CS7646 ML4T
Machine Learning
for Trading

PROJECT 8: STRATEGY EVALUATION

DUE DATE

11/22/2020 11:59PM Anywhere on Earth time

REVISIONS

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

– 10/28/2020 Gradescope updated to accept README.txt during file upload

– 11/17/2020 Update addEvidence() to add_evidence to align with code base

OVERVIEW

This assigment counts towards 20% of your overall grade.

- Indicator Selection:
  - Choose at least 3 indicators created and reported on P6.

- You can only use the indicators that were reported on P6. If you created an EMA function as part of MACD indicator, you may only use MACD as the indicator and not EMA. You may use EMA if you reported EMA as an indicator.
- Indicators can only be used once

- Implement a **Manual Strategy** (manual rule based trader) by:
  - Using your intuition and the indicators selected above, create a strategy and test it against a stock using your market

- Implement a **Strategy Learner**
  - You must draw on the learners you have created so far in the course. Your choices are:

1. **Classification**-based learner: Create a strategy using your Random Forest learner. Suggestions if you follow this approach: Classification_Trader_Hints. Important note, if you choose this method, you must set the leaf_size for your learner to 5 or greater. This is to avoid degenerate overfitting in-sample. For classification, you must convert your regression learner to use mode rather than mean (RTLearner, BagLearner).
2. **Reinforcement**-based learner: Create a Q-learning-based strategy using your Q-Learner. Read the Classification_Trader_Hints first, because many of the ideas there are relevant for the Q trader, then see Q_Trader_Hints. For Q-learning, use the same binning cuts for in-sample and out-of-sample.
3. **Optimization**-based learner: Create a scan-based strategy using an optimizer. Read the Classification_Trader_Hints first, because many of the ideas there are relevant for the Opto trader, then see Opto_Trader_Hints

- Regardless of your choice above, your learner should work in the following way:
  - In the training phase (e.g., add_evidence()) your learner will be provided with a stock symbol and a time period. It should use this data to learn a strategy. For instance, for a classification-based learner it will use this data to make predictions about future price changes.
  - In the testing phase (e.g., testPolicy()) your learner will be provided a symbol and a date range. All learning should be turned OFF during this phase.
  - You should use the same indicators as you use in the **Manual Strategy** in Strategy Learner so we can compare your results. You may optimize your indicators for time (vectorization).

- Your learner should return a trades DataFrame like it did in the last project. Here are some important requirements: Your testPolicy() method should be much faster than your add_evidence() method. The timeout requirements (see rubric) will be set accordingly. Multiple calls to your testPolicy() method should return exactly the same result.

Overall, your tasks for this project include:

- Build a **Manual Strategy** that combines a minimum of 3 out of the 5 indicators from Project 6.
- Build a **Strategy Learner** based on one of the learners described above that uses the same 3+ indicators.
- Test/debug the **Manual Strategy** and **Strategy Learner** on specific symbol/time period problems.
- Conduct experiments.

Write a report describing your **Manual Strategy**, **Strategy Learner** and **Experiments**.

TEMPLATE

- Download and install the files from this zip file File:Strategy_Evaluation_2020Fall.zip
- `ManualStrategy.py` Code implementing a ManualStrategy object (your **Manual Strategy**) in the `strategy_evaluation/` directory. It should implement testPolicy() which returns a trades data frame (see below). The main part of this code should call marketsimcode as necessary to generate the plots used in the report. NOTE: You will have to create this file yourself.
- Place your existing Q-Learner or RTLearner and BagLearner (Okay to include DTLearner as well if inheritance is involved) or OptimizationLearner into the `strategy_evaluation/` directory.
- Place your existing indicators.py into the `strategy_evaluation/` directory (NOTE: You can make changes to the indicators to properly work with both **Manual Strategy** and **Strategy Learner** but both strategies must use the same indicator code).
- Place your existing marketsimcode.py into the `strategy_evaluation/` directory (optional: if needed).
- `StrategyLearner.py` Code implementing a StrategyLearner object (your ML strategy) in the `strategy_evaluation` directory.

