Dustin McDonnell

***Part 4/Stock Bot***

I was feeling confident about this whole project. I really felt like I grew as a programmer, and I was proud of my new skills. Then I got to this section. I don’t remember who, but someone said they could get it done in a day, forty minutes even. I think I’m a humble guy, I tacked on an additional 6 hours to give myself leeway. Then I went two days over the time limit I set for myself, repeatedly bashed my head against the digital wall and ended up with a something that is simultaneously the best I can do and something I’m extremely unhappy with. Anyway, that’s the preamble.

The idea I had originally was to read the csv data from Yahoo Finance, run an RSI calculator and a moving average calculator on the data which would then be loaded into an ArrayList. This ArrayList would then be fed one date at a time into the Stock Bot which would use heuristics such as the RSI being over 70/ under 30 and differences between the moving average and closing prices. I personally have some experience with trading stocks, as I had a /r/wallstreetsbets phase, so I decided to use the closing price as the main comparison for the heuristics. I am aware traditionally you wouldn’t use the closing price to make trades, but certain banks allow you to trade after market hours, which I always thought was better anyway.

The real problem arose when I decided to use the csv reader from Part 3, which was the external library’s csv reader instead of my own. I already was having some problems with it, but I decided to press ahead with the goal of fine tuning my skills with external methods. What a fool I was.

***Methods in FilePickerPart3 class***

***filePicker()***

This is the same method used in parts 1 and 3. The only difference is the directory now refers to Stockbot. Chooses a file based off user input in the directory to be passed to other methods in the class that alter the data.

A screenshot of a computer program

Description automatically generated

***csvReader()***

This is a variation on Part 2’s csv reader that adds more headers to be read from. I had a huge issue here trying to read the RSI off csvs after it was added because of an Index out of bound error, on account of RSIs ignoring the first N elements in an array due to momentum. As I type it out and look at it, I realize I could have changed the loop and used an if statement to check if the reader was past the point where RSIs started. Though I’ve wound part 4 too tightly and I’m afraid to change it at this point.

A computer screen shot of a program

Description automatically generated

***Methods in ManipulateDataPart3***

***choseN()***

This uses a scanner to get the N from user Input for RSI then returns it for use in the actual RSI function. I could have included this in the RSI function, but I think the intent was to use this for multiple purposes before it went off the rails.

A computer screen with white text

Description automatically generated

***calculateRSI()***

This takes the integer from choseN() then uses that as the range to perform the calculations required to get the RSI over a certain time-period. Although traditionally the RSI is calculated over the last fourteen days, I did not want to hard code that as I thought it would be more useful if it were more flexible.

A computer screen shot of a program code

Description automatically generated

***smoothData()***

This is the same method used in part 3 for smoothing data with the values being returned to an ArrayList of an ArrayList of Strings. I thought this would be easier to navigate with each top-level array corresponding to a different date and all the actual values contained within being the same number for every top-level array. I probably bit off more than I could chew however, and I should not have had my first real experience with multi-dimensional arrays for such an important project. This method also adds more instructions for the user than previous versions as there are more scanner entries required from the user and I didn’t want it to be confusing.

A screen shot of a computer program

Description automatically generated

***totalCSV()***

This method combines the ArrayList<ArrayList<String>> previously mentioned and a second ArrayList of RSI values together into a new CSV which the Stock Bot would then perform calculations on. The second else statement exists because originally, I was going to have this be multipurpose before it ended up becoming very rigid and tightly wound and as a result the second else statement is never used. I am keeping it for my own purposes as I intend to try and improve it later. This project really made me curious if real developers plan out their projects ahead of time to avoid issues like this. I’ve heard an anecdote that Stephen King plans none of his novels and just free flow writes all his novels. After this project I can safely say I am not the Stephen King of programming.

The values are hardcoded in print Record because of my previously mentioned multi-level array idea, it worked but I think I could have improved the readability significantly by using loops.

A computer screen shot of a program code

Description automatically generated

***createFile()***

This is the exact same method used for parts 1 and 3, old reliable I have no issues with the functionality of this and I now consider it my mental bedrock. The only difference is the directory now refers to stock bot instead of parts 1 or 3. This method is used to create the File object that will be passed to the csv writer. It creates a new file at the specified path if the file does not already exist.

A computer screen shot of a program code

Description automatically generated

***Methods in the Running class.***

***running()***

Provides a simple user interface on startup allowing different methods to be accessed. Loops unless 0 is chosen as the index.

A computer screen shot of a program code

Description automatically generated

***Methods in the StockBot class.***

***stockBot()***

Provides an additional user interface if Stock Bot is chosen from the previous menu. The user is prompted to choose one of three algorithms. The first being an algorithm of my own creation which uses three different heuristics. The second is buy as much as you can initially and hold until the exit date. The final algorithm is the same as the second but is a more active approach to trading where you sell when up in your portfolio to hold on to gains.

