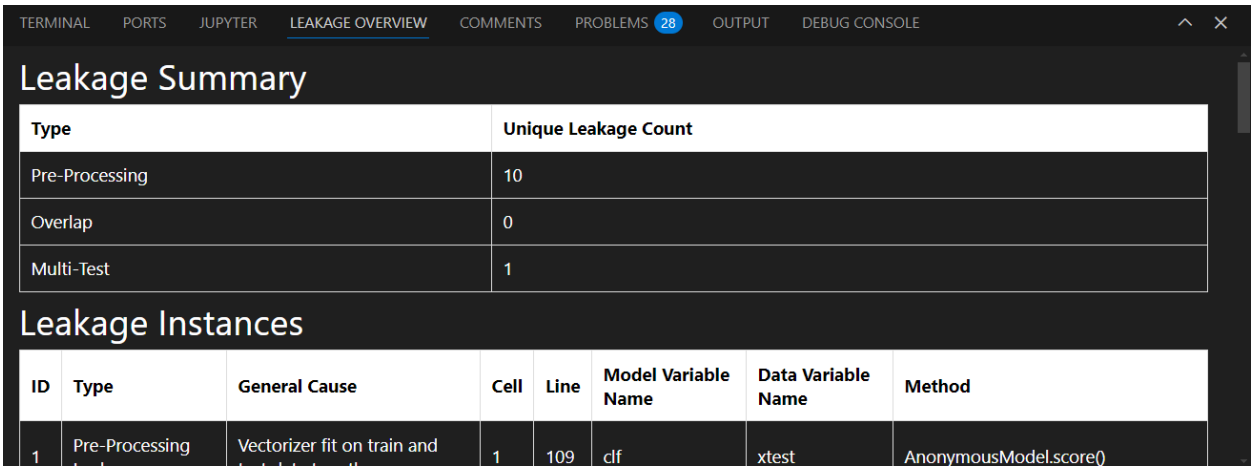


**Step 7:** Allow time for the extension to analyze the notebook for instances of data leakage. This may take a few minutes.

**Step 8:** Once the analysis is complete, you will receive a notification at the bottom right of VS Code.




**Step 9:** Review the "Leakage Overview" tab in the bottom panel of VS Code. It will show a summary of detected leakages and provide a detailed table of instances. Each instance can be examined by clicking on a row in the table.



# Fixing Data Leakage

**Step 1:** Navigate to a data leakage instance by selecting a row in the leakage instances table.

TERMINAL PORTS JUPYTER LEAKAGE OVERVIEW COMMENTS PROBLEMS 28 OUTPUT DEBUG CONSOLE

| | |
|---|---|
| Pre-Processing | 10 |
| Overlap | 0 |
| Multi-Test | 1 |

## Leakage Instances

| ID | Type | General Cause | Cell | Line | Model Variable Name | Data Variable Name | Method |
|---|---|---|---|---|---|---|---|
| 1 | Pre-Processing Leakage | Vectorizer fit on train and test data together | 1 | 109 | clf | xtest | AnonymousModel.score() |
| 2 | Pre-Processing Leakage | Vectorizer fit on train and test data together | 1 | 136 | clf (0) | Name Not Found | AnonymousModel.score() |

**Step 2:** The selected leakage instance will be highlighted in your Jupyter Notebook file.

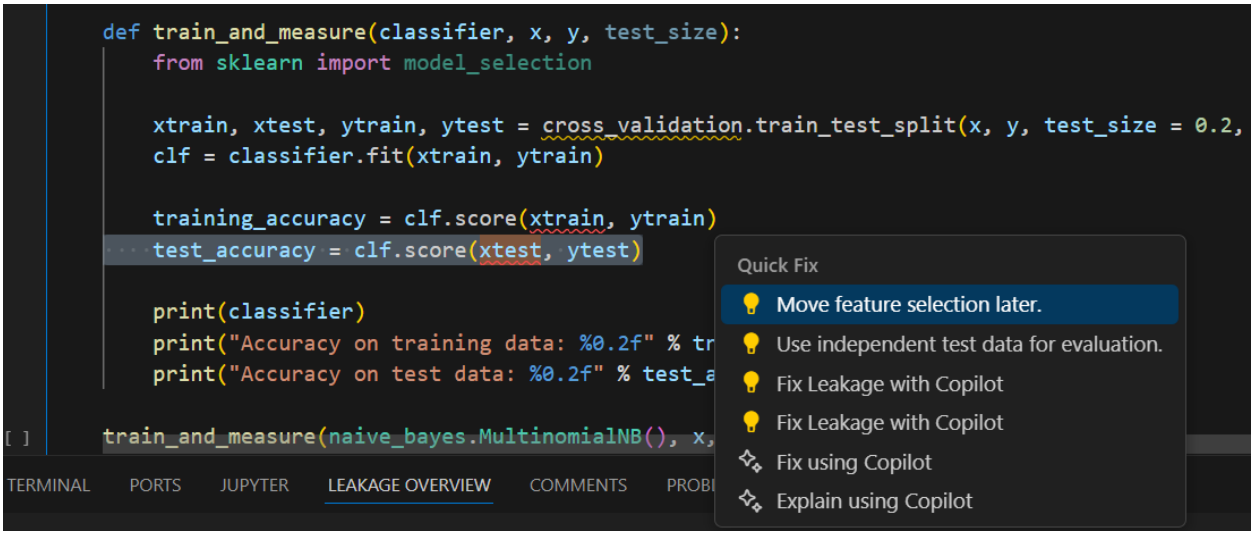
```
def train_and_measure(classifier, x, y, test_size):
    from sklearn import model_selection

    xtrain, xtest, ytrain, ytest = cross_validation.train_test_split(x, y, te
    clf = classifier.fit(xtrain, ytrain)

    training_accuracy = clf.score(xtrain, ytrain)
    test_accuracy = clf.score(xtest, ytest)

    print(classifier)
    print("Accuracy on training data: %0.2f" % training_accuracy)
    print("Accuracy on test data: %0.2f" % test_accuracy)
```

**Step 3:** Hover over the highlighted line with the red error to reveal the "Quick Fix" option.

```
y = (critics.fresh == 'fresh')

# In[10]:

def train_and_measure(classifi
    from sklearn import model_

    xtrain, xtest, ytrain, yte
    clf = classifier.fit(xtrai

    training_accuracy = clf.sc
    test_accuracy = clf.score(xtest, ytest)

    print(classifier)
    print("Accuracy on training data: %0.2f" % training_accuracy)
    print("Accuracy on test data: %0.2f" % test_accuracy)
```

custom-test2.ipynb(110, 31): Variable: xtest
custom-test2.ipynb(110, 31): Model: clf
custom-test2.ipynb(110, 31): Method: AnonymousModel.score()
Data Leakage: MultiTestLeakage MultiTestLeakage(dataLeakage)
custom-test2.ipynb(110, 31): Variable: xtest
custom-test2.ipynb(110, 31): Model: clf
custom-test2.ipynb(110, 31): Method: AnonymousModel.score()
(variable) xtest: Any
View Problem (Alt+F8) Quick Fix... (Ctrl+.) Fix using Copilot (Ctrl+I)

**Step 4:** Click on "Quick Fix" to see several potential solutions. Choose the most suitable option.




**Step 5:** Your Jupyter Notebook will be updated to remove the data leakage instance. Note that these fixes are rudimentary and might not always be the optimal solution. In this example, the cross_validation.train_test_split() was removed to indicate that it should be moved further down.