



PROJECT 2: OPTIMIZE SOMETHING



Table of Contents

- Overview
- About the Project
- Your Implementation
- Contents of Report
- Testing Recommendations

Submission Requirements

Grading Information

Development Guidelines

Optional Resources

REVISIONS

This assignment is subject to change up until 3 weeks prior to the due date. We do not anticipate changes; any changes will be logged in this section.

- May 19, 2022
 - Added the images directory to the provided starter code.

1. OVERVIEW

In this project, you will write software that evaluates and prepares portfolio metrics. You will submit the code for the project to Gradescope SUBMISSION. You will also submit to Canvas a chart as a 1-page report that compares two normalized portfolios.

1.1 Learning Objectives

The specific learning objectives for this assignment are focused on the following areas:

- **Portfolio Analysis Metrics:** Implement several metrics that are used in assessing the performance of a portfolio.
- **Optimization:** Learn how to use the SciPy optimization function to determine the stock allocations in a portfolio that maximize a specific portfolio metric.
- **Programming:** Each assignment will build upon one another. The techniques around portfolios, metrics, and optimization will play important roles in future projects.
- **Course Conduct:** Developing and testing code locally in the local Conda ml4t environment, submitting it for pre-validation in the Gradescope TESTING environment, and submitting it for grading in the Gradescope SUBMISSION environment.

2. ABOUT THE PROJECT

Revise the optimization.py code to return several portfolio statistics: stock allocations (allocs), cumulative return (cr), average daily return (adr), standard deviation of daily returns (sddr), and Sharpe ratio (sr). This project builds upon what you learned about portfolio performance metrics and optimizers to optimize a portfolio. That means that you will find how much of a portfolio's funds should be allocated to each stock to optimize its performance. While a portfolio can be optimized for many different metrics, in this version of the assignment we will maximize the Sharpe Ratio. You will also use the data cleansing techniques introduced in the course lectures.

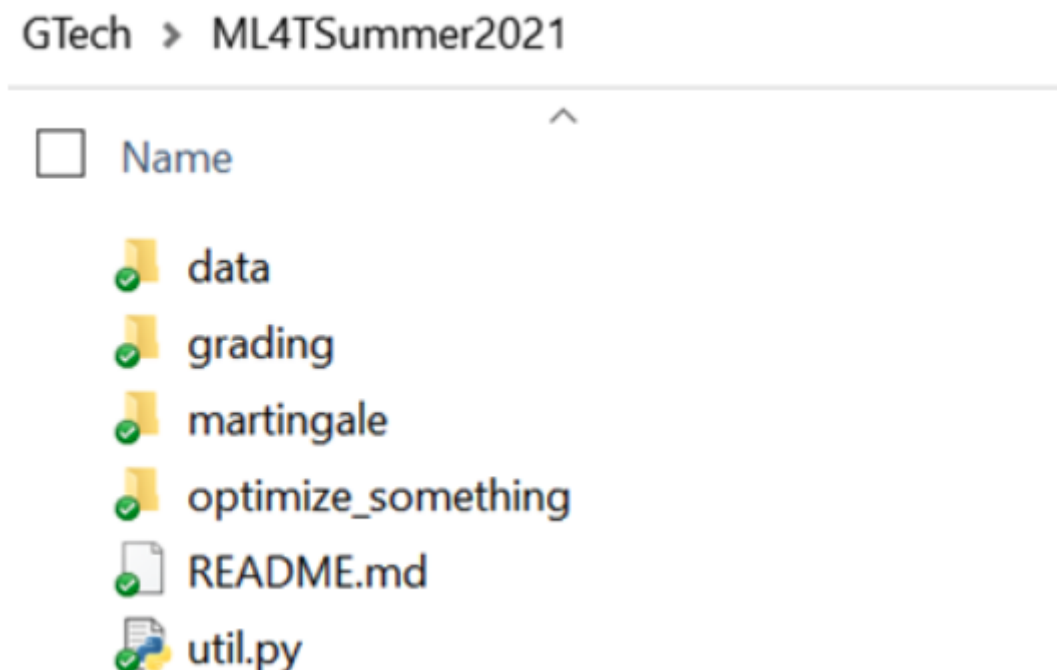
3. YOUR IMPLEMENTATION

Revise the code to implement a Python function named `optimize_portfolio()` in the file `optimization.py` that can find the optimal allocations for a given set of stocks. You should optimize for maximum Sharpe Ratio taking long positions only. Once the code is complete, include your chart in a report. Before the deadline, make sure to pre-validate your submission using Gradescope TESTING. Once you are satisfied with the results in testing, submit the code to Gradescope SUBMISSION. Only code submitted to Gradescope SUBMISSION will be graded. **If you submit your code to Gradescope TESTING and have not also submitted your code to Gradescope SUBMISSION, you will receive a zero (0).**

3.1 Getting Started

To make it easier to get started on the project and focus on the concepts involved, you will be given a starter framework. This framework assumes you have already set up the [local environment](#) and [ML4T Software](#). The framework for Project 2 can be obtained from: [Optimize_Something_2022Summer.zip](#).

