CS7646 ML4T
Machine Learning
for Trading

PROJECT 6: INDICATOR EVALUATION

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.

1 OVERVIEW

In this project, you will develop technical indicators and a Theoretically Optimal Strategy that will be the ground layer of a later project (i.e., project 8). The technical indicators you develop here will be utilized in your later project to devise an intuition-based trading strategy and a Machine Learning based trading strategy. The Theoretically Optimal Strategy will give a baseline to gauge your later project's performance. We hope Machine Learning will do better than your intuition, but who knows? You will submit the code for the project to Gradescope SUBMISSION. The report will be submitted to Canvas.

1.1 Learning Objectives

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

- **Indicators**: You will develop an understanding of various trading indicators and how they might be used to generate trading signals.
- **Optimal Strategy**: You will also develop an understanding of the upper bounds (or maximum) amount that can be earned through trading given a specific instrument and timeframe.

2 ABOUT THE PROJECT

Please keep in mind that the completion of this project is pivotal to Project 8 completion. The indicators that are selected here cannot be replaced in Project 8. We encourage spending time finding and researching indicators, including examining how they might later be combined to form trading strategies. Considering how multiple indicators might work together during Project 6 will help you complete the later project.

3 YOUR IMPLEMENTATION

This project has two main components: First, you will develop a theoretically optimal strategy (TOS), which represents the maximum amount your portfolio can theoretically return. Note that this strategy does not use any indicators. Second, you will research and identify five market indicators.

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).**

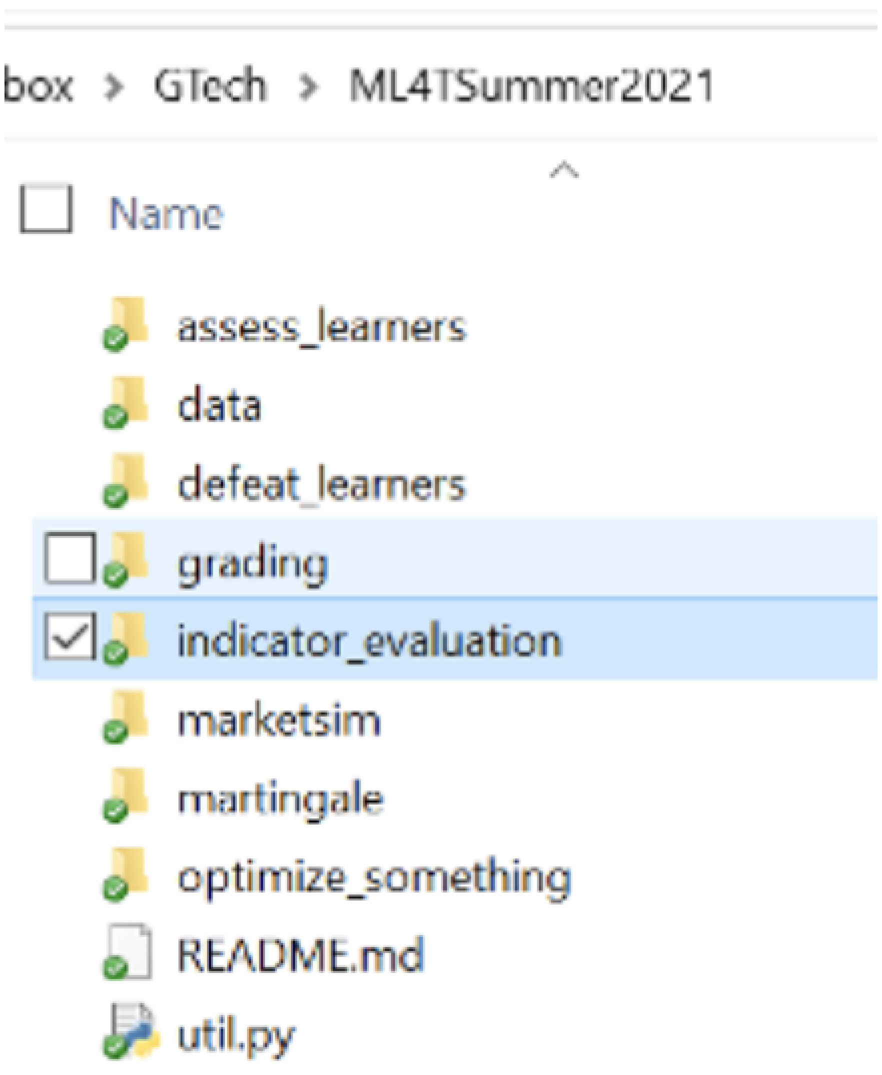
3.1 Getting Started

This framework assumes you have already set up the local environment and ML4T Software. There is no distributed template for this project.

