Not Over Thinking

Value Factor

Algorithmic Trading Strategy with Full Code

Haixiang

2024.02 | Vol 75.

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

STRATEGY & ECONOMIC RATIONALE

The investment universe contains all NYSE, AMEX, and NASDAQ stocks. To represent “value” investing, HML portfolio goes long high book-to-price stocks and short, low book-to-price stocks. In this strategy, we show the results for regular HML which is simply the average of the portfolio returns of HML small (which goes long cheap and short expensive only among small stocks) and HML large (which goes long cheap and short expensive only among large caps). The portfolio is equal-weighted and rebalanced monthly.

BUY	SELL
goes long cheap	short expensive only among small stocks

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equity
FINANCIAL INSTRUMENTS	Stocks
REGION	United States
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	1000
WEIGHTING	Equal weighting
LOOKBACK PERIODS	N/A
LONG/SHORT	Long & short

ALGORITHM

```
from AlgorithmImports import *
class Value(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.quantile = 5

        self.long = []
        self.short = []

        self.month = 12
        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(s
elf.symbol), self.Selection)

    def OnSecuritiesChanged(self, changes):
        for security in changes.AddedSecurities:
            security.SetFeeModel(CustomFeeModel())
            security.SetLeverage(5)
```

```
    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):
        sorted_by_market_cap = sorted([x for x in fine if x.ValuationRatios.PBRatio != 0 and \
                            ((x.SecurityReference.ExchangeId == "NYS") or (x.SecurityReference.
ExchangeId == "NAS") or (x.SecurityReference.ExchangeId == "ASE"))],
                            key = lambda x:x.MarketCap, reverse=True)

        top_by_market_cap = [x for x in sorted_by_market_cap[:self.coarse_count]]

        if len(top_by_market_cap) >= self.quantile:
            sorted_by_pb = sorted(top_by_market_cap, key = lambda x:(x.ValuationRatios.PBRatio),
 reverse=False)
            quantile = int(len(sorted_by_pb) / self.quantile)
            self.long = [i.Symbol for i in sorted_by_pb[:quantile]]
            self.short = [i.Symbol for i in sorted_by_pb[-quantile:]]

        return self.long + self.short

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

        # Trade execution.
        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.short:
                self.Liquidate(symbol)

        # Leveraged portfolio - 100% long, 100% short.
        for symbol in self.long:
            if symbol in data and data[symbol]:
                self.SetHoldings(symbol, 1 / len(self.long))

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

        self.long.clear()
        self.short.clear()

    def Selection(self):
        if self.month == 12:
            self.selection_flag = True

        self.month += 1
        if self.month > 12:
            self.month = 1
# 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

Total Trades	27884	Average Win	0.08%
Average Loss	-0.06%	Compounding Annual Return	6.997%
Drawdown	63.900%	Expectancy	0.208
Net Profit	382.546%	Sharpe Ratio	0.469
Probabilistic Sharpe Ratio	0.077%	Loss Rate	47%
Win Rate	53%	Profit-Loss Ratio	1.30
Alpha	0.065	Beta	-0.169
Annual Standard Deviation	0.117	Annual Variance	0.014
Information Ratio	-0.013	Tracking Error	0.221
Treynor Ratio	-0.325	Total Fees	$1051.74
Estimated Strategy Capacity	$12000.00	Lowest Capacity Asset	EVOP WUQSEZA2N1B9
Portfolio Turnover	0.52%		

Fig 2. Performance Metrics