- `experiment1.py and experiment2.py` Code conducting the experiments outlined below. NOTE: You will have to create this file yourself.
- `testproject.py` Code initializing/running all necessary files for the report. NOTE: You will have to create this file yourself.
- See "what to turn in" below for a list of files that should be submitted.
- To test your **Strategy Learner**, follow the instructions on Running the grading scripts

DATA DETAILS, DATES & RULES

- Use only the data provided for this course. You are not allowed to import external data.
- For your report, trade only the symbol JPM. This will enable us to more easily compare results. We will test your **Strategy Learner** with other symbols as well.
- You may use data from other symbols (such as SPY) to inform both your Manual Learner and **Strategy Learner**.
- The in-sample period is January 1, 2008 to December 31, 2009.
- The out-of-sample/testing period is January 1, 2010 to December 31 2011.
- Starting cash is $100,000.
- Allowable positions are: 1000 shares long, 1000 shares short, 0 shares.
- Only buy/sell is allowed. stops, trailing stops, stoploss or any other trading setup is not allowed.
- Benchmark: The performance of a portfolio starting with $100,000 cash, investing in 1000 shares of the symbol in use on the first trading day, and holding that position. Include transaction costs.
- There is no limit on leverage.
- Transaction costs:
  - ManualStrategy and StrategyLearner: Commission: $9.95, Impact: 0.005 (unless stated otherwise in an experiment).
  - Auto-Grader Commission will always be $0.00, Impact may vary, and will be passed in as a parameter to the learner.
- Minimize use of herrings.

TASKS

Implement Manual Rule-Based Trader

Not included in template. You will have to create this code file.

Create `ManualStrategy.py` and implement a set of rules using at a minimum of 3 indicators you created in Project 6 (NOTE: You can make changes to the indicators to properly work with both **Manual Strategy** and **Strategy Learner** but both strategies must use the same indicator code). Devise some simple logic using your indicators to enter and exit positions in the stock. All indicators must be used in some way to determine a buy/sell signal. You cannot use a single indicator for all signals.

A recommended approach is to create a single logical expression that yields a -1, 0, or 1, corresponding to a “short,” “out” or “long” position. Example usage is signal: If you are out of the stock, then a 1 would signal a BUY 1000 order. If you are long, a -1 would signal a SELL 2000 order. You don’t have to follow this advice though, so long as you follow the trading rules outlined above.

For the report we want a written description, not code, however, it is OK to augment your written description with a pseudocode figure.

You should tweak your rules as best you can to get the best performance possible during the in-sample period (do not peek at out-of-sample performance) and should include more than one trade. Use your rule-based strategy to generate a trades dataframe over the in-sample period.

We expect that your rule-based strategy should outperform the benchmark over the in-sample period.

Benchmark: The performance of a portfolio starting with $100,000 cash, investing in 1000 shares of JPM on the first trading day, and holding that position.

Your ManualStrategy should implement the following API:

```
df_trades = ms.testPolicy(symbol = "AAPL", sd=dt.datetime(2010, 1, 1), ed=dt.datetime(2011,12,31), s
```

Implement Strategy Learner

For this part of the project you should develop a learner that can learn a trading policy using your learner and the same indicators used in ManualStrategy (NOTE: You can make changes to the indicators to properly work with both **Manual Strategy** and **Strategy Learner** but both strategies must use the same indicator code). You should be able to use your Q-Learner or RTLearner from the earlier project directly. If you want to use the optimization approach, you will need to create new code for that. You will need to write code in `StrategyLearner.py` to "wrap" your learner appropriately to frame the trading problem for it. Utilize the template provided in `StrategyLearner.py`.

Your Strategy Learner should find the optimal parameters **that you choose to optimize** for each indicator and should result in more than one trade. Remember, the indicators used must match those used for Manual Strategy. However, optimization of the indicators does not need to match that of Manual Strategy. Example: for SMA the learner `could` find the optimal lookback window to use.

NOTE: Lookback windows are not required to be optimized. You can use the same window used in ManualStrategy if you wish.

