# Setup

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy
import seaborn as sns
import yfinance as yf
import arch
import riskfolio as rp

%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
tic=pd.read_excel('TIC.xlsx')
```

In [3]:
```
tic
```

Out[3]:

| | Nomi | Pesi | Ticker |
|---|---|---|---|
| 0 | S&P 500 | 0.21 | ^GSPC |
| 1 | CONV BOND | 0.07 | GCVB.L |
| 2 | EUR SHORT BOND | 0.15 | ERNE.L |
| 3 | INFL LINKED | 0.05 | EMI.MI |
| 4 | GOLD | 0.16 | GLD |
| 5 | MSCI WORLD | 0.20 | MWL=F |
| 6 | US TREASURY | 0.06 | SHV |
| 7 | COMMODITIES | 0.06 | DBC |
| 8 | EUR LONG BOND | 0.04 | BLV |

In [4]:
```
tickers=list(tic['Ticker'])
tickers.sort()
```

In [5]:
```
tickers
```

Out[5]:
```
['BLV', 'DBC', 'EMI.MI', 'ERNE.L', 'GCVB.L', 'GLD', 'MWL=F', 'SHV', '^GSPC']
```

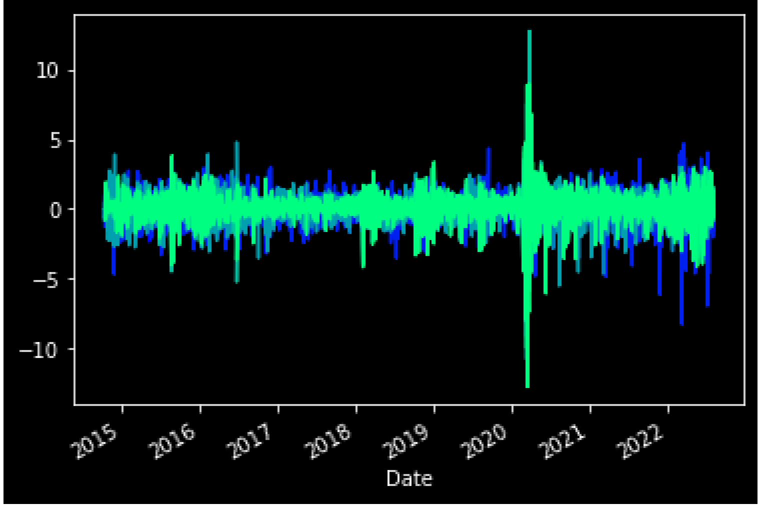
In [6]:
```
data=yf.download(tickers,interval='1d')[['Adj Close']]
```

```
[*********************100%***********************]  9 of 9 completed
```

In [7]:
```
data.dropna(inplace=True)
```

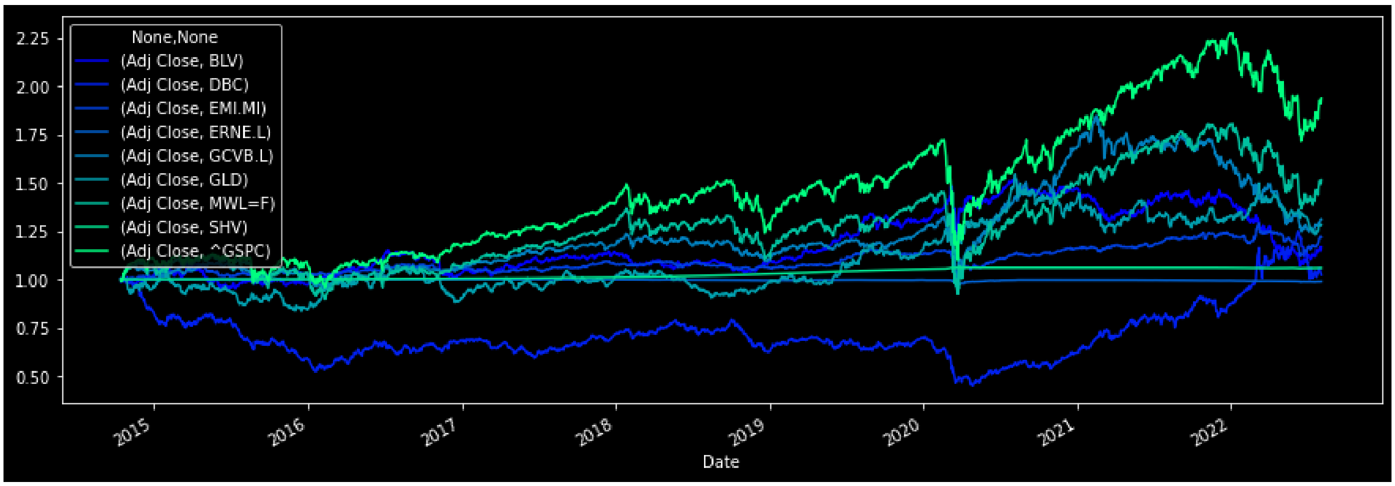
In [8]:
```
plt.style.use('dark_background')

