CS7646 ML4T
Machine Learning
for Trading

PROJECT 1: MARTINGALE

Table of Contents

REVISIONS

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

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

1. OVERVIEW

In this project, you will write software that will perform probabilistic experiments involving an American Roulette wheel. The project will help provide you with some initial feel for risk, probability, and "betting." Purchasing a stock is, after all, a bet that the stock will increase (or, in some cases, decrease) in value. You will submit the code for the project to Gradescope SUBMISSION. You will also submit to Canvas a report that discusses your experimental findings.

1.1 Learning Objectives

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

- **Mathematical Tools**: Developing an understanding of common probabilistic and statistical tools associated with machine learning, including *expectations*, *standard deviations, sampling, minimum values, maximum values,* and *convergence*.
- **Research**: Experience researching additional material (conceptual and programming) to ensure the successful completion of the assignment.
- **Programming & Academic Writing**: Each assignment will build upon one another. The techniques around experimentation, graphs, interpretation (and so on) will play important roles in this and future projects.
- **Course Conduct**: Developing and testing code locally in the local Conda ml4t environment, submitting it for pre-validation in the Gradescope TESTING environment, and submitting it for grading in the Gradescope SUBMISSION environment.

2. ABOUT THE PROJECT

In this project, you will build a Simple Gambling Simulator. Specifically, you will revise the code in the martingale.py file to simulate 1000 successive bets on the outcomes (i.e., spins) of the American roulette wheel using the betting scheme outlined in the pseudo-code below. Each series of 1000 successive bets are called an "episode." 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 will 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 below).

In this project, you will evaluate Professor Balch's actual betting strategy at roulette when he goes to Las Vegas.

Here is the pseudocode of the strategy:

```
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
```

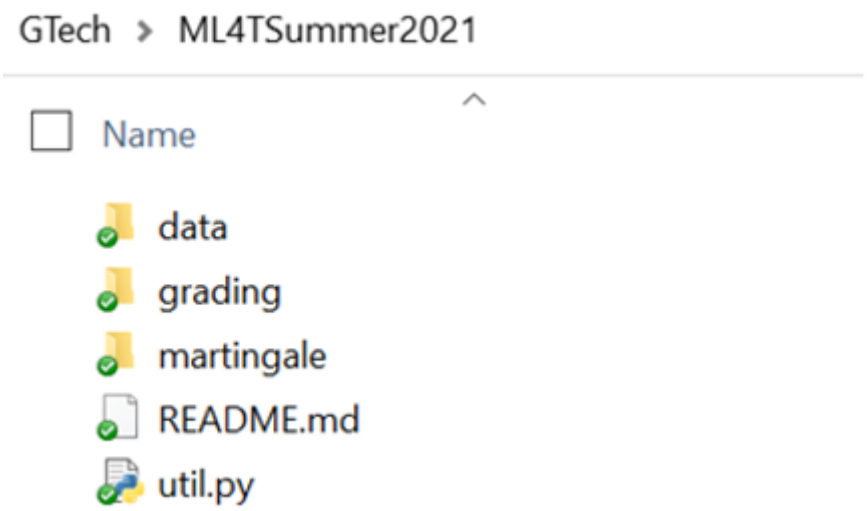
martingale_pseudocode hosted with ❤ by GitHub view raw

Additional 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 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.

3. YOUR IMPLEMENTATION

You will develop an implementation leveraging the pseudocode above that conducts several experiments. Conduct the following experiments, then write your report. Before the deadline, make sure to pre-validate your submission using Gradescope TESTING. Once you are satisfied with the results in testing, submit the code to Gradescope SUBMISSION. Only code submitted to Gradescope SUBMISSION will be graded. If you submit your code to Gradescope TESTING and have not also submitted your code to Gradescope SUBMISSION, you will receive a zero (0).

3.1 Getting Started



You will be given a starter framework to make it easier to get started on the project and focus on the concepts involved. This framework assumes you have already set up the local environment and ML4T Software. The framework for Project 1 can be obtained from: Martingale_2022Summer.zip. Extract its contents into the base directory (e.g., ML4T_2022Spr, although "ML4T_2021Summer" is shown in the image below). This will add a new " martingale " folder to the directory structure.

