CTA_Presentation

September 12, 2023

[2]:
```python
import numpy as np
import pandas as pd
import vectorbt as vbt
import quantstats as qs
import matplotlib.pyplot as plt

from binance import Client
from binance.enums import HistoricalKlinesType
```

0.0.1

[3]:
```python
#get binance data

client = Client()
value = client.get_historical_klines(symbol = 'BTCUSDT',
                                      interval = '15m',
                                      start_str = '2020-01-01',
                                      end_str = '2023-08-15',
                                      klines_type=HistoricalKlinesType.FUTURES)

columns_name = ['openTime', 'Open', 'High', 'Low', 'Close', 'Volume',␣
 ↪'closeTime',
                'quoteVolume', 'numTrade', 'takerBuyVolume',␣
 ↪'takerBuyQuoteVolume', 'ignore']

df = pd.DataFrame(value)
df.columns = columns_name
df['openTime']= pd.to_datetime(df['openTime'], unit='ms')
df = df.drop(['ignore', 'closeTime'], axis=1)
df = df.sort_values('openTime', ascending=True)
df = df.set_index('openTime')
df = df.astype(float)
df = df[~df.index.duplicated(keep='first')]
```

[4]:
```python
#split to in and out sample
price_all = df[['Open', 'High', 'Low', 'Close']]
price_is = price_all['2020-01-01':'2022-08-31'] # in sample data
```

```python
price_os = price_all['2022-09-01':'2023-08-15'] # out sample data
print(price_all)
print(price_is)
print(price_os)
```

```
                          Open      High       Low     Close
openTime
2020-01-01 00:00:00    7189.43   7190.52   7172.94   7176.26
2020-01-01 00:15:00    7176.22   7179.41   7170.69   7172.36
2020-01-01 00:30:00    7172.79   7179.45   7170.61   7174.83
2020-01-01 00:45:00    7174.51   7179.36   7170.15   7171.55
2020-01-01 01:00:00    7171.43   7188.77   7171.10   7186.60
…                        …        …        …        …
2023-08-14 23:00:00   29426.50  29426.60  29413.40  29416.00
2023-08-14 23:15:00   29416.00  29419.50  29413.90  29418.80
2023-08-14 23:30:00   29418.80  29432.40  29414.30  29428.00
2023-08-14 23:45:00   29428.00  29430.50  29416.90  29419.50
2023-08-15 00:00:00   29419.50  29433.00  29406.50  29429.20

[126913 rows x 4 columns]
                          Open      High       Low     Close
openTime
2020-01-01 00:00:00    7189.43   7190.52   7172.94   7176.26
2020-01-01 00:15:00    7176.22   7179.41   7170.69   7172.36
2020-01-01 00:30:00    7172.79   7179.45   7170.61   7174.83
2020-01-01 00:45:00    7174.51   7179.36   7170.15   7171.55
2020-01-01 01:00:00    7171.43   7188.77   7171.10   7186.60
…                        …        …        …        …
2022-08-31 22:45:00   20108.20  20160.40  20100.40  20148.20
2022-08-31 23:00:00   20148.30  20148.30  20033.60  20043.00
2022-08-31 23:15:00   20043.10  20061.50  20001.00  20004.80
2022-08-31 23:30:00   20004.80  20009.10  19959.80  19998.90
2022-08-31 23:45:00   19999.00  20047.40  19996.90  20041.50

[93504 rows x 4 columns]
                         Open     High      Low    Close
openTime
2022-09-01 00:00:00   20041.4  20126.2  19999.0  20099.4
2022-09-01 00:15:00   20099.5  20122.5  20051.5  20089.7
2022-09-01 00:30:00   20089.7  20112.7  19910.0  19914.9
2022-09-01 00:45:00   19914.9  20080.0  19910.0  20057.2
2022-09-01 01:00:00   20057.2  20122.9  20044.7  20059.6
…                       …       …       …       …
2023-08-14 23:00:00   29426.5  29426.6  29413.4  29416.0
2023-08-14 23:15:00   29416.0  29419.5  29413.9  29418.8
2023-08-14 23:30:00   29418.8  29432.4  29414.3  29428.0
2023-08-14 23:45:00   29428.0  29430.5  29416.9  29419.5
```

```
2023-08-15 00:00:00  29419.5  29433.0  29406.5  29429.2

[33409 rows x 4 columns]
```

0.0.2

[5]:
```
#double rsi as signal and filter
def rsi_long(close, rsi_window = 29, filter_window = 50):

    rsi_crossover = vbt.RSI.run(close, window=rsi_window).rsi_crossed_above(70).
 ↪to_numpy()
    rsi_crossunder = vbt.RSI.run(close, window=rsi_window).
 ↪rsi_crossed_below(30).to_numpy()

    rsi_filter = vbt.RSI.run(close, window = filter_window).rsi.to_numpy()

    signal = np.where(rsi_crossover & (rsi_filter > 70), 1, 0)
    signal = np.where(rsi_crossunder, -1, signal)

    return signal

MyInd = vbt.IndicatorFactory(
    class_name = 'rsi_long',
    short_name = 'long',
    input_names = ['close'],
    param_names = ['rsi_window', 'filter_window'],
    output_names = ['signals']
).from_apply_func(rsi_long, keep_pd=True)
```

0.0.3

[6]:
```
#run strategy and do parameter optimization on in sample data

win = np.arange(20, 50)
signal = MyInd.run(price_is['Close'] , rsi_window=win, filter_window=win,␣
 ↪param_product = True)

entries = signal.signals == 1
exits = signal.signals == -1

portfolio = vbt.Portfolio.from_signals(
    price_is['Close'],
    #price_is['Close'].shift(-1),
    entries,
    exits,
    freq = '15m',
    direction = 'longonly',
```

```
    sl_stop = 0.03,
    fees = 0.0015
)

print(portfolio.total_return().sort_values())
print(portfolio.sharpe_ratio().sort_values())
```

```
long_rsi_window  long_filter_window
20               20                   -0.592385
21               21                   -0.528170
22               41                   -0.449519
                 42                   -0.423551
                 43                   -0.394123
                                         …
49               21                    4.982664
32               34                    5.034642
49               33                    5.117464
                 20                    5.157155
                 38                    5.674729
Name: total_return, Length: 900, dtype: float64
long_rsi_window  long_filter_window
22               41                   -0.805691
                 42                   -0.721886
                 43                   -0.712548
21               48                   -0.612060
                 44                   -0.562227
                                         …
32               34                    1.777865
33               45                    1.821711
49               38                    1.823256
32               38                    1.857084
                 39                    1.877984
Name: sharpe_ratio, Length: 900, dtype: float64
```

[7]:

```
pf = pd.concat([portfolio.total_return(),  portfolio.sharpe_ratio()], axis=1)
```