rets=((np.log(data) - np.log(data.shift(1))).dropna()*100)

rets.plot(legend=False,cmap='winter')
```

Out[8]:
```
<AxesSubplot:xlabel='Date'>
```




In [9]:
```
(rets/100+1).cumprod().plot(figsize=(15,5),cmap='winter')
```

Out[9]:
```
<AxesSubplot:xlabel='Date'>
```




# Sample mean

Assumiamo che la sample mean sia informativa

In [10]:
```
rets.columns=tickers
model={}
result={}
for i in tickers:
    model[i]=arch.univariate.ConstantMean(rets.loc[:,i])
    result[i]=model[i].fit(disp="off")
```

# Monte Carlo

Assumiamo che i rendimenti seguano una distribuzione multivariata con t di student (con 8 gradi di libertà)

i.e. $R_t \sim MVT(\mu, \Sigma)$

Usiamo la fattorizzazione di Cholesky per trovare la matrice triangolare inferiore $L$ tale che $LL' = \Sigma$

Quindi i rendimenti degli asset possono essere descritti come:

$R_T = \mu + LZ_t$

Dove $Z_t \sim student\ t_{n=8}$

In [11]:
```
n_t=52
n_mc=5000
```

In [12]:
```
weights=np.array(tic.iloc[:,1])
```

# Sample mean

In [13]:
```
mu={}
for i in tickers:
    mu[i]=result[i].params['mu']/100
mu_df=pd.DataFrame.from_dict(mu,orient='index')
```

# Var/Cov Matrix con denoising and detoning

In [14]:
```
cov=rp.ParamsEstimation.covar_matrix(rets/100,method='fixed')
```

In [15]:
```
from datetime import datetime, timedelta
```

# Loop

In [16]:
```
portf_returns = np.full((n_t,n_mc),0.)

for i in range(0,n_mc):
    Z = np.random.standard_t(12,size=len(tickers)*n_t)
    Z = Z.reshape((len(tickers),n_t))
    L = np.linalg.cholesky(cov)
    wkrets=np.inner(L,np.transpose(Z))+np.array(mu_df)
    portf_r = np.cumprod(np.inner(weights,np.transpose(wkrets)) + 1)
    future_dates = [rets.index[-1] + timedelta(weeks=x) for x in range(0,n_t+1)]
    portf_returns[:,i] = portf_r
```

# Funzioni

In [17]:
```
def band(r,n_t=n_t):
    y2=r
    y1=1
    lt2=np.log(n_t)
    b=(y2-y1)/lt2
    a=y1

