CS7646: ML4T
Machine Learning for Trading

PROJECT 1: MARTINGALE

DUE DATE

08/30/2020 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 changes; any changes will be logged in this section.

OVERVIEW

This assigment counts towards 3% of your overall grade.

The purpose of this assignment is to get you started programming in Python right away and to help provide you some initial feel for risk, probability, and "betting." Purchasing a stock is, after all, a bet that the stock will increase in value.

In this project you will evaluate the actual betting strategy that Professor Balch uses at roulette when he goes to Las Vegas. Here it is:

Pseudocode

```
episode_winnings = $0
while episode_winnings < $80:
    won = False
    bet_amount = $1
    while not won
        wager bet_amount on black
        won = result of roulette wheel spin
        if won == True:
            episode_winnings = episode_winnings + bet_amount
```

```
        else:
            episode_winnings = episode_winnings - bet_amount
            bet_amount = bet_amount * 2
```

Here are some details regarding how roulette betting works: Betting on black (or red) is considered an "even money" bet. That means that if you bet N chips and win, you keep your N chips and you win another N chips. If you bet N chips and you lose then those N chips are lost. The odds of winning or losing depend on whether you're betting at an American wheel or a European wheel. For this project we will be assuming an **American wheel**. You can learn more about roulette and betting here: https://en.wikipedia.org/wiki/Roulette

TEMPLATE

Set up your development environment

First, if you haven't yet set up your software environment, follow the instructions here: ML4T_Software_Setup. This is the base directory structure used for all projects in the class, including supporting data and software will be set up correctly when you follow those instructions.

Then:

- Download the template code here: File:Martingale_2020Fall.zip
- Extract its contents into the base directory (ML4T_2020Fall)
- You should see the following directory structure:
  - `ML4T_2020Fall/`: Root directory for course
    - `data/`: Location of data
    - `grading/`: Grading libraries used by the individual grading scripts for each assignment.
    - `util.py`: Common utility library. This is the **only** allowed way to read in stock data. Only use the API methods provided here to read in stock data. Do NOT modify this file. For grading, we will use our own unmodified version.
    - README.md
    - `martingale/`: Root directory for this project
      - `martingale.py`: Main project file to use as a template for your code.

You should change only `martingale.py`. All of your code should be in that one file. Do not create additional files. It should always remain in and run from the directory `ML4T_2020Fall/martingale/`. Leave the copyright information at the top intact.

- Implement the necessary functions in `martingale/martingale.py`.
- To execute your martingale code for debugging purposes, run **`PYTHONPATH=..:. python martingale.py`** from the `martingale/` directory.

To test your code, we will be calling `__main__()` function only. Doing so should run all aspects of the program and output the desired charts.

TASKS

author() and gtid() function

Revise the code functions `author()` and `gtid()` to correctly include your GT Username and 9 digit GT ID respectively. Your GT Username should be something like `tbalch78` and your GTID is a 9 digit number. You should also update this information the comments section at the top.

Build a simple gambling simulator

Revise the code in `martingale.py` to simulate 1000 successive bets on spins of the roulette wheel using the betting scheme outlined above. You should test for the results of the betting events by making successive calls to the `get_spin_result(win_prob)` function. Note that you'll have to update the `win_prob` parameter according to the correct probability of winning. You can figure that out by thinking about how roulette works (see wikipedia link above).

Track your winnings by storing them in a numpy array. You might call that array `winnings` where `winnings[0]` should be set to 0 (just before the first spin). `Winnings[1]` should reflect the total winnings after the first spin and so on.

Chart Creation

For the following charts, and for all charts in this class you should use python's matplotlib library. Your submitted project should include all of the code necessary to generate the charts listed in your report. You should configure your code to write the figures to .png files. Do not allow your code to create a window that displays images (DO NOT use plt.show() and manually save as .png files). If it does you will receive a penalty.

Experiment 1: Explore the strategy and make some charts

Now we want you to run some experiments to determine how well the betting strategy works. The approach we're going to take is called Monte Carlo simulation where the idea is

to run a simulator over and over again with randomized inputs and to assess the results in aggregate. Review the "report" section below for specific properties of the strategy we want you to evaluate.

- Figure 1: Run your simple simulator 10 times and track the winnings, starting from 0 each time. Plot all 10 runs on one chart using `matplotlib` functions. The horizontal (X) axis should range from 0 to 300, the vertical (Y) axis should range from -256 to +100. Note that we will not be surprised if some of the plot lines are not visible because they exceed the vertical or horizontal scales.
- Figure 2: Run your simple simulator 1000 times. Remember that the simple simulator simulates 1000 successive bets, so you will be running that 1000 times. Plot the mean value of winnings for each round of spin using the same axis bounds as Figure 1. For example, you should take the mean of the first spin of all 1000 simulations. Add an additional line above and below the mean, at mean (+) standard deviation, and mean (-) standard deviation of the winnings at each point.
- Figure 3: Use the same data you used for Figure 2, but plot the median instead of the mean. Add an additional line above and below the median at median (+) standard deviation, and median (-) standard deviation of the winnings at each point.

For all of the above charts and experiments, if and when the target $80 winnings is reached, stop betting and allow the $80 value to persist from spin to spin (e.g., fill the data forward with a value of $80).

