Mia Watts April 26, 2024

**Stock Bot Documentation and Results**

One of the final pieces of the project was a Stock Bot in which we were to program, from scratch, a bot that would track stocks and tell users whether it was a good time to buy, sell, or wait. A .pdf file with instructions was used to program the Stock Bot and will be included for reference within the documentation files in GitHub.

**Current Result Graphs**

**Classes and Method Descriptions**

There are three classes associated with the current stock bot iteration. These classes run the bot, read data, or print a current date. The StockBot class is the one that runs the bot, the StockReader reads data and contains getter and setter methods, and the StockEvaluator returns a current stock result for a specific date (the closing and opening prices for the data). Each of these can come together to print information, calculate RSI (Relative Strength Index), and determine whether to buy, sell, or hold a stock.

* Methods in the StockBot class:
  + calculateRSI() – calculates the RSI for a given series of stock data. The RSI is calculated using formatting from the RSI site given within the instructions. The method takes two parameters, the first being the data list and the second being the quarter where the RSI is calculated (N value, or range or dates).
  + buySellHold() – determines whether the user should buy, sell, or hold their current stock based on the RSI that has been calculated. If the RSI is less than 30, buy. If the RSI is greater than 70, sell. Otherwise, hold. Takes RSI value as a parameter.
* Methods in the StockEvaluator class:
  + readData() – Takes a filename as a parameter and reads stock data from the file.
  + calculateMovingAverage() – Using the site provided within the instructions as guidance, the moving average was calculated to return an average across a specific range of values (very similar to a smoother function like those programmed within other parts of this project. I would like to reprogram a lot of this to work with my original smoother programs).
* Methods in the StockReader class:
  + getDate() – returns the date.
  + getOpenPrice() – returns the open price for the current date.
  + getClosePrice() – returns the closed price for the current date.
  + setDate() – sets the date to a new one passed as a parameter.
  + setOpenPrice() – sets the open price to a new one passed as a parameter.
  + setAdjClosePrice() – sets the closed price to a new one passed as a parameter.
  + toString() – this is an overridden method that prints the current stock data into the console in a formatted way. The date, closing price, and open price are displayed with labels when this method is called.

**Console Results**

When running the bot’s main java file entitled “StockBot.java,” results are printed to the console. The RSI value calculated when using an N value of 14 can be rounded to 45.2, thus meaning that currently, with the given data, it would be best to hold rather than buy or sell Nintendo stocks since these stocks are quite slow in movement. The results from the program are shown below.

A black screen with white text

Description automatically generated

Current results aside, the program is written to change the statement below the RSI value depending on what that value is. If the RSI value is less than 30, the program will tell the user to buy. If the RSI value is greater than 70, the program will tell the user to sell.

**Smoothing Results**

After calling the smoothing algorithm from the Java plotter, salter, and smoother algorithm part of the project (and editing it to work with the stock data), the closing prices (and any other prices placed into the method call), were changed to show the averages to the left and to the right of the data points. The graph produced is shown below.

As shown by the orange line, the smoothed results are closely related to the data but instead show the averages rather than the actual data points (more clearly seen near increment values 35-47 where jagged edges on the blue “close” line are now positive trending and smooth).