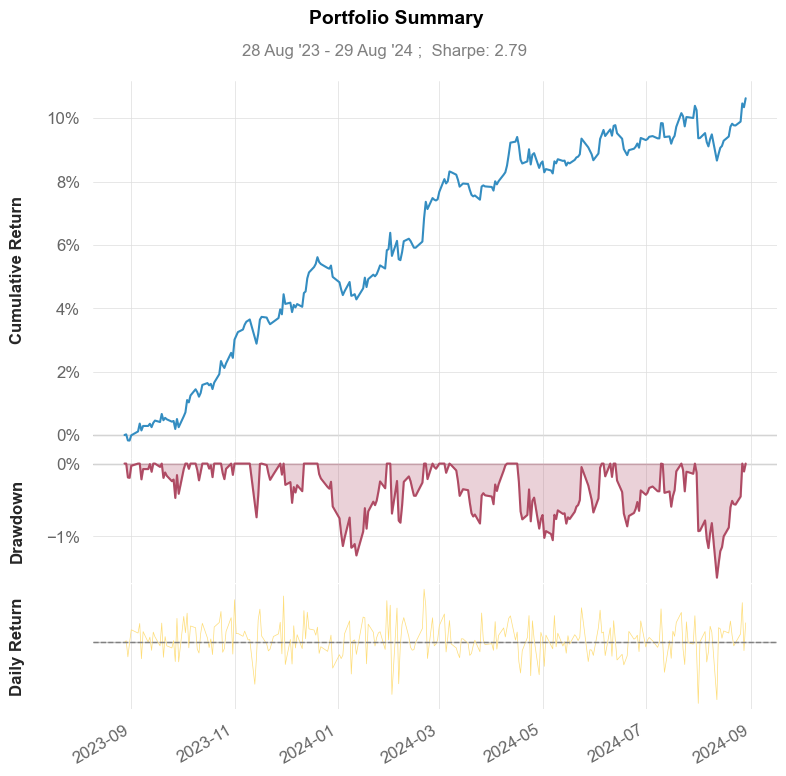
# IMPORTANT

We have done the two projects as given in the email; however it does not show the actual value I can bring with me in building out a systematic equities statistical arbitrage desk. As discussed with Costin on our first call, I have built a proprietary framework for auto alpha generation – which I have tested in the crypto markets that showed a live Sharpe of 5. This is something I would like to deploy within the equities space. I also believe, my framework can be very useful as I can single handedly deploy the infrastructure + get the strategy live generating PnL very quickly.

**A Jupyter notebook for the backtest is provided in /project\_1\_v3/ml\_strategy\_ls\_selected\_v2.ipynb**

I have partially modified my crypto system to auto-generated alpha signals and ran basic tests for equities and it’s showing very promising results. See performance below:

**Performance: 2.8 Sharpe with very low drawdown from Aug 2023-Aug 2024**

A summary of the approach is as follows:

1. Universe – Nasdaq 100
2. A subset of universe (tradeable universe) is created using a **proprietary approach based on Network Theory**. This approach tries to create a subset of the universe which is tradable and is referred to us ‘Community’. We utilize Louvain algorithm to identify these communities.
3. **Network Theory Advantage:** Unlike traditional correlation/cointegration approach which only compare a pair for linear relationship, network theory allows one to capture linear and non-linear relationships at the universe level. One can expand this to a multi-dimensional problem where similar stocks are selected by comparing their risk factor exposure (BARRA model).
4. **Auto Alpha (proprietary IP):** We have partially modified our crypto trading framework to auto generate some alphas. This approach can be very useful as I can come in and single handledly set up a trading desk, rather than having multiple researchers who will manually design alpha signals.
5. Auto alphas are then fed into lightgbm models to form predictions.
6. We form basic long/short portfolios that aim to capitalize on alpha generated from over/under performing stocks. In practice, this will be more sophisticated where the portfolio will be risk factor neutral and targeting specific risk profile.
7. Please note, we did not have intraday data which is required to carry out tests for resilience/persistence of the alpha signals – therefore the signals used here may not be necessarily ones I would use in practice.
8. **Also note – I have not provided any of my personal IP framework code.**

I would love to discuss this in call so we can go into details.

# Code Setup

* Environment: Created using Anaconda3, python 3.11, windows.
* Requirements.txt provided, please use pip install.
* **Please ensure that the current working directory is set to ‘Bluecrest’.**

# Data

**Must execute before running any projects.**

1. data\_pull\\get\_data.py: Pulls all the necessary data, S&P 500/Nasdaq 100 index and components.
2. The components are just a snapshot from wiki.
3. Saved in a SQLite3 database in /Bluecrest/equities\_data.db

# Project 1

To launch the app, from the ‘Bluecrest’ directory run:

*streamlit run project\_1\_v3\\app5.py*

This should launch the web-app on ‘localhost8501’.

The webapp allows one to do the following:

* **Market Data Visualizer:** Plot market data by index/symbol/dates
* **Traditional pair selection:** Hedge ratio calculated using Kalman filter rather than regression (which requires a window). Displays correlation and other metrics such as mean reversion speed (half life). This allows the user to select which pair they are interested in backtesting.
* **Pair Selection: Novel approach:** The alternative approach to pair selection here is using Network theory that groups similar stocks by understanding linear and non-linear structural patterns at a universe level, rather than pair level. This makes it more efficient to find similar stocks.
* **Backtester:** Most user inputs are self-explanatory.
  + In-sample inputs: Data for which the optimizer runs on (‘Run Optimizer’).
  + Out-Sample: Data for which we want to test whether the outputs from the optimizer (entry/exit thresholds) work well in the future or not.
  + Symbols: You can select from the above traditional/Network approach.

Below is a way to run the backtest:

1. First decide your in/out sample dates. Out-sample dates should be greater than in-sample date.
2. Choose your symbols you want to test.
3. Run ‘Form In-sample strategy’.
4. Run ‘Run optimizer’ and get the optimal entry/exit thresholds.
5. Enter the optimal thresholds into the appropriate boxes.
6. Run ‘Form strategy out-sample’.
7. Run ‘Run backtest’. This should output a table of the performance metrics for the out-sample dataset.

The backtesting is done using VectorBT library. **Please note this approach is very naïve when it comes to selecting** optimal entry/exit thresholds and is most definitely going to overfit to the in-sample data. In practice, the thresholds will need to be dynamic so it adjusts to the recent market conditions.

# Project 2

Relatively simple, did not build a GUI – please check the notebook provided with explanation on the approach.