```
import numpy as np
import pandas as pd
from datetime import datetime
import datetime as dt
import matplotlib.pyplot as plt
import yfinance as yf
import pyfolio as pf
```

```
/opt/anaconda3/lib/python3.8/site-packages/pyfolio/pos.py:26:
UserWarning: Module "zipline.assets" not found; mutltipliers will not
be applied to position notionals.
  warnings.warn(
```

```
class backtesting_crossover:

    def __init__(self, ticker, start_date, end_date, ma_short,
ma_long):
        self.ticker = ticker
        self.start_date = start_date
        self.end_date = end_date
        self.ma_short = ma_short
        self.ma_long = ma_long
        self.fetchdata()
        self.indicators()
        self.signals()
        self.positions()
        self.returns()
        self.analysis()

    def fetchdata(self):
        self.df = yf.download(self.ticker, self.start_date,
self.end_date)

    def indicators(self):
        self.df['ma_short'] = self.df['Adj
Close'].rolling(self.ma_short).mean()
        self.df['ma_long'] = self.df['Adj
Close'].rolling(self.ma_long).mean()
        self.df['ma_short_prev_day'] = self.df['ma_short'].shift(1)
        self.df['ma_long_prev_day'] = self.df['ma_long'].shift(1)
        self.df.dropna(inplace=True)

    def signals(self):
        self.df['signal'] = np.where((self.df['ma_short'] >
self.df['ma_long'])
                                     & (self.df['ma_short_prev_day'] <
self.df['ma_long_prev_day']),1,0)
        self.df['signal'] = np.where((self.df['ma_short'] <
self.df['ma_long'])
                                     & (self.df['ma_short_prev_day'] >
self.df['ma_long_prev_day']),-1,self.df['signal'])
```

```
    def positions(self):
        self.df['position'] = self.df['signal'].replace(to_replace =
0, method = 'ffill')
#       print(self.df[60:100])

    def returns(self):
        self.df['bnh_returns'] = np.log(self.df['Adj Close'] /
self.df['Adj Close'].shift(1))
        self.df['strategy_returns'] =
self.df['bnh_returns']*self.df['position'].shift(1)
        print('Total return: ', self.df['strategy_returns'].sum())
        return self.df['strategy_returns'].sum()

    def analysis(self):

self.df[['ma_short','ma_long','position']].plot(figsize=(15,6),grid =
True)
        plt.show()

self.df[['bnh_returns','strategy_returns']].cumsum().plot(figsize=(15,
6),grid = True)
        plt.show()
        pf.create_simple_tear_sheet(self.df['strategy_returns'])

end1 = dt.datetime(2021,10,31).date()
start1 = end1 - pd.Timedelta(days=3*252)

start1

datetime.date(2019, 10, 6)

backtesting_crossover('AAPL',start1,end1,10,20)