A screen shot of a computer program

Description automatically generated

***movingAverageCheck()***

Buying when the moving average is higher than the current price is not a guarantee to profit in the long run. However, it is safe and a higher moving average as far as I’m aware is an indicator in technical analysis that the stock will be moving up shortly. Personally, I think technical analysis is about as useful as horoscope analysis, but that opinion is up to the individual. This returns true when the moving average is higher than the closing price.

A computer screen with text

Description automatically generated

***fallingKnife()***

The saying is you can’t catch a falling knife. Sure, they can’t. But maybe we can? For the purposes of my stock bot if the closing price is lower than the opening price, we’re taking an extremely aggressive trading stance and treating that as a buy signal. This must be the floor price, right?

A screen shot of a computer code

Description automatically generated

***vibeCheck()***

This is the programming equivalent of a coin flip. The few successful traders I know, and my own admittedly limited success inform me that intuition is just as important as any technical analysis of stock, I can’t program intuition, but I can program a coin flip which is about the same odds as me making the right decision for buying and selling stock.

A screen shot of a computer

Description automatically generated

***rsiCheck()***

This is commented out and not currently in use with my trading algorithm, but I wanted to document it here to see my intention. It was supposed to read the previously calculated RSI for each day and send a sell signal if it was above 70, which traditionally means a stock is overvalued.

A computer screen with white text

Description automatically generated

The issue was previously mentioned that the csv reader kept throwing an out of bounds index error. I decided that 3 heuristics were fine for my purposes because as far as I know no one really has a successful trading algorithm anyway.

***buyAndHold()***

This is the only method that prior to testing I assumed would work for gaining money. You can’t beat the market you can just be in it, and time spent holding is the only surefire indicator of profit in the long term. If the market ever is permanently down that means America, no longer exists and if that’s the case I’m not really worried about stocks. This method takes the closing price of the first day buys as much as it can based off how the user designates the initial price. Then the stocks held are multiplied by the closing price on the final day being checked. We can skip all the interim days because we’re just holding and they really don’t matter.

A screen shot of a computer program

Description automatically generated

***buyAndHoldSometimes()***

This method is for the casual in all of us, the dabblers, the stock hobbyists. When you want to make money by spending time in the market, but you also want to play a little bit, if we’re being honest the stock market is just the best casino in the world and if programming stock options were in the range of my abilities this project’s scale would have spiraled out of control.

The logic for this method is the same as buyAndHold() except the interim days are checked to see if the portfolio is up 10 percent. If it is 20 percent of stocks held are liquidated and added back into liquid cash. No subsequent buying is performed past the first day.

A screen shot of a computer screen

Description automatically generated

***tradeEvaluator()***

I’ll admit it, this is the worst function I wrote for the entire project. It’s logic probably isn’t correct, its unnecessarily large, and the readability is lacking. I didn’t understand the importance of commenting until I wrote this, and I wrote this last. Troubleshooting was painful because following my own logic was borderline impossible without some way to follow what I was writing. The big picture logic is as such, the method is called and like the other algorithms a time period is specified for what the user wants to be checked to simulate day by day trading. Then the heuristics are all called and if they return true counter iterates on itself. The number counter is at determines whether to buy or sell stock. I wanted an aggressive trading stance because I just think that’s more fun, so if counter was at 0 it was a sell signal for 20 percent of the portfolio. If the counter was at 1 it was a hold signal. If the counter were at 2 or 3 it was a buy signal for 20 percent or 50 percent of your assets respectively.

A screen shot of a computer code

Description automatically generated

***tradeEvaluator() cont.***

Below is an example of one of the if statements depending on how many counters were available from the initial for loop. They’re all identical except for the amount bought/sold.

A black screen with text

Description automatically generated

***Methods contained in StockMain class.***

***stockMain()***

Creates a Stock Bot object then runs the stockBot() method.

A screen shot of a computer code

Description automatically generated

***Results***

Below is the CSV file after the data a moving average of 5 days and an RSI for the last 5 days were added in excel. This is not the entirety of the file, but the information all looks the same from here onwards. The RSI ignores the first N\*2 days, so no RSI is listed.

A table of numbers and numbers

Description automatically generated

This is the chart with open, close and moving average graphed for the entirety of the csv file. High and low are not included for readability and none of my heuristics take them into account.

A graph showing a line graph

Description automatically generated with medium confidence

The RSI was graphed separately for readability. The results are very swingy, but this is because RSI is typically calculated over a longer period of time than five days.