[8]:

```
#
import plotly.graph_objects as go

x = pf.loc[pf['total_return'] > 0].index.get_level_values(0)
y = pf.loc[pf['total_return'] > 0].index.get_level_values(1)
z = pf.loc[pf['total_return'] > 0]['total_return']
data = (pf.loc[pf['total_return'] > 0]['sharpe_ratio'])
scatter_data = go.Scatter3d(
    x=x,
    y=y,
    z=z,
```

```
    mode='markers',
    marker=dict(
        color=data,  #
        colorscale='Viridis',
        colorbar=dict(title='Sharpe')  #
    )
)

#
layout = go.Layout(
    title='3D Scatter Plot',
    scene=dict(
        xaxis=dict(title='rsi_window'),
        yaxis=dict(title='filter_window'),
        zaxis=dict(title='return')
    )
)

#    Figure
fig = go.Figure(data=[scatter_data], layout=layout)
fig.show()
fig.write_html(file = '3d_plot.html' , auto_open  = True)#
```

[9]:

```
#select strategies with best sharpe ratio
idx_best_10_sharpe = pf.loc[np.logical_and(pf.index.get_level_values(0) > 25,␣
↪pf.index.get_level_values(0) < pf.index.get_level_values(1))].sort_values(by␣
↪= 'sharpe_ratio', ascending=False).index[0:10].to_list()
idx_best_10_return = pf.loc[pf.index.get_level_values(0) > 25].sort_values(by =␣
↪'total_return', ascending=False).index[0:10].to_list()
idx_best_5_sharpe = pf.loc[np.logical_and(pf.index.get_level_values(0) > 25, pf.
↪index.get_level_values(0) < pf.index.get_level_values(1))].sort_values(by =␣
↪'sharpe_ratio', ascending=False).index[0:5].to_list()
#idx_best_5_sharpe = pf.loc[pf.index.get_level_values(0) > 25].sort_values(by =␣
↪'sharpe_ratio', ascending=False).index[0:5].to_list()
idx_best_5_return = pf.loc[pf.index.get_level_values(0) > 25].sort_values(by =␣
↪'total_return', ascending=False).index[0:5].to_list()
print("best 10 sharpe: " + str(idx_best_10_sharpe))
print("best 10 return: " + str(idx_best_10_return))
print("best 5 sharpe: " + str(idx_best_5_sharpe))
print("best 5 return: " + str(idx_best_5_return))
```

```
best 10 sharpe: [(32, 39), (32, 38), (33, 45), (32, 34), (44, 49), (33, 37),
(32, 44), (32, 41), (35, 49), (33, 48)]
best 10 return: [(49, 38), (49, 20), (49, 33), (32, 34), (49, 21), (49, 31),
(32, 38), (49, 32), (32, 39), (49, 36)]
best 5 sharpe: [(32, 39), (32, 38), (33, 45), (32, 34), (44, 49)]
best 5 return: [(49, 38), (49, 20), (49, 33), (32, 34), (49, 21)]
```

[10]:
```
entries_list = []
exits_list = []

for param in idx_best_5_sharpe:

    signal = MyInd.run(price_all['Close'] , rsi_window=param[0],␣
 ↪filter_window=param[1], param_product = True)

    entries = signal.signals == 1
    exits = signal.signals == -1
    entries_list.append(entries)
    exits_list.append(exits)

entries_df = pd.concat(entries_list, axis = 1)
exits_df = pd.concat(exits_list, axis = 1)

portfolio = vbt.Portfolio.from_signals(
    price_all['Close'].shift(-1),
    entries_df,
    exits_df,
    freq = '15m',
    direction = 'longonly',
    sl_stop = 0.03,
    fees = 0.0015,
    init_cash=10000
)

rett = portfolio.returns()

n = 72 * 6
rolling_risk = rett.rolling(n).std()
w_df_ = ((1/rolling_risk).T/(1/rolling_risk).sum(axis = 1)).T

rett['rp'] = (rett * w_df_).sum(axis = 1)
rett['eq'] = rett.iloc[:, :5].mean(axis = 1)

rett.cumsum().plot()

df_metrics = pd.DataFrame()
for col in rett.columns:
    df_metrics[col] = qs.reports.metrics(rett[col], mode='full', display=False,␣
 ↪periods_per_year = 4 * 24 * 365)

df_metrics.head(50)
```

[10]:
```
                          (32, 39, Close) (32, 38, Close) (33, 45, Close)  \
Start Period                   2020-01-01      2020-01-01      2020-01-01
```

End Period	2023-08-15	2023-08-15	2023-08-15
Risk-Free Rate	0	0	0
Time in Market	0.32	0.33	0.26
Cumulative Return	5.71	5.76	5.78
CAGR	0.44	0.44	0.44
Sharpe	1.63	1.59	1.79
Prob. Sharpe Ratio	1.0	1.0	1.0
Smart Sharpe	1.62	1.58	1.77
Sortino	2.29	2.24	2.56
Smart Sortino	2.28	2.23	2.53
Sortino/√2	1.62	1.58	1.81
Smart Sortino/√2	1.61	1.57	1.79
Omega	1.05	1.05	1.06
Max Drawdown	-0.25	-0.26	-0.21
Longest DD Days	337	329	292
Volatility (ann.)	0.36	0.38	0.32
Calmar	1.75	1.7	2.08
Skew	-1.46	-1.2	-0.82
Kurtosis	117.23	108.35	150.23
Expected Daily	0	0	0
Expected Monthly	0.04	0.04	0.04
Expected Yearly	0.61	0.61	0.61
Kelly Criterion	0.02	0.02	0.03
Risk of Ruin	0	0	0
Daily Value-at-Risk	0	0	0
Expected Shortfall (cVaR)	0	0	0
Max Consecutive Wins	12.0	12.0	12.0
Max Consecutive Losses	10.0	10.0	9.0
Gain/Pain Ratio	0.47	0.44	0.62
Gain/Pain (1M)	3.67	2.9	3.18
Payoff Ratio	1.01	1.01	1.01
Profit Factor	1.05	1.05	1.06
Common Sense Ratio	1.1	1.1	1.14
CPC Index	0.54	0.54	0.55
Tail Ratio	1.05	1.05	1.07
Outlier Win Ratio	14.67	14.28	16.81
Outlier Loss Ratio	2.74	2.77	2.43
MTD	-0.02	-0.02	-0.02
3M	0	-0.02	-0.05
6M	0.18	0.12	0.18
YTD	0.44	0.38	0.57
1Y	0.64	0.53	0.74
3Y (ann.)	0.42	0.39	0.45
5Y (ann.)	0.44	0.44	0.44
10Y (ann.)	0.44	0.44	0.44
All-time (ann.)	0.44	0.44	0.44
Best Day	0.05	0.05	0.05

