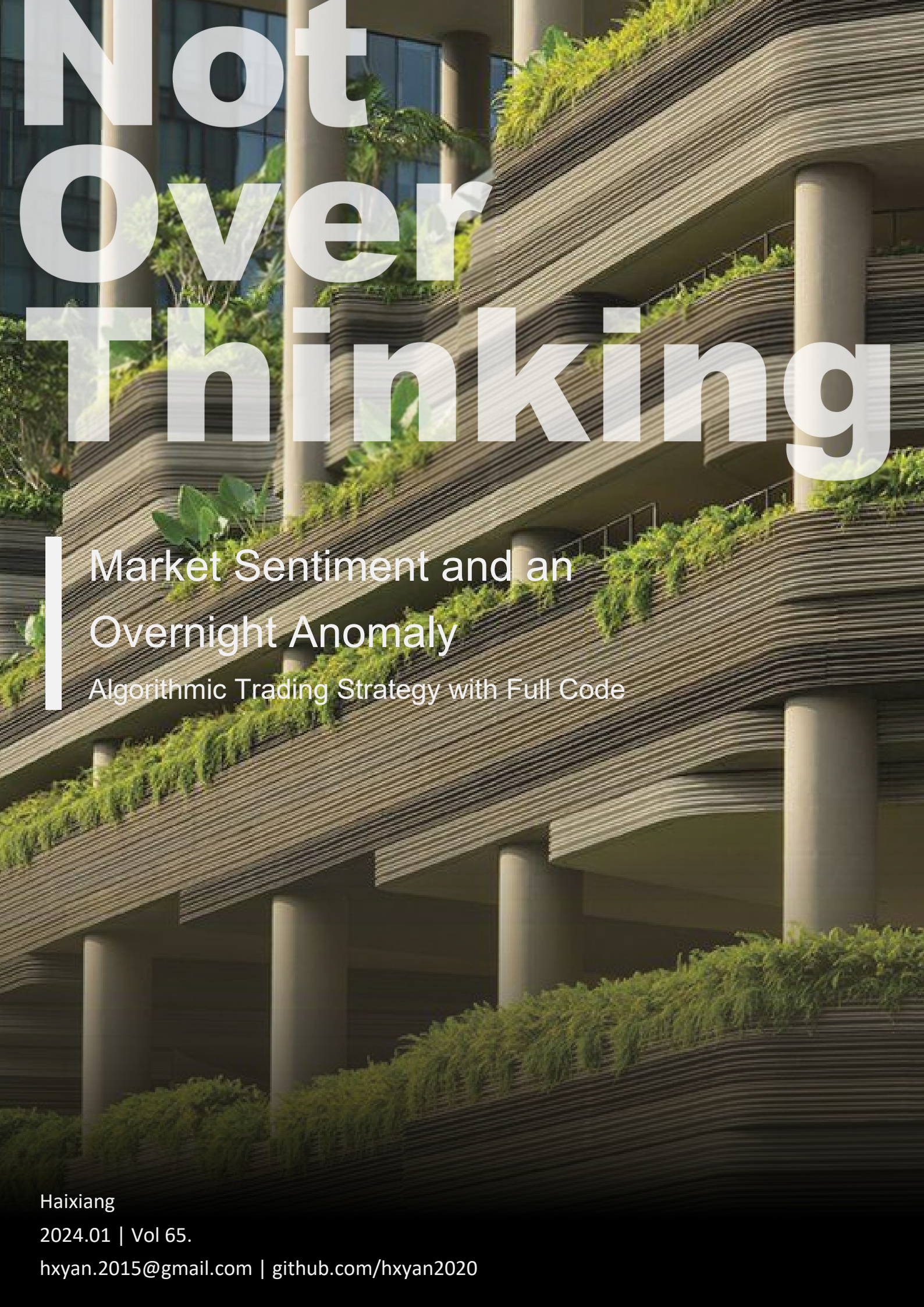

Not Over Thinking

Market Sentiment and an Overnight Anomaly

Algorithmic Trading Strategy with Full Code

Haixiang

2024.01 | Vol 65.

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

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of SPY ETF, and the price of SPY, price of VIX and Brain Market Sentiment (BMS) indicator are used to identify the market sentiment. The investor buys SPY ETF and holds it overnight; when the price of SPY is above its 20-day moving average, the price of VIX is below its moving average, and the value of the BMS indicator is greater than its 20-day moving average.

Note that the authors suggest using this strategy as an overlay when deciding whether to make a trade rather than using this system on its own.

BUY	SELL
(see above)	(see above)

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	ETFs
REGION	United States
PERIOD OF REBALANCING	Daily
NO. OF TRADED INSTRUMENTS	1
WEIGHTING	Equal weighting
LOOKBACK PERIODS	N/A
LONG/SHORT	Long only