    U= np.zeros([n_t + 1, 1])
    for t in range(0, int(n_t)+1):
      U[t] = a+b*np.log(t+1)
    U_df=pd.DataFrame(U,index=future_dates)
    return U_df
```

In [18]:
```
def trend(r,n_t=n_t):
    L= np.zeros([n_t + 1, 1])
    L[0]=1
    for t in range(1, int(n_t)+1):
      L[t] = L[t-1]*(r**(1/n_t))
    L_df=pd.DataFrame(L,index=future_dates)
    return L_df
```

# Grafico

In [19]:
```
ptf_returns=np.insert(portf_returns, 0, 1, axis=0)
```

In [20]:
```
hist=((rets@weights)/100+1).cumprod()
```

In [21]:
```
f = plt.figure()
f.set_figwidth(15)
f.set_figheight(10)

with sns.color_palette("winter"):
    plt.plot(hist[-1]*pd.DataFrame(ptf_returns,index=future_dates))


plt.plot(hist)

plt.plot(hist[-1]*band(np.quantile(portf_returns[-1],q=0.99)),color='white', label='99% 
plt.plot(hist[-1]*band(np.quantile(portf_returns[-1],q=0.01)),color='white',)
plt.plot(hist[-1]*band(np.quantile(portf_returns[-1],q=0.9)),color='white', label='80% p
plt.plot(hist[-1]*band(np.quantile(portf_returns[-1],q=0.1)),color='white', linestyle='d



plt.plot(hist[-1]*trend(np.quantile(portf_returns[-1],q=0.5)),color='white')
plt.title('Simulazione Monte Carlo Portafoglio All Weather - Prossime 52 settimane')
plt.legend()
plt.plot()
```

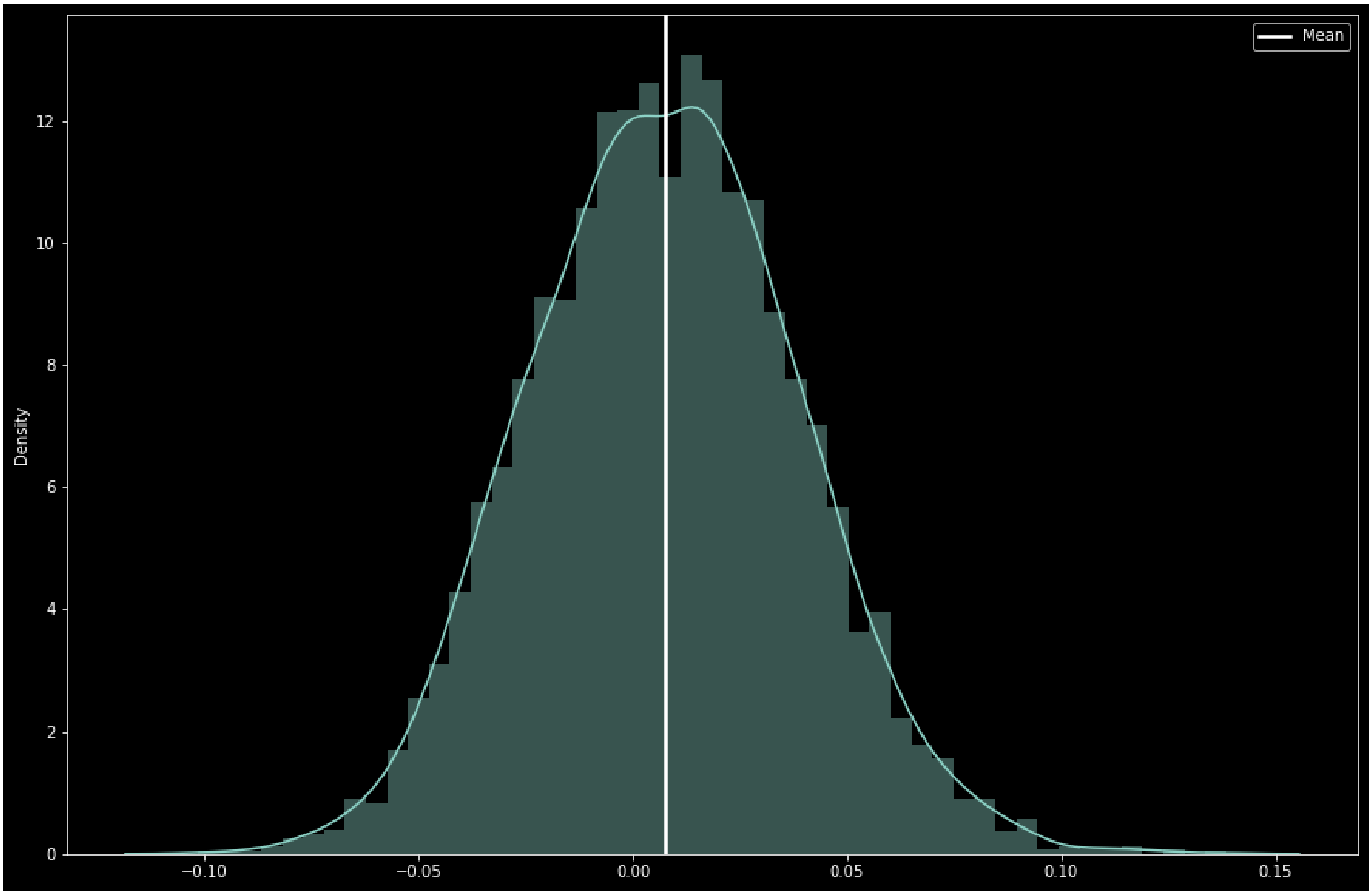
Out[21]: []


In [22]:
```
print('Nel 99% dei casi non dovresti perdere più del: ' +str((np.quantile(portf_returns[
print('Nel 99% dei casi non dovresti guadagnare più del: ' +str((np.quantile(portf_retur
```