[**********100%**********]  1 of 1 completed
                 Open       High        Low      Close  Adj Close
Volume  \
Date

2020-01-31  80.232498  80.669998  77.072502  77.377502  76.244957
199588400
2020-02-03  76.074997  78.372498  75.555000  77.165001  76.035568
173788400
2020-02-04  78.827499  79.910004  78.407501  79.712502  78.545776
136616400
2020-02-05  80.879997  81.190002  79.737503  80.362503  79.186272
118826800
2020-02-06  80.642502  81.305000  80.065002  81.302498  80.112495
105425600
2020-02-07  80.592499  80.849998  79.500000  80.007500  79.023567
117684000
```

2020-02-10 78.544998 80.387497 78.462502 80.387497 79.398880
109348800
2020-02-11 80.900002 80.974998 79.677498 79.902496 78.919846
94323200
2020-02-12 80.367500 81.805000 80.367500 81.800003 80.794022
113730400
2020-02-13 81.047501 81.555000 80.837502 81.217499 80.218674
94747600
2020-02-14 81.184998 81.495003 80.712502 81.237503 80.238434
80113600
2020-02-18 78.839996 79.937500 78.652496 79.750000 78.769226
152531200
2020-02-19 80.000000 81.142502 80.000000 80.904999 79.910027
93984000
2020-02-20 80.657501 81.162498 79.552498 80.074997 79.090225
100566000
2020-02-21 79.654999 80.112503 77.625000 78.262497 77.300026
129554000
2020-02-24 74.315002 76.044998 72.307503 74.544998 73.628235
222195200
2020-02-25 75.237503 75.632500 71.532501 72.019997 71.134293
230673600
2020-02-26 71.632500 74.470001 71.625000 73.162498 72.262749
198054800
2020-02-27 70.275002 71.500000 68.239998 68.379997 67.539047
320605600
2020-02-28 64.315002 69.602501 64.092499 68.339996 67.499542
426510000
2020-03-02 70.570000 75.360001 69.430000 74.702499 73.783806
341397200
2020-03-03 75.917503 76.000000 71.449997 72.330002 71.440483
319475600
2020-03-04 74.110001 75.849998 73.282501 75.684998 74.754227
219178400
2020-03-05 73.879997 74.887497 72.852501 73.230003 72.329407
187572800
2020-03-06 70.500000 72.705002 70.307503 72.257500 71.368881
226176800
2020-03-09 65.937500 69.522499 65.750000 66.542503 65.724152
286744800
2020-03-10 69.285004 71.610001 67.342499 71.334999 70.457718
285290000
2020-03-11 69.347504 70.305000 67.964996 68.857498 68.010689
255598800
2020-03-12 63.985001 67.500000 62.000000 62.057499 61.294308
418474000
2020-03-13 66.222504 69.980003 63.237499 69.492500 68.637886
370732000
2020-03-16 60.487499 64.769997 60.000000 60.552502 59.807823
322423600

```
2020-03-17  61.877499  64.402496  59.599998  63.215000  62.437576
324056000
2020-03-18  59.942501  62.500000  59.279999  61.667500  60.909103
300233600
2020-03-19  61.847500  63.209999  60.652500  61.195000  60.442421
