Not Over Thinking

Crude Oil Predicts Equity Returns

Algorithmic Trading Strategy with Full Code

Haixiang

2023.07 | Vol 15.

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

STRATEGY & ECONOMIC RATIONALE

Minor changes have no significant impact on the outcomes, and various oil types like Brent, WTI, and Dubai can be employed. The original research paper on this phenomenon uses Arab Light crude oil.

The regression equation employs monthly oil returns as an independent variable and equity returns as a dependent variable. The model is updated monthly, and data from the previous month is included. Based on the results of the regression analysis and dependent on the previous month's oil price change, the investor can determine if the expected stock market return for a specific month will exceed or fall below the risk-free rate.

BUY	SELL
Fully invested in market portfolio if expected return is higher (aka bull market)	Invest in cash if expected return is lower (aka bull market)

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Equities
FINANCIAL INSTRUMENTS	CFDs, ETFs, funds, futures
REGION	Global
PERIOD OF REBALANCING	Monthly
NO. OF TRADED INSTRUMENTS	1
LOOKBACK PERIODS	1 Month
HOLDING PERIODS	Depends
LONG/SHORT	Long Only

ALGORITHM

```
from data_tools import QuantpediaFutures, QuandlValue, CustomFeeModel
from AlgorithmImports import *
import numpy as np
from collections import deque
from scipy import stats

class CrudeOilPredictsEquityReturns(QCAlgorithm):

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

        self.data = {}

        self.symbols = [
            "CME_ES1",  # E-mini S&P 500 Futures, Continuous Contract #1
            "CME_CL1"   # Crude Oil Futures, Continuous Contract #1
        ]

        self.cash = self.AddEquity('SHY', Resolution.Daily).Symbol

        self.risk_free_rate = self.AddData(QuandlValue, 'FRED/DGS3MO', Resolution.Daily).Symbol
```

```
        # Monhtly price data.
        self.data = {}

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

        self.recent_month = -1

    def OnData(self, data):
        rebalance_flag = False

        for symbol in self.symbols:
            if symbol in data:
                if self.recent_month != self.Time.month:
                    rebalance_flag = True

                if data[symbol]:
                    price = data[symbol].Value
                    self.data[symbol].append(price)

        if rebalance_flag:
