Enhanced Stock Prediction and Algorithmic Trading System Using Cointegration Testing and Stock Chains

-Ryan Mennmeier
-Sept 2024

Introduction

In the highly competitive and volatile technology sector, accurately predicting stock price movements is critical for investors, financial analysts, and portfolio managers. However, traditional prediction methods often fall short due to the complex nature of financial markets and the numerous external factors influencing stock prices. In 2023, global market volatility in the tech sector contributed to an estimated $1 trillion in investor losses, underscoring the need for more advanced tools. This project, "Enhanced Stock Prediction and Algorithmic Trading System Using Cointegration Testing and Stock Chains," aims to bridge this gap by developing a robust, integrated predictive modeling system that combines advanced machine learning techniques, such as GRU and Transformer models, with cointegration testing to form stock chains for key technology companies like NVIDIA, Apple, Google, Microsoft, and Amazon. By incorporating exogenous data and secondary stocks for cointegration, this system seeks to improve forecast accuracy, providing actionable insights that can enhance decision-making and optimize investment strategies, with the potential to reduce financial risk by up to 30%, according to recent studies on the efficacy of machine learning in stock prediction.

Brief Resulting Overview

As time became critical I was not able to witness some of the key planned components of this project. At the time of this write-up summary the selected Stock Chains have been completed, though the output has not been fully discerned. Some of the more advanced and planned models in GRU and Transformers have yet to be implemented, as we need to develop our trading strategy based off of the Stock Chains and then perform backtesting first. There are plans to continue this project as I am very interested in the outcomes and how this project progresses.

Speaking of which, one of the highlights of this project was seeing the results of the cointegration tests. Taking core stock data and then performing cointegration tests with our secondary stock data and getting positive results (in various degrees) was really neat to see, as cointegration often has a very low success rate. It was also interesting to see which of our core stocks cointegrated more than others with the same set of secondary stocks, as well as which core stocks cointegrated with some of the exogenous features that were used.

I have identified several areas of improvement just in the portion of the project I have completed, and I have a plan to implement these revised strategies when next I continue to work on this project.

Key Intended Stakeholders

-Investors
-Financial Analysts
-Portfolio Managers
-Data Science Team Members

Implementation details and more decision results can be found below in the GitHub repository link:

GitHub Repo Link

https://github.com/Taliwat/Portfolio-Proj-WK/tree/main/DL-Strategies

Approach

Data Acquisition and Wrangling

Data for this project was acquired entirely from the **yfinance library**, with two separate downloads and then saving and generating my csv files once I had acquired the requisite data history. I then used the original csv files to create my additional files as I moved into further phases of my project. There are many other finance APIs available but quite a few come with costs associated per usage (at least for the data you really would want to use). For the project thus far yfinance proved to be an efficient use-case as it is free and had the data we required.

Storytelling and Inferential Statistics

In this project I picked the following “core stocks”, listed and then represented by their stock ticker:

-Apple [AAPL]
-Amazon [AMZN]
-Google [GOOG]*
-Mastercard [MA]
-Meta [META]
-Microsoft [MSFT]
-Nvidia [NVDA]
-Pfizer [PFE]
-Procter & Gamble [PG]
-Tesla [TSLA]

* Google has two separate stock tickers in GOOG and GOOGL. One is Class A with voting rights and the other is Class C with no voting rights. For the purposes of this project we are just using the Class A version.

For each core stock I downloaded a 5 year history from 01-01-2019, and ending 08-15-2024. I wanted my core stocks to be from entities that were widely known, as well as had a bit of diversity amongst their sectors as a group. Each core stock has its own EDA notebook for individual analysis, as well as its own baseline modeling notebook. Each core stock will act as the root node in the Decision Tree and stock chain, so at this time it is important that they remain separated.

Regarding the actual data wrangling I kept the data download whole so I could perform basic operations and checks on it. I looked for missing values and imputed and interpolated as necessary so that I had a complete csv file to work with upon getting to my EDA stage as well as baseline modeling.

Regarding EDA I wanted to maintain consistency in my process so it would be easier to see any trends amongst the plots I produced, so I performed the same plots in each notebook for each core stock. Results were observed independently.

