

Running pytest initially

```
(mlip) rajashreedahal@Rajashrees-Laptop lab7 % cd MLIP_Lab6
(mlip) rajashreedahal@Rajashrees-Laptop MLIP_Lab6 % pytest
===== test session starts =====
platform darwin -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: /Users/rajashreedahal/Desktop/CS594/lab7/MLIP_Lab6
collected 2 items

test_utility.py .F [100%]

===== FAILURES =====
test_data_split

feature_target_sample = (   furnishingstatus_furnished furnishingstatus_unfurnished
0                        True                        False
1                        False                       True, 0    13300000
1      12250000
Name: price, dtype: int64)

def test_data_split(feature_target_sample):
    return_tuple = data_split(*feature_target_sample)
    # TODO test if the length of return_tuple is 4
    > raise NotImplemented
E      TypeError: exceptions must derive from BaseException

test_utility.py:41: TypeError
===== short test summary info =====
FAILED test_utility.py::test_data_split - TypeError: exceptions must derive from BaseException
===== 1 failed, 1 passed in 9.53s =====
(mlip) rajashreedahal@Rajashrees-Laptop MLIP_Lab6 %
```

After adding assertion criteria:

```
===== 1 failed, 1 passed in 9.53s =====
(mlip) rajashreedahal@Rajashrees-Laptop MLIP_Lab6 % pytest
===== test session starts =====
platform darwin -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: /Users/rajashreedahal/Desktop/CS594/lab7/MLIP_Lab6
collected 2 items

test_utility.py .. [100%]

===== 2 passed in 0.68s =====
(mlip) rajashreedahal@Rajashrees-Laptop MLIP_Lab6 %
```

Showing jenkins output:

Jenkins

macOS | mliplab6 #3 Console [Jenkins] | mliplab6 #5 Console [Jenkins] | Pytest Test Implementation... | Facebook | CS594 - Google Drive | Lab7 - Google Docs

Dashboard > mliplab6 > #5

Status

Changes

Console Output

Edit Build Information

Delete build '#5'

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Console Output

Download Copy View as plain text

Started by user Rajashree Dahal
Obtained Jenkinsfile from git https://github.com/RajashreeDahal4/MLIP_Lab6.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/rajashreedahal/.jenkins/workspace/mliplab6
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /Users/rajashreedahal/.jenkins/workspace/mliplab6/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.mliplab6.url https://github.com/RajashreeDahal4/MLIP_Lab6.git # timeout=10
Fetching upstream changes from https://github.com/RajashreeDahal4/MLIP_Lab6.git
> git --version # timeout=10
> git --version # 'git version 2.39.3 (Apple Git-146)'
> git fetch --tags --force --progress -- https://github.com/RajashreeDahal4/MLIP_Lab6.git +refs/heads/*:refs/remotes/mliplab6/* # timeout=10
> git rev-parse refs/remotes/mliplab6/main^{commit} # timeout=10
Checking out Revision 89289f74bc6fea76b221411ae968d616df11a41c (refs/remotes/mliplab6/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 89289f74bc6fea76b221411ae968d616df11a41c # timeout=10
Commit message: "trying to fix issue with pytest output"
> git rev-list --no-walk ad51e739aec780882fcb2948664d988cbf0cfa36 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
In C or Java, we can compile our program in this step
In Python, we can build our package here or skip this step
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh

```
Test Step: We run testing tool like pytest here
===== test session starts =====
platform darwin -- Python 3.13.2, pytest-8.3.5, pluggy-1.5.0
rootdir: /Users/rajashreedahal/.jenkins/workspace/mliplab6
collected 2 items

test_utility.py ..                                     [100%]

===== 2 passed in 0.83s =====
pytest runned
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
In this step, we deploy our porject
[Pipeline] echo
Depending on the context, we may publish the project artifact or upload pickle files
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```