            self.recent_month = self.Time.month

        rf_rate = 0
        if self.Securities[self.risk_free_rate].GetLastData() and (self.Time.date() - self.Secu
rities[self.risk_free_rate].GetLastData().Time.date()).days < 5:
            rf_rate = self.Securities[self.risk_free_rate].Price
        else:
            return

        if self.Securities[self.cash].GetLastData() and (self.Time.date() - self.Securities[sel
f.cash].GetLastData().Time.date()).days >= 5:
            return

        market_prices = np.array(self.data[self.symbols[0]])
        oil_prices = np.array(self.data[self.symbols[1]])

        # At least one year of data is ready.
        if len(market_prices) < 13 or len(oil_prices) < 13:
            return

        # Trim price series lenghts.
        min_size = min(len(market_prices), len(oil_prices))
        market_prices = market_prices[-min_size:]
        oil_prices = oil_prices[-min_size:]

        market_returns = (market_prices[1:] - market_prices[:-1]) / market_prices[:-1]
        oil_returns = (oil_prices[1:] - oil_prices[:-1]) / oil_prices[:-1]

        # Simple Linear Regression
        # Y = C + (M * X)
        # Y = α + (β * X)

        # Y = Dependent variable (output/outcome/prediction/estimation)
        # C/α = Constant (Y-Intercept)
        # M/β = Slope of the regression line (the effect that X has on Y)
        # X = Independent variable (input variable used in the prediction of Y)

        slope, intercept, r_value, p_value, std_err = stats.linregress(oil_returns[:-1], market
_returns[1:])
```

```
X = oil_returns[-1]
expected_market_return = intercept + (slope * X)

if expected_market_return > rf_rate:
    self.SetHoldings(self.symbols[0], 1)
else:
    if self.Securities[self.cash].Price != 0:
        self.SetHoldings(self.cash, 1)
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	8.114%	Sharpe Ratio	0.686
Total Trades	4799	Average Win	0.12%
Average Loss	-0.13%	Compounding Annual Return	11.682%
Drawdown	33.900%	Expectancy	0.417
Net Profit	282.387%	Loss Rate	27%
Win Rate	73%	Profit-Loss Ratio	0.93
Alpha	0.019	Beta	0.736
Annual Standard Deviation	0.129	Annual Variance	0.017
Information Ratio	-0.08	Tracking Error	0.081
Treynor Ratio	0.12	Total Fees	$294.71
Estimated Strategy Capacity	$9200000.00	Lowest Capacity Asset	WCBO R735QTJ8XC9X

Fig 2. Performance Metrics

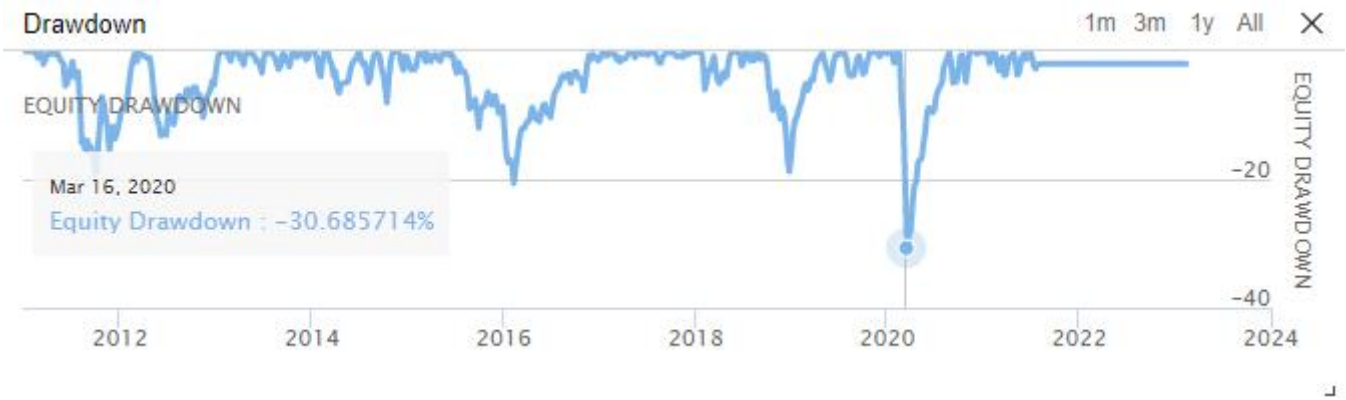

Fig 3. Drawdown

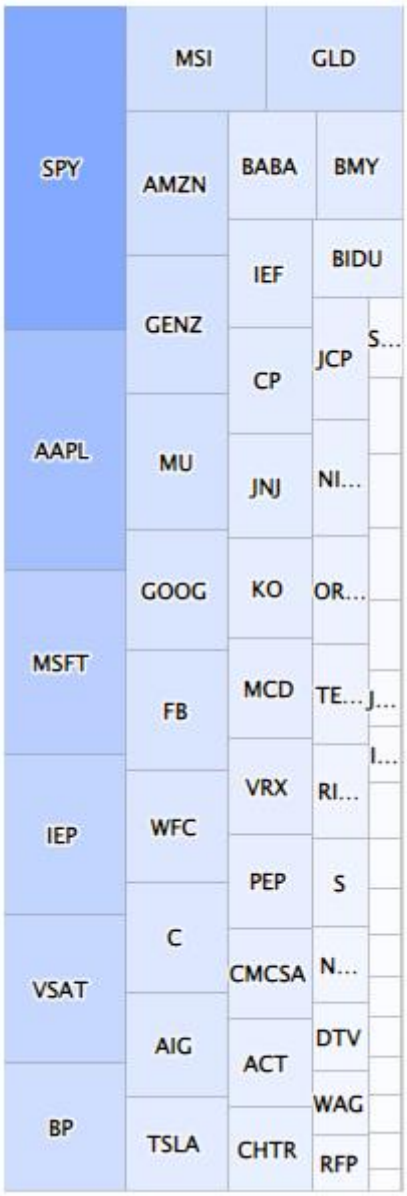

Fig 4. Assets Sales Volume