NOMURA ASSIGNMENT

Overview

The goal of the assignment was to design a mean-reverting, long-only strategy that assumed a total position of 100 across 500 component stocks of the CSI500 index. The target is to beat the Buy and Hold strategy, which takes the 500 stocks equally-weighted.

Data Description

We have 30-min bar data from 01/04/2022 to 31/07/2022 for all the 500 constituent stocks of the CSI500 index, as of 01/01/2021. Of this, 01/04/2022 to 30/06/2022 is taken as in-sample, while 01/07/2022 to 31/07/2022 is taken as out-sample data.

I downloaded the in-sample and out-sample data in different folders to avoid confusion. (Code present in the Data Extraction.py file.)

Buy and Hold Strategy

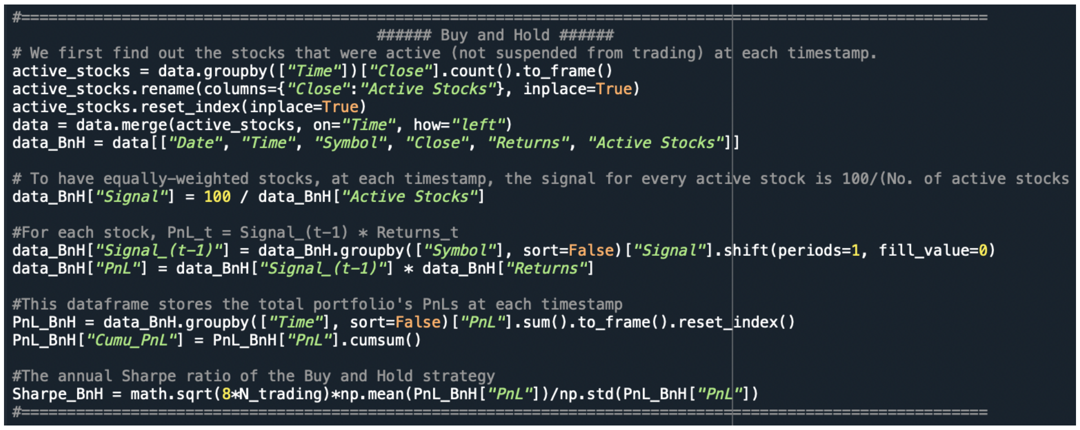
Out of the 500 stocks, there were 13 stocks who had did not trade on the exchange due to trading suspension or other reasons. Hence, for the Buy and Hold strategy, the holdings would still be equally distributed, just among the stocks that were active at that given timestamp. So, the holding $h_{i,t}$, holding of stock i at time t is

$$h_{i,t} \ = \ 100 \,/\, (No.\ of\ Active\ Stocks)_t$$

Given the returns $r_{i,\,t}$ of stock i at time t, $PnL_{i,t}$ and the Portfolio PnL, PnL_t can be written as

$$PnL_t \ = \ \sum_{i=1}^{500} PnL_{i,t} \ = \ \sum_{i=1}^{500} r_{i,t} \ * \ h_{i,t-1}$$

We can calculate cumulative returns and Sharpe ratio of the Buy and Hold strategy using PnL_t



```
#==========================================================================================
                                    ###### Buy and Hold ######
# We first find out the stocks that were active (not suspended from trading) at each timestamp.
active_stocks = data.groupby(["Time"])["Close"].count().to_frame()
active_stocks.rename(columns={"Close":"Active Stocks"}, inplace=True)
active_stocks.reset_index(inplace=True)
data = data.merge(active_stocks, on="Time", how="left")
data_BnH = data[["Date", "Time", "Symbol", "Close", "Returns", "Active Stocks"]]

# To have equally-weighted stocks, at each timestamp, the signal for every active stock is 100/(No. of active stocks
data_BnH["Signal"] = 100 / data_BnH["Active Stocks"]

#For each stock, PnL_t = Signal_(t-1) * Returns_t
data_BnH["Signal_(t-1)"] = data_BnH.groupby(["Symbol"], sort=False)["Signal"].shift(periods=1, fill_value=0)
data_BnH["PnL"] = data_BnH["Signal_(t-1)"] * data_BnH["Returns"]

#This dataframe stores the total portfolio's PnLs at each timestamp
PnL_BnH = data_BnH.groupby(["Time"], sort=False)["PnL"].sum().to_frame().reset_index()
PnL_BnH["Cumu_PnL"] = PnL_BnH["PnL"].cumsum()

#The annual Sharpe ratio of the Buy and Hold strategy
Sharpe_BnH = math.sqrt(8*N_trading)*np.mean(PnL_BnH["PnL"])/np.std(PnL_BnH["PnL"])
#==========================================================================================
```