Feature optimization was key as once the original features were customized and lagged we had 153 features in our dataset. I created a separate notebook that performed PCA, RFE (Recursive Feature Elimination), VIF (Variance Inflation Factor), and MIS (Mutual Information Scores) to determine the optimal number of features for my data in regards to my dependent variable in ‘Close’, then filter them down according to relevance and their collinearity. I wanted to make sure that the features I ended up with though still predicted well with the target so I introduced MIS at the bottom as a check for the final list. I would repeat this process over and over, filtering and adding features that were over the VIF threshold or that predicted unreasonably low until I (finally) acquired my desired amount of features that met my VIF threshold AND predicted well.

Baseline Modeling

For the baseline model it was also performed individually on each core stock we had, so there are ten separate analyses or notebooks. I chose Linear Regression with Ridge for this baseline model, and incorporated the optimized feature list (25 of them in quantity) previously derived in my other notebook as my X. Tabular results for their metrics can be found below, rounded to the nearest ten-thousandths:

Ticker	Train RMSE	Test RMSE	Train MAPE	Test MAPE	Train R2	Test R2	Train Adj R2	Test Adj R2
AAPL	0.0006	0.0007	0.6 %	1.0 %	0.999	0.999	0.999	0.999
AMZN	0.001	0.001	2.1 %	0.3 %	0.999	0.999	0.999	0.999
GOOG	0.074	0.09	68 %	40 %	0.95	0.50	0.95	0.50
MA	0.0007	0.0008	0.05 %	0.05 %	0.999	0.999	0.999	0.999
META	0.0017	0.0018	0.6 %	0.1 %	0.999	0.999	0.999	0.999
MSFT	0.0005	0.0008	0.15 %	0.05 %	0.999	0.999	0.999	0.999
NVDA	0.002	0.007	0.08 %	8.2 %	0.999	0.999	0.999	0.999
PFE	0.0005	0.0005	0.04 %	0.03 %	0.999	0.999	0.999	0.999
PG	0.0004	0.0004	0.7 %	0.1 %	0.999	0.999	0.999	0.999
TSLA	0.01	0.01	1.6 %	1.5 %	0.999	0.996	0.999	0.996

These results can be summarized in the following:

-Using optimized features with Ridge allowed for an enhanced version of the Linear Regression that provided a setup for our dataset(s) that accounted for variance, collinearity, overfitting, and predictive power.

-Utilizing the optimized features instead of the base 153 provided the Ridge a much better assessment of which features to regularize, and provided better output metrics in the end.

-Note the anomaly metrics we have in GOOG (Google) and to a point other Test MAPE readings for other core stocks. GOOG performed much worse than its counterparts, however I take this as a good thing as this sort of variant data actually gives credence to the otherwise excellent data metrics from our other core stocks in that the process is more natural instead of superfluous. If the above-average metrics had been consistent across the board I would have wanted to look into the data again for suspect data leakage or overfitting. Using a subset of features that are optimized reduces the doubt involved with this notion, and provides faith that we can continue even with the GOOG output especially knowing that with further phases of modeling that it will improve.