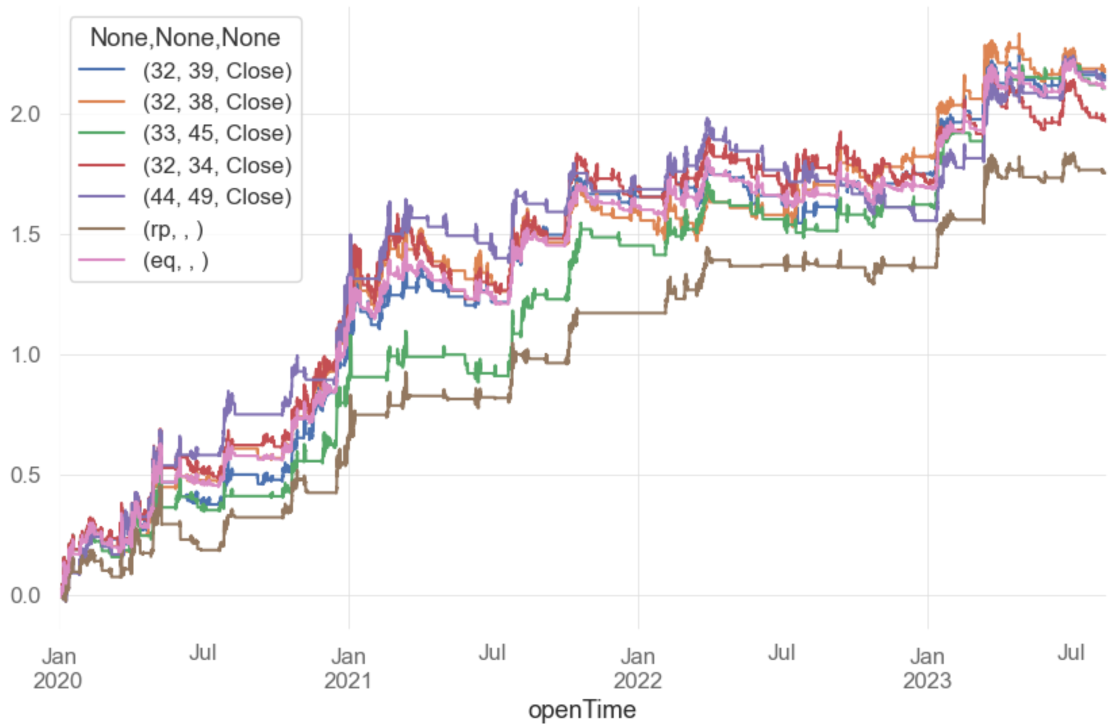
Worst Day	-0.09	-0.09	-0.09
Best Month	0.27	0.29	0.34

	(32, 34, Close)	(44, 49, Close)	(rp, ,) \
Start Period	2020-01-01	2020-01-01	2020-01-01
End Period	2023-08-15	2023-08-15	2023-08-15
Risk-Free Rate	0	0	0
Time in Market	0.41	0.31	0.28
Cumulative Return	4.32	5.75	3.95
CAGR	0.38	0.44	0.36
Sharpe	1.35	1.67	1.67
Prob. Sharpe Ratio	0.99	1.0	1.0
Smart Sharpe	1.34	1.65	1.64
Sortino	1.9	2.33	2.35
Smart Sortino	1.89	2.31	2.31
Sortino/√2	1.34	1.65	1.66
Smart Sortino/√2	1.34	1.63	1.63
Omega	1.04	1.05	1.06
Max Drawdown	-0.34	-0.36	-0.24
Longest DD Days	307	351	292
Volatility (ann.)	0.4	0.35	0.29
Calmar	1.1	1.22	1.48
Skew	-1.02	-1.52	-1.77
Kurtosis	86.59	132.02	203.31
Expected Daily	0	0	0
Expected Monthly	0.04	0.04	0.04
Expected Yearly	0.52	0.61	0.49
Kelly Criterion	0.02	0.03	0.03
Risk of Ruin	0	0	0
Daily Value-at-Risk	0	0	0
Expected Shortfall (cVaR)	0	0	0
Max Consecutive Wins	12.0	12.0	12.0
Max Consecutive Losses	10.0	10.0	10.0
Gain/Pain Ratio	0.31	0.52	0.66
Gain/Pain (1M)	2.02	2.8	4.13
Payoff Ratio	1.01	1.0	1.01
Profit Factor	1.04	1.05	1.06
Common Sense Ratio	1.07	1.11	1.16
CPC Index	0.53	0.54	0.55
Tail Ratio	1.03	1.05	1.09
Outlier Win Ratio	12.16	14.95	17.65
Outlier Loss Ratio	3.07	2.68	2.7
MTD	-0.02	-0.03	-0.01
3M	-0.03	0.05	0.01
6M	0.01	0.4	0.19
YTD	0.25	0.73	0.44
1Y	0.12	0.46	0.42

3Y (ann.)	0.3	0.34	0.37
5Y (ann.)	0.38	0.44	0.36
10Y (ann.)	0.38	0.44	0.36
All-time (ann.)	0.38	0.44	0.36
Best Day	0.05	0.05	0.05
Worst Day	-0.09	-0.09	-0.09
Best Month	0.29	0.49	0.23

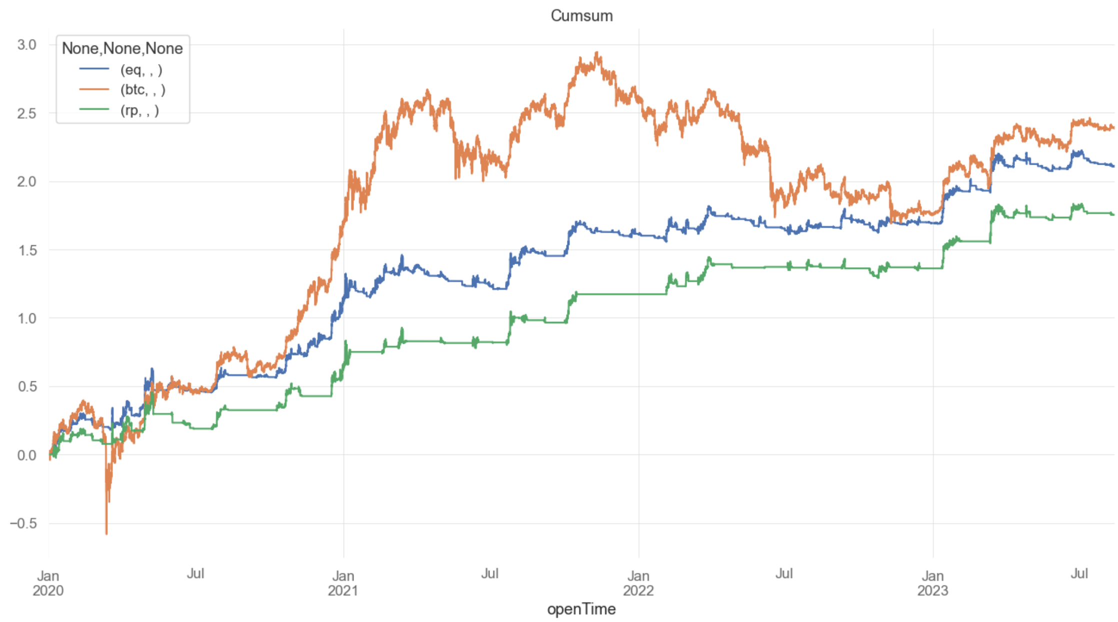
	(eq, ,)
Start Period	2020-01-01
End Period	2023-08-15
Risk-Free Rate	0
Time in Market	0.5
Cumulative Return	5.71
CAGR	0.44
Sharpe	1.74
Prob. Sharpe Ratio	1.0
Smart Sharpe	1.73
Sortino	2.46
Smart Sortino	2.44
Sortino/√2	1.74
Smart Sortino/√2	1.72
Omega	1.05
Max Drawdown	-0.23
Longest DD Days	292
Volatility (ann.)	0.33
Calmar	1.9
Skew	-1.4
Kurtosis	135.35
Expected Daily	0
Expected Monthly	0.04
Expected Yearly	0.61
Kelly Criterion	0.02
Risk of Ruin	0
Daily Value-at-Risk	0
Expected Shortfall (cVaR)	0
Max Consecutive Wins	12.0
Max Consecutive Losses	10.0
Gain/Pain Ratio	0.46
Gain/Pain (1M)	3.48
Payoff Ratio	1.02
Profit Factor	1.05
Common Sense Ratio	1.1
CPC Index	0.54
Tail Ratio	1.05
Outlier Win Ratio	11.56
Outlier Loss Ratio	3.73

```
MTD                          -0.02
3M                           -0.01
6M                            0.17
YTD                           0.47
1Y                            0.49
3Y (ann.)                     0.39
5Y (ann.)                     0.44
10Y (ann.)                    0.44
All-time (ann.)               0.44
Best Day                      0.05
Worst Day                    -0.09
Best Month                    0.27
```