Your StrategyLearner should implement the following API:

```
import StrategyLearner as sl
learner = sl.StrategyLearner(verbose = False, impact = 0.0, commission=0.0) # constructor
learner.add_evidence(symbol = "AAPL", sd=dt.datetime(2008,1,1), ed=dt.datetime(2009,12,31), sv = 100
df_trades = learner.testPolicy(symbol = "AAPL", sd=dt.datetime(2010,1,1), ed=dt.datetime(2011,12,31)
```

The input parameters are:

- verbose: if False do not generate any output
- impact: The market impact of each transaction
- commission: The commission amount charged.

- symbol: The stock symbol to train 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 data frame whose values represent trades for each day. 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 when switching from long to short or short to long so long as net holdings are constrained to -1000, 0, and 1000.

Implement Experiment 1

Not included in template. You will have to create this code file.

Compare your Manual Strategy with your Strategy Learner in-sample trading JPM. Create a chart that shows:

- Value of the ManualStrategy portfolio (normalized to 1.0 at the start)
- Value of the StrategyLearner portfolio (normalized to 1.0 at the start)
- Value of the Benchmark portfolio (normalized to 1.0 at the start)

The code that implements this experiment and generates the relevant charts and data should be submitted as `experiment1.py`.

See DATA DETAILS, DATES & RULES section above for commission and impact information.

Implement Experiment 2

Not included in template. You will have to create this code file.

Conduct an experiment with your StrategyLearner that shows how changing the value of impact should affect in-sample trading behavior (use at least two metrics). Trade JPM on the in-sample period with a commission of $0.00.

The code that implements this experiment and generates the relevant charts and data should be submitted as experiment2.py.

See the 'Report' section on Experiment 2 for more details.

Implement Test Project

Execution Limit: 10 minutes

Not included in template. You will have to create this code file.