Mean-Reverting Strategy

A mean-reverting strategy goes long on assets that it considers oversold and expect to rebound up, as the market has a tendency to revert back to its long-time average. Mean-reverting strategies work very well in range-bound markets, but face difficulties in highly trending markets as the price has a lower tendency to come back to its long-term mean.

Since we have to maintain a constant total position of 100 across all 500 stocks for each timestamp t, we have

$$\sum_{i=1}^{i=500} h_{i,t})_{final} = 100$$

where $h_{i,t})_{final}$ is the final holding of stock i at time t.

We will calculate holdings for each stock individually using some features on its data. Then, holding of each stock at a given time will be scaled such that the total holding at each timestamp remains 100.

Here, we have taken the following features to determine how much an individual stock is overbought or oversold, hence measuring its tendency to rebound up.

1. Bollinger Bands
2. RSI (Relative Strength Index)
3. MACD (Moving Average Convergence/Divergence)
4. ADX (Average Directional Index)

Appropriate Buy and Square-off points are set for each of the features to generate an individual signal series for each feature. The signals of the features will be later combined to get a final signal series for the stock.

Feature Analysis

We have taken the following 4 features for our mean-reverting strategy:

1. Bollinger Bands

   Bollinger Bands is an indicator that is based on volatility. The market hypothesis is that 95% of the time, price of an asset stays between the 2 Bollinger Bands, which are 2-SD deviations from the Moving Average of the asset price.
   To get our Bollinger bands, we calculate the 20-period moving average and moving std deviation.

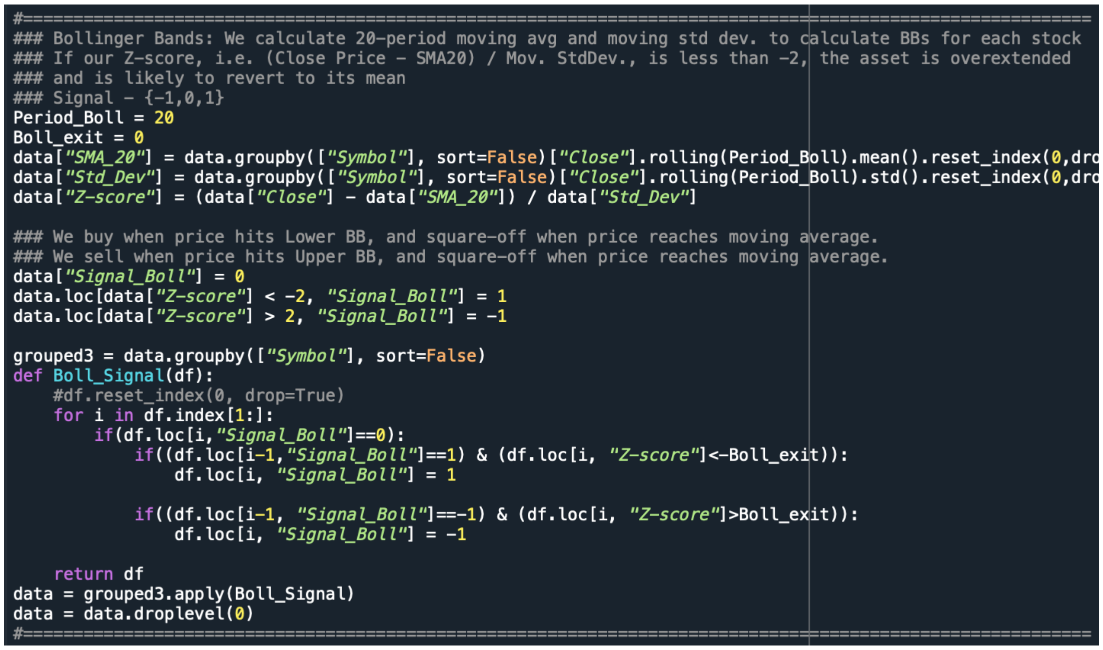
$$BOL_{upper} = SMA20(Price) + 2 * \sigma_{20}(Price)$$
$$BOL_{lower} = SMA20(Price) - 2 * \sigma_{20}(Price)$$