[11]:
```
rett['btc'] = df['Close'].pct_change()
fig, ax = plt.subplots(figsize=(16,8))
rett[['eq', 'btc', 'rp']].cumsum().plot(ax=ax, title = 'Cumsum')
```

[11]: <Axes: title={'center': 'Cumsum'}, xlabel='openTime'>



[12]:
```
qs_metrics = pd.DataFrame()

days = 365
compounded = True

qs_metrics["benchmark"] = qs.reports.metrics(rett['btc'], mode='full',␣
↪display=False, periods_per_year =4 * 24 * days, compounded = compounded)
qs_metrics["eq_in_sample"] = qs.reports.metrics(rett['eq']['2020-01-01':
↪'2022-08-31'], mode='full', display=False, periods_per_year =4 * 24 * days,␣
↪compounded = compounded)
qs_metrics["eq_out_sample"] = qs.reports.metrics(rett['eq']['2022-09-01':
↪'2023-08-15'], mode='full', display=False, periods_per_year =4 * 24 * days,␣
↪compounded = compounded)
qs_metrics["eq_all"] = qs.reports.metrics(rett['eq'], mode='full',␣
↪display=False, periods_per_year =4 * 24 * days, compounded = compounded)
qs_metrics["rp_in_sample"] = qs.reports.metrics(rett['rp']['2020-01-01':
↪'2022-08-31'], mode='full', display=False, periods_per_year =4 * 24 * days,␣
↪compounded = compounded)
qs_metrics["rp_out_sample"] = qs.reports.metrics(rett['rp']['2022-09-01':
↪'2023-08-15'], mode='full', display=False, periods_per_year =4 * 24 * days,␣
↪compounded = compounded)
qs_metrics["rp_all"] = qs.reports.metrics(rett['rp'], mode='full',␣
↪display=False, periods_per_year =4 * 24 * days, compounded = compounded)
```

[13]:
```
day_rett = rett.resample("1D").sum()
print(day_rett)
```

```
mean = np.mean(day_rett['eq'])*252
print(mean)
std = np.std(day_rett['eq']) * (252**0.5)
sharpe = round(mean/std, 4)
print(sharpe)
```

```
                  32                  33        32    44        rp        eq  \
                  39        38        45        34    49
               Close     Close     Close     Close Close
openTime
2020-01-01  0.000000  0.000000  0.000000  0.000000   0.0  0.000000  0.000000
2020-01-02  0.000000  0.000000  0.000000  0.000000   0.0  0.000000  0.000000
2020-01-03  0.015027  0.015027  0.015027  0.015027   0.0  0.000000  0.012022
2020-01-04  0.006498  0.006498  0.006498  0.006498   0.0  0.000000  0.005198
2020-01-05 -0.000351 -0.000351 -0.000351 -0.000351   0.0  0.000000 -0.000281
…                …         …         …         …   …         …         …
2023-08-11 -0.001718 -0.001718  0.000000 -0.001718   0.0 -0.000738 -0.001031
2023-08-12  0.000757  0.000757  0.000000  0.000757   0.0 -0.000117  0.000454
2023-08-13 -0.003412 -0.003412  0.000000 -0.003412   0.0  0.000000 -0.002047
2023-08-14 -0.005565 -0.005565  0.000000 -0.005565   0.0  0.000000 -0.003339
2023-08-15  0.000000  0.000000  0.000000  0.000000   0.0  0.000000  0.000000

                 btc



openTime
2020-01-01  0.003038
2020-01-02 -0.033071
2020-01-03  0.054045
2020-01-04  0.001337
2020-01-05  0.000647
…                …
2023-08-11 -0.000926
2023-08-12  0.000182
2023-08-13 -0.004330
2023-08-14  0.004380
2023-08-15  0.000330

[1323 rows x 8 columns]
0.40092613005346633
1.4529
```

[14]:
```
qs_metrics.head(60)
```

[14]:
```
                          benchmark eq_in_sample eq_out_sample      eq_all  \
Start Period             2020-01-01   2020-01-01    2022-09-01  2020-01-01
End Period               2023-08-15   2022-08-31    2023-08-15  2023-08-15
```

Risk-Free Rate	0	0	0	0
Time in Market	1.0	0.49	0.53	0.5
Cumulative Return	3.1	3.45	0.51	5.71
CAGR	0.31	0.47	0.35	0.44
Sharpe	0.9	1.77	1.7	1.74
Prob. Sharpe Ratio	0.96	1.0	0.95	1.0
Smart Sharpe	0.89	1.77	1.61	1.73
Sortino	1.28	2.48	2.48	2.46
Smart Sortino	1.27	2.48	2.34	2.44
Sortino/$\sqrt{2}$	0.9	1.75	1.75	1.74
Smart Sortino/$\sqrt{2}$	0.9	1.75	1.66	1.72
Omega	1.02	1.05	1.05	1.05
Max Drawdown	-0.77	-0.23	-0.16	-0.23
Longest DD Days	643	167	123	292
Volatility (ann.)	0.74	0.35	0.27	0.33
Calmar	0.4	2.05	2.11	1.9
Skew	1.25	-1.7	0.48	-1.4
Kurtosis	155.08	133.91	104.14	135.35
Expected Daily	0	0	0	0
Expected Monthly	0.03	0.05	0.03	0.04
Expected Yearly	0.42	0.65	0.23	0.61
Kelly Criterion	0.01	0.02	0.02	0.02
Risk of Ruin	0	0	0	0
Daily Value-at-Risk	-0.01	0	0	0
Expected Shortfall (cVaR)	-0.01	0	0	0
Max Consecutive Wins	14.0	12.0	11.0	12.0
Max Consecutive Losses	13.0	10.0	10.0	10.0
Gain/Pain Ratio	0.16	0.46	0.45	0.46
Gain/Pain (1M)	1.01	3.95	2.43	3.48
Payoff Ratio	1.01	1.02	1.04	1.02
Profit Factor	1.02	1.05	1.05	1.05
Common Sense Ratio	1.03	1.11	1.11	1.1
CPC Index	0.51	0.54	0.55	0.54
Tail Ratio	1.01	1.06	1.06	1.05
Outlier Win Ratio	4.79	11.5	11.6	11.56
Outlier Loss Ratio	4.81	3.57	4.08	3.73
MTD	0.01	-0.01	-0.02	-0.02
3M	0.09	-0.07	-0.01	-0.01
6M	0.33	-0.04	0.17	0.17
YTD	0.78	0.02	0.47	0.47
1Y	0.21	0.17	0.51	0.49
3Y (ann.)	0.27	0.47	0.35	0.39
5Y (ann.)	0.31	0.47	0.35	0.44
10Y (ann.)	0.31	0.47	0.35	0.44
All-time (ann.)	0.31	0.47	0.35	0.44
Best Day	0.24	0.05	0.04	0.05
Worst Day	-0.13	-0.09	-0.05	-0.09

