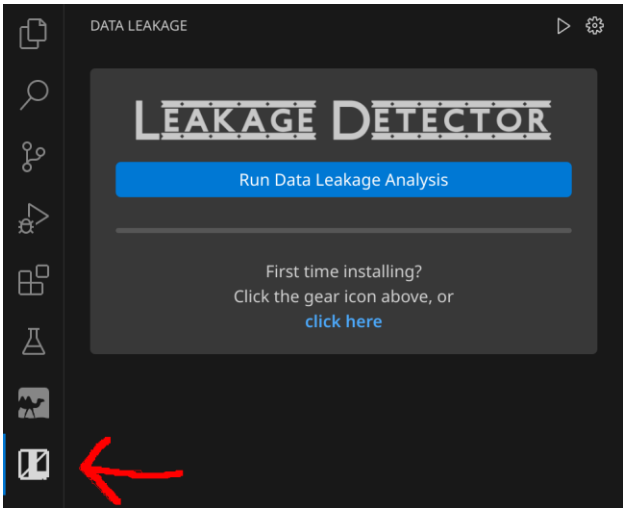
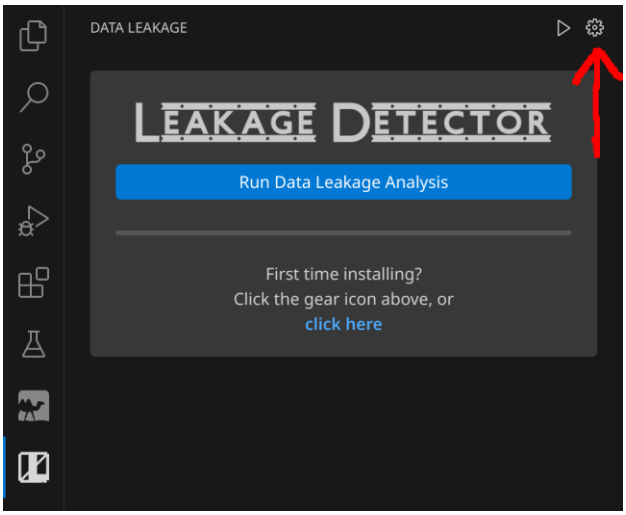
Disclaimer: This extension is compatible with both .venv virtual environments and Conda environments for analyzing Jupyter Notebooks for data leakage.

Guide to Running the Data Leakage Extension in VS Code

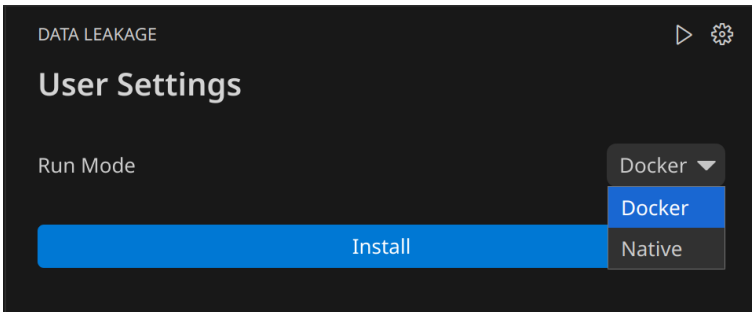
Step 1: Launch Visual Studio Code. From the activity bar on the left, choose the "Data Leakage" extension.



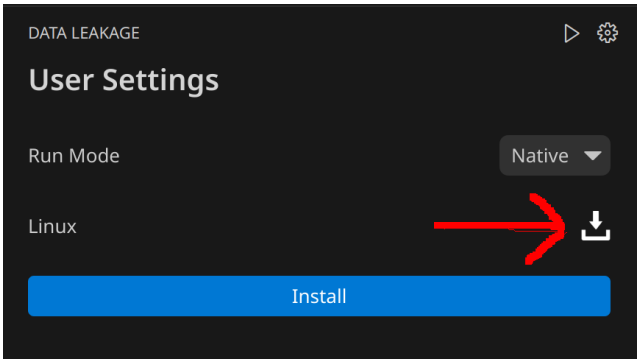
Step 2: Click on the settings icon to adjust the run settings.