You will have access to the ML4T/Data directory data, but you should use ONLY the API functions in util.py to read it. You may create a new folder called indicator_evaluation to contain your code for this project.

You should create the following code files for submission. They should contain ALL code from you that is necessary to run your evaluations.

- **indicators.py**
  Code implementing your indicators as functions that operate on DataFrames. There is no defined API for indicators.py, but when it runs, the main method should generate the charts that will illustrate your indicators in the report.
- **marketsimcode.py**
  An improved version of your marketsim code accepts a "trades" DataFrame (instead of a file). More info on the trades data frame is below. It is OK not to submit this file if you have subsumed its functionality into one of your other required code files. *This file has a different name and a slightly different setup than your previous project. However, that solution can be used with several edits for the new requirements.*
- **TheoreticallyOptimalStrategy.py**
  Code implementing a TheoreticallyOptimalStrategy (details below). It should implement testPolicy(), which returns a trades data frame (see below).
- **testproject.py**
  This file should be considered the entry point to the project. The *if "__name__" == "__main__":* section of the code will call the testPolicy function in TheoreticallyOptimalStrategy, as well as your indicators and marketsimcode as needed, to generate the plots and statistics for your report (more details below).



3.2 Data, Details, Dates & Rules

For both sections:

- Use only the data provided for this course. You are not allowed to import external data.
- Add an author() function to each file.
- For your report, use only the symbol JPM.
- Use the time period January 1, 2008, to December 31, 2009.
- Starting cash is $100,000.

Theoretically Optimal Strategy only:

- Allowable positions are 1000 shares long, 1000 shares short, 0 shares. (You may trade up to 2000 shares at a time as long as your positions are 1000 shares long or 1000 shares short.)
- Benchmark: The performance of a portfolio starting with $100,000 cash, investing in 1000 shares of JPM, and holding that position.

- Transaction costs for TheoreticallyOptimalStrategy:
    - Commission: $0.00
    - Impact: 0.00.

3.3 Implement Part 1: Theoretically Optimal Strategy

In the Theoretically Optimal Strategy, assume that you can see the future. You are constrained by the portfolio size and order limits as specified above. Create a set of trades representing the best a strategy could possibly do during the in-sample period using JPM. We have you do this to have an idea of an upper bound on performance, which can be referenced in Project 8.

Note: The Theoretically Optimal Strategy does not use the indicators developed in the next section.

Use a revised market simulator based on the one you wrote earlier in the course to determine the portfolio valuation. For this activity, use $0.00 and 0.0 for commissions and impact, respectively.

Provide a chart that reports:

- Benchmark (see definition above) normalized to 1.0 at the start: Plot as a **purple line**.
- Value of the theoretically optimal portfolio (normalized to 1.0 at the start): Plot as a **red line**.

You should also create a table in your report that shows to 6 digits to the right of the decimal:

- Cumulative return of the benchmark and portfolio
- Stdev of daily returns of benchmark and portfolio
- Mean of daily returns of benchmark and portfolio

Your TOS should implement a function called ‘testPolicy()’ as follows:

```
testPolicy(symbol=”AAPL”, sd=dt.datetime(2010, 1, 1), ed=dt.datetime(2011,12,31),
```

The input parameters are:

- symbol: the stock symbol to act on
- sd: A DateTime object that represents the start date
- ed: A DateTime object that represents the end date
- sv: Start value of the portfolio

The output result is:

- df_trades: A single column data frame, indexed by date, whose values represent trades for each trading day (from the start date to the end date of a given period). Legal values are +1000.0 indicating a BUY of 1000 shares, -1000.0 indicating a SELL of 1000 shares, and 0.0 indicating NOTHING. Values of +2000 and -2000 for trades are also legal so long as net holdings are constrained to -1000, 0, and 1000. *Note: The format of this data frame differs from the one developed in a prior project*.

