Luke Marolda: 15-112 Term Project Proposal

Project Description

The name of my project is Intellitrade. It is an interactive stock trading game, in which a user can choose from a variety of historical stock data, and can play a game in which the goal is to make as much money possible by trading that stock using technical analysis. The main component of the project is the user interface, which will display the stock's price action over the period of a year, as well as a variety of basic technical indicators for your aid. You will be able to buy and sell, view your account information (profit/loss, etc.), view the stock's price graph at different speeds and different zoom levels, view simple moving averages with a period of your choice, view exponential moving averages with a period of your choice, view the moving average convergence index, and the RSI indicator. Keep in mind, all of this is accomplished without any external modules. Additionally, you will be able to pause and take notes/draw support and resistance lines on the charts, and the game will have an AI versus mode, where you play against the computer which will be buying and selling the stock based on a basic machine learning model from data provided by the indicators.

Competitive Analysis

The first project I will be analyzing is EBITA: The Historical Stock Trading Game by Gabriel Rodan. This game has some similarities and differences to my personal project. One key difference is his use of google finance and panda modules to support his game, as I am planning to create mine with the use of no external modules in order to challenge myself and add more algorithmic complexion. One similarity between our projects is the graphing feature of historical price data, however, there are still many differences. Both of our price action graphs have the ability to zoom in and out, and present the data at different rates. However, my price action is displayed differently, as his includes all data points for a full year, where mine has the ability to take different sizes of data, which is more commonly used on a traditional trading platform. Additionally, my project makes use of many more technical indicators, and provides more customizability on which indicators and drawings you an include, which makes playing the game much easier for a novice investor. His game includes options trading, which mine does not, and his also includes a graph at the end of the game, analyzing your performance, which mine will not. However, mine will have an AI versus mode which will add another level of complexity and competitive aspect to the game. I also am hoping to make my project more visually appealing than his, making it more enjoyable to play, and making the player feel like a real technical investor.

Another project similar to mine is PayOff, by Kunal Jobanputra. Our projects are less similar than the previously described, but they still share some key commonalities. One similarity is the graphing features for certain stocks, but his uses real data compared to my historical data, and his charts are created via an external module, compared to mine which uses no modules. His game has a feature which uses machine learning to predict a stock's value over the next certain amount of days. Although my project will also feature something similar, mine uses a less complex machine learning model, but is much more interactive as the AI versus mode actually buys and sells the stocks to achieve a maximum P/L ratio, whereas his version does not. Additionally, it is clear that my game focuses much more on the displays of price action and indicators, and tries to replicate a real trading platform, whereas his project seems more like an app that would accompany a trader.

Structural Plan

My project will be organized using the CMU graphics package and will have one main file with the math indicator, stock graphing, and user interface functions. An additional fire in the same directory will be used to house the csv to list function, which takes all of the csv files containing my stock data and translates them into usable data as a 2D list database.

In my main file, my functions will be split into two main categories: technical functions, and visual interface functions. My technical functions will include the bulk of the algorithmic complexity for the project, including the math indicator functions which return data points for indicators of my choosing, the graphing functions, which include the scaling function which takes raw data, and returns points to be graphed on the user interface, the account information functions, which include calculating current profit and loss, etc, and the AI versus mode functions, which include buying and selling calls based on a set of conditional rules as described in the algorithmic plan.

The visual interface functions are less algorithmically complex and are the typical Tkinter animations functions used to create the interface. These functions take the data found from all of my technical functions and presents it in an aesthetically pleasing manor. These functions also include all of the buttons and interactive pieces of the interface, which will control the model.

Algorithmic Plan

In my opinion, the most complicated portion of my project is the graphing of a stock's price action and indicators without using any help from external modules. The complexity lies in the scaling of the stock data. For example, one stock may range from $100-150 a share during a given period, whereas a different stock may range between $1-5 a share over the same range. It is a little bit tricky to get build a graphing algorithm that properly plots your data despite these differences in ranges, and also can display the price action at different speeds and zoom levels.

To solve this, I created my own graphing algorithm that solves all of these issues. Essentially, the bulk of the work is done by a scaling function, that uses proportions to take the ranges of stock data and plot them appropriately in one coherent window. Essentially, the algorithm takes a range of data for a stock over a specified period of days, and then finds the local maximum and minimum values of the stock over that period of time and finds the range between those values. Then, for every point of price data for the stock in that period, it calculates the percentage of change in relation to that calculated range, and then multiplies that percentage by the height of the window in which you want to display the stock price action. This will map each small range of data with proper proportions in the user interface, ensuring that all stocks and data for those stocks are presented with the correct proportions in their window. Then, to actually show the price data over time, the scaling algorithm just moves through the 2D list database of stock info, and slowly increments through all the data, increasing the starting and ending positions of the period of data you are looking at, which is how the program gets the price action to move across the screen coherently. This allows the user to change two things, the size of the period of data you are looking at, which allows for the "zoom in" and "zoom out" feature of the graph, and the rate at which the algorithm moves through the stock data, which allows for the "speed up" and "slow down" feature of how fast the price action is displayed.