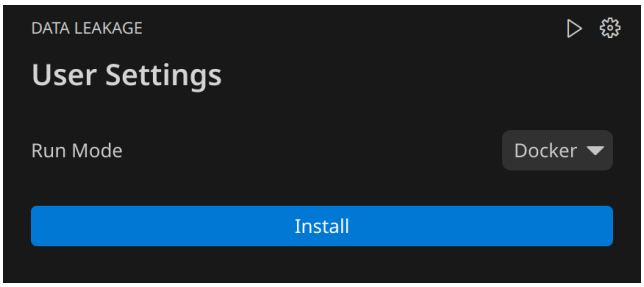
Step 3: Choose your preferred run mode from the dropdown menu, either "Native" or "Docker."



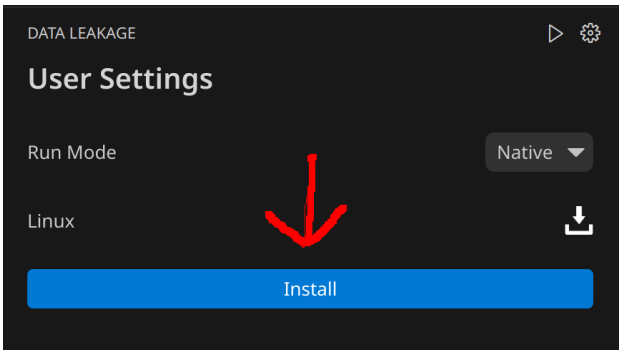
- **3.1 Native Mode:** This mode uses a downloaded binary specific to your operating system. Click the download icon beside your OS to get the binary. If you opt for Docker, skip this step.



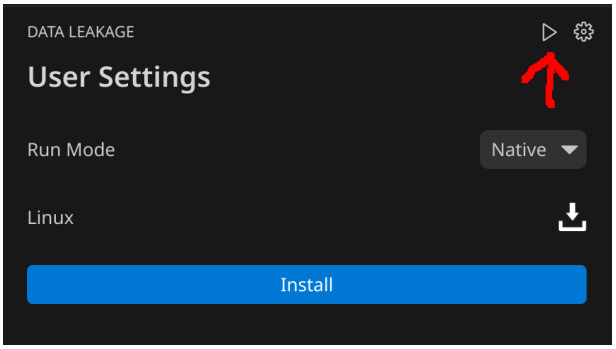
- **3.2 Docker Mode:** This mode automatically installs the Docker image and sets up the container. Ensure Docker Desktop is running in the background.



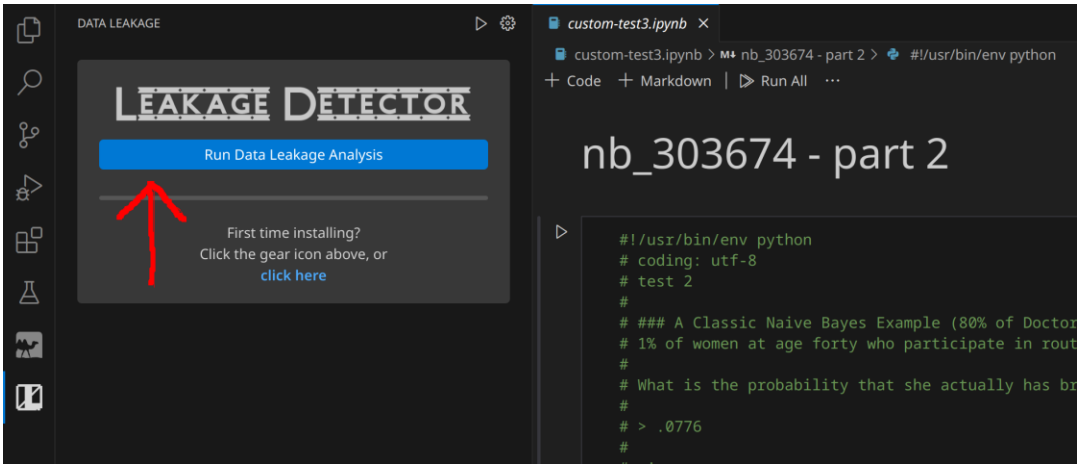
Step 4: If you downloaded the binary, extract this zipped binary to a directory of your choice. Click "Install" and select the extracted binary from your file directory.



