Not Over Thinking

Net Current Asset Value Effect
Algorithmic Trading Strategy with Full Code

Haixiang
2023.08 | Vol 28.
hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of all stocks on the London Exchange. Companies with more than one class of ordinary shares and foreign companies are excluded. Also excluded are companies on the lightly regulated markets and companies which belong to the financial sector. The portfolio of stocks is formed annually in July. Only those stocks with an NCAV/MV higher than 1.5 are included in the NCAV/MV portfolio. This Buy-and-hold portfolio is held for one year. Stocks are weighted equally.

BUY	SELL
Only those stocks with an NCAV/MV higher than 1.5 are included in the NCAV/MV portfolio.	The opposite

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	Stocks
REGION	Europe
PERIOD OF REBALANCING	Yearly
NO. OF TRADED INSTRUMENTS	100
WEIGHTING	Equal weighting
LOOKBACK PERIODS	Annual
LONG/SHORT	Long only

ALGORITHM

```python
class NetCurrentAssetValueEffect(QCAlgorithm):

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

        self.symbol = self.AddEquity('SPY', Resolution.Daily).Symbol

        self.coarse_count = 3000

        self.long = []

        self.selection_flag = False
        self.UniverseSettings.Resolution = Resolution.Daily
        self.AddUniverse(self.CoarseSelectionFunction, self.FineSelectionFunction)

        self.Schedule.On(self.DateRules.MonthEnd(self.symbol),
self.TimeRules.AfterMarketOpen(self.symbol), self.Selection)

    def OnSecuritiesChanged(self, changes):
        for security in changes.AddedSecurities:
```

```python
        security.SetFeeModel(CustomFeeModel())
        security.SetLeverage(10)

    def CoarseSelectionFunction(self, coarse):
        if not self.selection_flag:
            return Universe.Unchanged

        selected = [x.Symbol for x in coarse if x.HasFundamentalData and x.Market == 'usa']
        return selected

    def FineSelectionFunction(self, fine):
        fine = [x for x in fine if x.EarningReports.BasicAverageShares.ThreeMonths > 0 and
x.MarketCap != 0 and x.ValuationRatios.WorkingCapitalPerShare != 0]
        sorted_by_market_cap = sorted(fine, key = lambda x:x.MarketCap, reverse=True)
        top_by_market_cap = [x for x in sorted_by_market_cap[:self.coarse_count]]

        # NCAV/MV calc.
        self.long = [x.Symbol for x in top_by_market_cap if
((x.ValuationRatios.WorkingCapitalPerShare *
x.EarningReports.BasicAverageShares.ThreeMonths) / x.MarketCap) > 1.5]

        return self.long

    def OnData(self, data):
        if not self.selection_flag:
            return
        self.selection_flag = False

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

        for symbol in self.long:
            if symbol in data and data[symbol]:
                self.SetHoldings(symbol, 1 / len(self.long))

        self.long.clear()

    def Selection(self):
        if self.Time.month == 6:
            self.selection_flag = True

# 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

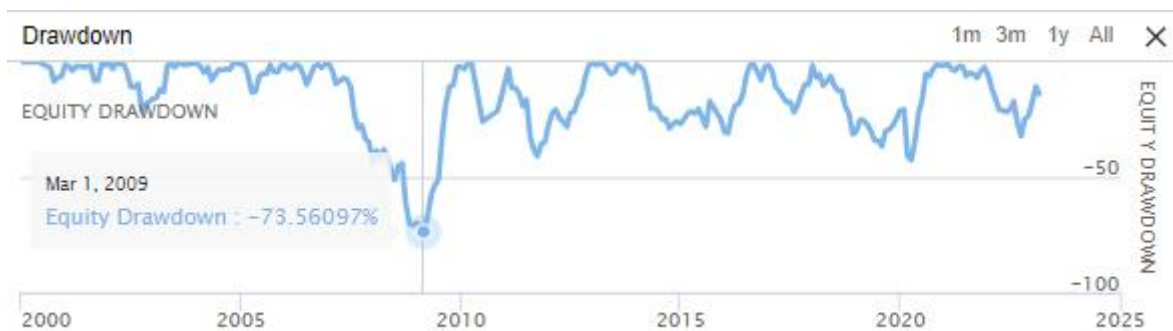|			
PSR	19.863%	Sharpe Ratio	0.943
Total Trades	1225	Average Win	1.87%
Average Loss	-0.81%	Compounding Annual Return	27.369%
Drawdown	77.100%	Expectancy	1.286
Net Profit	27409.184%	Loss Rate	31%
Win Rate	69%	Profit-Loss Ratio	2.30
Alpha	0.165	Beta	0.836
Annual Standard Deviation	0.224	Annual Variance	0.05
Information Ratio	0.859	Tracking Error	0.181
Treynor Ratio	0.253	Total Fees	$9640.19
Estimated Strategy Capacity	$110000.00	Lowest Capacity Asset	CO UHRIGHQMWXNP

Fig 2. Performance Metrics

Fig 3. Drawdown

Fig 4. Assets Sales Volume