CS7646 ML4T
Machine Learning
for Trading

PROJECT 6: INDICATOR EVALUATION

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.

ABOUT THE PROJECT

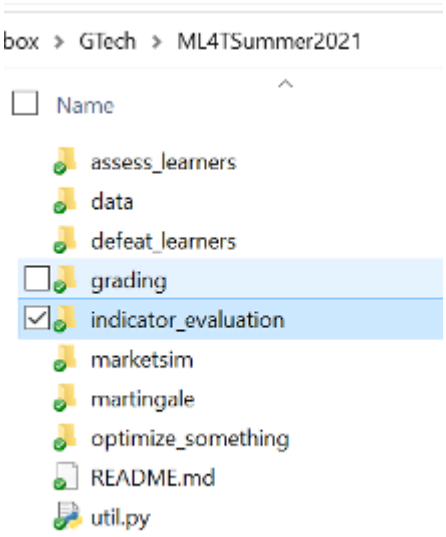
In this project, you will develop technical indicators and a Theoretically Optimal Strategy that will be the ground layer of a later project. The technical indicators you develop here will be utilized in your later project to devise an intuition-based trading strategy and a Machine Learning based trading strategy. 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 in Gradescope SUBMISSION. The report will be submitted to Canvas.

Please keep in mind that the completion of this project is pivotal to Project 8 completion. The indicators selected here cannot be replaced in Project 8. We encourage spending time finding and research 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.

STARTER CODE

Please note that there is no starting .zip file associated with this project. You should create a directory for your code in ml4t/indicator_evaluation. The directory structure should align with the course environment framework, as discussed on the local environment and ML4T Software pages.

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

- **indicators.py** Code implementing your indicators as functions that operate on DataFrames. The "main" method in indicators.py should generate the charts that 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 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).
- **README.txt** This is a text file that describes each .py file and provides instructions describing how to run your code.

DATA DETAILS, DATES & RULES

- 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.
- Allowable positions are 1000 shares long, 1000 shares short, 0 shares. (You may trade up to 2000 shares at a time as long as you maintain these holding requirements.)
- 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.

TASKS & REQUIREMENTS

Part 1: 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]).

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])
```

Another good indicator worth considering is momentum.

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

You can use util.py to read any of the columns in the stock symbol files.

It is usually worthwhile to standardize the resulting values (see https://en.wikipedia.org/wiki/Standard_score).

You are allowed to use up to two indicators presented and coded in the lectures (SMA, Bollinger Bands, RSI), but the other three will need to come from outside the class material (momentum is allowed to be used).

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.

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 an indicator in Project 8 unless it is explicitly identified in Project 6. 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.

Part 2: Theoretical 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: Theoretically Optimal Strategy does not use the indicators developed in the previous section. Some may find it useful to work on Part 2 of the assignment before beginning Part 1.

Use the 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 **green line.**
- Value of the theoretically optimal portfolio (normalized to 1.0 at the start): Plot as a **red line**

You should also report in your report:

- 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(2011,1
```

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:

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

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

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.
```

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

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 run:

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

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.

Part 5: Implement README.txt

You must also create a README.txt file that has**:**

- Description of what each python file is for/does.
- Explicit instructions on how to properly run your code.

TECHNICAL REQUIREMENTS

- Your project must be coded in Python 3.6. and run in the Gradescope SUBMISSION environment.
- When utilizing any example order files, the code must run in less than 10 seconds per test case.
- 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.
- 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).

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

Note: The Sharpe ratio uses the sample standard deviation.

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 **report.pdf**.

Part 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.

Part 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. The performance metrics should include cumulative returns, standard deviation of daily returns, and the mean of daily returns for both the benchmark and portfolio. Performance metrics must include 4 digits to the right of the decimal point (e.g., 98.1234)

TESTING

There is no locally provided grading / pre-validation script for this assignment. You are encouraged to perform any unit tests necessary to instill confidence in your implementation.

You are encouraged to submit your files to Gradescope TESTING, where some basic pre-validation tests will be performed against the code. Gradescope TESTING does not grade your assignment. No credit will be given for coding assignments that fail in Gradescope SUBMISSION and failed to pass this pre-validation in Gradescope TESTING.

You are allowed unlimited resubmissions to Gradescope TESTING. Please refer to the Gradescope Instructions for more information.

SUBMISSION INSTRUCTIONS

This is an individual assignment. All work you submit should be your own. Ensure to cite any sources you reference and use quotes and in-line citations to mark any direct quotes. Citations within the code should be captured as comments.

Late work is 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.

Report Submission

Complete your report using the JDF format, then save your submission as a PDF. The report is to be submitted as **report.pdf**. Assignments should be submitted to the corresponding assignment submission page in Canvas. You should submit a single PDF for the report portion of the assignment. Please submit the following file to Canvas in PDF format only:

- **report.pdf**

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

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

Code Submission

This class uses Gradescope, a server-side autograder, to evaluate your code submission. No credit will be given for code that does not run in the Gradescope SUBMISSION environment. Students are encouraged to leverage Gradescope TESTING before submitting an assignment for grading.

Please submit the following files to Gradescope SUBMISSION:

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

Do not submit any other files.

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

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. The submitted code is run as a batch job after the project deadline. 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 everything 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.

RUBRIC

Report

- General
    - Neatness (up to 5 points deduction if not).
    - Bonus for exceptionally well-written reports (up to 2 points)
    - Is the required report provided (-100 if not)
- Indicators
    - Are there five different indicators where you may only use two from the set discussed in the lectures (i.e., no more than two from the set [SMA, Bollinger Bands, RSI])? (-15 points each if not)
    - Does the submitted code indicators.py properly reflect the indicators provided in the report (up to -75 points if not)
    - Individual Indicators (up to 15 points potential deductions per indicator):
    - Is there a compelling description of why the indicator might work (-5 if not)
    - Is the indicator described in sufficient detail that someone else could reproduce it? (-5 points if not)
    - Is there a chart for the indicator that properly illustrates its operation, including a properly labeled axis and legend? (up to -5 points if not)
- Theoretically optimal (up to 20 points potential deductions):
    - Is the methodology described correct and convincing? (-10 points if not)
    - Is the chart correct (dates and equity curve), including properly labeled axis and legend (up to -10 points if not)
    - The historical value of benchmark normalized to 1.0, plotted with a green line (-5 if not)
    - The historical value of portfolio normalized to 1.0, plotted with a red line (-5 if not)
    - Are the reported performance criteria correct? See the appropriate section for required statistics. (-2 points for each item if not)

Code

- Is the required code provided, including code to recreate the charts and usage of correct trades DataFrame? () (up to -100 if not)
- All charts must be created and saved using Python code. DO NOT use plt.show() (up to -100 if all charts are not created or if plt.show() is used)

Auto-Grader

- No auto-grader

ALLOWED

- Your code may use the standard Python libraries, NumPy, SciPy, matplotlib, and Pandas libraries. Be sure you are using the correct versions as stated on the ML4T Local Environment Setup page.
- Code provided by the instructor or is allowed by the instructor to be shared.
- Code that displays "warning" messages to the terminal or console.
- Students are allowed to share charts in the pinned Students Charts thread alone.
- A herring.

PROHIBITED

- You may not use any libraries not listed in the "allowed" section above.
- You may not use any other method of reading data besides util.py.
- You may not modify or copy code in util.py.
- You may not use any code you did not write yourself.
- You may not use the Python os library/module. Code in Gradescope SUBMISSION must not generate any output to the screen/console/terminal (other than run-time "warning" messages) when verbose = False.
- Code must not create extra directories or files.
- Code must not use absolute import statements, such as: *from folder_name import TheoreticalOptimalStrategy*, where "folder_name" is the path/name of a folder or directory.
- You must **not** share tables and statistics.

RELEVANT RESOURCES

You may find the following resources useful in completing the project or 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.