271857200
2020-03-20  61.794998  62.957500  57.000000  57.310001  56.605198
401693200
2020-03-23  57.020000  57.125000  53.152500  56.092499  55.402672
336752800
2020-03-24  59.090000  61.922501  58.575001  61.720001  60.960964
287531200
2020-03-25  62.687500  64.562500  61.075001  61.380001  60.625141
303602000
2020-03-26  61.630001  64.669998  61.590000  64.610001  63.815422
252087200
2020-03-27  63.187500  63.967499  61.762501  61.935001  61.173317
204216800

             ma_short    ma_long  ma_short_prev_day  ma_long_prev_day
signal  \
Date

2020-01-31  78.210757  76.980160          78.351910         76.867336
0
2020-02-03  77.962692  77.118481          78.210757         76.980160
0
2020-02-04  78.018858  77.353121          77.962692         77.118481
0
2020-02-05  78.111237  77.637152          78.018858         77.353121
0
2020-02-06  78.258547  77.908372          78.111237         77.637152
0
2020-02-07  78.319627  78.045824          78.258547         77.908372
0
2020-02-10  78.648814  78.193421          78.319627         78.045824
0
2020-02-11  78.714796  78.235404          78.648814         78.193421
0
2020-02-12  78.804380  78.423812          78.714796         78.235404
0
2020-02-13  78.848006  78.599958          78.804380         78.423812
0
2020-02-14  79.247353  78.729055          78.848006         78.599958
0
2020-02-18  79.520719  78.741706          79.247353         78.729055
0
2020-02-19  79.657144  78.838001          79.520719         78.741706
0
2020-02-20  79.647540  78.879388          79.657144         78.838001
```

0
2020-02-21 79.366293 78.812420 79.647540 78.879388
0
2020-02-24 78.826759 78.573193 79.366293 78.812420
0
2020-02-25 78.000301 78.324557 78.826759 78.573193
-1
2020-02-26 77.334591 78.024694 78.000301 78.324557
0
2020-02-27 76.009093 77.406737 77.334591 78.024694
0
2020-02-28 74.737180 76.792593 76.009093 77.406737
0
2020-03-02 74.091718 76.669535 74.737180 76.792593
0
2020-03-03 73.358843 76.439781 74.091718 76.669535
0
2020-03-04 72.843263 76.250204 73.358843 76.439781
0
2020-03-05 72.167181 75.907360 72.843263 76.250204
0
2020-03-06 71.574067 75.470180 72.167181 75.907360
0
2020-03-09 70.783659 74.805209 71.574067 75.470180
0
2020-03-10 70.716001 74.358151 70.783659 74.805209
0
2020-03-11 70.290795 73.812693 70.716001 74.358151
0
2020-03-12 69.666321 72.837707 70.290795 73.812693
0
2020-03-13 69.780156 72.258668 69.666321 72.837707
0
2020-03-16 68.382557 71.237137 69.780156 72.258668
0
2020-03-17 67.482267 70.420555 68.382557 71.237137
0
2020-03-18 66.097754 69.470509 67.482267 70.420555
0