If the price moves below the Lower BB, the asset is oversold and highly likely to rebound up. We can calculate the Z-score of the price to compare its value with the BBs.

$$Z_{score} = (Price - SMA20(Price)) / \sigma_{20}(Price)$$

- Buy when Z-score moves below -2, Square-off at Z=0
- Sell when Z-score moves above 2, Square-off at Z=0

The Bollinger Signal, as defined above and implemented below, has a range {-1,0,1}.



```
#==========================================================================================
### Bollinger Bands: We calculate 20-period moving avg and moving std dev. to calculate BBs for each stock
### If our Z-score, i.e. (Close Price - SMA20) / Mov. StdDev., is less than -2, the asset is overextended
### and is likely to revert to its mean
### Signal - {-1,0,1}
Period_Boll = 20
Boll_exit = 0
data["SMA_20"] = data.groupby(["Symbol"], sort=False)["Close"].rolling(Period_Boll).mean().reset_index(0,dro
data["Std_Dev"] = data.groupby(["Symbol"], sort=False)["Close"].rolling(Period_Boll).std().reset_index(0,dro
data["Z-score"] = (data["Close"] - data["SMA_20"]) / data["Std_Dev"]

### We buy when price hits Lower BB, and square-off when price reaches moving average.
### We sell when price hits Upper BB, and square-off when price reaches moving average.
data["Signal_Boll"] = 0
data.loc[data["Z-score"] < -2, "Signal_Boll"] = 1
data.loc[data["Z-score"] > 2, "Signal_Boll"] = -1

grouped3 = data.groupby(["Symbol"], sort=False)
def Boll_Signal(df):
    #df.reset_index(0, drop=True)
    for i in df.index[1:]:
        if(df.loc[i,"Signal_Boll"]==0):
            if((df.loc[i-1,"Signal_Boll"]==1) & (df.loc[i, "Z-score"]<-Boll_exit)):
                df.loc[i, "Signal_Boll"] = 1

            if((df.loc[i-1, "Signal_Boll"]==-1) & (df.loc[i, "Z-score"]>Boll_exit)):
                df.loc[i, "Signal_Boll"] = -1

    return df
data = grouped3.apply(Boll_Signal)
data = data.droplevel(0)
#==========================================================================================
```

Note: Even though I have taken a short signal here and the strategy is long only, this is intentional and will be handled when calculating the final signal, which has to be positive only.