Best Month	0.47	0.27	0.26	0.27
Worst Month	-0.37	-0.09	-0.07	-0.09
Best Year	3.03	1.86	0.47	1.86
Worst Year	-0.64	0.02	0.02	0.05
Avg. Drawdown	-0.03	-0.02	-0.02	-0.02
Avg. Drawdown Days	6	5	6	6
Recovery Factor	3.1	7.21	2.73	9.15
Ulcer Index	0.44	0.11	0.09	0.11
Serenity Index	0.03	0.16	0.06	0.2
Avg. Up Month	0.2	0.12	0.09	0.11
Avg. Down Month	-0.13	-0.03	-0.04	-0.03

	rp_in_sample	rp_out_sample	rp_all
Start Period	2020-01-01	2022-09-01	2020-01-01
End Period	2022-08-31	2023-08-15	2023-08-15
Risk-Free Rate	0	0	0
Time in Market	0.26	0.32	0.28
Cumulative Return	2.47	0.43	3.95
CAGR	0.38	0.29	0.36
Sharpe	1.7	1.58	1.67
Prob. Sharpe Ratio	1.0	0.94	1.0
Smart Sharpe	1.69	1.5	1.64
Sortino	2.38	2.28	2.35
Smart Sortino	2.36	2.17	2.31
Sortino/√2	1.68	1.61	1.66
Smart Sortino/√2	1.67	1.53	1.63
Omega	1.06	1.06	1.06
Max Drawdown	-0.24	-0.13	-0.24
Longest DD Days	167	146	292
Volatility (ann.)	0.3	0.26	0.29
Calmar	1.57	2.19	1.48
Skew	-2.23	0.38	-1.77
Kurtosis	212.83	130.81	203.31
Expected Daily	0	0	0
Expected Monthly	0.04	0.03	0.04
Expected Yearly	0.51	0.19	0.49
Kelly Criterion	0.03	0.03	0.03
Risk of Ruin	0	0	0
Daily Value-at-Risk	0	0	0
Expected Shortfall (cVaR)	0	0	0
Max Consecutive Wins	12.0	10.0	12.0
Max Consecutive Losses	10.0	9.0	10.0
Gain/Pain Ratio	0.7	0.56	0.66
Gain/Pain (1M)	4.25	3.76	4.13
Payoff Ratio	1.01	1.02	1.01
Profit Factor	1.06	1.06	1.06
Common Sense Ratio	1.17	1.15	1.16

```
CPC Index                   0.55     0.55     0.55
Tail Ratio                  1.1      1.09     1.09
Outlier Win Ratio           18.04    15.63    17.65
Outlier Loss Ratio          2.56     2.96     2.7
MTD                         -0.02    -0.01    -0.01
3M                          0        0.01     0.01
6M                          0.06     0.19     0.19
YTD                         0.19     0.44     0.44
1Y                          0.43     0.43     0.42
3Y (ann.)                   0.38     0.29     0.37
5Y (ann.)                   0.38     0.29     0.36
10Y (ann.)                  0.38     0.29     0.36
All-time (ann.)             0.38     0.29     0.36
Best Day                    0.05     0.04     0.05
Worst Day                   -0.09    -0.05    -0.09
Best Month                  0.23     0.22     0.23
Worst Month                 -0.1     -0.04    -0.1
Best Year                   0.78     0.44     0.78
Worst Year                  0.19     -0.01    0.18
Avg. Drawdown               -0.02    -0.02    -0.02
Avg. Drawdown Days          7        8        7
Recovery Factor             5.66     2.89     7.26
Ulcer Index                 0.1      0.07     0.09
Serenity Index              0.14     0.07     0.19
Avg. Up Month               0.09     0.1      0.1
Avg. Down Month             -0.04    -0.02    -0.03
```

[15]:
```
qs.reports.full(rett['eq'], benchmark = rett['btc'], periods_per_year= 24 * 4
    *365, compounded=False, active_returns=False)
```

```
<IPython.core.display.HTML object>

                           Benchmark    Strategy
-------------------------  -----------  -----------
Start Period               2020-01-03   2020-01-03
End Period                 2023-08-15   2023-08-15
Risk-Free Rate             0.0%         0.0%
Time in Market             100.0%       50.0%

Total Return               241.44%      210.49%
CAGR                      26.44%       24.17%

Sharpe                     0.91         1.75
Prob. Sharpe Ratio         95.79%       99.95%
Smart Sharpe               0.9          1.73
Sortino                    1.29         2.46
Smart Sortino              1.28         2.44
Sortino/√2                 0.91         1.74
```

Smart Sortino/√2	0.9	1.73
Omega	1.05	1.05
Max Drawdown	-77.27%	-23.0%
Longest DD Days	643	292
Volatility (ann.)	73.71%	33.34%
R^2	0.0	0.0
Information Ratio	-0.0	-0.0
Calmar	0.41	1.91
Skew	1.25	-1.4
Kurtosis	154.93	135.12
Expected Daily %	0.0%	0.0%
Expected Monthly %	3.55%	4.36%
Expected Yearly %	10.27%	48.12%
Kelly Criterion	2.05%	2.36%
Risk of Ruin	0.0%	0.0%
Daily Value-at-Risk	-0.65%	-0.29%
Expected Shortfall (cVaR)	-0.65%	-0.29%
Max Consecutive Wins	14	12
Max Consecutive Losses	13	10
Gain/Pain Ratio	0.16	0.46
Gain/Pain (1M)	1.02	3.48
Payoff Ratio	1.03	1.02
Profit Factor	1.02	1.05
Common Sense Ratio	1.03	1.1
CPC Index	0.53	0.54
Tail Ratio	1.01	1.05
Outlier Win Ratio	3.57	17.49
Outlier Loss Ratio	3.54	5.82
MTD	0.8%	-1.93%
3M	10.32%	-0.43%
6M	32.96%	18.02%
YTD	63.54%	41.26%
1Y	30.24%	43.38%
3Y (ann.)	27.07%	24.69%
5Y (ann.)	26.44%	24.17%
10Y (ann.)	26.44%	24.17%
All-time (ann.)	26.44%	24.17%
Best Day	23.69%	4.63%
Worst Day	-12.58%	-9.09%
Best Month	40.92%	25.65%
Worst Month	-37.34%	-9.4%
Best Year	171.92%	113.12%

