Predicting when it is profitable to do a "straddle" on stocks in the S&P 500 index

For my CS021 final project, I chose to try and find a profitable trading strategy using Python. To do this, my first thought was to train a neural network to predict when to buy or sell stocks. Upon further research, I discovered that this is very hard to do given the amount of noise in stock market data. To work my way around this I found a subset of stock market activity and stock market data called options trading. I looked at the many strategies traders use for options trading and found one that seemed like it would be a good fit called the “straddle strategy”. This strategy involves buying a call option (a contract to be able to buy a set number of shares of a stock at an agreed upon price) and a put option (a contract to be able to sell a set number of shares of a stock at an agreed upon price). With a call, the trader is betting on the price of the stock going up, so that they can buy the shares at the lower agreed upon price from the contract. With a put option, the trader is betting on the price of the stock going down, so that they can sell the shares at the higher agreed upon price from the contract. When these options are bought together in a “straddle”, they offset each other and the trader profits if the price of the stock moves sharply in either direction. This strategy presents a unique set of data that I found is useful for training a neural network model. To train the model, I first process the data using the process_data program. This program pulls out relevant information from the raw data I downloaded from Cboe Global Markets, such as volatility, break-even prices, and whether the strategy was profitable for a particular stock at a particular time. I then train the model using the model program and test its accuracy against random prediction accuracy, guessing 1 (buy) everytime, and guessing 0 (don’t buy) everytime. Then, I use the market_test program to test the network’s performance over a certain time period in the actual market. I make the same comparisons with its performance as I did with its accuracy and show the results in a graph. Based on the results from this and from when I compared the accuracies, it is clear that the model outperforms random guessing, buying every time, and not buying every time.

References

alpacahq/alpaca-trade-api-python. (2020, December 3). GitHub. https://github.com/alpacahq/alpaca-trade-api-python

API reference — pandas 1.1.4 documentation. (n.d.). Pandas.Pydata.org. Retrieved December 4, 2020, from https://pandas.pydata.org/docs/reference/index.html

https://www.facebook.com/jason.brownlee.39. (2019, July 4). *Keras Tutorial: Develop Your First Neural Network in Python Step-By-Step*. Machine Learning Mastery. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

NumPy Reference — NumPy v1.19 Manual. (n.d.). Numpy.org. Retrieved December 4, 2020, from https://numpy.org/doc/stable/reference/index.html

Stack Overflow - Where Developers Learn, Share, & Build Careers. (2019). Stack Overflow. https://stackoverflow.com/

Team, K. (n.d.). *Keras documentation: Keras API reference*. Keras.Io. https://keras.io/api/