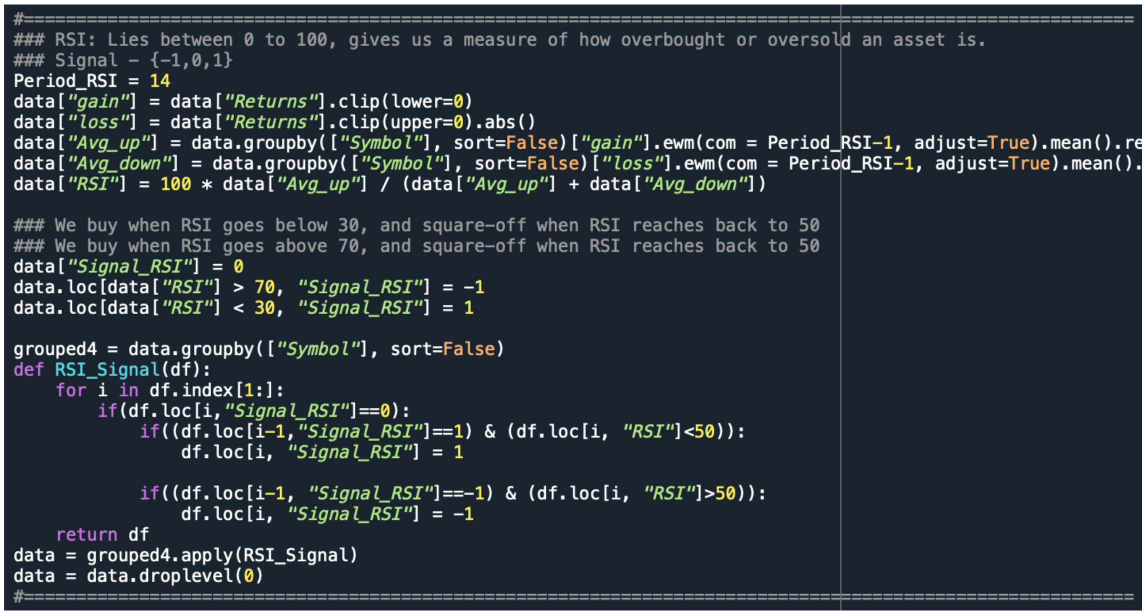
2. RSI

RSI gives us a measure of how overbought or oversold an asset is. RSI measures the speed and magnitude of a security's recent price changes, comparing the strength of a security on its gaining days to its strength on its losing days. It lies between 0 to 100.

Generally, an RSI < 30 means the asset is oversold, while an RSI > 70 indicates an overbought situation for the asset

- Buy when RSI moves below 30, square-off at RSI = 50
- Sell when RSI moves above 70, square-off at RSI = 50

The RSI Signal, as defined above and implemented below, has a range {-1,0,1}.



```
#===============================================================================================================
### RSI: Lies between 0 to 100, gives us a measure of how overbought or oversold an asset is.
### Signal - {-1,0,1}
Period_RSI = 14
data["gain"] = data["Returns"].clip(lower=0)
data["loss"] = data["Returns"].clip(upper=0).abs()
data["Avg_up"] = data.groupby(["Symbol"], sort=False)["gain"].ewm(com = Period_RSI-1, adjust=True).mean().re
data["Avg_down"] = data.groupby(["Symbol"], sort=False)["loss"].ewm(com = Period_RSI-1, adjust=True).mean().
data["RSI"] = 100 * data["Avg_up"] / (data["Avg_up"] + data["Avg_down"])

### We buy when RSI goes below 30, and square-off when RSI reaches back to 50
### We buy when RSI goes above 70, and square-off when RSI reaches back to 50
data["Signal_RSI"] = 0
data.loc[data["RSI"] > 70, "Signal_RSI"] = -1
data.loc[data["RSI"] < 30, "Signal_RSI"] = 1

grouped4 = data.groupby(["Symbol"], sort=False)
def RSI_Signal(df):
    for i in df.index[1:]:
        if(df.loc[i,"Signal_RSI"]==0):
            if((df.loc[i-1,"Signal_RSI"]==1) & (df.loc[i, "RSI"]<50)):
                df.loc[i, "Signal_RSI"] = 1

            if((df.loc[i-1, "Signal_RSI"]==-1) & (df.loc[i, "RSI"]>50)):
                df.loc[i, "Signal_RSI"] = -1
    return df
data = grouped4.apply(RSI_Signal)
data = data.droplevel(0)
#===============================================================================================================
```

Similar to Bollinger Bands, RSI also has a short signal, which will be handled when calculating the final signal series.

3. <u>MACD</u>

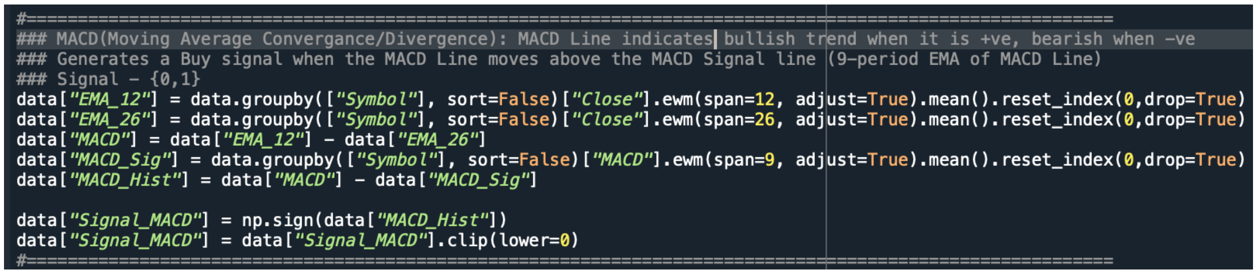
MACD is simply the difference between a short-term and a longer-term EMA of the price. The MACD Line indicates a bullish trend when it is +ve, bearish when -ve.

$$MACD = EMA12(Price) - EMA26(Price)$$

MACD is technically a trend following indicator. To use it as a reversion feature, we take the MACD Signal Line, which is a 9-period EMA of the MACD.