Create `testproject.py` and implement the necessary calls (following each respective API) to **Manual Strategy.py**, **StrategyLearner.py**, **experiment1.py** and **experiment2.py** with the appropriate parameters to run everything needed for the report in a single Python call:

```
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.

Implement author() function/method

Deduction if not implemented.

You should implement a function called `author()` that returns your Georgia Tech user ID as a string in all python files. 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.

Create README

Deduction if not present or instructions to run your code is incorrect

Create a file named `README.txt` that describes:

- Each file you are submitting.
- Step-by-step instructions to run your code.

Report

Answer the following prompt in a maximum of 10 pages (excluding references) in JDF format. 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 and tables to support each of your answers. Charts and tables should be generated by the code and saved to files. Charts should be properly annotated with legible and appropriately named labels, titles, and legends. Tables should be properly annotated with column names. When numbers are presented in tables, ensure a sufficiently level of numeric precision is provided.

Please address each of these points / questions, the questions asked in the Project 8 wiki, and the items stated in the Project 8 rubric in your report. The total number of charts may not exceed 10 charts. The report is to be submitted as **report.pdf**.

Introduction: ~ 0.5 pages

(Optional)

The report should briefly describe the paper's justification. While the introduction may assume that the reader has some domain knowledge, it should assume that the reader is unfamiliar with the specifics of the assignment. The introduction should also present an initial hypothesis (or hypotheses).

Indicator Overview: ~1 page

Briefly describe the indicators you used to devise your Manual Strategy and Strategy Learner. You must use a minimum of 3 indicators of the 5 you implemented in Project 6. At a minimum, for each indicator discuss the following:

- Discuss the parameters for each indicator that are optimized in both **Manual Strategy** and **Strategy Learner**.

Hint: If you used Bollinger Bands in Project 6 and want to use that indicator here, you can replace it with BB %B, which should work better with this assignment.

Manual Strategy: ~3 pages

Describe how you combined your indicators to create an overall signal. **Explain** *how* and *why* you decide to enter and exit your positions? Why do you believe (or not) that this is an effective strategy? Create a chart that shows, in-sample:

- **Benchmark** (starting with $100,000 cash, investing in 1000 shares of JPM and holding that position): **Green line**
- Performance of **Manual Strategy**: **Red line**
- Both should be **normalized to 1.0** at the start.
- Vertical **blue lines** indicating LONG entry points.
- Vertical **black lines** indicating SHORT entry points.

Compare the performance of your Manual Strategy versus the benchmark for the in-sample and out-of-sample time periods. Provide a chart to support the discussion.

Evaluate the performance of your strategy in the out-of-sample period. Note that you should not train or tweak your approach on this data. You should use the classification learned using the in-sample data only. Create a chart that shows, out-of-sample.

- **Benchmark** (starting with $100,000 cash, investing in 1000 shares of JPM and holding that position): **Green line**
- Performance of **Manual Strategy**: **Red line**
- Both should be **normalized to 1.0** at the start.
- Vertical **blue lines** indicating LONG entry points.
- Vertical **black lines** indicating SHORT entry points.

Create a table that summarizes the performance of the stock, and the Manual Strategy for both in-sample and out-of-sample periods. **Explain WHY these differences occur**. At minimum, the table must include:

- Cumulative return of the benchmark and Manual Strategy portfolio
- STDEV of daily returns of benchmark and Manual Strategy portfolio
- Mean of daily returns of benchmark and Manual Strategy portfolio

Strategy Learner: ~1.5 pages

The centerpiece of this section should be the description of how you utilized your learner to determine trades:

- Describe the steps you took to frame the trading problem as a learning problem for your learner.
- Describe the hyperparameters, their values, and how they were determined.
- Describe how you discretized (standardized) or otherwise adjusted your data. If this was not performed or necessary, explain why.

Experiment 1 (Manual Strategy / Strategy Learner): ~1.5 pages

Describe your experiment in detail: This includes any assumptions, the initial experimental hypothesis, parameter values, an any other information that would enable an informed reader to setup and repeat the experiment.

Describe, interpret, and summarize the outcome of your experiment. Would you expect this relative result every time with in-sample data? Explain why or why not.

Create a chart that shows:

- Value of the ManualStrategy portfolio (normalized to 1.0 at the start)
- Value of the StrategyLearner portfolio (normalized to 1.0 at the start)
- Value of the Benchmark portfolio (normalized to 1.0 at the start)

Experiment 2 (Strategy Learner): ~1.5 pages

Provide a hypothesis regarding how changing the value of impact should affect in-sample trading behavior and results (provide at least two metrics, assessing a minimum of 3 different measurements for each metric).

Your descriptions should be stated clearly enough that an informed reader could conduct the experiment and reproduce the results without referencing your code.

References: ~0.25 pages

(Optional)