Extended Modeling

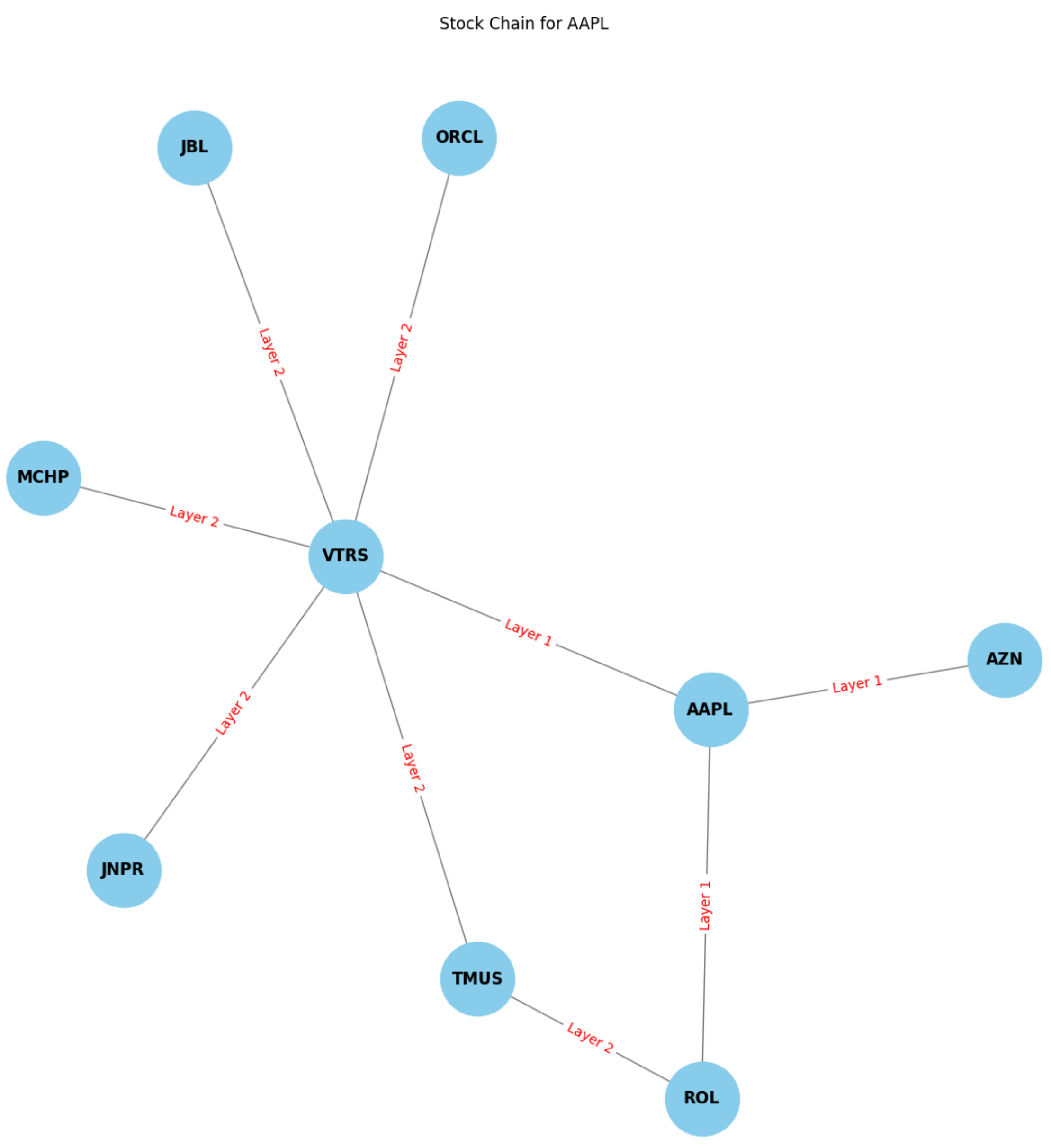
In this phase we moved onto cointegration testing and the construction of our stock chains. For cointegration testing I was very excited about this portion as I wanted to see how the generated list of secondary stocks tested with the core stocks we had been working with. I first performed an ADFuller test to make sure each core stock was stationary, and tested for cointegration on both our secondary stock list and exogenous data separately first. Results varied, and can be found in the below table:

Stock Ticker	Sec Stock Pairs	Exo Feature Pairs
AAPL	3	0
AMZN	1	0
GOOG	6	3
MA	40	5
META	0	0
MSFT	8	0
NVDA	4	0
PFE	0	1
PG	33	4
TSLA	5	0

To be noted that **the above results are only for the Layer 1 cointegration**, with no modifiers. The p_value threshold is set at the default at 0.05 and there are no limits on how many pairs can be created in this first layer, as it runs through until no pairs can be found further.

As I performed tests on successive layers it became necessary to put such limitations on the p_value modifier as well as the maximum pairs allowed per test pair so that the chain could stand a chance of being 'pruned' and stopping sooner rather than later. This proved easier said than done for even the smaller sampled core stock pairing test samples. **With the p_value threshold set all the way down to 0.01, and max pairs allowed per test pair at 3**, I still decided to stop my chains at Layer 4 as they were continually cointegrating. Part of this was due to pairs cointegrating with themselves in an A-B scenario where in one layer A would cointegrate with B, then in the next layer where B was the test pair B would then cointegrate back with A, and then repeated. I attempted to put in some gatekeeping logic into my main function to prevent this from happening as well as trying to keep unique pairs, however I was unsuccessful at this time from doing so. I will go back at a later time and look more into it.