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

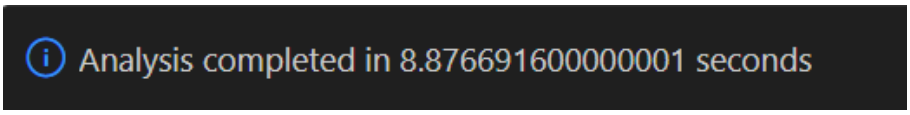


Step 6: Open a Jupyter Notebook file in the active tab of VS Code. In the extension window, click "Run Data Leakage Analysis" to start the process.

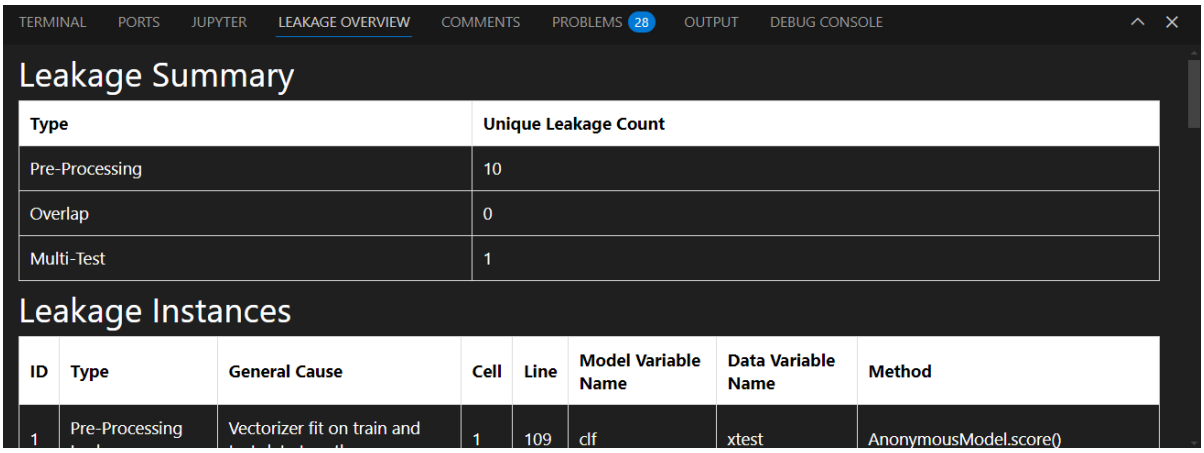


Step 7: Allow time for the extension to analyze the notebook for instances of data leakage. This may take a few minutes.

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



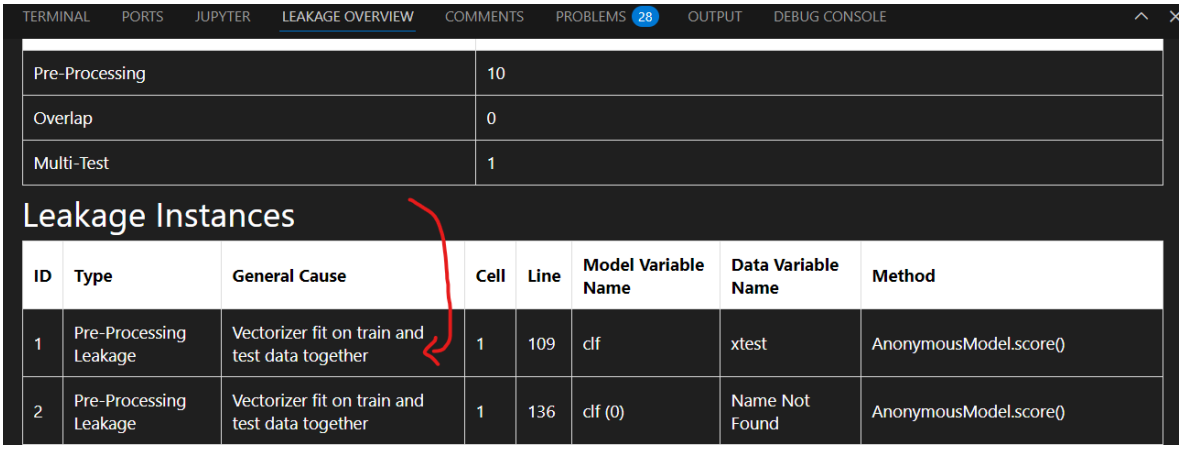
Step 9: Review the "Leakage Overview" tab in the bottom panel of VS Code. It will show a summary of detected leakages and provide a detailed table of instances. Each instance can be examined by clicking on a row in the table.