Within the martingale folder is a single file:

martingale.py

You will modify the martingale.py file to implement the necessary functionality for this assignment. The existing code in the martingale.py file may contain ideas for functions and methods that could be used in your implementations. This file must remain and run from within the martingale directory using the following command:

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

martingale_execution hosted with ❤ by GitHub view raw

3.2 Experiment 1 – Explore the strategy and create some charts

In this experiment, you will develop code that performs experiments using Professor Balch's original betting strategy. You will run some experiments to determine how well the betting strategy works. The approach we are going to take is called Monte Carlo

simulation. The idea is to run a simulator repeatedly with randomized inputs and assess the results in aggregate. Your implementation will produce the following charts (i.e., figures):

- **Figure 1**: Run your simple simulator 10 episodes and track the winnings, starting from 0 each time. Plot all 10 episodes on one chart using Matplotlib functions. The horizontal (X) axis must range from 0 to 300, the vertical (Y) axis must range from –256 to +100. 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 episodes. (Remember that 1000 successive bets are one episode.) Plot the mean value of winnings for each spin round using the same axis bounds as Figure 1. For example, you should take the mean of the first spin of all 1000 episodes. Add an additional line above and below the mean, at mean plus standard deviation, and mean minus 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 to represent the median plus standard deviation and median minus standard deviation of the winnings at each point.

For all the above figures and experiments, if the target of $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).

The charts created by the experiments must be included in your report, along with your supporting analysis and discussion. All charts must be properly titled, have appropriate axis labels, use consistent axis ranges, and have legends.

3.3 Experiment 2 – A more realistic gambling simulator

You may have noticed that the original strategy performed in experiment 1 works well, maybe better than you expected. One reason for this is that we were allowing the gambler to use an unlimited bankroll. In this experiment, we retain the upper limit of $80 in winning retained but make things more realistic by giving the gambler a $256 bankroll. This will require a modification to the original strategy since if he or she runs out of money: bzzt, that's it. Repeat the experiments, as above, with this new condition. Note that once the player has lost all their money (i.e., episode_winnings reach -256), stop betting and fill that number (-256) forward. An important corner case to handle is the situation where the next bet should be $N, but you only have $M (where M<N). Since you cannot bet more than you have, be sure you only bet $M. Here are the two charts to create:

- **Figure 4**: Run your realistic simulator 1000 episodes and track the winnings, starting from 0 each time. 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 plus standard deviation and mean minus 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 to represent the median plus standard deviation and median minus standard deviation of the winnings at each point.

The charts created by the experiments will be included in your report, along with your supporting analysis and discussion. All charts must be properly titled, have appropriate axis labels, use consistent axis ranges, and have legends. You should use python's Matplotlib library.

3.4 Technical Requirements

The following technical requirements apply to this assignment:

1. The martingale.py file must implement this API specification.
2. All winnings must be tracked 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). The entry in winnings[1] should reflect the total winnings after the first spin and so on.
3. Use the population standard deviation. The standard deviation is plotted as two lines, the upper standard deviation (e.g., mean +stdev) and the lower standard deviation (e.g., mean –stdev).
4. You may set a specific random seed for this assignment. If a specific random seed is used, it can only be called once, and it must use your GT ID as the numeric value.
5. The implementation may optionally write text, statistics, and/or tables to a single file named p1_results.txt or p1_results.html.

3.5 Notes and Hints

3.5.1 Structuring the NumPy Array

	1 Spin [0]	2 Spin[1]	3 Spin [2]	4 Spin [3]	... ...	1001 Spin [1000]
1 Episode [0]	0	1	2	3	...	50
2 Episode [1]	0	-1	1	0	...	15
3 Episode [2]	0	1	0	2	...	80
Mean	0	0.3333	1.0000	1.6667	...	48.3333
StDev	0	0.9428	0.8165	1.2472	...	26.5623

Hint: One way to think about structuring the NumPy array for holding winnings is illustrated below. Each episode consists of 1000 spins plus the initial value in the first column.

3.5.2 Expectations

An expectation can be thought of as an arithmetic mean and is written as:

$$E[X] = \sum_x xP[X = x]$$

This equation essentially says: given a distribution, X (for example, all of the rows in an experiment), the expectation is the value x multiplied by its probability of finding x in that distribution for all possible x values. Sources (useful for deeper dives): Foundation of Machine Learning (appendix C), Mitchell's Machine Learning (chapter 5.3), and Probabilistic Machine Learning (chapter 2.2.5).

4 CONTENTS OF REPORT (100 POINTS)

In addition to submitting your agent to Gradescope, you will also write up a report describing your experimental hypothesis, design, and findings. The assignment requires the production and evaluation of the empirical results. While the results and analysis must be based on experimental observation, the analysis can also be supported using theoretical or mathematical proof. Your submitted project should include all the code necessary to generate the charts presented in your report. **Up to -30 points in deductions will be applied to the report score for unmet implementation requirements or code that fails to run.**

Your report must use the JDF format, which specifies font sizes and margins that should not be altered. Charts must be generated by the code (including any desired annotations) and saved as .png files to the project directory in the images subdirectory.

