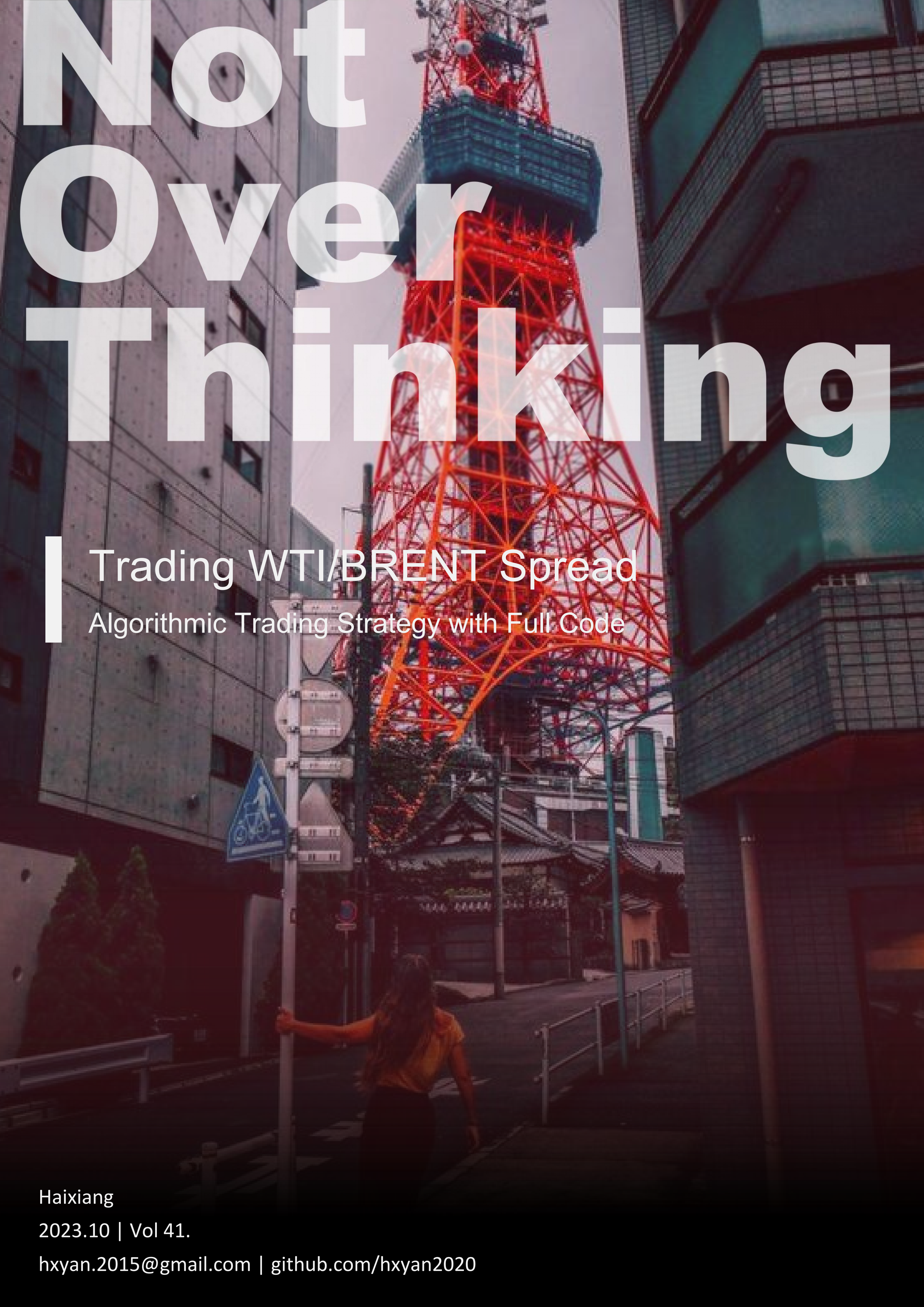


# Not Over Thinking

## Trading WTI/BRENT Spread

Algorithmic Trading Strategy with Full Code

Haixiang

2023.10 | Vol 41.