Your testproject.py code should call testPolicy() as a function within TheoreticallyOptimalStrategy as follows:

```python
import TheoreticallyOptimalStrategy as tos
df_trades = tos.testPolicy(symbol = "JPM", sd=dt.datetime(2008, 1, 1), ed=dt.date
```

tos_example.py hosted with ❤ by **GitHub** **view raw**

The df_trades result can be used with your market simulation code to generate the necessary statistics.

3.4 Implement Part 2: Technical Indicators

Develop and describe 5 technical indicators. You may find our lecture on time series processing, the Technical Analysis video, and the vectorize_me PowerPoint to be helpful. For each indicator, you should create a single, compelling chart (with proper title, legend, and axis labels) that illustrates the indicator (you can use sub-plots to showcase different aspects of the indicator).

For example, you might create a chart showing the stock's price history, along with "helper data" (such as upper and lower Bollinger Bands) and the value of the indicator itself. Another example: If you were using price/SMA as an indicator, you would want to create a chart with 3 lines: Price, SMA, Price/SMA. To facilitate visualization of the indicator, you might normalize the data to 1.0 at the start of the date range (i.e., divide price[t] by price[0]).

You must pick at least 3 indicators from the list below. These indicators have been shown to have good results in project 8. The other 2 indicators may be from the list as well, or any that you are familiar with or find online (see the optional resources for suggested sources). All of these indicators should be researched for details on their implementation and effectiveness. Be sure to research the different window sizes and parameters, this will make Project 8 less daunting.

- Bollinger Bands
- Moving Averages and their derivatives; this includes SMA, EMA, WMA, DEMA, TEMA
- RSI
- Momentum or Rate of Change (only one of these two can be used).
- Stochastic Indicator
- Percentage Price Indicator
- CCI
- MACD

You may only pick two from the indicators given in the class resources of RSI, SMA, and Bollinger Bands.

There is no defined API that must be adhered to in indicators.py. You can define the functions and their parameters so that your code is correct and optimized. Hint: We recommend either creating a run() method in indicators.py that will run all indicators and create graphs or subsuming that logic into testproject.py.

In your report (described below), a description of each indicator should enable someone to reproduce it just by reading the description. We want a written detailed description here, not code. However, it is OK to augment your written description with a **pseudocode** figure. **Do NOT copy/paste code parts here as a description**.

You should have already successfully coded the Bollinger Band feature:

```
bb_value[t] = (price[t] - SMA[t])/(2 * stdev[t])
```

bb_value.py hosted with ❤ by **GitHub** **view raw**

Another good indicator worth considering is momentum.

```
momentum[t] = (price[t]/price[t-N]) - 1
```

momentum.py hosted with ❤ by **GitHub** **view raw**

Use util.py to read any of the columns in the stock symbol files.

It is usually worthwhile to standardize the resulting values (see Standard Score).

The indicators should return results that can be interpreted as actionable buy/sell signals. For example, Bollinger Bands alone does not give an actionable signal to buy/sell easily framed for a learner, but BBP (or %B) does.

3.4.1 Planning Ahead

In Project 8, you will need to use the same indicators you will choose in this project. **You will not be able to switch indicators in Project 8**. Some indicators are built using other indicators and/or return multiple results vectors (e.g., MACD uses EMA and returns MACD and Signal vectors). While such indicators are okay to use in Project 6, please keep in mind that Project 8 will require that each indicator return one results vector. In this case, MACD would need to be modified for Project 8 to return your own custom results vector that somehow combines the MACD and Signal vectors, or it would need to be modified to return only one of those vectors. If you use an indicator in Project 6 that returns multiple results vectors, we recommend taking an additional step of determining how you might modify the indicator to return one results vector for use in Project 8. **While Project 6 doesn't need to code the indicators this way, it is required for Project 8**.