The other most complex algorithm for my project is the AI versus mode. However, since I mathematically defined all of my indicators in their own functions, it will be easy to tell the computer to trade based on a set of rules analyzing data from these indicators. Here is the simplified version of my model: buy whenever two different period SMA's cross over one another, and sell at the next found cross. In addition, this buying and selling can only occur when the MACD has a positive value (or negative, depending on if it is shorting the stock), and potentially when the RSI is above the 70 level or below the 30 level (once again, depending on if the computer is shorting the stock). This is essentially a set of rules the computer will follow, and base its trades off of those rules in order to maximize profitability.

Timeline Plan

Here is a general timeline of when I plan to complete the most complex and basic features of my project:

<u>TP0 (by Monday, November 23rd):</u>

A this time, I must have completed my proposed project design, basic sketches and descriptions of the project's features, collected all of the stock data I will be using, coded basic versions of the functions which take my CSV stock data and convert it to a 2D list database, and the math functions which calculate simple moving averages and exponential moving averages for a function, as well as the moving average convergence divergence index.

TP1 (by Monday, November 30th):
At this time, my main goal is to complete my graphing algorithm and a basic user interface, which properly graphs a stock's price action over time, with the ability to change speed and zoom settings, and which also correctly displays a variety of indicators based on my math functions. This is the most complex part of my project, so completing this now will put me in a solid position for the rest of the project. I also aim to have made some progress by now on the simpler aspects of my project, such as game over screen once the stock data has finished.

TP2 (by Saturday, December 5th):
By this time, I must have completed my minimum viable project. This includes reading through CSV files to get data with past stock prices, an interactive interface, where users can take notes/highlight support and resistance lines and other trends, a graphic display on stock price overtime (without module), and a basic game AI versus mode that can compete with the player by following a set of rules based on data acquired from the math indicators.

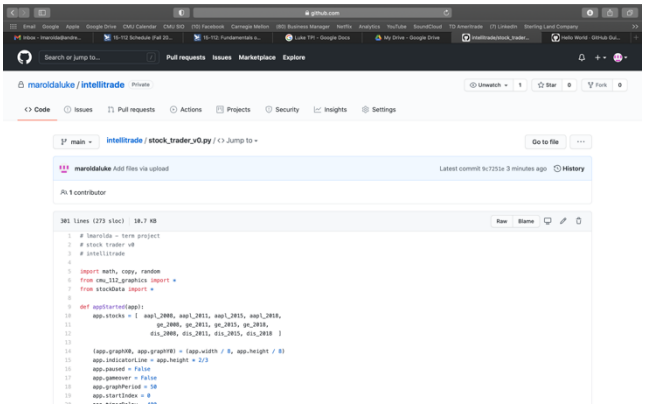
TP3 (by Wednesday, December 9th):
At this point, I must have a fully polished working game with minimal bugs. This includes fixing any bugs or issues with my code, improving the visual aesthetic of the platform, adding more complexity to my graphing and AI versus algorithms, and adding any other cool features I want, such as more technical indicators or more historic stock data to choose from.

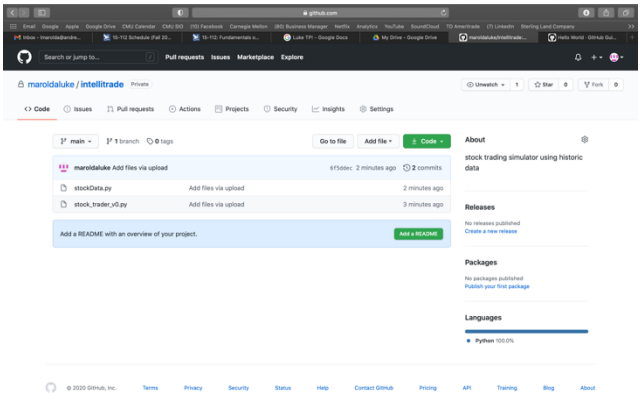
Module List

I will be using no external modules for this project!

Version Control Plan

To back up my code, I created a GitHub repository to store the code and information for my project. Whenever I make serious project on my code, I will upload my work to this repository, which will keep my code safe. It is a private repository so nobody else can view it but me. Here are some pictures:





TP2 Update:

Since TP1, I went from 350 to 1100 lines of code, and created a fully functional user interface for the game. The only main change is that I am going to add the AI mode in between TP2 and TP3, because as my project grew, I realized the AI mode was much less important, so I want to finish the interface and make sure I have a working game before adding that.

TP3 Update:

Since TP2, I added the AI versus mode, a logo for the game which is imported as an image, a game description tab which explains a bit about the technical indicators, and then some minor UI modifications. The most impressive of these changes is certainly the AI mode, which automatically trades the current stock using a basic SMA cross and MACD signal strategy, which has a low alpha but is good for the game's purposes. The AI is profitable around 60% of the time, which is solid considering the trading strategy I implemented.