Extract its contents into the base directory (e.g., `ML4T_2022Summer`). This will add a new folder called “`optimize_something`” to the directory structure.



Within the `optimize_something` folder are two files:

- `optimization.py`

- `grade_optimization.py`

You will modify the `optimization.py` file to implement the necessary functionality for this assignment. You must implement the `optimize_portfolio()` function. The existing code in the `optimization.py` file is best thought of as “stub” or “starter” code that may contain ideas for functions and methods that might be used in your implementation. This file must remain and run from within the `optimize_something` directory using the following command:

```
1 PYTHONPATH=../:. python optimization.py
```

`optimize_something_command` hosted with ❤ by GitHub

[view raw](#)

The `grade_optimization.py` file is discussed in the Testing Recommendations section below.

Note: You can leverage the functions in the now deprecated (no longer assigned) [assess portfolio](#) project that assessed the value of a portfolio with a given set of allocations. This is given only as a potential tool to use while completing the project to ensure you have your portfolio statistic calculations correct. Note that this code was written for Python 2.7 and may need to be updated to run with Python 3.6.

3.2 Implement the `optimize_portfolio` function

The function should accept as input a list of symbols as well as start and end dates and **return a list of floats as a one-dimensional Numpy array** that represents the allocations to each of the equities. You can take advantage of routines developed in the optional `assess portfolio` (see note under Starter Code) project to compute daily portfolio value and statistics, and then by using cut-and-paste to move the code from the project for the necessary functions into `optimization.py`. Otherwise, you will need to create the functions within `optimization.py`.

You are given the following inputs for optimizing a portfolio:

- A date range to select the historical data to use (specified by a start and end date)
- Symbols for equities (e.g., GOOG, AAPL, GLD, XOM). Note: You should support any symbol in the data directory. Your code should support any number of assets ≥ 2 . Do not hardcode a specific number (e.g., 4) as the number of assets.

Your goal is to find allocations to the symbols that optimize the criterion given above. Assume 252 trading days in a year and a risk-free return of 0.0 per day.

You will produce a single chart, as a .png file called "Figure1.png", comparing the optimal normalized portfolio with SPY using the following portfolio parameters:

- Start Date: 2008-06-01, End Date: 2009-06-01, Symbols: ['IBM', 'X', 'GLD', 'JPM'].

While four (4) symbols are used in the portfolio associated with Figure 1, the implementation must be robust enough to handle any number of symbols.

The chart must be correct, be properly titled, have appropriate axis labels, use consistent axis ranges, and have legends. You should use python's Matplotlib library.

The implementation will be evaluated using 10 test cases that will validate that the sum of the allocations to 1.0 (with a ± 0.02 tolerance), the individual allocations range from 0.0 to 1.0 (with a ± 0.02 tolerance), and that the standard deviation is correct (within 5% of the reference implementation).

3.2.1 Example

```
1 import datetime as dt
2 allocs, cr, adr, sddr, sr =
3     optimize_portfolio(sd=datetime(2008,1,1), ed=datetime(2009,1,1),
4     syms=['GOOG', 'AAPL', 'GLD', 'XOM'], gen_plot=False)
```

optimize_portfolio_snippet hosted with ❤ by GitHub

[view raw](#)

Where the returned output is:

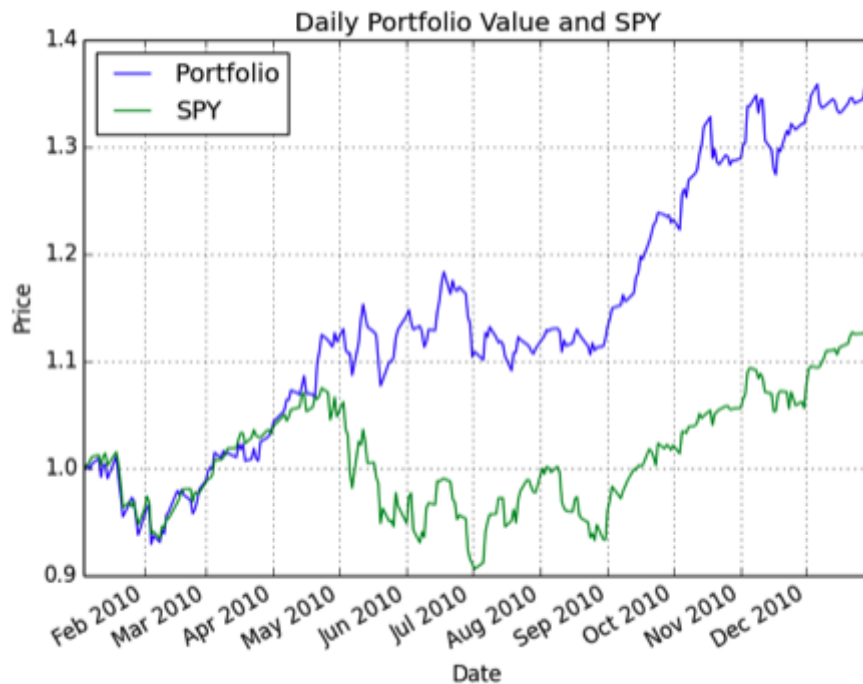
- allocs: A 1-d NumPy NDAarray of allocations to the stocks. All the allocations must be between 0.0 and 1.0 and they must sum to 1.0. Some of the allocations may be 0.
- cr: Cumulative return
- adr: Average daily return
- sddr: Standard deviation of daily return
- sr: Sharpe ratio

The input parameters are:

- sd: A DateTime object that represents the start date
- ed: A DateTime object that represents the end date

- `syms`: A list of symbols that make up the portfolio (note that your code should support any symbol in the data directory)
- `gen_plot`: When `gen_plot=True` the implementation must generate and save the chart (as a .png file) that is found in your report. The Autograder will call your code with `gen_plot=False` (when conducting the performance tests) and `gen_plot=True` (when generating the chart found in your report).

An example chart that you might create for assessing your optimizer.



3.2.2 Suggestions

- Refer to comments in the provided helper code for pointers regarding how to implement the SciPy minimize function. In order to specify bounds and constraints when using the `scipy.optimize` module, you'll need to use a special syntax explained here: <http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>
- For bounds, you simply need to pass in a sequence of 2-tuples (`<low>`, `<high>`). Just remember that you need to supply as many tuples as the number of stocks in your portfolio.
- For constraints, it's a little tricky. You need to pass in a sequence of dicts (dictionaries), one dictionary per constraint. Each dictionary must specify the type of constraint (`'eq'` for equality, or `'ineq'` for inequality), and a function that returns 0 only when the input satisfies the constraint (this is the same input that is supplied to your evaluation function). E.g. to constrain the sum of all

values in the input array to be less than 50, you could pass in the following (lambdas are just anonymous functions defined on-the-spot):

```
constraints = ({ 'type': 'ineq', 'fun': lambda inputs: 50.0 - np.sum(inputs) })
```

- Use a uniform allocation of $1/n$ to each of the n assets as your initial guess.

3.3 Technical Requirements

The following technical requirements apply to this assignment

1. The optimization.py file must implement this [API specification](#) (including the default arguments).
2. Use only the functions provided in the util.py file for reading historical stock data provided in the ../data folder. Do NOT modify, copy, or move the util.py file or its functions.
3. Your code must run in less than 5 seconds per test case in the university-provided Gradescope SUBMISSION environment.
4. A random seed may not be set for this assignment.
5. Watermarked Charts (i.e., where the GT Username appears over the lines) can be shared in the designated pinned (e.g., "Project 2- Student Charts") thread alone. Charts presented in reports or submitted for grading must not contain watermarks.
6. You must never directly use (or indirectly use via a util.py function) Matplotlib's plot.show function in the Gradescope environment. While you may test locally with this function, a deduction will be applied if it is used in the code submitted for grading.

4 CONTENTS OF THE REPORT

In addition to submitting your code to Gradescope, you will also produce a report that will consist of a single page that contains the chart generated by your implementation. The charts must be generated by the code and saved by the code to a file. The chart can be imported (without additional post-processing) into the report document. The chart will be evaluated on its correctness (i.e., shape, scale, range, normalized lines, current number of lines) and textual completeness (i.e., appropriate title, legends, axis labels). All textual elements must be readable and non-overlapping.