The charts should be imported (without additional post-processing or editing) into the report document. Charts must be properly annotated with legible and appropriately named labels, titles, and legends. All charts must use the same axis bounds.

Answer the following prompts and include the required charts in a maximum of 7 pages (excluding references) in JDF format. Any content beyond 7 pages will not be considered for a grade. The analysis and responses must be supported by experimental evidence:

- **Question 1**: In Experiment 1, based on the experiment results calculate and provide the estimated probability of winning $80 within 1000 sequential bets. Thoroughly explain your reasoning for the answer using the experiment output. Your explanation should NOT be based on estimates from visually inspecting your plots, but from analyzing any output from your simulation.
- **Question 2**: In Experiment 1, what is the estimated expected value of winnings after 1000 sequential bets? Thoroughly explain your reasoning for the answer.
- **Question 3**: In Experiment 1, do the upper standard deviation line (mean + stdev) and lower standard deviation line (mean – stdev) reach a maximum (or minimum) value and then stabilize? Do the standard deviation lines converge as the number of sequential bets increases? Thoroughly explain why it does or does not.
- **Question 4**: In Experiment 2, based on the experiment results calculate and provide the estimated probability of winning $80 within 1000 sequential bets. Thoroughly explain your reasoning for the answer using the experiment output. Your explanation should NOT be based on estimates from visually inspecting your plots, but from analyzing any output from your simulation.
- **Question 5**: In Experiment 2, what is the estimated expected value of winnings after 1000 sequential bets? Thoroughly explain your reasoning for the answer.
- **Question 6**: In Experiment 2, do the upper standard deviation line (mean + stdev) and lower standard deviation line (mean – stdev) reach a maximum (or minimum) value and then stabilize? Do the standard deviation lines converge as the number of sequential bets increases? Thoroughly explain why it does or does not.
- **Question 7**: What are some of the benefits of using expected values when conducting experiments instead of simply using the result of one specific random episode?

Recommendation: If the upper and lower standard deviation lines do not converge and/or stabilize, explain why they do not. If they converge and/or stabilize, also discuss the value(s) at which they do so.

5 TESTING RECOMMENDATIONS

A separate local testing script is not provided for this assignment. You are encouraged to perform any tests necessary to instill confidence that the code will run properly when submitted for grading and will produce the required results. To run and test that the file will run from within the martingale directory, use the command given below at the command prompt from within the martingale directory:

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

martingale_execution hosted with ❤ by GitHub view raw

Note: Once submitted for grading, we will use the above command to call the "__main__" section only. The program should run in its entirety and produce the necessary output and charts.

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 do not pass this pre-validation.

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 **p1_martingale_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 to Canvas in PDF format only:

p1_martingale_report.pdf

Do not submit any other files. Charts must be included in the report, not submitted as separate files. Also note that when we run your submitted code, it should produce and save all five (5) figures as .png files in the project directory.

You are allowed unlimited submissions of the **p1_martingale_report.pdf** file to **Canvas**.

6.2 Code Submission

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

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

martingale.py

Do not submit any other files.

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

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

7 GRADING INFORMATION

Your report is worth 100% of your grade. As such, it will be graded on a 100-point scale coinciding with a rubric design to mirror the questions above (see rubric below). Make sure to answer the questions in the Project Wiki and properly address all aspects of the Rubric. The code submitted to Gradescope SUBMISSION is run as a batch job after the project deadline. **All points for the assignment will be returned in the Canvas report score**.

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

- 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)
- Is each of the charts provided, correct, and includes a title, labeled axes, and legend? (Up to -8 points for each if incorrect)

7.1.2 Code

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

- If the code crashes when run. (-10 points; up to a max of -30 points)
- If the code does not produce appropriate charts that are saved as .png files. (-10 points each instance; up to a max of -20 points)
- If the code displays any charts in a window or screen. (-10 points each instance; up to a max of -20 points)
- If the code saves in a directory outside the project directory. (up to a max of –20 points)
- If the implemented code does not reflect the project requirements. (-10 points; up to a max of -30 points)

7.1.3 Auto-Grader (Private Grading Script)

- A private grading script is not used for 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 Development Recommendations, Guidelines, and Rules apply to this project:

- N/A

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.

Wikipedia (accessed 2021), Expected Value

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

James, D. Witten, T. Hastie, R. Tibshirani (2017), An Introduction to Statistical Learning (Chapter 2)

Murphy, (2021), Probabilistic Machine Learning: An Introduction (Chapter 2)