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

## STRATEGY & ECONOMIC RATIONALE

A 20-day moving average of WTI/Brent spread is calculated each day. If the current spread value is above SMA 20, then we enter a short position in the spread on close (betting that the spread will decrease to the fair value represented by SMA 20). The trade is closed at the close of the trading day when the spread crosses below fair value. If the current spread value is below SMA 20, then we enter a long position betting that the spread will increase, and the trade is closed at the close of the trading day when the spread crosses above fair value.

| BUY | SELL |
|---|---|
| current spread value is below SMA 20 | The opposite |

## PARAMETER & VARIABLES

| PARAMETER | VALUE |
|---|---|
| MARKETS TRADED | Commodity |
| FINANCIAL INSTRUMENTS | Futures |
| REGION | Global |
| PERIOD OF REBALANCING | Daily |
| NO. OF TRADED INSTRUMENTS | 2 |
| WEIGHTING | Equal weighting |
| LOOKBACK PERIODS | 20 days |
| LONG/SHORT | Long only |

## ALGORITHM

```python
class WTIBRENTSpread(QCAlgorithm):

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

        self.symbols = [
            "ICE_WT1",  # WTI Crude Futures, Continuous Contract
            "ICE_B1"    # Brent Crude Oil Futures, Continuous Contract
        ]

        self.spread = RollingWindow[float](20)

        for symbol in self.symbols:
            data = self.AddData(QuantpediaFutures, symbol, Resolution.Daily)
            data.SetLeverage(5)
            data.SetFeeModel(CustomFeeModel())

    def OnData(self, data):
        symbol1 = self.Symbol(self.symbols[0])
        symbol2 = self.Symbol(self.symbols[1])
        if symbol1 in data.Keys and symbol2 in data.Keys and data[symbol1] and
data[symbol2]:
            price1 = data[symbol1].Price
            price2 = data[symbol2].Price

            if price1 != 0 and price2 != 0:
                spread = price1 - price2
                self.spread.Add(spread)

        # MA calculation.
        if self.spread.IsReady:
            if (self.Time.date() -
self.Securities[symbol1].GetLastData().Time.date()).days < 5 and (self.Time.date() -
self.Securities[symbol2].GetLastData().Time.date()).days < 5:
                spreads = [x for x in self.spread]
                spread_ma20 = sum(spreads) / len(spreads)

                current_spread = spreads[0]

                if current_spread > spread_ma20:
                    self.SetHoldings(symbol1, -1)
                    self.SetHoldings(symbol2, 1)
                elif current_spread < spread_ma20:
                    self.SetHoldings(symbol1, 1)
                    self.SetHoldings(symbol2, -1)
            else:
                self.Liquidate()

# Quantpedia data.
# NOTE: IMPORTANT: Data order must be ascending (datewise)
class QuantpediaFutures(PythonData):
    def GetSource(self, config, date, isLiveMode):
        return
SubscriptionDataSource("data.quantpedia.com/backtesting_data/futures/{0}.csv".format(confi
g.Symbol.Value), SubscriptionTransportMedium.RemoteFile, FileFormat.Csv)

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

        if not line[0].isdigit(): return None
        split = line.split(';')

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

        return data

# Custom fee model.
class CustomFeeModel(FeeModel):
    def GetOrderFee(self, parameters):
        fee = parameters.Security.Price * parameters.Order.AbsoluteQuantity * 0.00005
        return OrderFee(CashAmount(fee, "USD"))
```

