Not Over Thinking

Federal Open Market Committee Meeting Effect in Stocks

Algorithmic Trading Strategy with Full Code

Haixiang

2023.09 | Vol 35.

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

STRATEGY & ECONOMIC RATIONALE

The investor is invested in stocks during FOMC meetings (going long S&P 500 ETF, fund, future, or CFD on a close one day before the meeting and closing position on close after the meeting). Otherwise, he is invested in cash during the remaining days. The strategy has very low exposure to the stock market (8 days during the average year); therefore, it can be very easily leveraged to gain very significant returns.

BUY	SELL
S&P 500 ETF, fund, future, or CFD on a close one day before the meeting and closing position on close after the meeting	The opposite

PARAMETER & VARIABLES

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

ALGORITHM

```
from AlgorithmImports import *
from pandas.tseries.offsets import BDay
from datetime import datetime

class FederalOpenMarketCommitteeMeetingEffectinStocks(QCAlgorithm):

    def Initialize(self) -> None:
        self.SetStartDate(2000, 1, 1)
        self.SetCash(100000)

        self.market:Symbol = self.AddEquity("SPY", Resolution.Minute).Symbol

        self.fed_days_symbol:Symbol = self.AddData(FedDays, 'fed_days', Resolution.Daily,
TimeZones.NewYork).Symbol
        self.SetWarmUp(1, Resolution.Daily)
        FedDays.set_algo(self)

        self.recent_day:int = -1

    def OnData(self, data:Slice) -> None:
        if self.IsWarmingUp: return
        if self.fed_days_symbol in data and data[self.fed_days_symbol]:
            self.Log(f"New FOMC meeting data arrived: {self.Time}; submitting an MOC 
order...")

            # new fed day data arrived
            quantity:float = self.CalculateOrderQuantity(self.market, 1.)
            self.MarketOnCloseOrder(self.market, quantity)

            self.recent_day = self.Time.day
        else:
            # other new minute resolution data arrived
            if self.Portfolio[self.market].Invested:
                if self.Time.day != self.recent_day:
                    self.recent_day = self.Time.day

                    self.Log(f"FOMC meeting day; submitting an MOC order to close opened 
position...")

                    self.MarketOnCloseOrder(self.market, -
self.Portfolio[self.market].Quantity)

class FedDays(PythonData):
    algo = None

    @staticmethod
    def set_algo(algo):
        FedDays.algo = algo

    def GetSource(self, config:SubscriptionDataConfig, date:datetime, isLiveMode:bool) -> 
SubscriptionDataSource:
        if isLiveMode:
            # FedDays.algo.Log(f"Edited GetSource date {FedDays.algo.Time}")
            return 
SubscriptionDataSource("https://data.quantpedia.com/backtesting_data/economic/fed_days.jso
n", SubscriptionTransportMedium.RemoteFile, FileFormat.UnfoldingCollection)

        return 
SubscriptionDataSource("https://data.quantpedia.com/backtesting_data/economic/fed_days.csv
", SubscriptionTransportMedium.RemoteFile)

    def Reader(self, config:SubscriptionDataConfig, line:str, date:datetime, 
isLiveMode:bool) -> BaseData:
        if isLiveMode:
            try:
                # FedDays.algo.Log(f"Reader")

                objects = []
                data = json.loads(line)
                end_time = None

                for index, sample in enumerate(data):
                    custom_data = FedDays()
                    custom_data.Symbol = config.Symbol

                    custom_data.Time = (datetime.strptime(str(sample["fed_date"]), "%Y-%m-
%d") - BDay(1)).replace(hour=9, minute=31)
                    # FedDays.algo.Log(f"{custom_data.Time}")

                    end_time = custom_data.Time
                    objects.append(custom_data)

                return BaseDataCollection(end_time, config.Symbol, objects)

            except ValueError:
                # FedDays.algo.Log(f"Reader Error")
                return None
        else:
            if not (line.strip() and line[0].isdigit()):
                return None

            custom = FedDays()
            custom.Symbol = config.Symbol

            custom.Time = (datetime.strptime(line, "%Y-%m-%d") - BDay(1)).replace(hour=9,
minute=31)
            custom.Value = 0.
            custom["fed_date_str"] = line

            return custom
```