You may not use stand-alone indicators with different parameters in Project 8 (e.g., SMA(5) and SMA(30)). This means someone who wants to implement a strategy that uses different values for an indicator (e.g., a Golden Cross that uses two SMA calls with different parameters) will need to create a Golden_Cross indicator that returns a single results vector, but internally the indicator can use two SMA calls with different parameters). This Golden_Cross indicator would need to be defined in Project 6 to be used in Project 8.

Indicator selection in project 8 is limited to the indicators explicitly identified and researched in Project 6, with few exceptions. Note that an indicator like MACD uses EMA as part of its computation. If you want to use EMA in addition to using MACD, then EMA would need to be explicitly identified as one of the five indicators.

3.5 Part 3: Implement author() function (deduction if not implemented)

You should implement a function called author() that returns your Georgia Tech user ID as a string in each .py file. This is the ID you use to log into Canvas. It is not your 9 digit student number. Here is an example of how you might implement author():

```
def author():
  return 'tb34' # replace tb34 with your Georgia Tech username.
```

author_example.py hosted with ❤ by **GitHub** **view raw**

Implementing this method correctly does not provide any points, but there will be a penalty for not implementing it.

3.6 Part 4: Implement Test Project

Create testproject.py and implement the necessary calls (following each respective API) to indicators.py and TheoreticallyOptimalStrategy.py, with the appropriate parameters to run everything needed for the report in a single Python call. You may also want to call your market simulation code to compute statistics. The file will be invoked using the command:

```
PYTHONPATH=../:. python testproject.py
```

testproject_call hosted with ❤ by **GitHub** **view raw**

This is to have a single–entry point to test your code against the report. Calling testproject.py should run all assigned tasks and output all necessary charts and statistics for your report.

3.7 Technical Requirements

The following technical requirements apply to this assignment

1. Use only the functions in util.py to read in stock data. Only use the API methods provided in that file. Please note that util.py is considered part of the environment and should not be moved, modified, or copied. For grading, we will use our own unmodified version.
2. An indicator can only be used once with a specific value (e.g., SMA(12)). If you need to use multiple values, consider creating a custom indicator (e.g., my_SMA(12,50), which internally uses SMA(12) and SMA(50) before returning a single results vector).
3. The implementation may optionally write text, statistics, and/or tables to a single file named p6_results.txt or p6_results.html.
4. The Sharpe ratio uses the **sample standard deviation**.

3.8 Hints & Recommendations

- The "secret" regarding leverage and a "secret date" discussed in the YouTube lecture do not apply and should be ignored.

4 CONTENTS OF REPORT

In addition to submitting your code to Gradescope, you will also produce a report. Your report should use JDF format and has a maximum of 10 pages. Any content beyond 10 pages will not be considered for a grade. Ten pages is a maximum, not a target; our recommended per-section lengths intentionally add to less than 10 pages to leave you room to decide where to delve into more detail. This length is intentionally set, expecting that your submission will include diagrams, drawings, pictures, etc. These should be incorporated into the body of the paper unless specifically required to be included in an appendix.

The JDF format specifies font sizes and margins, which should not be altered. Include charts to support each of your answers. Charts should also be generated by the code and saved to files. Charts should be properly annotated with legible and appropriately named labels, titles, and legends.

Please address each of these points/questions in your report. The report is to be submitted as **p6_indicatorsTOS_report.pdf**.

4.1 Indicators: ~ 5 pages (estimated)

At a minimum, address each of the following for each indicator:

- Introduce and describe each indicator you use in sufficient detail that someone else could reproduce it. (The indicator can be described as a mathematical equation or as pseudo-code).
- Provide a compelling description regarding why that indicator might work and how it could be used. Be sure to describe how they create buy and sell signals (i.e., explain how the indicator could be used alone and/or in conjunction with other indicators to generate buy/sell signals).
- Provide one or more charts that convey how each indicator works compellingly. (up to 3 charts per indicator).

The total number of charts for Part 1 must not exceed 10 charts.

4.2 Theoretically Optimal Strategy (TOS) ~ 1.5 pages (estimated)

For the Theoretically Optimal Strategy, at a minimum, address each of the following:

- Describe how you created the strategy and any assumptions you had to make to make it work.