## BACKTESTING PERFORMANCE

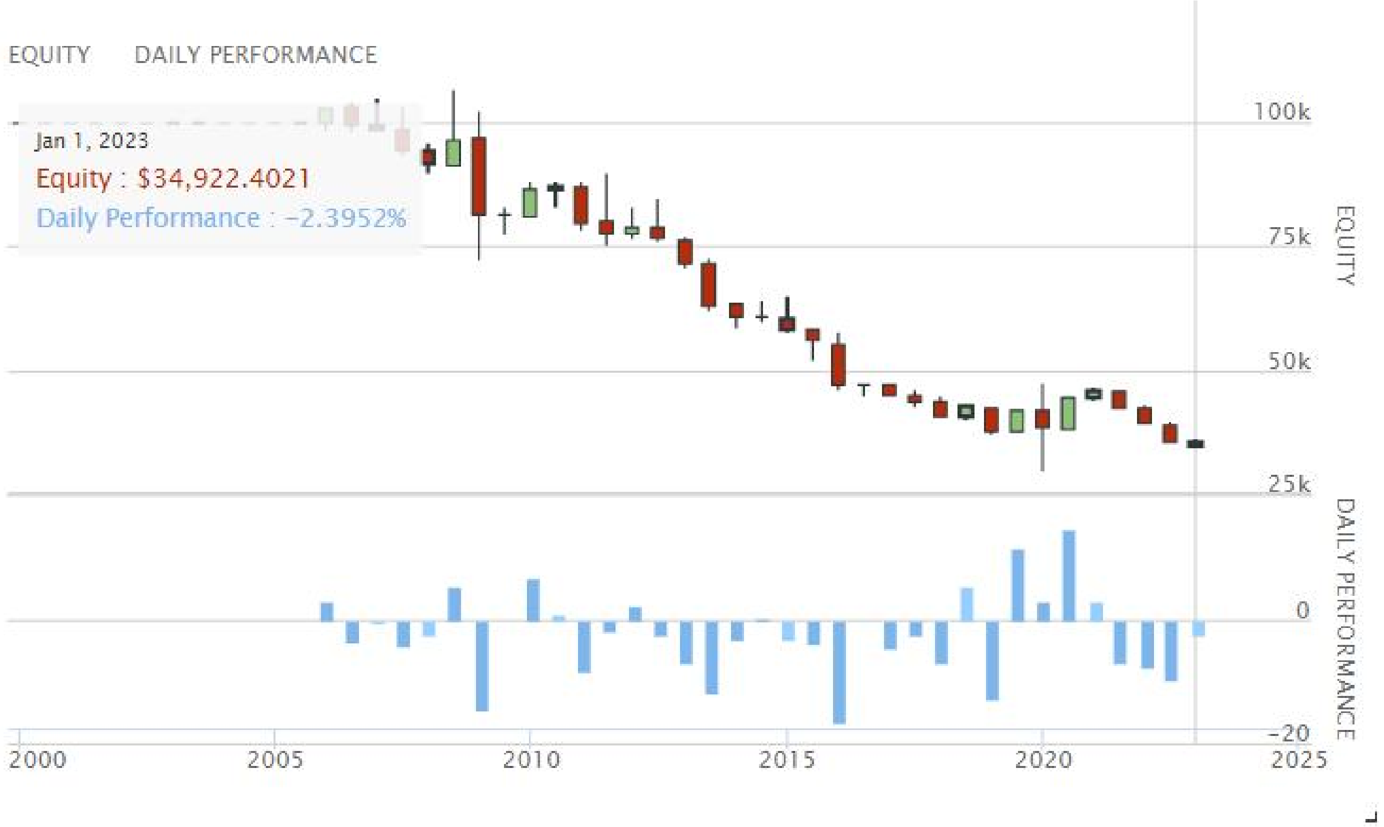
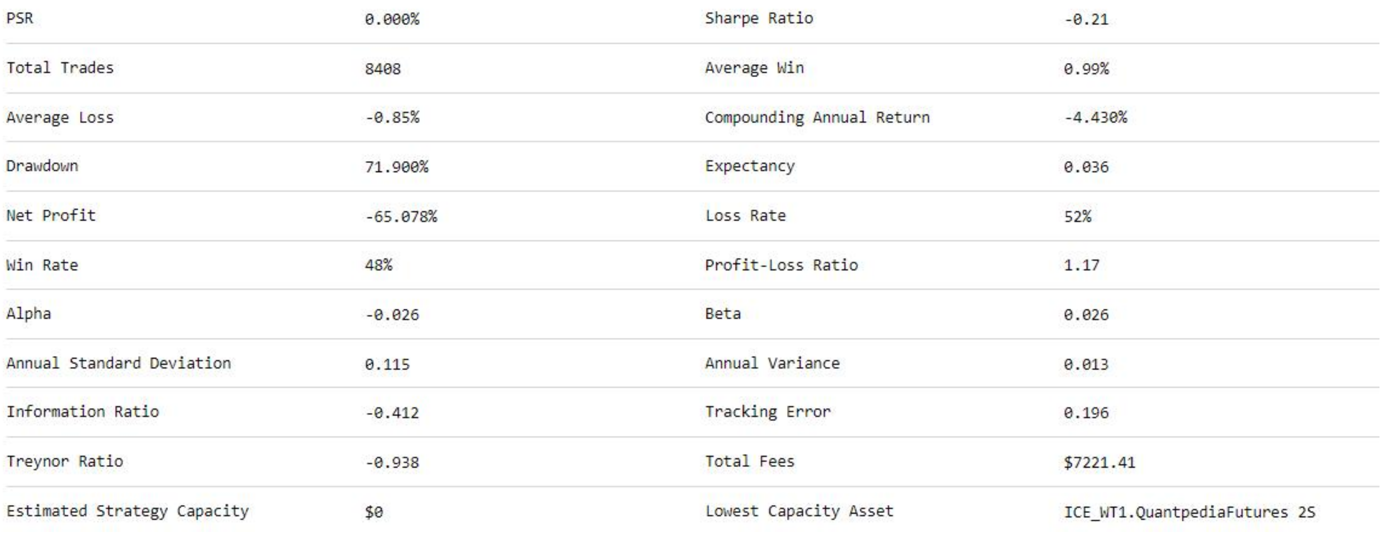