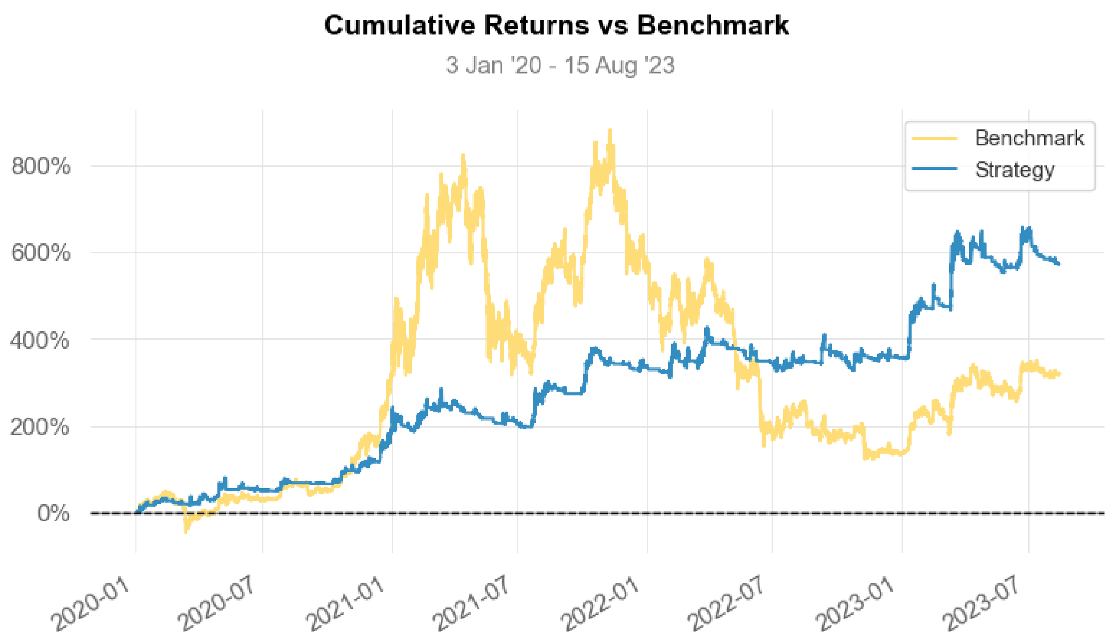
Worst Year	-82.34%	7.83%
Avg. Drawdown	-2.54%	-1.98%
Avg. Drawdown Days	6	6
Recovery Factor	3.12	9.15
Ulcer Index	0.44	0.11
Serenity Index	0.03	0.2
Avg. Up Month	22.21%	12.45%
Avg. Down Month	-13.44%	-3.69%
Win Days %	50.27%	50.62%
Win Month %	56.82%	59.09%
Win Quarter %	73.33%	73.33%
Win Year %	75.0%	100.0%
Beta	-	0.0
Alpha	-	0.58
Correlation	-	0.12%
Treynor Ratio	-	1045265.69%

None

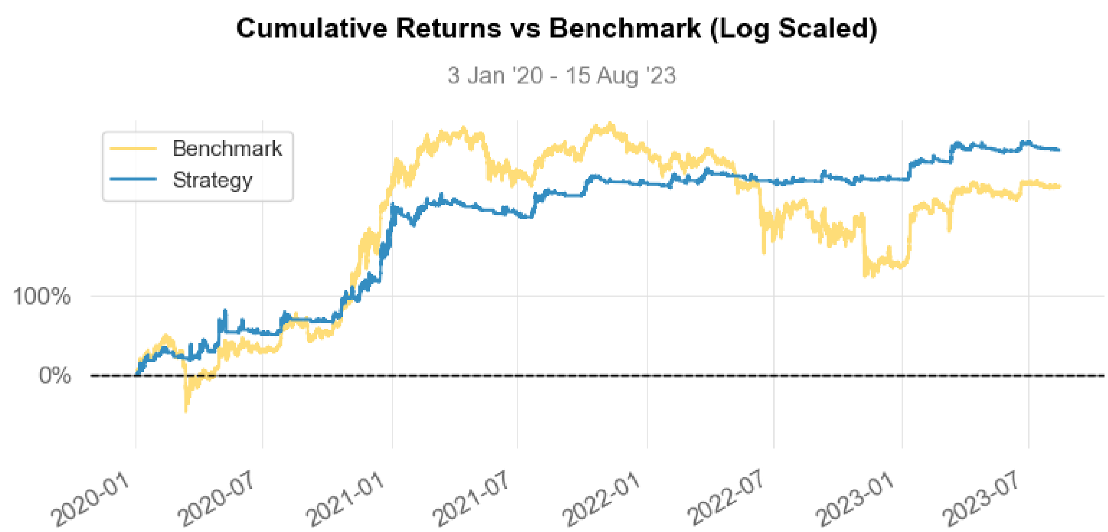
<IPython.core.display.HTML object>

	Start	Valley	End	Days	Max Drawdown	99% Max Drawdown
1	2021-03-13	2021-07-10	2021-08-08	148	-23.002339	-22.871998
2	2022-03-28	2022-07-06	2023-01-13	292	-18.947326	-18.558040
3	2020-05-08	2020-07-20	2020-10-21	167	-16.904152	-16.618903
4	2021-01-03	2021-02-02	2021-02-19	48	-16.684764	-16.020205
5	2021-10-20	2022-02-04	2022-02-08	111	-14.503154	-13.744971