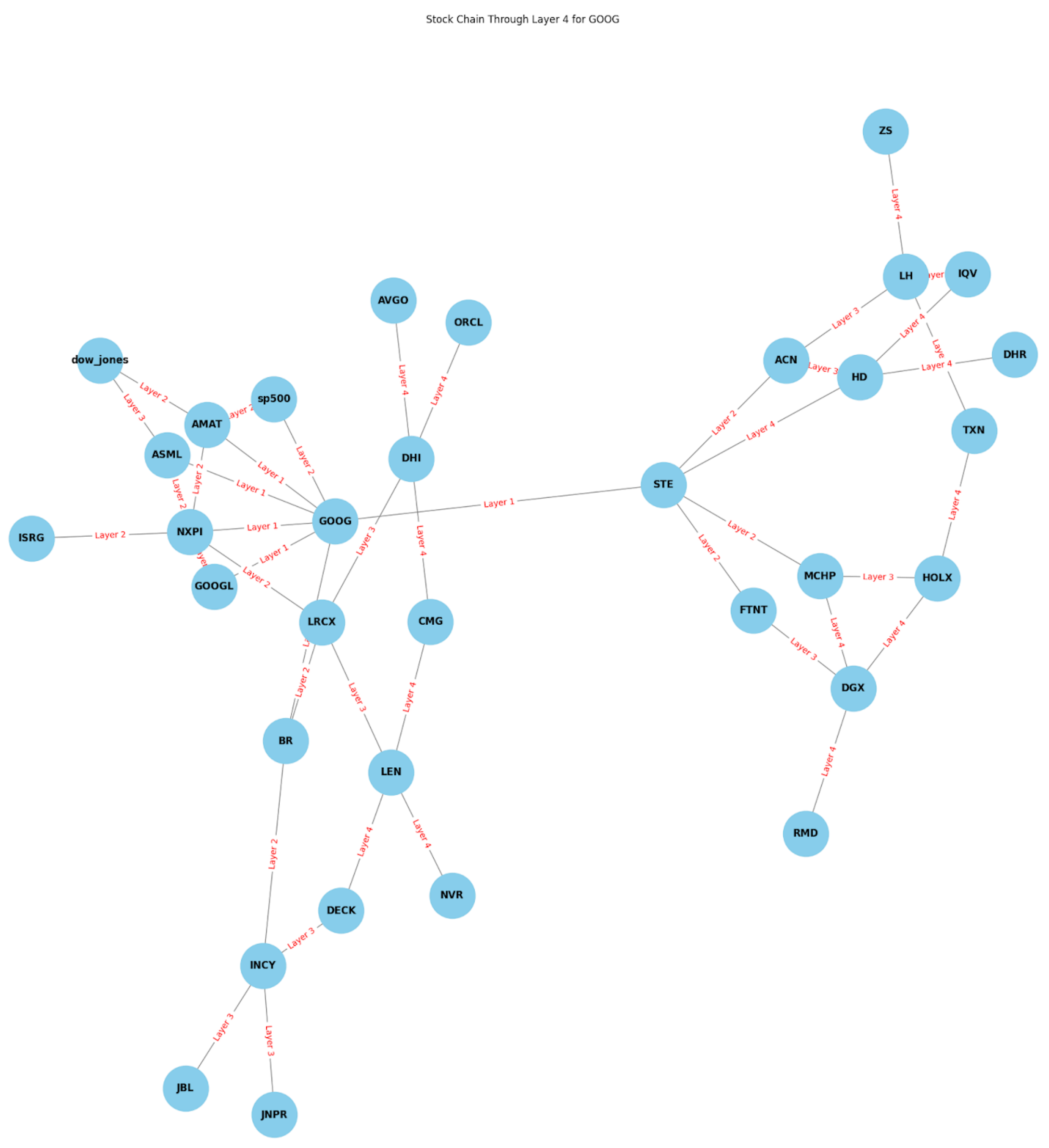
Speaking more about the stock chains themselves, I kept track of my output from layer to layer through Layer 4 (my line in the sand), then wanted to graph the output. I used networkx with matplotlib. I didn't do all of the original core stocks, as looking again at the chart above some were unnecessary or wouldn't be conducive to a good plot. For example AMZN, META, and PFE have either no cointegrated pairs or too few to warrant pursuing further layering (I even tried AMZN and it didn't cointegrate further). On the flipside of things MA and PG have just too many at this time in Layer 1 to create a presentable graph chain (they have 45 and 37 pairs in their first layer, respectively) so I stayed with the conservative core stocks, and they still proved challenging to keep contained through Layer 4. I pursued AAPL, GOOG, and NVDA to the finish and will post them below:



-AAPL Stock Chain Through Layer 4-

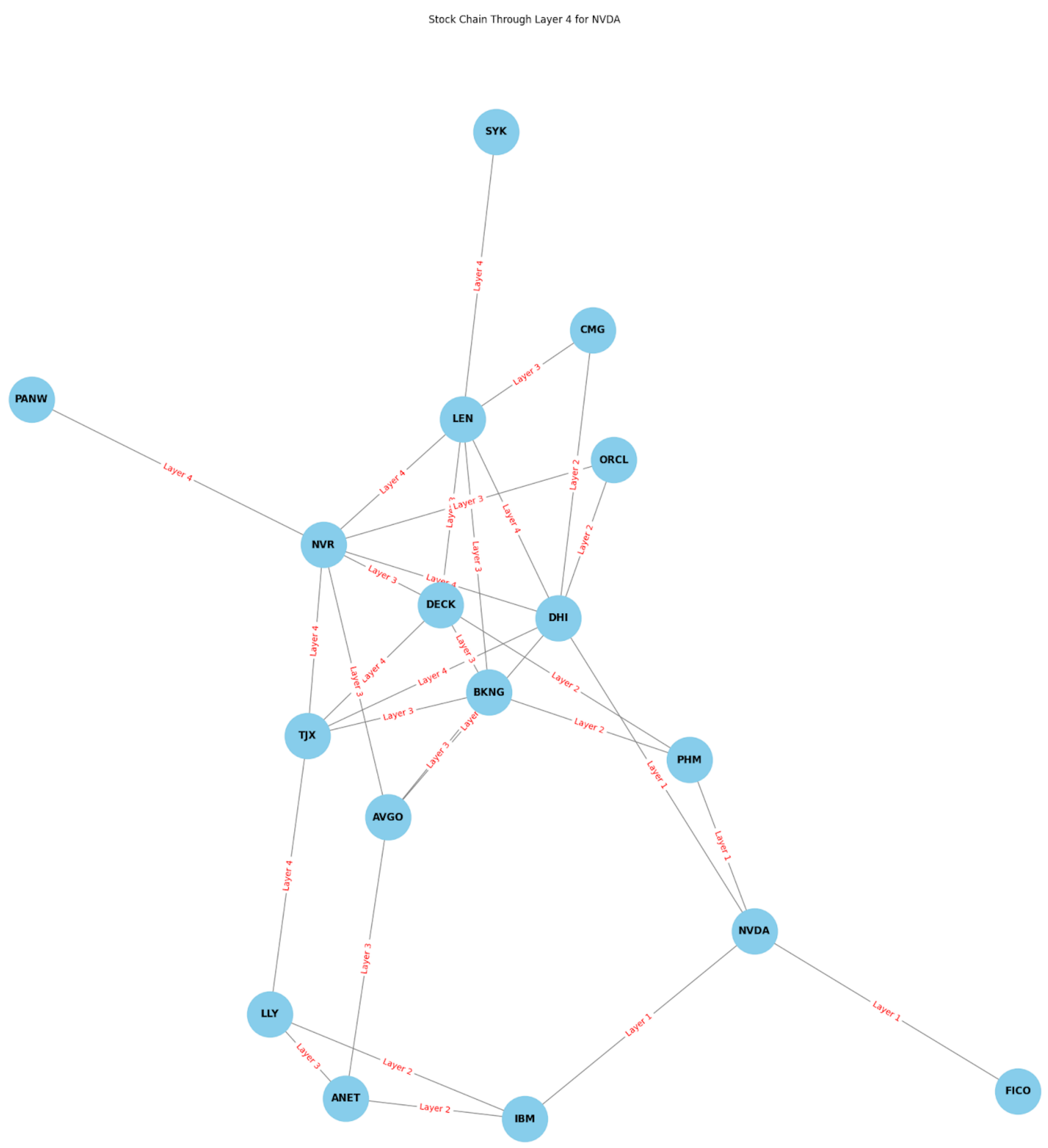
Here is the best tree in my opinion as it's not too convoluted or messy. Nice and simple. You can see the denotations showing the relationships between each node and the Layer # in which the cointegration was formed. It is interesting to see the cointegrated relationship between AAPL and its Layer 1 pairs in ROL and VTRS, and that they each share the same pair in TMUS in Layer 2.

