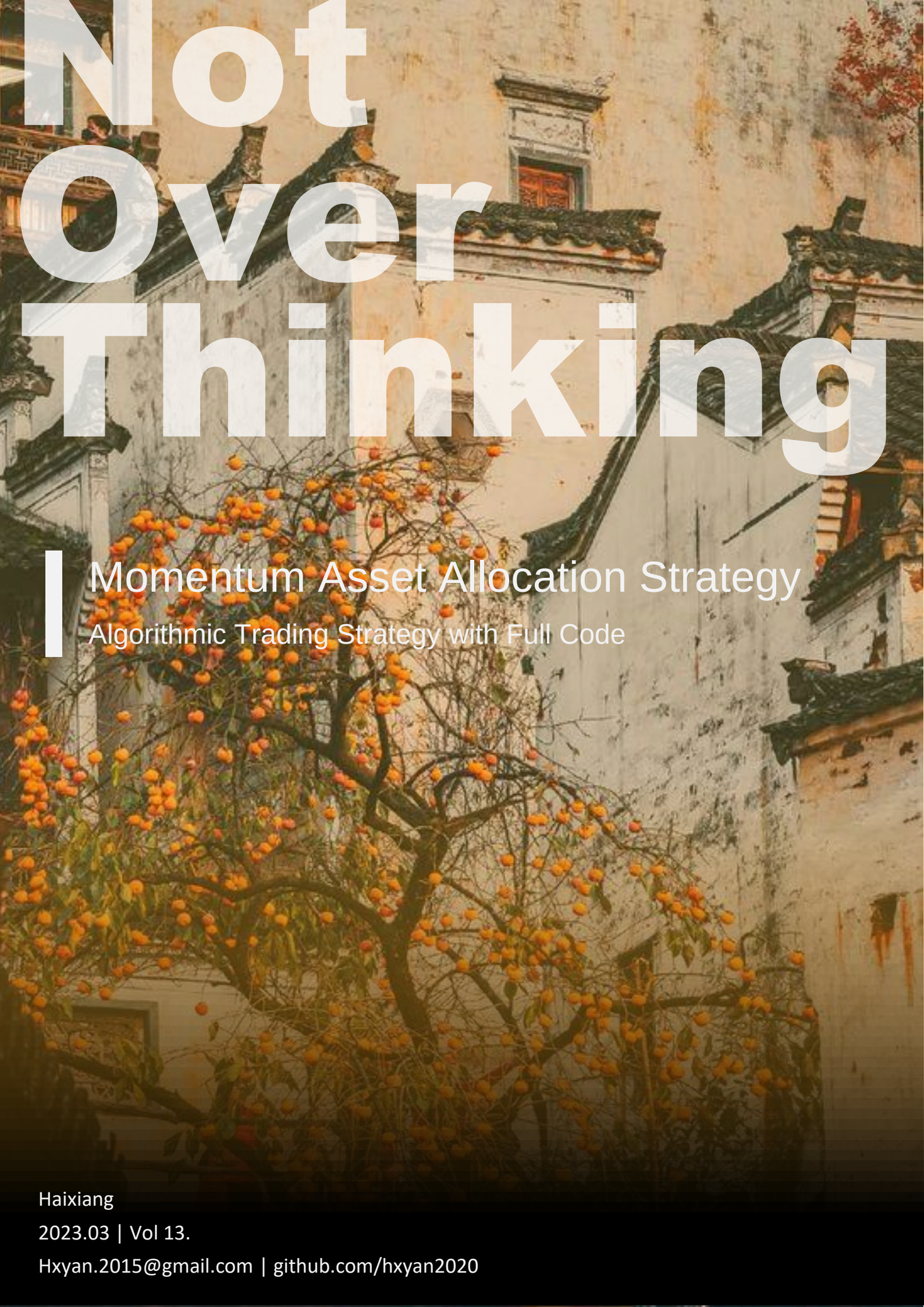

Not Over Thinking

Momentum Asset Allocation Strategy

Algorithmic Trading Strategy with Full Code

Haixiang

2023.03 | Vol 13.

Hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of 5 ETFs (SPY – US stocks, EFA – foreign stocks, BND – bonds, VNQ – REITs, GSG – commodities). Pick 3 ETFs with the strongest 12-month momentum into portfolio and weight them equally. Hold for one month and then rebalance.

BUY	SELL
Pick 3 ETFs with the strongest 12-month momentum into portfolio. Hold for one month and then rebalance.	When the ETF falls to rank top 3 based on momentum indicator

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	bonds, commodities, equities, REITs
FINANCIAL INSTRUMENTS	CFDs, ETFs, funds, futures
REGION	Global
PERIOD OF REBALANCING	1 month
NO. OF TRADED INSTRUMENTS	5
WEIGHTING	Equal Weighting
LOOKBACK PERIODS	12 months
HOLDING PERIODS	1 month
LONG/SHORT	Long Only

ALGORITHM

```
class MomentumAssetAllocationStrategy(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2000, 1, 1)
        self.SetCash(100000)

        self.data:dict[str, RateOfChange] = {}
        ## create a dictionary by calling self.data attribute, key = str, which is the
symbol, value = Rate of Change

        period:int = 12 * 21  ## lookback period = 12 months * 21 days
        self.SetWarmUp(period, Resolution.Daily)

        self.traded_count:int = 3  ## only buy the top 3 ETFs based on momentum

        self.symbols:List[str] = ["SPY", "EFA", "IEF", "VNQ", "GSG"]

        for symbol in self.symbols:
            self.AddEquity(symbol, Resolution.Minute)
            self.data[symbol] = self.ROC(symbol, period, Resolution.Daily)
            ## ROC = Rate of Change for a day > extend to the period length (12 * 21) will
become a MA line
```

```
        self.recent_month:int = -1

    def OnData(self, data):
        if self.IsWarmingUp: return

        if not (self.Time.hour == 9 and self.Time.minute == 31):
            return

        self.Log(f"Market Open Time: {self.Time}")

        # rebalance once a month
        if self.Time.month == self.recent_month:
            return
        self.recent_month = self.Time.month

        self.Log(f"New monthly rebalance...")

        # debug/log info
        selected:dict[str, RateOfChange] = {}  ## created a “selected” dictionary