Fixing Data Leakage

Data leakage can be resolved through our manual Quick Fix solution or through the GitHub Copilot VS Code extension with its AI-based solution.

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



Pre-Processing	10
Overlap	0
Multi-Test	1

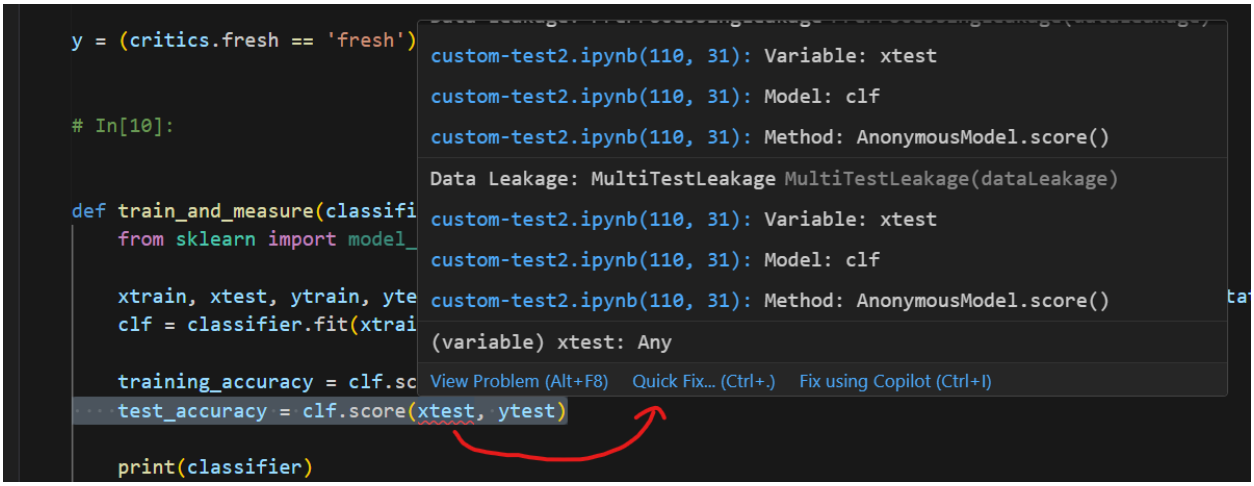
Leakage Instances

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()