BACKTESTING PERFORMANCE

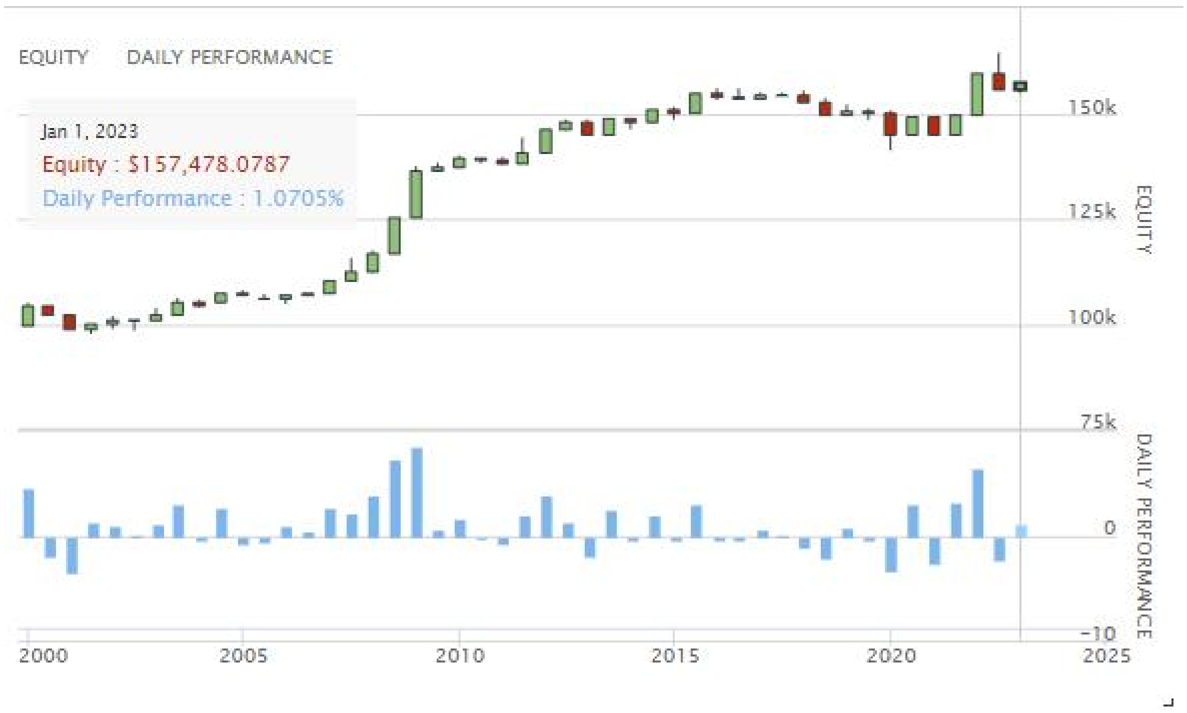
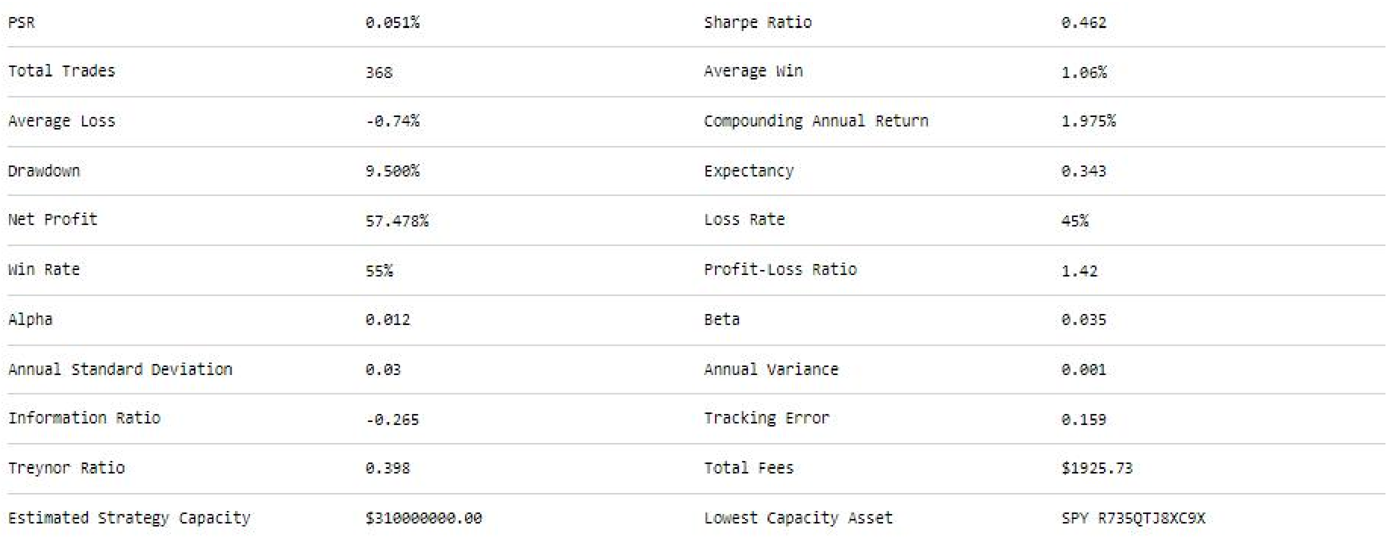