References should be placed at the end of the paper in a dedicated section. Reference lists should be numbered and organized alphabetically by first author's last name. If multiple papers have the same author(s) and year, you may append a letter to the end of the year to allow differentiated in-line text (e.g. Joyner, 2018a and Joyner, 2018b in the section above). If multiple papers have the same author(s), list them in chronological order starting with the older paper. Only works that are cited in-line should be included in the reference list. The reference list does not count against the length requirements.

Submission Instructions

Complete your assignment using JDF, then save your submission as a PDF. Assignments should be submitted to the corresponding assignment submission page in Canvas. You should submit a single PDF for this assignment. Be sure to following the instructions in Canvas for the report and code submissions.

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.

Late work is not accepted without advanced agreement except in cases of medical or family emergencies. In the case of such an emergency, please immediately contact the Dean of Students followed by contacting the "Instructors" through a Piazza private post.

WHAT TO TURN IN

Be sure to follow these instructions diligently! No zip files.

Canvas:

Submit the following files (only) via Canvas before the deadline:

- Project 8: Strategy Evaluation (Report)
  - Your report as report.pdf.

Gradescope:

(SUBMISSION) Project 8: Strategy Evaluation:

- Your code as <Implemented Learner>.py, ManualStrategy.py, StrategyLearner.py, indicators.py, experiment1.py, experiment2.py, marketsimcode.py (optional if needed) and testproject.py
  - <Implemented Learner.py> refers to: QLearner.py or RTLearner (okay to submit DTLearner if using inheritance) and BagLearner.py, and OptimizeLearner.py.
- README.txt

Do not submit any other files.

You are only allowed 3 submissions to **(SUBMISSION) Project 8: Strategy Evaluation** but unlimited resubmissions are allowed on **(TESTING) Project 8: Strategy Evaluation.**

Note that Gradescope does **not** grade your assignment live; instead, it pre-validates that it will run against our batch autograder that we will run after the deadline. There will be **no** credit given for coding assignments that do not pass this pre-validation.

Refer to the Gradescope Instructions for more information.

RUBRIC

Report [30 Points]

- Is the report especially well written (up to 2 point bonus)
- Does the strategy utilize a minimum of 3 indicators? (up to -30 points)
- Does the report description match the code? (up to -10 points)

Indicator Overview (2 Points)

- Are the indicators used in Manual Strategy and Strategy Learner briefly described (up to -2 points)

Manual Strategy (6 Points)

- Is the trading strategy described with clarity and in sufficient detail that someone else could reproduce it? (up to -4 points)
- Does the manual trading system provide higher cumulative return than the benchmark over the in-sample time period? (-2 points if not)
- Are differences between the in-sample and out-of-sample performances appropriately explained (up to -4 points)
- Is the required table present and correct (up to -2 points)
- Did the student use the correct symbol? (-2 points)
- Did the student use the correct date periods? (-2 points)
- Does the strategy obey holding constraints (-6 points)
- Does the provided chart(s) include:
  - Value of benchmark normalized to 1.0 with green line (-1 point)
  - Value of portfolio normalized to 1.0 with red line (-1 point)
  - Are vertical lines, appropriately colored, included to indicate entries (-1 point)

Strategy Learner (6 Points)

- Is the method by which the learner is utilized to create a trading strategy described sufficiently clearly that an informed reader could reproduce the results without referencing your code? (up to -5 points)
- Did the student choose to use optimization based learning, and did their learning strategy beat the Manual Strategy (+1 point)

Experiment 1 (10 Points)

- Is the required experiment explained well? (up to -7 points)
- Is the required experiment compellingly supported the required chart? (-3 points)

Experiment 2 (6 Points)

- Is the required experiment explained well? (up to -4 points)
- Is the required experiment compellingly supported with tabular or graphical data?? (-2 points)

Code

- Does the submitted code indicators.py properly reflect the indicators provided in the report (up to -30 points if not)
- Are the indicators used for Manual Strategy and Strategy Learner the same as the ones used in Project 6? (up to -30 point if not)
- Does the submitted code and report reflect an understanding of the subject matter and follow the assignment directions? (up to -30 points if not)
- Missing author method. (up to -5 for each missing with max -10 points)
- Missing or incorrect instructions in the README.txt file (-10 points)
- Does testproject.py produce all included charts, with one run and within the time limit without any manipulation? (up to -30 points if not)
- Are the charts created and saved using Python code(DO NOT use plt.show() and manually save your charts)? (up to -30 points if not)

Code & Report

Submission of code and report (up to 100 points deductions):

- Is the required code provided, including code to recreate the charts and usage of correct trades data frame. (up to -100 if not)
- Is the required report provided (up to -100 if not)

Auto-Grader [70 Points]

We will test Strategy Learner in the following situations:

- Training / in-sample: January 1, 2008 to December 31 2009.
- Testing / out-of-sample: January 1, 2010 to December 31 2011.
- Symbols: ML4T-220, AAPL, UNH, SINE_FAST_NOISE
- Starting value: $100,000
- Benchmark: Buy 1000 shares on the first trading day, Sell 1000 shares on the last day.