- Buy when the MACD is greater than MACD Signal Line
- Square-off when MACD goes below the MACD Signal Line

Since this has only one critical point, the range of the MACD Signal/Holding is {0,1}



```
#===============================================================================================================
### MACD(Moving Average Convergance/Divergence): MACD Line indicates bullish trend when it is +ve, bearish when -ve
### Generates a Buy signal when the MACD Line moves above the MACD Signal line (9-period EMA of MACD Line)
### Signal - {0,1}
data["EMA_12"] = data.groupby(["Symbol"], sort=False)["Close"].ewm(span=12, adjust=True).mean().reset_index(0,drop=True)
data["EMA_26"] = data.groupby(["Symbol"], sort=False)["Close"].ewm(span=26, adjust=True).mean().reset_index(0,drop=True)
data["MACD"] = data["EMA_12"] - data["EMA_26"]
data["MACD_Sig"] = data.groupby(["Symbol"], sort=False)["MACD"].ewm(span=9, adjust=True).mean().reset_index(0,drop=True)
data["MACD_Hist"] = data["MACD"] - data["MACD_Sig"]

data["Signal_MACD"] = np.sign(data["MACD_Hist"])
data["Signal_MACD"] = data["Signal_MACD"].clip(lower=0)
#===============================================================================================================
```

\

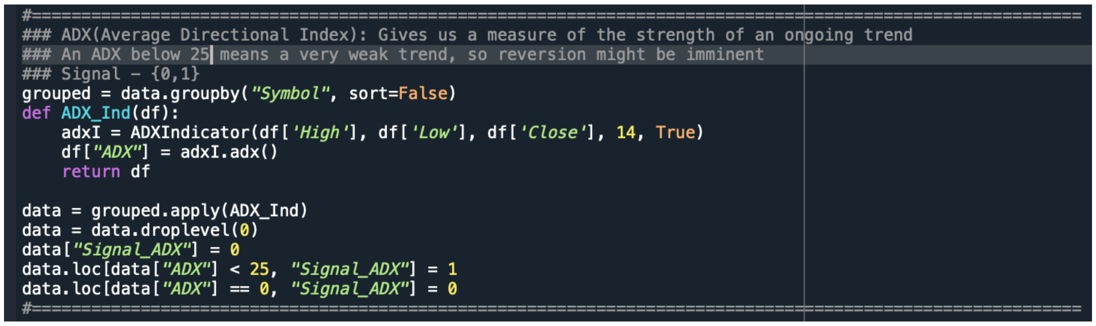
4. ADX

The ADX (Average Directional Index) is used to quantify the strength of an ongoing trend. ADX lies between 0 to 100. It is non-directional, so it depicts the strength of an upward and downward trend in a similar fashion

Usually, an ADX above 25 is considered to a condition of fairly strong trending in the asset price, while an ADX below 25 might signify a loss in trend strength and a possible chance for a reversion.

- Buy when ADX falls below 25
- Square-off when ADX rises back above 25

Since we have no short signal here as well like the MACD, the range of the ADX signal is {0,1}



```
#==========================================================================================================
### ADX(Average Directional Index): Gives us a measure of the strength of an ongoing trend
### An ADX below 25 means a very weak trend, so reversion might be imminent
### Signal - {0,1}
grouped = data.groupby("Symbol", sort=False)
def ADX_Ind(df):
    adxI = ADXIndicator(df['High'], df['Low'], df['Close'], 14, True)
    df["ADX"] = adxI.adx()
    return df

data = grouped.apply(ADX_Ind)
data = data.droplevel(0)
data["Signal_ADX"] = 0
data.loc[data["ADX"] < 25, "Signal_ADX"] = 1
data.loc[data["ADX"] == 0, "Signal_ADX"] = 0
#==========================================================================================================
```

Signal Aggregation

The signal for every stock at each time is simply the sum of the respective signals from the 4 features, capped at a lower limit of 0 since we have to make a long-only strategy.