- Describe the strategy in a way that someone else could evaluate and/or implement it.
- Provide a chart that illustrates the TOS performance versus the benchmark.
- Provide a table that documents the benchmark and TOS performance metrics. These metrics should include cumulative returns, the standard deviation of daily returns, and the mean of daily returns for both the benchmark and portfolio. Performance metrics must include 6 digits to the right of the decimal point (e.g., 98.123456)

5 TESTING RECOMMENDATIONS

There is no locally provided grading / pre-validation script for this assignment. You are encouraged to perform any tests necessary to instill confidence in your implementation, ensure that the code will run properly when submitted for grading and that it will produce the required results.

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. **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, 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.

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. The report is to be submitted as **p6_indicatorsTOS_report.pdf**. Assignments 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(s) to Canvas in PDF format only:

p6_indicatorsTOS_report.pdf

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

You are allowed unlimited submissions of the **p6_indicatorsTOS_report.pdf** on 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 files to Gradescope **SUBMISSION**:

- **testproject.py**
- **indicators.py**
- **TheoreticallyOptimalStrategy.py**
- **marketsimcode.py** (optional file)

Do not submit any other files.

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

7 GRADING INFORMATION

Your report and code will be graded using a rubric design to mirror the questions above. Make sure to answer those questions in the report and ensure the code meets the project requirements. 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 [100 points + up to 2-point bonus]

The following adjustments will be applied to the report:

General

- If the report is not neat (up to -5 points).
- If the required report is not provided (-100 points)
- Bonus for exceptionally well-written reports (up to +2 points)

Indicators

- If there are not five different indicators (where you may only use two from the set discussed in the lectures [SMA, Bollinger Bands, RSI]) (-15 points each)
- If the submitted code in the indicators.py file does not properly reflect the indicators provided in the report (up to -75 points)
- Individual Indicators (up to 15 points potential deductions per indicator):
  - If there is not a compelling description of why the indicator might work (-5 points)

  - If the indicator is not described in sufficient detail that someone else could reproduce it (-5 points)
  - If there is not a chart for the indicator that properly illustrates its operation, including a properly labeled title, axis, and legend (up to -5 points)

Theoretically optimal (up to 20 points potential deductions):

- If the methodology described is not correct and convincing (-10 points)
- If the chart is not correct (dates and equity curve), including properly labeled axis and legend (up to -10 points)
- If the historical value of the benchmark is not normalized to 1.0 or is not plotted with a purple line (-5 points)
- If the historical value of the portfolio is not normalized to 1.0 or is not plotted with a red line (-5 points)
- If the reported performance criteria are incorrect (See the appropriate section in the instructions above for required statistics). (-2 points for each item)

7.1.2 Code

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

- If the required code is not provided, (including code to recreate the charts and usage of correct trades DataFrame) (up to -100 points)
- If all charts are not created and saved using Python code. (up to -100 points)
- If any charts are displayed to a screen/window/terminal in the Gradescope Submission environment. (up to -100 points) *Note: DO NOT use plt.show()*
- If the code saves in a directory outside the project directory. (up to a max of –100 points)

7.1.3 Autograder (Private Grading Script) [0 Points]

There is no auto-grader score associated with this project.

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 Course Development Recommendations, Guidelines, and Rules apply to this project:

- Watermarked charts may be shared in the dedicated discussion forum mega-thread alone.
- You may set a specific random seed for this assignment. If a specific random seed is used, it must only be called once within a test_code() function in the testproject.py file and it must use your GT ID as the numeric value.

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.

- Stockchart.com School (Technical Analysis Introduction)
- Stockchart.com Technical Indicators
- StockCharts has some good information on Trading Strategies and Models.
- Investopedia
- TA Ameritrade Technical Analysis Introduction Lessons (pick the ones you think are most useful)
- A good introduction to technical analysis
- 10 Indicators Every Trader Should Know
- Investopedia’s Introduction to Technical Analysis
- Technical Analysis of the Financial Markets by John Murphy.
- Technical Analysis Explained by Martin Pring.