ALGORITHM

```
from AlgorithmImports import *# endregion
class MarketSentimentAndAnOvernightAnomaly(QCAlgorithm):

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

        self.period:int = 20 # sma period

        self.weight:float = 0
        self.price_data:dict = {}

        self.spy_symbol:Symbol = self.AddEquity('SPY', Resolution.Minute).Symbol
        self.vix_symbol:Symbol = self.AddData(QuandlVix, 'CBOE/VIX', Resolution.Daily).Symbol
# starts in 2004
        self.bms_symbol:Symbol = self.AddData(QuantpediaBMS, 'BMS_GLOBAL', Resolution.Daily).Symbol
# starts in 2018

        for symbol in [self.spy_symbol, self.vix_symbol, self.bms_symbol]:
            self.price_data[symbol] = RollingWindow[float](self.period)

    def OnData(self, data: Slice):
        # calculate signal from SPY 16 minutes before close
        if self.spy_symbol in data and data[self.spy_symbol] and self.Time.hour == 15 and self.Time.
minute == 44:
            weight:float = 0.

            for symbol in [self.spy_symbol, self.vix_symbol, self.bms_symbol]:
```

```
                # trade only sub-strategies with underlying data available
                if self.Securities[symbol].GetLastData() and (self.Time.date() - self.Securities[sym
bol].GetLastData().Time.date()).days <= 3:
                    price:float = self.Securities[symbol].GetLastData().Price
                    rolling_window:RollingWindow = self.price_data[symbol]
                    if rolling_window.IsReady and self.GetSignal(price, rolling_window, True if symb
ol != self.vix_symbol else False):
                        weight += (1 / 3)

                    rolling_window.Add(price)

            q:int = int((self.Portfolio.TotalPortfolioValue * weight) / data[self.spy_symbol].Value)
            if q != 0:
                self.MarketOnCloseOrder(self.spy_symbol, q)
                self.MarketOnOpenOrder(self.spy_symbol, -q)

    def GetSignal(self, curr_value:float, rolling_window:RollingWindow, signal_above_sma:bool) -> bo
ol:
        prices:list[float] = [x for x in rolling_window]
        moving_average:float = sum(prices) / len(prices)

        result:bool = False
        if signal_above_sma and (curr_value > moving_average):
            result = True
        elif not signal_above_sma and (curr_value < moving_average):
            result = True

        return result
# Quantpedia data.# NOTE: IMPORTANT: Data order must be ascending (datewise)class QuantpediaBMS(Pyth
onData):
    def GetSource(self, config, date, isLiveMode):
        return SubscriptionDataSource("data.quantpedia.com/backtesting_data/index/{0}.csv".format(co
nfig.Symbol.Value), SubscriptionTransportMedium.RemoteFile, FileFormat.Csv)

    def Reader(self, config, line, date, isLiveMode):
        data:QuantpediaBMS = QuantpediaBMS()
        data.Symbol = config.Symbol

        if not line[0].isdigit(): return None

        split:list = line.split(',')

        data.Time = datetime.strptime(split[0], "%Y-%m-%d") + timedelta(days=1)
        data.Value = float(split[2])

        return data
class QuandlVix(PythonQuandl):
    def __init__(self):
        self.ValueColumnName = "VIX Close"
```

BACKTESTING PERFORMANCE

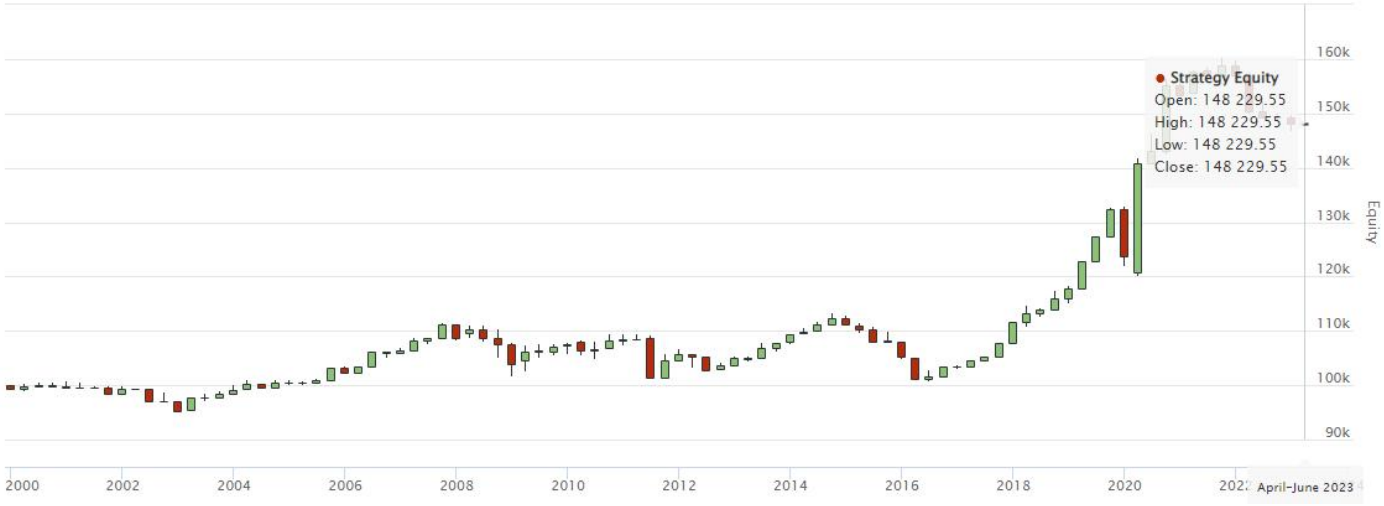

Fig 1. Overall Performance

Total Trades	7949	Average Win	0.20%
Average Loss	-0.20%	Compounding Annual Return	1.704%
Drawdown	11.000%	Expectancy	0.051
Net Profit	48.171%	Sharpe Ratio	0.348
Probabilistic Sharpe Ratio	0.007%	Loss Rate	47%
Win Rate	53%	Profit-Loss Ratio	0.99
Alpha	0.008	Beta	0.082
Annual Standard Deviation	0.036	Annual Variance	0.001
Information Ratio	-0.299	Tracking Error	0.152
Treynor Ratio	0.151	Total Fees	$18412.00
Estimated Strategy Capacity	$600000000.00	Lowest Capacity Asset	SPY R735QTJ8XC9X
Portfolio Turnover	51.62%		

Fig 2. Performance Metrics