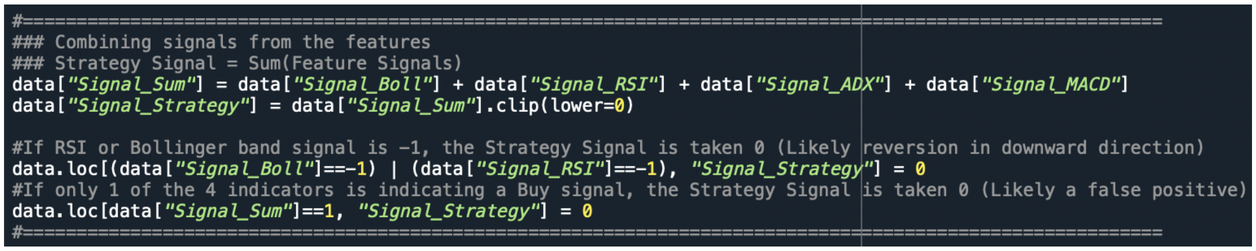
$$Signal_{strategy} = Signal_{Bollinger} + Signal_{RSI} + Signal_{MACD} + Signal_{ADX}$$

After testing the in-sample and out-sample data and comparing Sharpe ratio values, there are 2 additional conditions while calculating Strategy Signal that improve the robustness of the model.

We put $Signal_{strategy} = 0$ when

a) $Signal_{Bollinger}$ *or* $Signal_{RSI}$ are equal to -1 (Likely reversion in the downward direction)

b) Sum of feature signals = 1, i.e. only 1 of the 4 features indicate a Buy signal (Likely a false positive)



```python
#==========================================================================================================
### Combining signals from the features
### Strategy Signal = Sum(Feature Signals)
data["Signal_Sum"] = data["Signal_Boll"] + data["Signal_RSI"] + data["Signal_ADX"] + data["Signal_MACD"]
data["Signal_Strategy"] = data["Signal_Sum"].clip(lower=0)

#If RSI or Bollinger band signal is -1, the Strategy Signal is taken 0 (Likely reversion in downward direction)
data.loc[(data["Signal_Boll"]==-1) | (data["Signal_RSI"]==-1), "Signal_Strategy"] = 0
#If only 1 of the 4 indicators is indicating a Buy signal, the Strategy Signal is taken 0 (Likely a false positive)
data.loc[data["Signal_Sum"]==1, "Signal_Strategy"] = 0
#==========================================================================================================
```

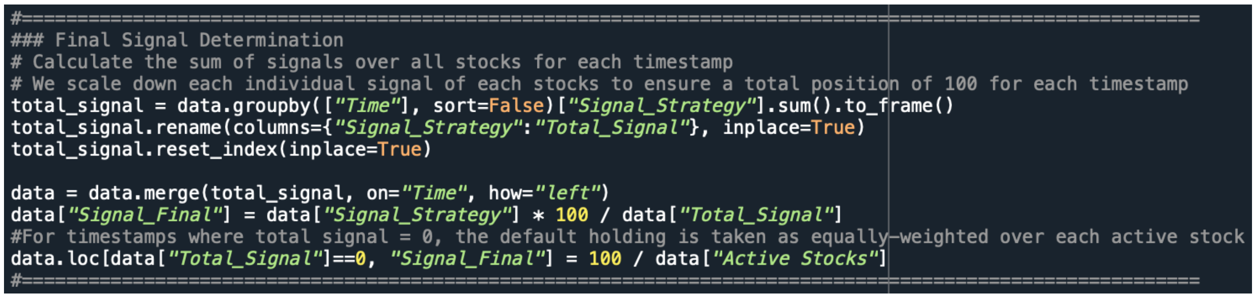
Final Signal Calculation

Let $h_{i,t}$ and $h_{i,t})_{norm}$ be the holdings and scaled final holdings for stock i and time t. This scaling ensures that the total position of all stocks combined is always equal to 100.

$$h_{i,t})_{norm} = (100 \,/\, \sum_{i=1}^{500} h_{i,t}) \; * \; h_{i,t}$$

. For timestamps where all $h_{i,t} = 0$, the default holdings is

$$h_{i,t})_{norm} = 100 \,/\, (No.\ of\ Active\ Stocks)_t$$



```python
#==============================================================================================================
### Final Signal Determination
# Calculate the sum of signals over all stocks for each timestamp
# We scale down each individual signal of each stocks to ensure a total position of 100 for each timestamp
total_signal = data.groupby(["Time"], sort=False)["Signal_Strategy"].sum().to_frame()
total_signal.rename(columns={"Signal_Strategy":"Total_Signal"}, inplace=True)
total_signal.reset_index(inplace=True)

data = data.merge(total_signal, on="Time", how="left")
data["Signal_Final"] = data["Signal_Strategy"] * 100 / data["Total_Signal"]
#For timestamps where total signal = 0, the default holding is taken as equally-weighted over each active stock
data.loc[data["Total_Signal"]==0, "Signal_Final"] = 100 / data["Active Stocks"]
#==============================================================================================================
```

