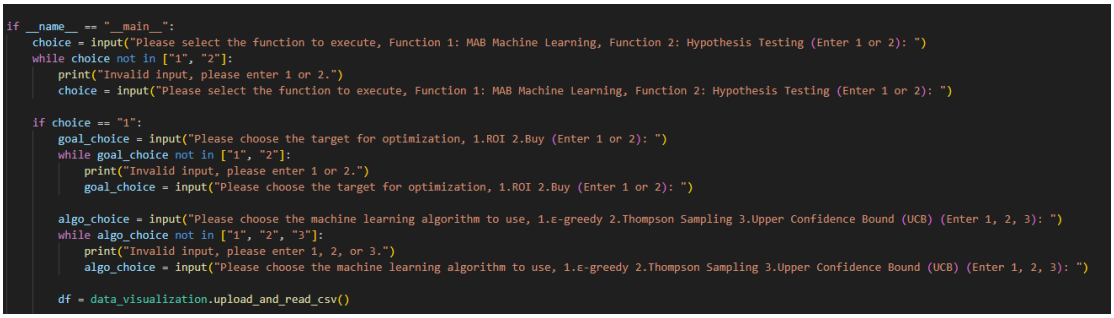
Clean Code Development

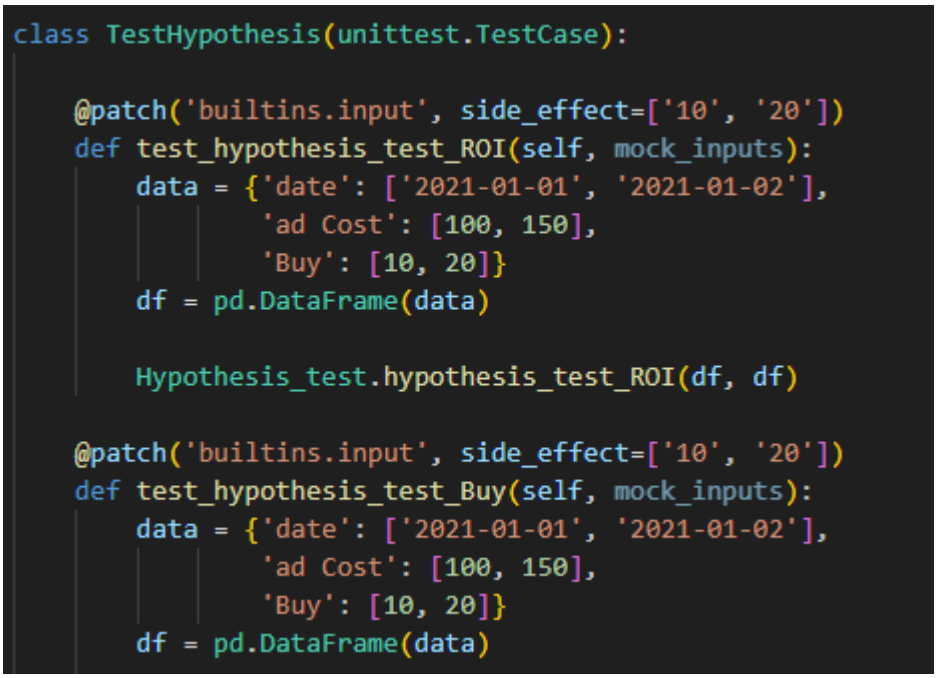
1. Meaningful variables



```
if __name__ == "__main__":
    choice = input("Please select the function to execute, Function 1: MAB Machine Learning, Function 2: Hypothesis Testing (Enter 1 or 2): ")
    while choice not in ["1", "2"]:
        print("Invalid input, please enter 1 or 2.")
        choice = input("Please select the function to execute, Function 1: MAB Machine Learning, Function 2: Hypothesis Testing (Enter 1 or 2): ")

    if choice == "1":
        goal_choice = input("Please choose the target for optimization, 1.ROI 2.Buy (Enter 1 or 2): ")
        while goal_choice not in ["1", "2"]:
            print("Invalid input, please enter 1 or 2.")
            goal_choice = input("Please choose the target for optimization, 1.ROI 2.Buy (Enter 1 or 2): ")

        algo_choice = input("Please choose the machine learning algorithm to use, 1.ε-greedy 2.Thompson Sampling 3.Upper Confidence Bound (UCB) (Enter 1, 2, 3): ")
        while algo_choice not in ["1", "2", "3"]:
            print("Invalid input, please enter 1, 2, or 3.")
            algo_choice = input("Please choose the machine learning algorithm to use, 1.ε-greedy 2.Thompson Sampling 3.Upper Confidence Bound (UCB) (Enter 1, 2, 3): ")

        df = data_visualization.upload_and_read_csv()
```

Meaningful variables help me to understand how the code works

2. Write some unitest



```
class TestHypothesis(unittest.TestCase):

    @patch('builtins.input', side_effect=['10', '20'])
    def test_hypothesis_test_ROI(self, mock_inputs):
        data = {'date': ['2021-01-01', '2021-01-02'],
                'ad Cost': [100, 150],
                'Buy': [10, 20]}
        df = pd.DataFrame(data)

        Hypothesis_test.hypothesis_test_ROI(df, df)

    @patch('builtins.input', side_effect=['10', '20'])
    def test_hypothesis_test_Buy(self, mock_inputs):
        data = {'date': ['2021-01-01', '2021-01-02'],
                'ad Cost': [100, 150],
                'Buy': [10, 20]}
        df = pd.DataFrame(data)
```

Writing unit tests ensures that small parts of the application, such as functions or methods, are tested individually to guarantee correct behavior.

3. Write some clear comments: Clear comments help clarify where explanatory variables are not self-evident.
4. Remove dead code: Removing dead code helps prevent the application from opening incorrect files, thus avoiding errors in the application.
5. Use Metrics: Metrics can help me detect code quality and performance, as well as identify potential security vulnerabilities.
6. Construct Build Management: Build Management can help me deploy CI/CD, allowing me to automate the detection of any issues with the code.