2020-03-19 64.909056 68.538119 66.097754 69.470509
0
2020-03-20 63.432687 67.503377 64.909056 68.538119
0
2020-03-23 62.400539 66.592099 63.432687 67.503377
0
2020-03-24 61.450864 66.083433 62.400539 66.592099
0
2020-03-25 60.712309 65.501552 61.450864 66.083433
0
2020-03-26 60.964421 65.315371 60.712309 65.501552

```
0
2020-03-27  60.217964  64.999060          60.964421         65.315371
0

            position
Date
2020-01-31         0
2020-02-03         0
2020-02-04         0
2020-02-05         0
2020-02-06         0
2020-02-07         0
2020-02-10         0
2020-02-11         0
2020-02-12         0
2020-02-13         0
2020-02-14         0
2020-02-18         0
2020-02-19         0
2020-02-20         0
2020-02-21         0
2020-02-24         0
2020-02-25        -1
2020-02-26        -1
2020-02-27        -1
2020-02-28        -1
2020-03-02        -1
2020-03-03        -1
2020-03-04        -1
2020-03-05        -1
2020-03-06        -1
2020-03-09        -1
2020-03-10        -1
2020-03-11        -1
2020-03-12        -1
2020-03-13        -1
2020-03-16        -1
2020-03-17        -1
2020-03-18        -1
2020-03-19        -1
2020-03-20        -1
2020-03-23        -1
2020-03-24        -1
2020-03-25        -1
2020-03-26        -1
2020-03-27        -1
Total return:  0.6045105009822337

<__main__.backtesting_crossover at 0x127470f10>
```

```
fast_ma_list = [5,10,15,20]
slow_ma_list = [25,50,100]

fast_ma = []
slow_ma = []
net_returns = []

for i in fast_ma_list:
    for j in slow_ma_list:
        print('For',i,j)
        a = backtesting_crossover('AAPL', start1, end1, i, j)
        fast_ma.append(i)
        slow_ma.append(j)
        net_returns.append(a.returns())
```

```
For 5 25
[**********************100%**********************]  1 of 1 completed
Total return:  0.7970909846803547
Total return:  0.7970909846803547
For 5 50
[**********************100%**********************]  1 of 1 completed
Total return:  0.22548275799238157
Total return:  0.22548275799238157
For 5 100
[**********************100%**********************]  1 of 1 completed
Total return:  0.467317301181575
Total return:  0.467317301181575
For 10 25
[**********************100%**********************]  1 of 1 completed
Total return:  0.5663468952131981
Total return:  0.5663468952131981
For 10 50
[**********************100%**********************]  1 of 1 completed
Total return:  0.23738469343371246
Total return:  0.23738469343371246
For 10 100
[**********************100%**********************]  1 of 1 completed
Total return:  0.14237396464073154
Total return:  0.14237396464073154
For 15 25