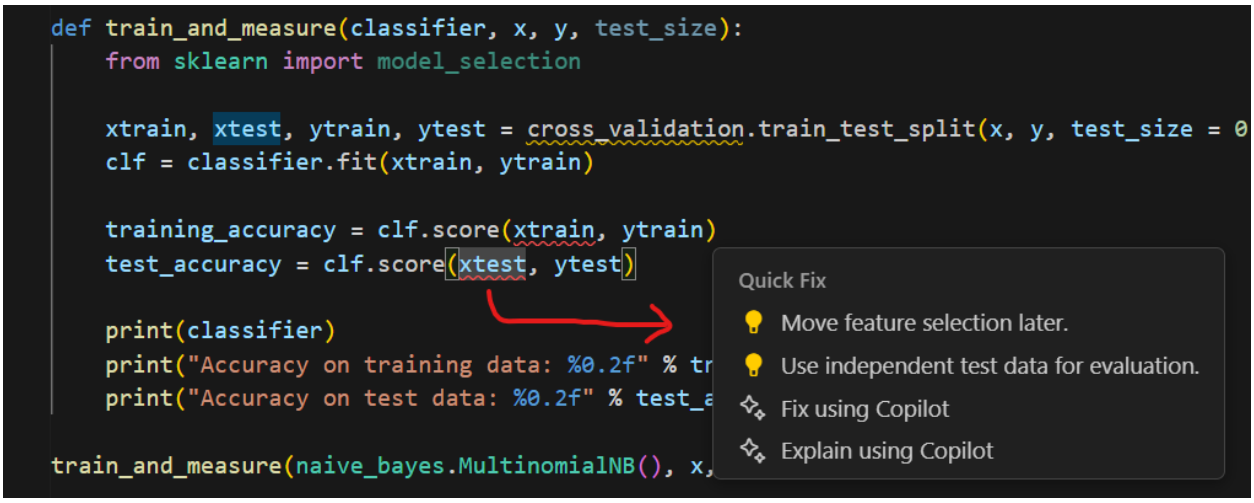
Step 2: The selected leakage instance will be highlighted in your Jupyter Notebook file.

```
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.

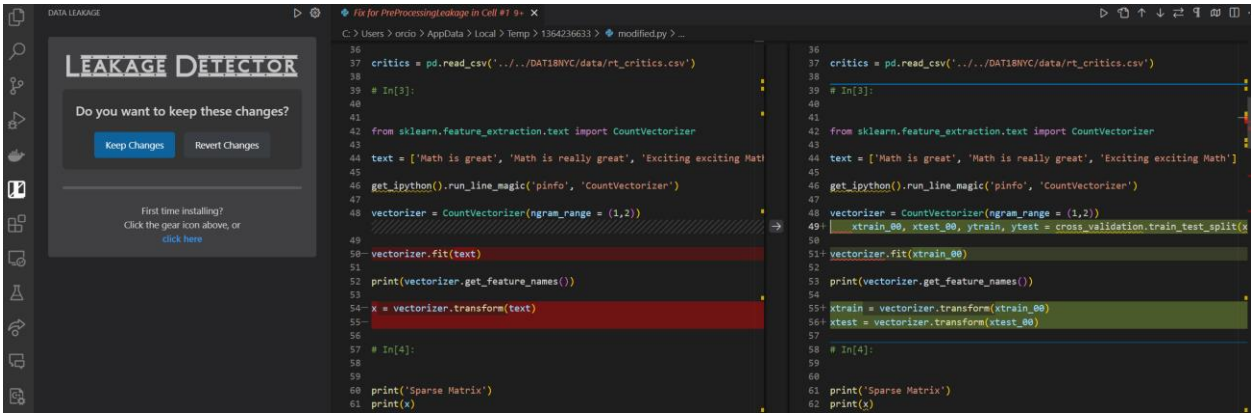


Step 4: Click on "Quick Fix" to see several potential solutions. Then, you may select the light bulb icon to perform the **manual** Quick Fix or select the option “Fix using Copilot” to perform Copilot’s **AI-based** Quick Fix. You must have the GitHub Copilot VS Code extension to fix using Copilot, which is discussed in the installation guide. These options attempt to resolve the data leakage.



Manual Quick Fix

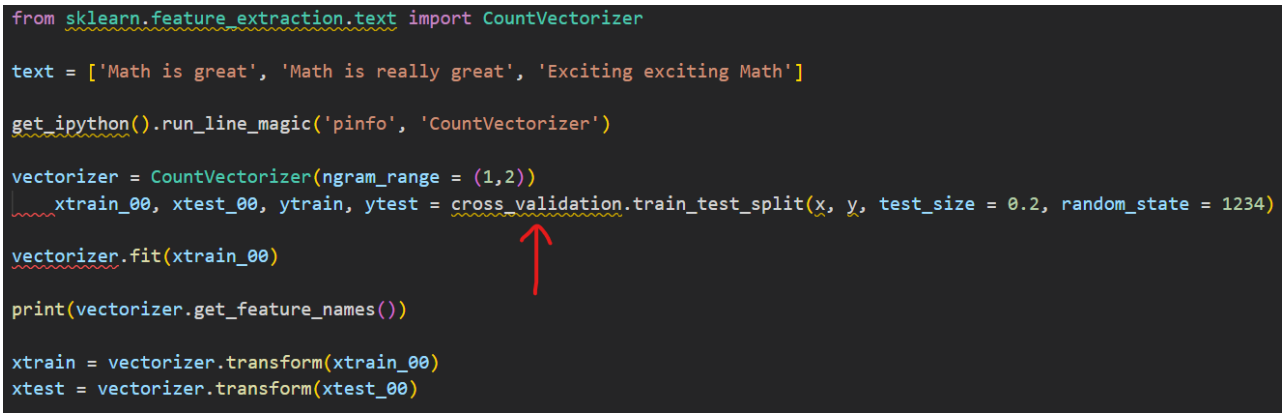
Select any light bulb icon to perform the manual Quick Fix. Next, a diff file is automatically generated for your reference, as shown below.



Then, our VS Code extension in the primary sidebar prompts you to accept or reject the Quick Fix changes. The left file of the diff displays your Jupyter Notebook, and the right file of the diff displays the proposed changes. If changes are accepted, the Jupyter Notebook will be updated to remove the data leakage; otherwise, your file contents will remain the same. Note that these fixes might not always be the optimal solution.

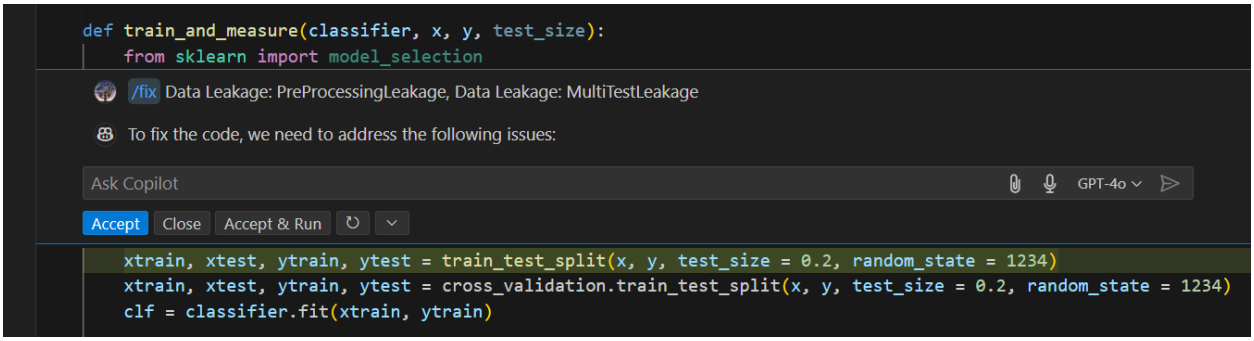
Disclaimer: When you click one of the light bulb icons to Quick Fix, your file will automatically be edited according to the proposed changes but clicking "Revert Changes" in the primary sidebar will rewrite the Jupyter Notebook to its original file content before the manual Quick Fix. The Jupyter Notebook may show a yellow circle next to its file name, but the file contents remain the same.

In this example, we selected the “Move feature selection later” Quick Fix option, so we are performing Quick Fix on an instance of preprocessing leakage. When the user keeps these changes for Quick Fix, the cross_validation.train_test_split() will move out of the train_and_measure() function into a higher position in the cell, along with other changes.



Copilot Quick Fix

Select the option “Fix using Copilot” to perform GitHub Copilot’s AI-based Quick Fix. This will prompt a Copilot window to "Accept”, “Close” (reject), or “Accept & Run.”



- The “Accept” option will accept Copilot’s code changes marked in green and may prompt the window again to fix other code related to the same data leakage.
- The “Close” option will close the Copilot window and restore your code.
- The “Accept & Run” option will accept Copilot’s code changes marked in green and run the Jupyter Notebook cell where the data leakage resides.

Please be aware that while GitHub Copilot can provide helpful suggestions, it might occasionally generate incorrect or suboptimal code solutions. Always review its recommendations critically before applying them.