5 TESTING RECOMMENDATIONS

To test your code, we will call the `optimize_portfolio()` function only. You are encouraged to perform any tests necessary to build confidence that the code will run properly when submitted for grading and will produce the required results. To run and test that the file will run from within the `optimize_something` directory, use the command given below at the command prompt from within the `optimize_something` directory:


```
1 PYTHONPATH=../:. python optimization.py
```

optimization_script hosted with  by GitHub

[view raw](#)

Additionally, we provide the `grade_optimization.py` file that can be used for lightweight testing. This local grading/pre-validation script is the same script that will be run when the code is submitted to Gradescope TESTING. To run and test that the file will run from within the `optimize_something` directory, use the command:

```
1 PYTHONPATH=../:. python grade_optimization.py
```

grade_optimization_script hosted with  by GitHub

[view raw](#)

In addition to testing on your local machine, you are encouraged to submit your files to Gradescope TESTING, where some basic pre-validation tests will be performed against the code. No credit will be given for coding assignments that do not pass this pre-validation. While you may see a result like the image below, **Gradescope TESTING does not grade your assignment.** The Gradescope TESTING script is not a complete test suite and does not match the more stringent private grader that is used in Gradescope SUBMISSION. Thus, the maximum Gradescope TESTING score of 64, while instructional, does not represent the minimum score one can expect when the assignment is graded using the private grading script. You are encouraged to develop additional tests to ensure that all project requirements are met.

You are allowed **unlimited** resubmissions to Gradescope **TESTING**. Please refer to the [Gradescope Instructions](#) for more information.

Check submitted files (1.0/1.0)

All required files submitted!

all tests (1.0/1.0)

```


```
--- Summary ---
Tests passed: 8 out of 8

--- Details ---
Test #0: passed
Test #1: passed
Test #2: passed
Test #3: passed
Test #4: passed
Test #5: passed
Test #6: passed
Test #7: passed

```


```

AUTOGRADER SCORE
64.0 / 64.0

PASSED TESTS
Check submitted files (1.0/1.0)
all tests (1.0/1.0)

6 SUBMISSION REQUIREMENTS

This is an individual assignment. All work you submit should be your own. Make sure to cite any sources you reference and use quotes and in-line citations to mark any direct quotes.

Assignment due dates in your time zone can be found by looking at the Project in the Assignment menu item in Canvas (ensure your Canvas time zone settings are set up properly). This date is 23:59 AOE converted to your time zone. Late submissions are allowed for a penalty. The times and penalties are as follows:

- -10% Late Penalty: +1 Hour late: submitted by 00:59 AOE (next day)
- -25% Late Penalty: +12 Hours Late: submitted by 11:59 AOE (next day)
- -50% Late Penalty: +24 Hours Late: submitted by 23:59 AOE (next day)
- -100% Late Penalty: > 24+ Late: submitted after 23:59 AOE (next day)

Assignments received after Monday at 23:59 AOE (even if only by a few seconds) are not accepted without advanced agreement except in cases of medical or family emergencies. In the case of such an emergency, please contact the [Dean of Students](#).

6.1 Report Submission

Complete your report using the [JDF](#) format, then save your submission as a PDF. Name your report **p2_optimization_report.pdf**. Reports should be submitted to the corresponding assignment submission page in Canvas. You should submit a single PDF for this assignment. Please submit the following file to Canvas in PDF format only:

p2_optimization_report.pdf

Do not submit any other files. The chart must be included in the report, not submitted as a separate file. Also note that when we run your submitted code, it should generate the chart. Not submitting a report will result in a penalty.

Project 2: Optimize Something (Report)
Project 2

Jun 7 by 9am

Your submission must contain 20 words or more. → !

After submission, you may see a message like the one below. This is okay, as your project 2 report submission is retained within Canvas and this warning can be ignored.

You are allowed unlimited submissions of the **p2_optimization_report.pdf** file to **Canvas**.

6.2 Code Submission

This class uses Gradescope, a server-side auto-grader, to evaluate your code submission. No credit will be given for code that does not run in this environment and students are encouraged to leverage Gradescope TESTING prior to submitting an assignment for grading. **Only code submitted to Gradescope SUBMISSION will be graded. If you submit your code to Gradescope TESTING and have not also submitted your code to Gradescope SUBMISSION, you will receive a zero (0).**

Please submit the following file to Gradescope **SUBMISSION**:

optimization.py

Do not submit any other files.

You will see a message like this (immediately below) if your code was correctly submitted to Gradescope SUBMISSION.