Fig 1. Overall Performance



PSR	0.051%	Sharpe Ratio	0.462
Total Trades	368	Average Win	1.06%
Average Loss	-0.74%	Compounding Annual Return	1.975%
Drawdown	9.500%	Expectancy	0.343
Net Profit	57.478%	Loss Rate	45%
Win Rate	55%	Profit-Loss Ratio	1.42
Alpha	0.012	Beta	0.035
Annual Standard Deviation	0.03	Annual Variance	0.001
Information Ratio	-0.265	Tracking Error	0.159
Treynor Ratio	0.398	Total Fees	$1925.73
Estimated Strategy Capacity	$310000000.00	Lowest Capacity Asset	SPY R735QTJ8XC9X

Fig 2. Performance Metrics

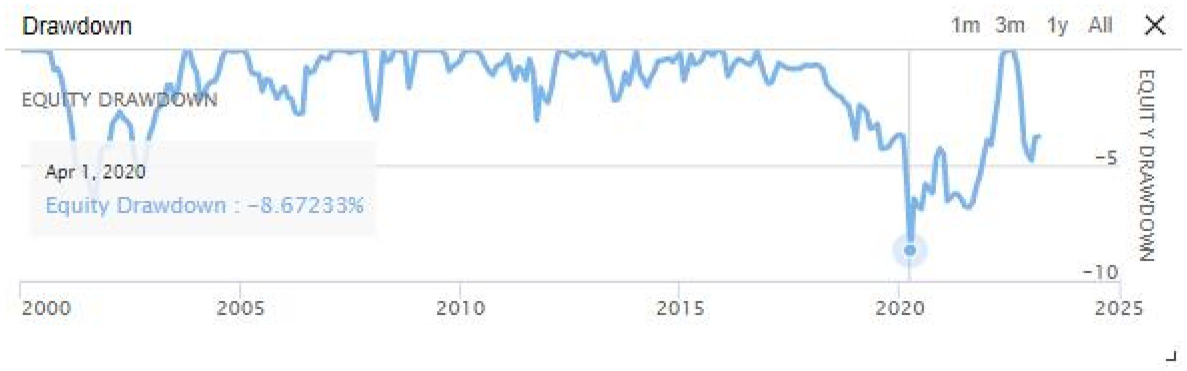


Fig 3. Drawdown