-GOOG Stock Chain Through Layer 4-



Here in Google's stock chain even with just 9 starting Layer 1 pairs the chain becomes increasingly more complex. Relationships are now becoming intertwined (described very well by networkx) and you can see the clear separation of relationships once the Layer 1 pair STE is shown, as nothing routes back to Google after that it is its own little community.

-NVDA Stock Chain Through Layer 4-



Finally here in NVDA's stock chain graph you can see that there is a lot of gravitation towards several of the pairs and that they share pair relationships with each other, as they keep cointegrating back and forth. It will be interesting to see in later aspects of this project how these relationships develop inside of a trading strategy and how it forecasts.

Conclusions and Future Work

This section will be fairly lengthy, as I have only completed about half of what I set out to in this project. I also have uncovered a lot of “what-if” scenarios in the topics I have done so far. Presented in the form of a bullet list:

- In conclusion, this is a very promising start I believe to this project. We successfully developed stock chains with our core stocks we used all through this project. Exogenous features even showed to integrate at different levels and layers (more work can be done here). The development of the stock chains harkens back to the main problem established in our proposal, in that with the stock chains we can now look between known cointegrated pairs to see what those inter-dependencies are and input those in our coming trading strategy for testing. This pending trading strategy, with the insights derived from the stock chains, will prove to be more robust and efficient because of the work accomplished creating said stock chains.

- Develop Trading Strategy from the Continuation and Finalization of Stock Chains
-this one is pretty big, and will take a steep time investment. However the development of both an aggregate trading strategy and one per stock chain will be paramount for our next step(s).

- Perform Backtesting on the Trading Strategies
-this is where we will finally start to see how successful our work is. Yes we looked at a baseline performance marker in our Linear Regression models, but with Backtesting we get to see how our completed stock chains work as a unit with a customized trading strategy.

- Perform GRU on Stock Chains for Forecasting
-once satisfied with the Backtesting results (or to a perfunctory degree), send each chain through GRU to forecast how they would do going forward.

- Keep customizing the project and adding to the data, by for example changing the data to 10 years of history instead of the 5 years we have now to encompass more market events and to make my core stocks more robust. Also I would like to add more core stocks to the pool, and increase the secondary stock pool etc. I also in hindsight would want the cointegration and stock chain development to be performed with a sector bias, (each stock ticker has its own sector) to see if the strength of its chain and the performance therein increases with in-sector cointegration pairs.

- I would also like to move this project to the Cloud (service notwithstanding between GCP, AWS, and Azure), and hook in real-time data, so that the models are updated based on the day's market data. Some of the feature optimization methods (speaking to Boruta/Random Forest here) were not able to be performed because of the massive amount of time needed; if Cloud is enabled this process can be run autonomously until completion to compare results.

Recommendations For The Clients

Going forward I would recommend the following based on the work performed thus far and the outcomes achieved:

- I would recommend moving forward with the trading strategy development, starting first with the AAPL stock chain then moving into the GOOG and NVDA chains. Develop separate strategies for each then backtest appropriately. Prune out the less efficient nodes and pairs in the chain to make the chain more efficient and forecast with GRU.
- Once the stock chain(s) have been optimized, run in a simulator (paper trade) for a term of at least ~3 months to establish key metrics such as Sharpe's ratio (balance of risk and aggression in your strategy), and monitor performance and make adjustments to your strategy as you go.
- I would recommend finally a thorough check of all the data and models, on a regular basis especially for overfitting as most of our technical indicators used are derived from our target in Close Price. On that note new and different indicators could be developed and implemented that provide value that aren't as closely related to our target, and run in our feature optimization notebook to see how they compare.