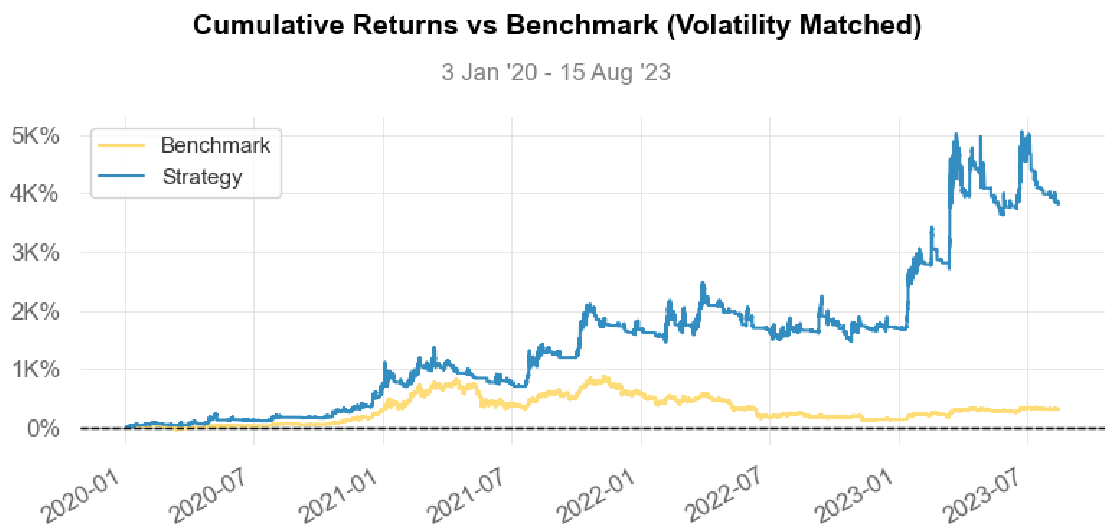
<IPython.core.display.HTML object>



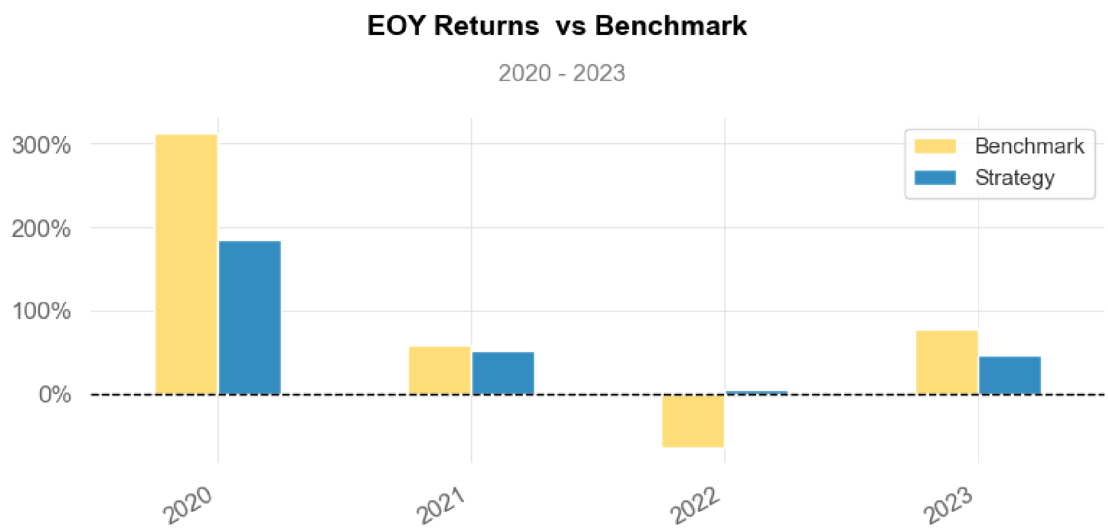
Cumulative Returns vs Benchmark
3 Jan '20 - 15 Aug '23
Benchmark
Strategy
800%
600%
400%
200%
0%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07



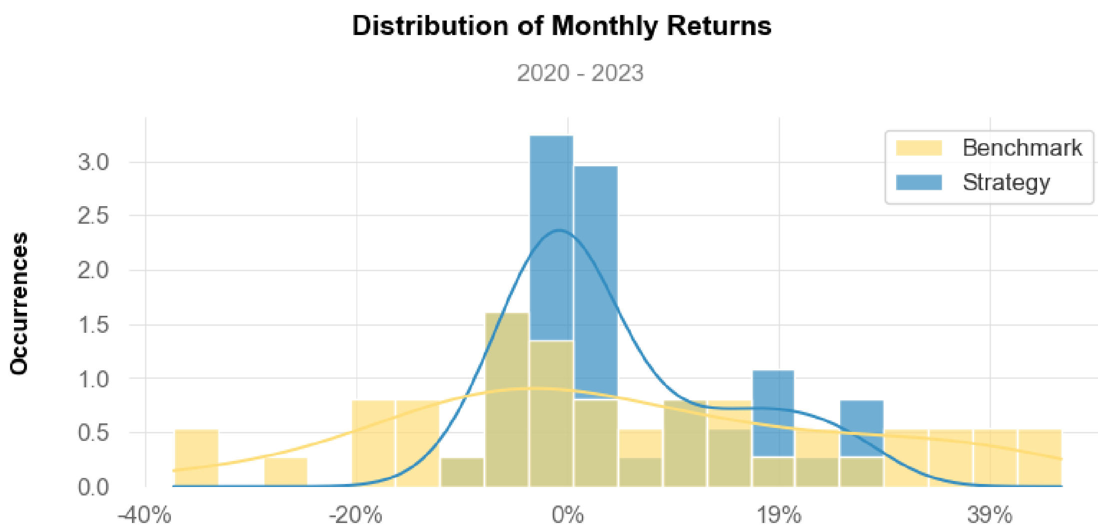
Cumulative Returns vs Benchmark (Log Scaled)
3 Jan '20 - 15 Aug '23
Benchmark
Strategy
100%
0%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

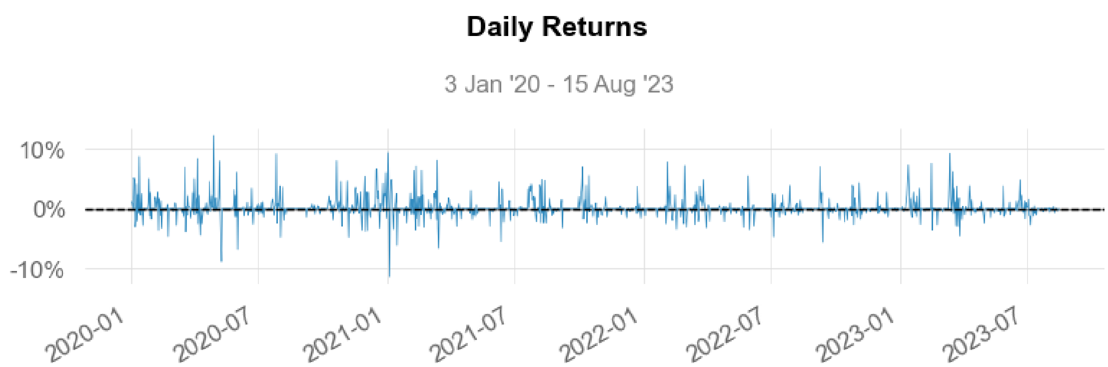


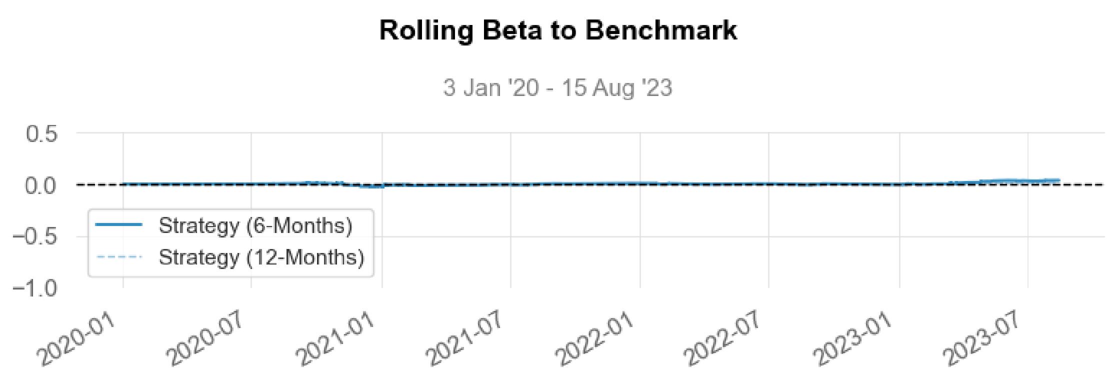
Cumulative Returns vs Benchmark (Volatility Matched)
3 Jan '20 - 15 Aug '23
Benchmark
Strategy
5K%
4K%
3K%
2K%
1K%
0%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