PnL Calculation

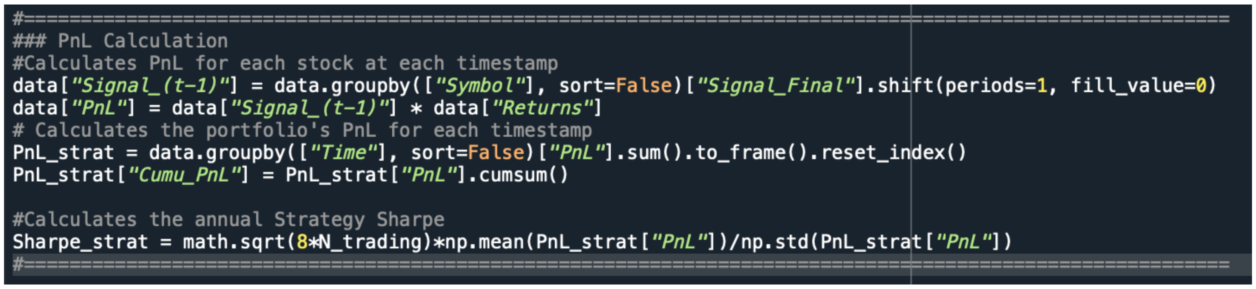
Let $r_{i,t}$ be returns of stock i at time t. The $PnL_{i,t}$ is

$$PnL_{i,t} = r_{i,t} \; * \; h_{i,t-1})_{norm}$$

To calculate the portfolio's PnL at each timestamp t,

$$PnL_t = \sum_{i=1}^{500} PnL_{i,t}$$

We can use the portfolio PnL to calculate the Strategy Sharpe ratio and strategy cumulative returns.



```
#==========================================================================================================
### PnL Calculation
#Calculates PnL for each stock at each timestamp
data["Signal_(t-1)"] = data.groupby(["Symbol"], sort=False)["Signal_Final"].shift(periods=1, fill_value=0)
data["PnL"] = data["Signal_(t-1)"] * data["Returns"]
# Calculates the portfolio's PnL for each timestamp
PnL_strat = data.groupby(["Time"], sort=False)["PnL"].sum().to_frame().reset_index()
PnL_strat["Cumu_PnL"] = PnL_strat["PnL"].cumsum()

#Calculates the annual Strategy Sharpe
Sharpe_strat = math.sqrt(8*N_trading)*np.mean(PnL_strat["PnL"])/np.std(PnL_strat["PnL"])
#==========================================================================================================
```

Ideas for Improvement

There are several ideas I have that can be used to increase the efficiency and robustness of this strategy.

1) We can apply a regression or Machine Learning technique to assign weights to the signals of different features. This technique would allow more weight to be given to the more effective feature.

$$Signal_{strategy} = \sum_{i=1}^{n_{features}} w_{feature_i} * Signal_{feature_i}$$

2) We can try to put in a dynamic stoploss using volatility. Bollinger Bands might come handy for this, as volatility can be estimated by the Bollinger Bands “squeeze”.

3) Instead of having just {0,1} as our $Signal_{feature}$, we can make a percentile-based Signal that would give stronger signal value for more favourable condition. For eg, currently both a Z_{score} of -2 or -3 are given the same +1 signal. However, the Z_{score} = -3 case is much more likely to rebound up.

4) We need to add an appropriate transaction cost for our strategy for an accurate performance evaluation of the strategy.

Measures of Strategy Performance

The most common metrics for quantifying how good a strategy is are

1) Sharpe ratio:

   Sharpe = Average (R - R_f) / StdDev(R)

2) Sortino ratio:

   Sortino = Average (R - R_f) / StdDev(Negative Returns)

Results

First, we compare the mean-reverting strategy performance to the performance of each individual feature, just to verify that the strategy performs better than any of the standalone feature.