A graph showing a graph

Description automatically generated with medium confidence

***Menu that appears on Program Launch***

A black background with white text

Description automatically generated

***Using the initial Yahoo Finance CSV and calculating the moving average and RSI***

I chose a moving average of 20 days and an RSI of 14 days.

A screenshot of a computer program

Description automatically generated

***Apple***

***Running Buy and Hold***

Here I chose the previously created Apple csv and ran the buy and hold algorithm over 200 days from the start of the csv with the final portfolio value being 114975. A 10.4 percent profit over 2/3’s of the year is really good and beats the average market return of about 10% percent a year.

A screenshot of a computer

Description automatically generated

***Running the Trade Algorithm***

The trade algorithm is run on the same file with the same starting balance over the same time period.

A screenshot of a computer program

Description automatically generated

A black background with text

Description automatically generated

Those are not all of the values listed for the sake of brevity, just the final week returned. The actual program returns all the values in the terminal. A final balance of 124072, which is beating the buy and hold algorithm which was a shock to me. Then again one of the heuristics is chance so it could just be luck. We will run all the algorithms on another stock to confirm these results.

***Running Buy and Hold Sometimes***

This algorithm is run with the same parameters as the last two.

A screen shot of a computer

Description automatically generated

We actually lose money here, which is not surprising to me. The vast majority of people do not make the correct decisions to beat the market, myself included.

***GameStop***

All the following algorithms will be run with the parameters of 10 days RSI, 10 days Moving Average, 200000 in initial balance, and over 150 days.

***Buy and Hold***

We have lost about a quarter of our portfolio. Which is incredibly painful, but this is not surprising. GameStop is a meme stock and once the public lost interest in it it returned to its true value.

A screenshot of a computer

Description automatically generated

We can see these results are consistent with GME’s stock price. There is an obvious downward trend over a long period of time.

A graph of a stock market

Description automatically generated with medium confidence

***Trade Algorithm***

Much like buy and hold we lost money almost the entire time, which was consistent with the trend but somehow the last day we made a gain of 40000 dollars, which seems incorrect. This seems to indicate that the stock somehow had a spike in price which is plausible based on the graphs below.

A black screen with white text

Description automatically generated

***GameStop small price spike 11/27-11/30***

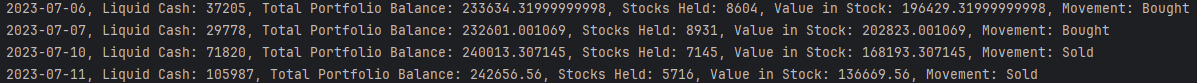
A screenshot of a graph

Description automatically generated

A graph with red line and numbers

Description automatically generated

Except the last day of the trade algorithm shows that we have almost no liquidity and our entire worth is in our stock value, which means buy and hold should also have had roughly the same value. There are periods where the trade algorithm has more value in its portfolio.



So, it is possible that it just had more stocks meaning the price spike was much more beneficial than buy and hold.

***Buy and Sell Sometimes***

This just consistently loses money, the moral of the story is don’t play with your stocks. Buy and forget about it or you lose money.

A screen shot of a computer

Description automatically generated

I will be running the algorithm’s on one more additional stock to test if the trading algorithm is actually working or not. I am beginning to suspect that there is a coding issue somewhere. It is incredibly unlikely/borderline impossible that I have accidentally stumbled on to a working trade algorithm.

***Reddit***

Reddit just recently went public with the stock only being available for purchase for 25 days. It will be interesting to see how the significantly shorter time period affects the trading algorithms.

The parameters will be 15000 dollars, 5 days moving average, 5 days RSI, and 25 days’ time period.

***Buy and Hold***



This lost money but not much and is consistent with the price from when Reddit went public to the current day.

***Trade Algorithm***

This finally lost money, I was beginning to get very concerned that it was completely broken. I still suspect that there is a logic error somewhere.

A black background with white text

Description automatically generated

***Buy and Hold Sometimes***

Once again Buy and Hold sometimes performed the worst, this doesn’t surprise me, and I suspected it would be the case.

A screen shot of white text

Description automatically generated

In order of performance for all three tests the algorithms go Trading Algorithm> Buy and Hold>Buy and Hold Sometimes. That being said, if I had to pick a method to trade, I would still pick Buy and Hold as that’s the only one historically proven to work over long periods of time and the logic is not suspect. I know for a fact there was no way I could do something that consistently made money on the stock market, when there are people significantly smarter and with exponentially more knowledge than I will ever have who have been attempting to do so. It was a relief when it failed to make money on the GameStop test.

I learned a lot from this project and going forward I hope that I make things I’m actually proud of instead of something like this. Even if this was the best I could do.