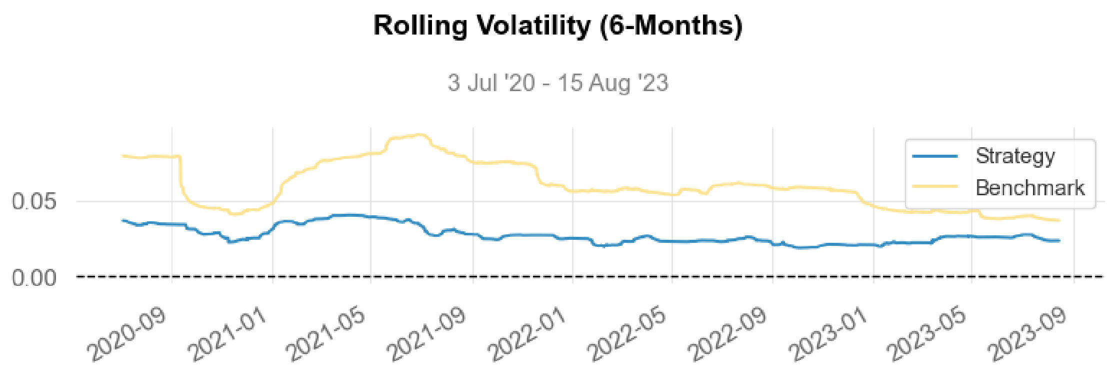


EOY Returns vs Benchmark
2020 - 2023
Benchmark
Strategy
300%
200%
100%
0%
2020
2021
2022
2023


Distribution of Monthly Returns
2020 - 2023
Occurrences
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-40%
-20%
0%
19%
39%
Benchmark
Strategy


Daily Returns
3 Jan '20 - 15 Aug '23
10%
0%
-10%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

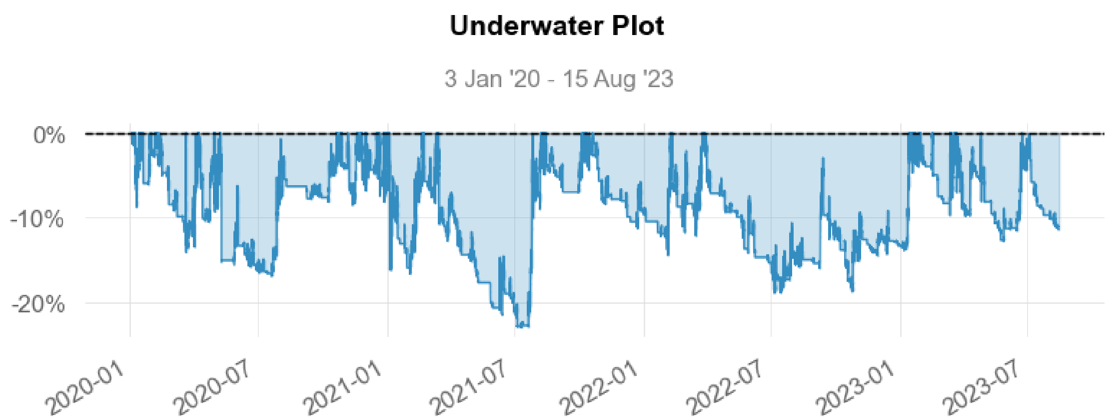

Rolling Beta to Benchmark
3 Jan '20 - 15 Aug '23
0.5
0.0
−0.5
−1.0
Strategy (6-Months)
Strategy (12-Months)
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07


Rolling Volatility (6-Months)
3 Jul '20 - 15 Aug '23
Strategy
Benchmark
0.05
0.00
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09

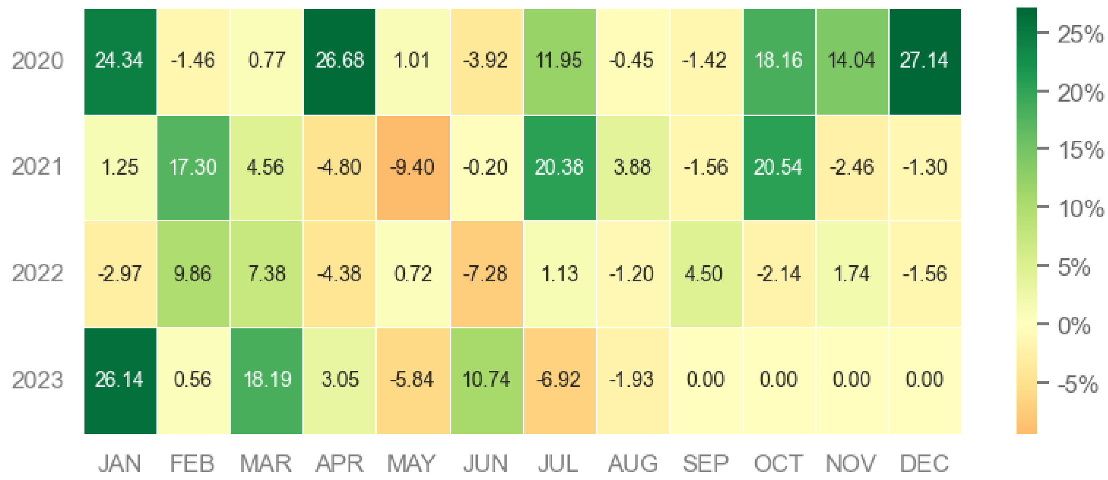

Rolling Sharpe (6-Months)
3 Jul '20 - 15 Aug '23
0.40
0.20
0.00
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09


Rolling Sortino (6-Months)
3 Jul '20 - 15 Aug '23
0.50
0.25
0.00
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09


Strategy - Worst 5 Drawdown Periods
3 Jan '20 - 15 Aug '23
600%
400%
200%
0%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07


Underwater Plot
3 Jan '20 - 15 Aug '23
0%
-10%
-20%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

Strategy - Monthly Returns (%)



Strategy - Return Quantiles

3 Jan '20 - 15 Aug '23