Policy	Bollinger Bands	RSI	ADX	MACD	M-R Strategy
Sharpe ratio	-1.0463	-0.9152	0.9602	0.0572	1.0644

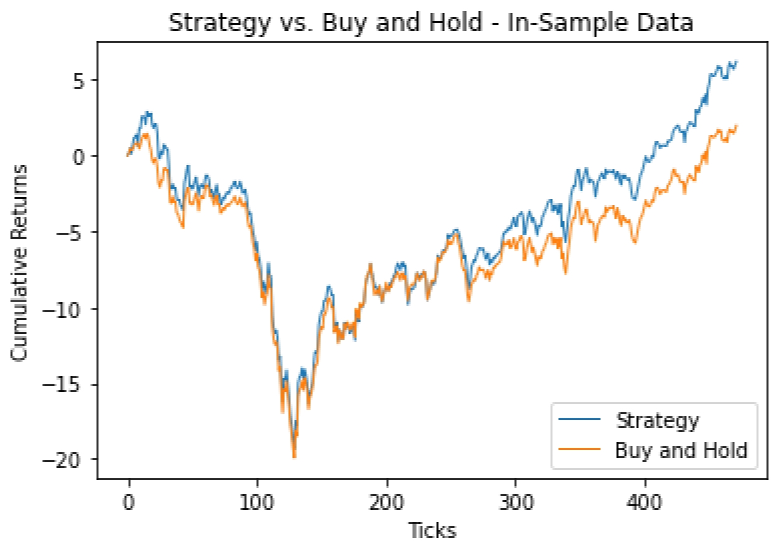
Clearly, the M-R strategy has a higher Sharpe ratio than any single feature. Combining these features highly increases their performance.

Strategy v/s Buy and Hold

Now, we compare the mean-reverting strategy performance to the performance of Buy and Hold strategy.

In-Sample

Policy	M-R Strategy	Buy and Hold
Sharpe ratio	1.0644	0.3577
Sortino ratio	1.3982	0.4951
Returns	6.223%	1.985%

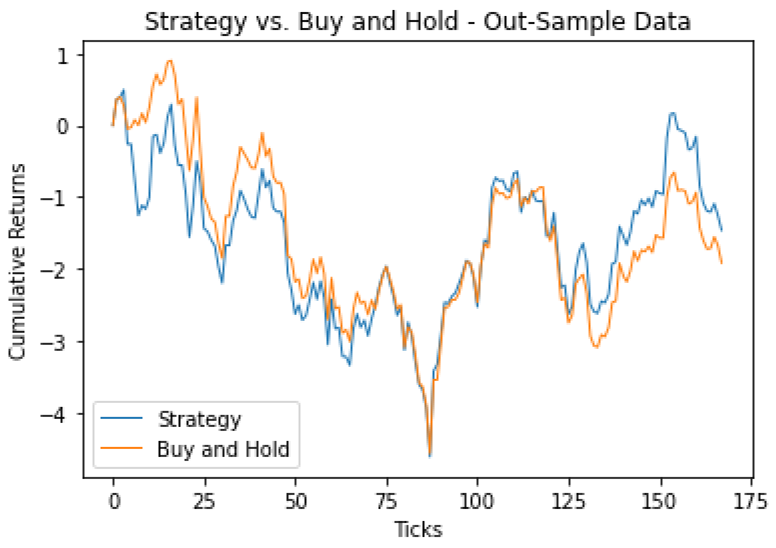


Clearly, the mean-reverting strategy has superior performance across all metrics. The Sharpe ratio, Sortino ratio and percentage returns all exceed the Buy & Hold strategy by almost **200%.**

Even in an adverse market condition, the MR strategy never underperformed with respect to the Buy and Hold, having it clearly surpass it.

<u>Out-sample</u>

Policy	M-R Strategy	Buy and Hold
Sharpe ratio	-1.1934	-1.7189
Sortino ratio	-1.8752	-2.5383
Returns	-1.461%	-1.915%



Although the strategy delivers a negative sharpe in the out-sample time window, it still performs much better than the conventional Buy and Hold.

<u>Summary</u>

It is evident from the analysis and results that the constructed Mean-Reverting strategy outperforms the standard Buy and Hold, since it takes the market dynamics into account while making trade decisions.

Folder Details

- Data Extraction.py: Contains the code for the data pulling from baostock
- Strategy.py: Contains the code for running the in-sample test, with the Sharpe values and Cumulative return plots
- Strategy-OutSample.py: Contains the code for running the out-sample test