The screenshot displays the 'Autograder Results' interface. It features a 'Results' tab and a 'Code' tab. The 'Results' tab shows a score of 0.0 out of 1.0. Below the score, there is a list of 'PASSED TESTS' including 'grader executed test (1.0/1.0)', 'Check submitted files (1.0/1.0)', and 'Check submission number (1.0/1.0)'. The 'Code' tab shows the submitted code.

Important: You are allowed a MAXIMUM of three (3) code submissions to Gradescope SUBMISSION.

7 GRADING INFORMATION

Your report is worth 20% of your grade. As such, it will be graded on a 20-point scale coinciding with a rubric design to mirror the criteria stated above. The submitted code

(which is worth 80% of your grade) is run as a batch job after the project deadline. The code will be graded using a rubric design to mirror the implementation details above, where each of the 10 implementation test cases is worth 8 points. Deductions will be applied for unmet implementation requirements or code that fails to run.

We do not provide an explicit set timeline for returning grades, except that all assignments and exams will be graded before the institute deadline (end of the term). As will be the case throughout the term, the grading team will work as quickly as possible to provide project feedback and grades.

Once grades are released, any grade-related matters must follow the [Assignment Follow-Up guidelines and process](#) alone. Regrading will only be undertaken in cases where there has been a genuine error or misunderstanding. Please note that requests will be denied if they are not submitted using the Summer 2022 form or do not fall within the timeframes specified on the [Assignment Follow-Up](#) page.

7.1 Grading Rubric

7.1.1 Report [20 Points]

Each chart line (i.e., SPY, Portfolio) is worth 10 points each. Deductions will be applied if any of the following occur:

- If no chart is provided or the chart is total nonsense. (-20 points).
- If a plot is entirely incorrect (e.g., wrong shape, wrong time period) (-10 points).
- If a plot is partly incorrect (e.g., chart begins correctly but midway or after is incorrect) (-5 points).
- 1) If the chart is not normalized (-10 points) **XOR** 2) If the chart data scale is substantially wrong (e.g., the right shape that is normalized to 1 but the max/end values are wrong – for example, 3.5 instead of 2.5). (-5 points)
- If the chart is missing a required data series (i.e., the chart is missing the portfolio line or the SPY line) (-10 points per missing data series).
- If the chart is missing labels, text, or a legend; or if the labels, text, or legend are unreadable (e.g., too small, labels overlap a lot). (-2 points per instance).

The report score range is from a minimum of zero (0) to 20 points.

7.1.2 Code

Code deductions will be applied if any of the following occur:

- If the code does not produce appropriate charts that are saved as .png files. (-10 points)
- If the code displays any charts in a window or screen. (-10 points)
- If the code saves in a directory outside the project directory. (up to a max of -20 points)

7.1.3 Auto-Grader (Private Grading Script) [80 Points]

We will test your code against 10 cases (8 points per case). Each case will be deemed "correct" if:

- $\text{sum}(\text{allocations}) = 1.0 \pm 0.02$ (+2 points)
- Each allocation is between 0 and 1.0 ± 0.02 (negative allocations are allowed if they are very small) (+2 points)
- Standard deviation of daily return of the allocated portfolio is within 5% of our reference solution or lower (+4 points)

There is no partial credit for the per-test-case points breakdown above if outside the threshold. If you are within the threshold, you get the point allocation, outside the threshold you get 0. The total score is additive, with a minimum score of zero (0) and a maximum score of 80.

8 DEVELOPMENT GUIDELINES (ALLOWED & PROHIBITED)

See the [Course Development Recommendations, Guidelines, and Rules](#) for the complete list of requirements applicable to all course assignments. **The Project Technical Requirements are grouped into three sections: Always Allowed, Prohibited with Some Exceptions, and Always Prohibited.**

The following exemptions to the Development Recommendations, Guidelines, and Rules apply to this project:

- Watermarked charts may be shared in the dedicated discussion forum thread alone.

9 OPTIONAL RESOURCES

Although the use of these or other resources is not required; some may find them useful in completing the project or in providing an in-depth discussion of the material.

Sharpe (1994), [The Sharpe Ratio](#)

[\(Deprecated\) Assess Portfolio Files](#)

SciPy.org, [scipy.optimize.minimize](#)

TutorialsPoint.com, [What are default arguments in Python?](#)

Hilpisch (1st or 2nd edition), [Python for Finance](#) (Chapter 9 – Mathematical Tools)

Martelli, A. Ravenscroft, and S. Holden (2017), [Python in a Nutshell, 3rd Edition](#)