        for symbol, roc in self.data.items():
            data_ready:bool = bool(symbol in data and data[symbol])
            roc_ready:bool = bool(roc.IsReady)
            ## to check if the two items under data object (aka symbol and roc) is ready
by using Boolean to indicate

            self.Log(f"Data for {symbol} are present: {data_ready}")
            self.Log(f"ROC for {symbol} IsReady: {roc_ready}")

            if data_ready and roc_ready:  ## if both are ready, add it into selected
                selected[symbol] = roc

        sorted_by_momentum:List = sorted(selected.items(), key = lambda x:
x[1].Current.Value, reverse = True)
        # sorted_by_momentum = sorted([x for x in self.data.items() if x[1].IsReady and
x[0] in data and data[x[0]]], key = lambda x: x[1].Current.Value, reverse = True)
        self.Log(f"Number of assets to sort: {len(sorted_by_momentum)}; at least
{self.traded_count} needed.")  ## self.traded_count is defined as 3

        long:List[str] = []

        if len(sorted_by_momentum) >= self.traded_count:
            long = [x[0] for x in sorted_by_momentum][:self.traded_count]

        invested:List[str] = [x.Key.Value for x in self.Portfolio if x.Value.Invested]
        for symbol in invested:
            if symbol not in long:
                self.Liquidate(symbol)

        self.Log(f"Selected long leg for next month: {long}")
        ## returning the list of long securities
        for symbol in long:
```

```
        self.SetHoldings(symbol, 1 / len(long))
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	0.068%	Sharpe Ratio	0.449
Total Trades	744	Average Win	0.63%
Average Loss	-2.09%	Compounding Annual Return	6.182%
Drawdown	47.800%	Expectancy	0.169
Net Profit	299.804%	Loss Rate	10%
Win Rate	90%	Profit-Loss Ratio	0.30
Alpha	0.02	Beta	0.498
Annual Standard Deviation	0.108	Annual Variance	0.012
Information Ratio	-0.088	Tracking Error	0.108
Treynor Ratio	0.097	Total Fees	$1518.49
Estimated Strategy Capacity	$660000.00	Lowest Capacity Asset	GSG TKH7EPK7SRC5

Fig 2. Performance Metrics

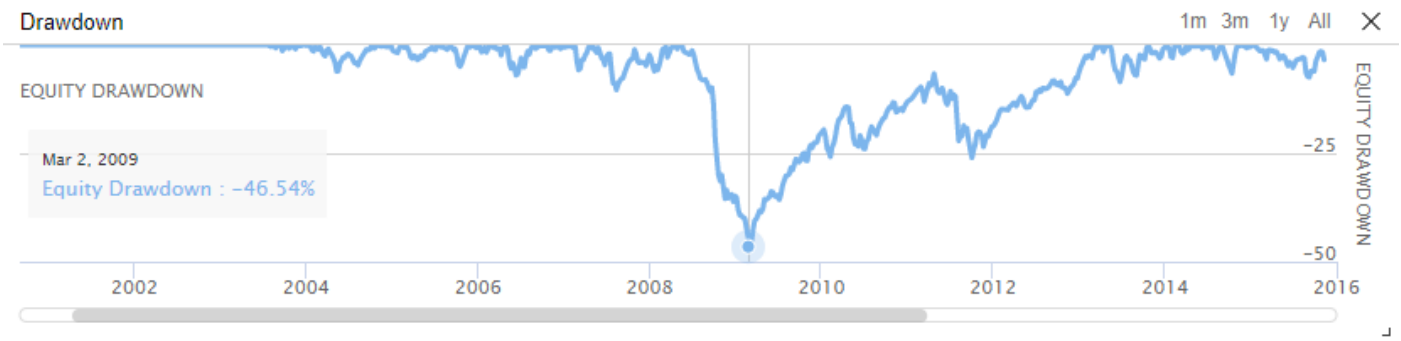

Fig 3. Drawdown

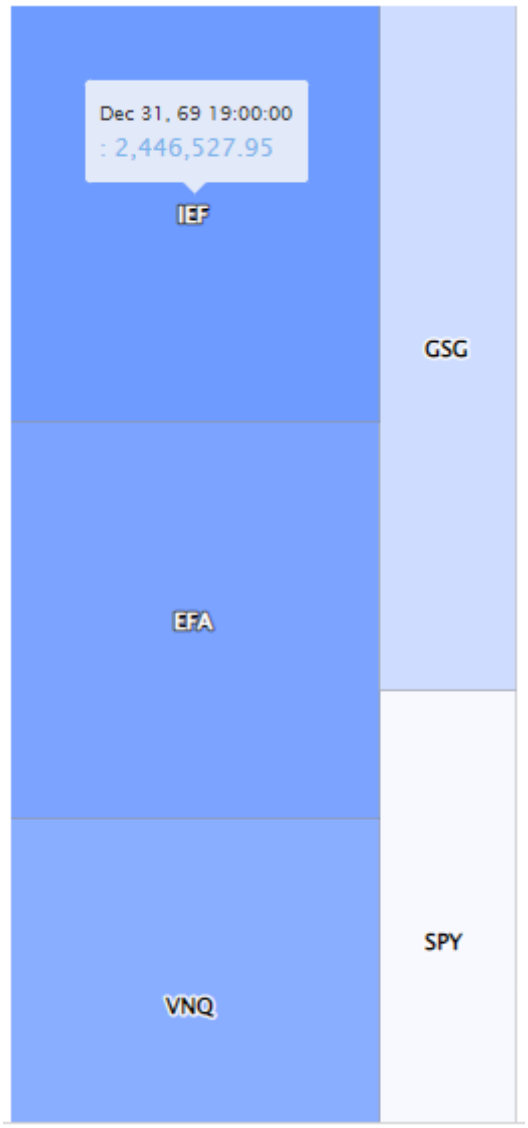

Fig 4. Assets Sales Volume