Victor Morgan
10/19/25
CSC-570
Dr. Osunmakinde

Conversion to my AI field of choice

For my Multi-Linear regressor, I chose to create a program that after being trained on a set of data provides users the option and ability to predict the closing value of a stock. This follows my theme of finance, and would allow users to determine if they should sell their stock before the end of the day, depending on if it's estimated to go up, down, or remain the same.

The PEAS are as follows:

Performance : The accuracy of the program's ability to predict closing prices, as measured by the R^2, MSE, and the displayed line of best fit.
Environment : The pre-loaded stock data csv, **Netflix.csv** that the model trains on, and the **stockBatch.csvs** that the model accepts from users to predict new close prices with.
Actuators : Predicting the closing stock value price for new entries. Displaying previous stock growth visually. Allowing model evaluation from **stockBatch.csv** at the user's request.
Sensors : The csv file used to train and test the model, as well as the input space for users and the additional csv files users may upload.

As I'm dealing with stocks, I didn't want to just encode the dates. I instead parsed them, so that they're interpreted by the model as intended. I separated the year, month, and day, and passed them to the dataframe used to train and test the model. I did the same for user input, when they manually predict closing prices, or do it automatically by uploading a file. I also check the date formatting of the file and handle potential errors accordingly, so the whole program doesn't crash if a csv file has an incompatible dating format.

Feature Engineering

For this assignment I continued to focus on the user-friendly aspect of how my AI should function, continuing to make my program more flexible, and resilient to improperly formatted user data.

I allow the user to request feedback as they see fit on the new data the model has performed a prediction on. This only works if the user has entered a csv file with the actual closing prices enclosed. If the user did not enter a file with the closing prices, they're able to try again when they upload their next batch file.

I've also implemented some code to stop the consistent checking of a new file. After waiting and iterating 5 times, the code will now ask the user if they want to continue running the program. If yes, the program will continue for another 5 iterations, if not it will terminate.

Lastly I implemented a line plot rather than a scatter plot to represent the growth of the stock over time, based on the initial csv file provided. This allows the user to see the stock as they typically would, making this a more finance compatible application.

Conclusion

This model builds on all of the previous ones I've worked on. It includes extensive user input validation for resilience and ease of use. It also features checking of files provided, and catches certain errors that are likely to occur. I decided to continue to give the user the choice of what they want to do with the program, this way the program can meet their needs as best as possible. I prefer semiautonomous programs for that very reason, as I might not fully anticipate all of the user's needs and wants.

When I took CSC-170L with you, I learned that the point of any program is to make the user's life as easy as possible. I learned the program should be interactive, and I've always had an increase in my program's communication with the user since then. I ensure the user knows what's going on. If something doesn't work, I output the error message. If the user's input is invalid, I let them know and provide a guide for how they should format their text. This is to reduce confusion as much as possible and keep their interaction light and easy.