[**********************100%**********************]  1 of 1 completed
Total return:  0.24361695637555247
Total return:  0.24361695637555247
For 15 50
[**********************100%**********************]  1 of 1 completed
Total return:  0.11306826993005523
Total return:  0.11306826993005523
For 15 100
[**********************100%**********************]  1 of 1 completed
Total return:  0.14542100995223914
```

```
Total return:  0.14542100995223914
For 20 25
[*********************100%***********************]  1 of 1 completed
Total return:  0.3164024433020568
Total return:  0.3164024433020568
For 20 50
[*********************100%***********************]  1 of 1 completed
Total return:  0.1722267996251069
Total return:  0.1722267996251069
For 20 100
[*********************100%***********************]  1 of 1 completed
Total return:  0.1266164676534186
Total return:  0.1266164676534186
```

```
results = pd.DataFrame({'fast ma': fast_ma, 'slow ma': slow_ma, 'net
returns': net_returns})
results
```

	fast ma	slow ma	net returns
0	5	25	0.797091
1	5	50	0.225483
2	5	100	0.467317
3	10	25	0.566347
4	10	50	0.237385
5	10	100	0.142374
6	15	25	0.243617
7	15	50	0.113068
8	15	100	0.145421
9	20	25	0.316402
10	20	50	0.172227
11	20	100	0.126616

```
results.sort_values(by = 'net returns', ascending = False)
```

	fast ma	slow ma	net returns
0	5	25	0.797091
3	10	25	0.566347
2	5	100	0.467317
9	20	25	0.316402
6	15	25	0.243617
4	10	50	0.237385
1	5	50	0.225483
10	20	50	0.172227
8	15	100	0.145421
5	10	100	0.142374
11	20	100	0.126616
7	15	50	0.113068

```
stock_list = ['PFE', 'F', 'LCID', 'NVDA', 'PTON', 'AMD', 'MRNA',
'UBER', 'BAC', 'INTC', 'NIO', 'AAPL']
```

```
stock_name = []
net_returns = []

for stock in stock_list:
    print('Backtesting result for', stock)
    a = backtesting_crossover(stock, start1, end1, 5, 25)
    stock_name.append(stock)
    net_returns.append(a.returns())
```

```
Backtesting result for PFE
[**********************100%**********************]  1 of 1 completed
Total return:  0.49010888138576847
Total return:  0.49010888138576847
Backtesting result for F
[**********************100%**********************]  1 of 1 completed
Total return:  0.7912078443872355
Total return:  0.7912078443872355
Backtesting result for LCID
[**********************100%**********************]  1 of 1 completed
Total return:  1.66860313684292
Total return:  1.66860313684292
Backtesting result for NVDA
[**********************100%**********************]  1 of 1 completed
Total return:  -0.1884478859616186
Total return:  -0.1884478859616186
Backtesting result for PTON