Experiment 2: A more realistic gambling simulator

You may have noticed that the strategy works pretty well, maybe better than you expected. One reason for this is that we were allowing the gambler to use an unlimited bank roll. In this experiment we're going to make things more realistic by giving the gambler a $256 bank roll. If he or she runs out of money, bzzt, that's it. Repeat the experiments above with this new condition. Note that once the player has lost all of their money (i.e., episode_winnings reaches -256) stop betting and fill that number (-256) forward. An important corner case to be sure you handle is the situation where the next bet should be $N, but you only have $M (where M<N). Make sure you only bet $M. Here are the two charts to create:

- Figure 4: Run your realistic simulator 1000 times. Plot the mean value of winnings for each spin using the same axis bounds as Figure 1. Add an additional line above and below the mean at mean (+) standard deviation, and mean (-) standard deviation of the winnings at each point.
- Figure 5: Use the same data you used for Figure 4, but plot the median instead of the mean. Add an additional line above and below the median at median (+) standard deviation, and median (-) standard deviation of the winnings at each point.

CONTENTS OF REPORT

Answer the following prompt in a maximum of 7 pages (excluding references) in JDF format. Any content beyond 7 pages will not be considered for a grade. Seven pages is a maximum, not a target; our recommended per-section lengths intentionally add to less than seven 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. Charts should 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**.

Experiment 1: ~ 1 page

Question 1: In Experiment 1, *based off the experiment results* calculate the estimated probability of winning $80 within 1000 sequential bets. *Explain your reasoning for the answer* using the experiment thoroughly (not based on plots).

Question 2: In Experiment 1, *what is the estimated expected value* of winnings after 1000 sequential bets? *Thoroughly explain your reasoning for the answer*. See the following Wikipedia entry to learn about expected value: https://en.wikipedia.org/wiki/Expected_value

Question 3: In Question 3, *do the **(**mean + standard**)** deviation line and **(**mean – standard deviation**)** line reach a maximum value then stabilize? Do the lines converge* as the number of sequential bets increases? *Explain why it does or does not* thoroughly.

Hint: If it does not converge and/or stabilized, explain why it does not. If it does converge and/or stabilize, also discuss the value(s) at which it does so.

Experiment 2: ~ 1 page

Question 4: In Experiment 2, *based off the experiment results* calculate the estimated probability of winning $80 within 1000 sequential bets. *Explain your reasoning for the answer* using the experiment thoroughly. (not based on plots).

Question 5: In Experiment 2, *what is the estimated expected value* of our winnings after 1000 sequential bets? *Thoroughly explain your reasoning for the answer* (not based on plots).

Question 6: In Question 6, *do the **(**mean + standard**)** deviation line and **(**mean – standard deviation**)** line reach a maximum value then stabilize? Do the lines converge* as the number of sequential bets increases? *Explain why it does or does not* thoroughly.

Hint: If it does not converge and/or stabilized, explain why it does not. If it does converge and/or stabilize, also discuss the value(s) at which it does so.

Charts: ~ 2.5 pages

(If the required charts are included in the Experiment sections above, then this section is optional)

Include Figure 1 through 5.

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.

Report 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

Canvas:

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

- Project 1: Martingale (Report)
  - Your report as `report.pdf`

Do not submit any other files. Note that your charts should be included in the report, not submitted as separate files. Also note that if we run your submitted code, it should generate all 5 figures as png files. Not submitting a report will result in a 0 for the assignment.

Unlimited resubmissions are allowed up to the deadline for the project.

Gradescope (TO BE UP SOON):

- Project 1: Martingale (Code)
  - Your code as `martingale.py`

Do not submit any other files.

You are only allowed 3 submissions to **Project 1: Martingale (Code)** but unlimited resubmissions are allowed on **(TESTING) Project 1.**

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.

RUBRIC

Report

- Are the questions answered correctly? (Up to -5 points for each incorrect answer)
- Is the reasoning for each question correct and supported by the evidence thoroughly? (Up to -5 points for each if incorrect)
- Are each of the charts provided and correct and include labeled axes and legend? (Up to -8 points for each if incorrect)

Code

- Does the code run without crashing? (-10 points if not)
- Does the code generate appropriate charts written to png files? **Specifically, DO NOT use plt.show() and then manually save your figures. The figures should be created and saved using Python code** (-10 points each up to a max of -20 if not)
- Does the implemented code reflect the project requirements? (Up to -10 points if not)

Auto-Grader

No auto-grader

REQUIRED, ALLOWED & PROHIBITED

Required:

- Your project must be coded in Python 3.6.x.
- ~~Your code must run on one of the university-provided computers (e.g. buffet01.cc.gatech.edu).~~ This requirement is not enforced for this first project, but will be for future projects.
- 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:

- Your code may use standard Python libraries (except os).
- 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.

Prohibited:

- Any use of global variables.
- Any libraries not listed in the "allowed" section above.
- Use of Python's os module.
- Any use of plot.show()
- Absolute import statements of the **current** project folder such as `from 'martingale' import XXXX or import martingale.XXXX`
- Extra directories (manually or code created)
- Extra files not listed in "WHAT TO TURN IN"
- Any code you did not write yourself (except for provided/allowed by the instructor).
- Knights who say "neeee."

FAQ

- **Q:** I'm using a Mac. When I try to plot, I get a strange exception with a stack trace, including a mention of libtk and tkinter.
- **A:** Run the following commands from the terminal to change the default back end

```
$ mkdir -p ~/.matplotlib
$ echo "backend: TkAgg" > ~/.matplotlib/matplotlibrc
```