- Commissions = $0.00, impact = 0.00

We expect the following outcomes in evaluating your system:

- For ML4T-220
  - add_evidence() completes without crashing within 25 seconds: 1 points
  - testPolicy() completes in-sample within 5 seconds: 2 points
  - testPolicy() returns same result when called in-sample twice: 2 points
  - testPolicy() returns an in-sample result with cumulative return greater than 100%: 5 points
  - testPolicy() returns an out-of-sample result with cumulative return greater than 100%: 5 points
- For AAPL
  - add_evidence() completes without crashing within 25 seconds: 1 points
  - testPolicy() completes in-sample within 5 seconds: 2 points
  - testPolicy() returns same result when called in-sample twice: 2 points
  - testPolicy() returns an in-sample result with cumulative return greater than benchmark: 5 points
  - testPolicy() returns an out-of-sample result within 5 seconds: 5 points
- For SINE_FAST_NOISE
  - add_evidence() completes without crashing within 25 seconds: 1 points
  - testPolicy() completes in-sample within 5 seconds: 2 points
  - testPolicy() returns same result when called in-sample twice: 2 points
  - testPolicy() returns an in-sample result with cumulative return greater than 200%: 5 points
  - testPolicy() returns an out-of-sample result within 5 seconds: 5 points
- For UNH
  - add_evidence() completes without crashing within 25 seconds: 1 points
  - testPolicy() completes in-sample within 5 seconds: 2 points
  - testPolicy() returns same result when called in-sample twice: 2 points
  - testPolicy() returns an in-sample result with cumulative return greater than benchmark: 5 points
  - testPolicy() returns an out-of-sample result within 5 seconds: 5 points

- Withheld test case 1: In-sample test case for an unknown symbol.
  - If any part of code crashes: 0 points awarded.
  - testPolicy() returns an in-sample result with cumulative return greater than benchmark: 5 points
- Withheld test case 2: In-sample test case to verify that strategy accounts for different values of `impact`
  - If any part of code crashes: 0 points awarded.
  - Learner returns different trades when impact value is significantly different: 5 points

We reserve the right to use different time periods if necessary to reduce auto grading time.

- IMPORTANT NOTES
  - For achieving the required cumulative return, recall that `cr = (portval[-1]/portval[0]) - 1.0`
  - The requirement that consecutive calls to `testPolicy()` produce the same output for the same input means that you **cannot** update, train, or tune your learner in this method. For example, a solution that uses Q-Learning should use `querySetState()` and **not**`query()` in `testPolicy()`. Updating, training, and tuning (`query()`) is fine inside add_evidence`()`.
  - Your learner should **not** select different hyper-parameters based on the **symbol**. Hyper-parameters include (but are not limited to) things like features, discretization size, sub-learning methods (for ensemble learners). Tuning using cross-validation or otherwise pre-processing the **data** is OK, things like `if symbol=="UNH"` are **not OK**. There will be a withheld test case that checks your code on a valid symbol that is not one of the four listed above.
  - Presence of code like `if symbol=="UNH"` will result in a 20 point penalty.
  - When evaluating the trades generated by your learner, we **will** consider transaction costs (market impact).

REQUIRED, ALLOWED & PROHIBITED

Required:

- Your project must be coded in Python 3.6.x.

- Your code must be submitted to Gradescope in the appropriate Gradescope assignment.
- Use only the API functions in util.py to read data. Do NOT modify this file or copy the code from it into other files. For grading, we will use our own unmodified version.
- All charts must be generated in Python, and you must provide the code you used.
- No external learning libraries allowed.
- Reference any code used in the "Allowed" section in your code. At minimum it should have the link/filename/video name of where it came from.

Allowed:

- You can develop your code on your personal machine, but it must also run successfully on Gradescope.
- Your code may use standard Python libraries.
- You may use the NumPy, SciPy, matplotlib and Pandas libraries. Be sure you are using the correct versions.
- Code provided by the instructor or allowed by the instructor to be shared.
- A herring.

Prohibited:

- Generating charts using a method other than Python.
- Any other method of reading data besides util.py
- Modifying (or depending on modifications to) util.py.
- Any libraries not listed in the "allowed" section above
- Any use of plot.show()
- Absolute import statements of the **current** project folder such as `from strategy_evaluation import XXXX` or `import strategy_evaluation.XXXX`
- Any Classes (other than Random) that create their own instance variables for later use (e.g., learners like kdtree).
- Print statements outside "verbose" checks (they significantly slow down auto grading).
- Any code you did not write yourself.

FAQ

- **Q:** I want to read some other values from the data besides just adjusted close, how can I do that?

  A: Look carefully at util.py and you will see that you can query for other values.
- **Q:** Are we only allowed one position at a time?

  A: You can be in one of three states: -1000 shares, +1000 shares, 0 shares.
- **Q:** Are we required to trade in only 1000 share blocks? (and have no more than 1000 shares long or short at a time?

  A: You can trade up to 2000 shares at a time as long as you maintain the requirement of holding 1000, 0 or -1000 shares.
- **Q:** Are we limited to leverage of 2.0 on the portfolio?

  A: There is no limit on leverage.

- **Q:** Are we only allowed one position at a time?

  A: You can be in one of three states: -1000 shares, +1000 shares, 0 shares.