[**********************100%**********************]  1 of 1 completed
Total return:  1.480369329997603
Total return:  1.480369329997603
Backtesting result for AMD
[**********************100%**********************]  1 of 1 completed
Total return:  -0.15596749011429242
Total return:  -0.15596749011429242
Backtesting result for MRNA
[**********************100%**********************]  1 of 1 completed
Total return:  0.34844010652841473
Total return:  0.34844010652841473
Backtesting result for UBER
[**********************100%**********************]  1 of 1 completed
Total return:  -1.4254927234956187
Total return:  -1.4254927234956187
Backtesting result for BAC
[**********************100%**********************]  1 of 1 completed
Total return:  -0.07178115696885906
Total return:  -0.07178115696885906
Backtesting result for INTC
[**********************100%**********************]  1 of 1 completed
Total return:  -0.2555154402566164
Total return:  -0.2555154402566164
Backtesting result for NIO
[**********************100%**********************]  1 of 1 completed
```

```
Total return:  2.6628262997783927
Total return:  2.6628262997783927
Backtesting result for AAPL
[**********************100%***********************]  1 of 1 completed
Total return:  0.7970914438972815
Total return:  0.7970914438972815
```

```
results1 = pd.DataFrame({'Stock': stock_name, 'Net Returns':
net_returns})
results1
```

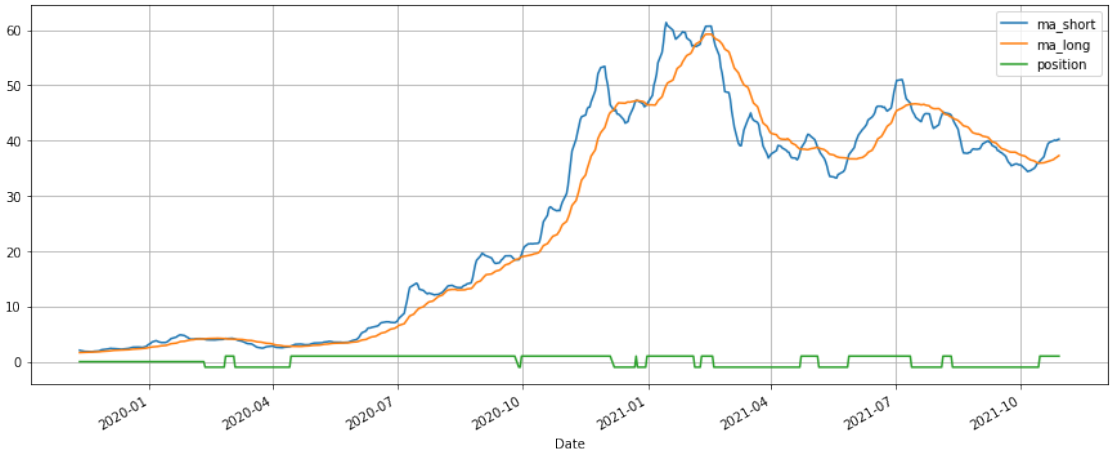
	Stock	Net Returns
0	PFE	0.490109
1	F	0.791208
2	LCID	1.668603
3	NVDA	-0.188448
4	PTON	1.480369
5	AMD	-0.155967
6	MRNA	0.348440
7	UBER	-1.425493
8	BAC	-0.071781
9	INTC	-0.255515
10	NIO	2.662826
11	AAPL	0.797091

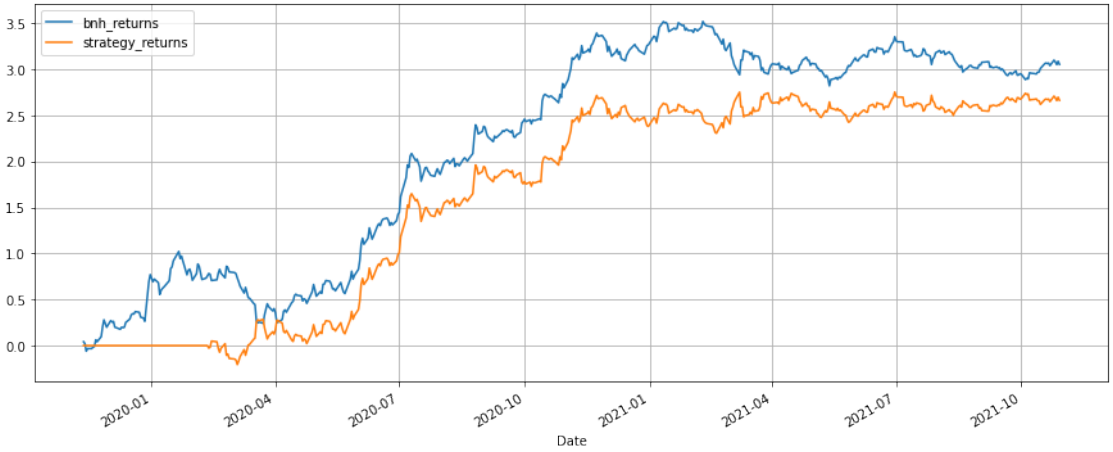
```
results1.sort_values(by = 'Net Returns', ascending = False)
```

	Stock	Net Returns
10	NIO	2.662826
2	LCID	1.668603
4	PTON	1.480369
11	AAPL	0.797091
1	F	0.791208
0	PFE	0.490109
6	MRNA	0.348440
8	BAC	-0.071781
5	AMD	-0.155967
3	NVDA	-0.188448
9	INTC	-0.255515
7	UBER	-1.425493

```
backtesting_crossover('NIO',start1,end1,5,25)
```

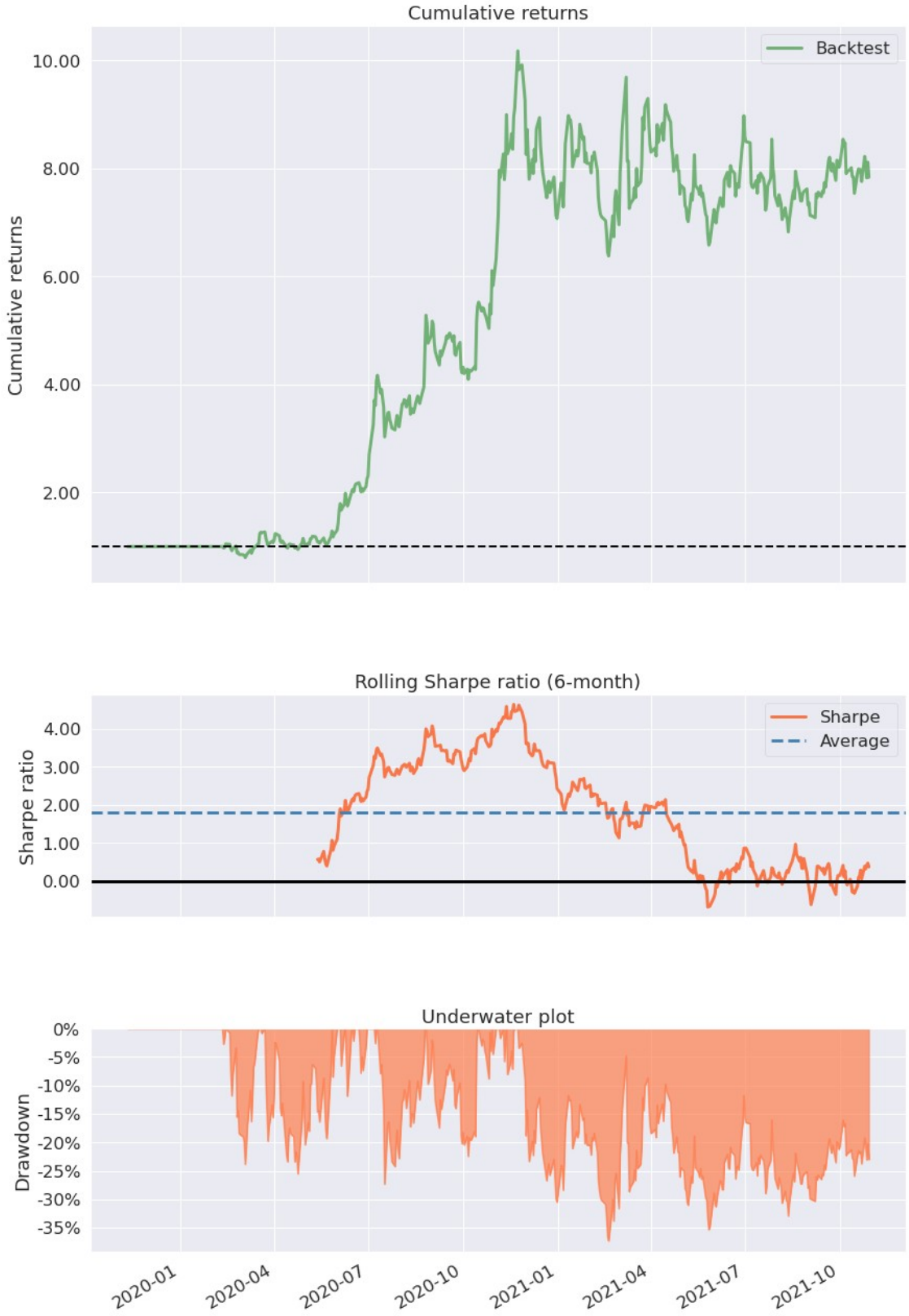
```
[**********************100%***********************]  1 of 1 completed
Total return:  2.6628262997783927
```






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

```
<__main__.backtesting_crossover at 0x1281b0dc0>
```


Cumulative returns
Backtest
10.00
8.00
6.00
4.00
2.00
Cumulative returns
Rolling Sharpe ratio (6-month)
Sharpe
Average
4.00
3.00
2.00
1.00
0.00
Sharpe ratio
Underwater plot
0%
-5%
-10%
-15%
-20%
-25%
-30%
-35%
Drawdown
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10