*Fig 1. Overall Performance*



| | | | |
|---|---|---|---|
| PSR | 0.000% | Sharpe Ratio | -0.21 |
| Total Trades | 8408 | Average Win | 0.99% |
| Average Loss | -0.85% | Compounding Annual Return | -4.430% |
| Drawdown | 71.900% | Expectancy | 0.036 |
| Net Profit | -65.078% | Loss Rate | 52% |
| Win Rate | 48% | Profit-Loss Ratio | 1.17 |
| Alpha | -0.026 | Beta | 0.026 |
| Annual Standard Deviation | 0.115 | Annual Variance | 0.013 |
| Information Ratio | -0.412 | Tracking Error | 0.196 |
| Treynor Ratio | -0.938 | Total Fees | $7221.41 |
| Estimated Strategy Capacity | $0 | Lowest Capacity Asset | ICE_WT1.QuantpediaFutures 2S |

*Fig 2. Performance Metrics*

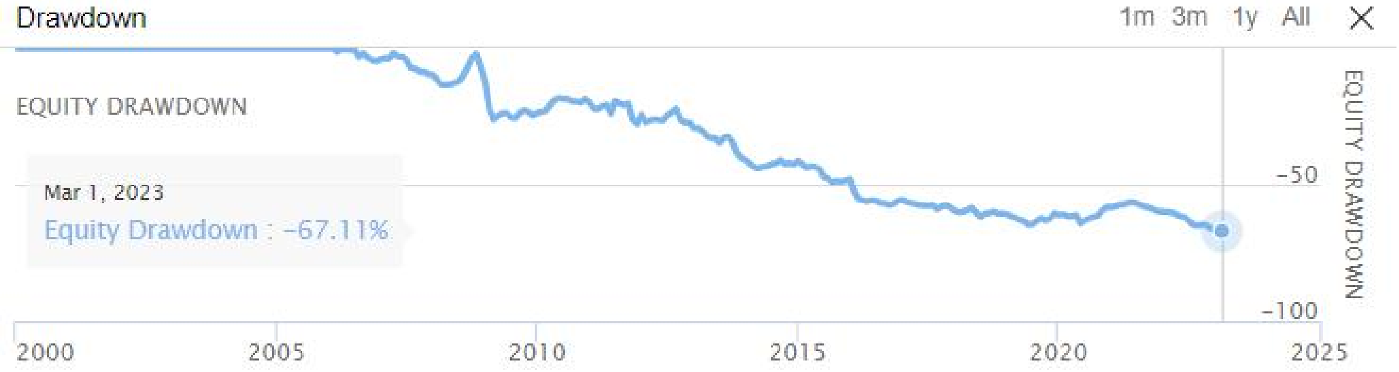


*Fig 3. Drawdown*