```
Nel 99% dei casi non dovresti perdere più del: -6.251346273772729
Nel 99% dei casi non dovresti guadagnare più del: 8.12534416325783
```

In [23]:
```
f = plt.figure()
f.set_figwidth(15)
f.set_figheight(10)


sns.distplot(portf_returns[-1]-1)

plt.axvline((portf_returns[-1]-1).mean(), c='white', ls='-', lw=2.5, label='Mean')
plt.legend()
```

Out[23]:
```
<matplotlib.legend.Legend at 0x253560c22e0>
```




# Analisi storica All Weather

In [24]:
```
import pyfolio
```

In [33]:
```
pyfolio.create_full_tear_sheet(hist.pct_change().dropna(),benchmark_rets=rets['^GSPC']/1
```

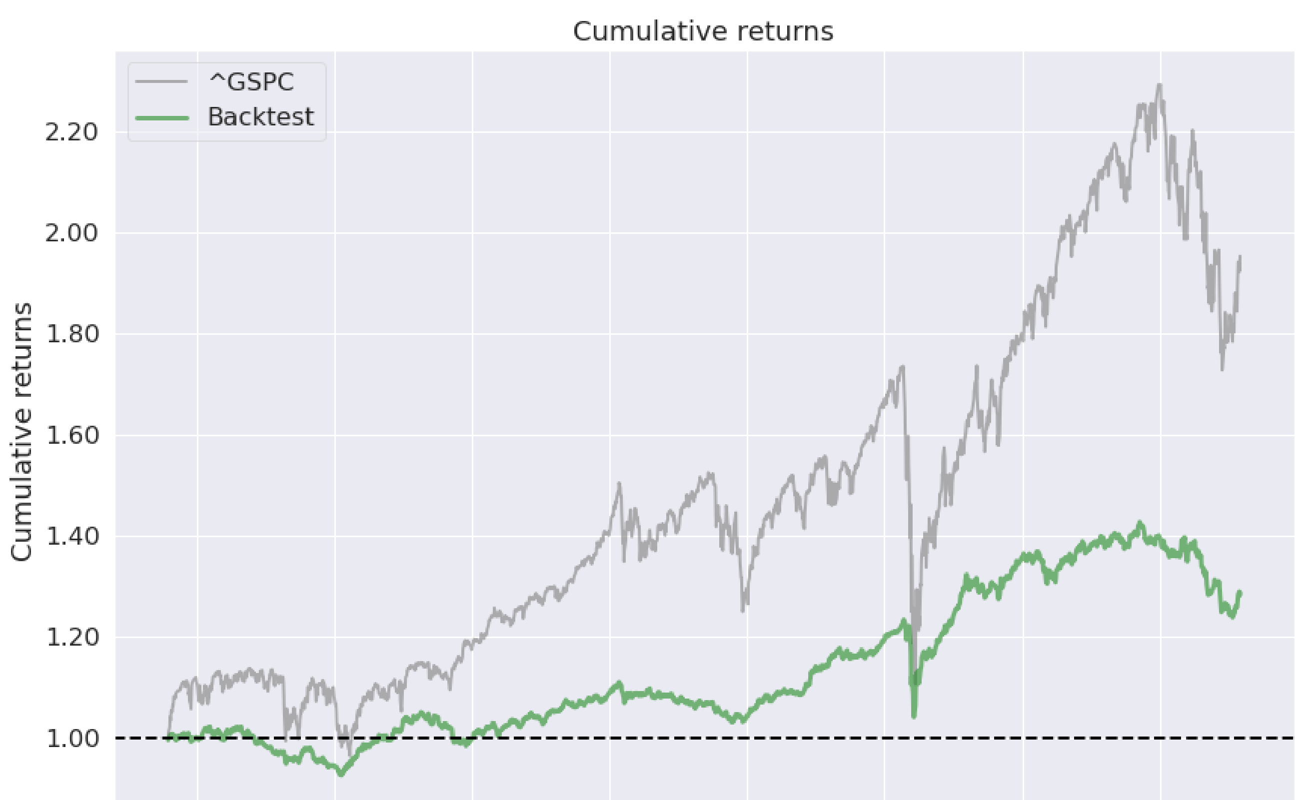
| | |
|---|---|
| **Start date** | 2014-10-16 |
| **End date** | 2022-08-03 |
| **Total months** | 90 |

| | Backtest |
|---|---|
| **Annual return** | 3.414% |
| **Cumulative returns** | 28.85% |
| **Annual volatility** | 6.467% |

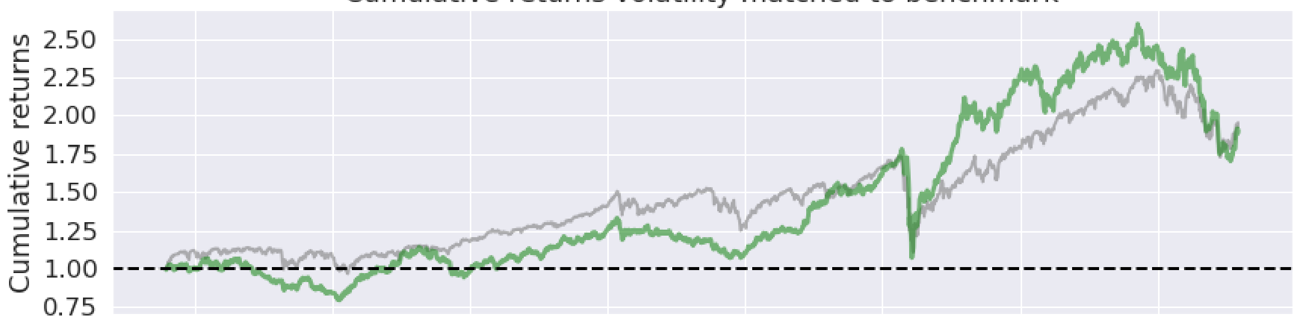
| | |
|---|---|
| **Sharpe ratio** | 0.55 |
| **Calmar ratio** | 0.22 |
| **Stability** | 0.86 |
| **Max drawdown** | -15.616% |
| **Omega ratio** | 1.11 |
| **Sortino ratio** | 0.75 |
| **Skew** | -1.49 |
| **Kurtosis** | 23.00 |
| **Tail ratio** | 0.99 |
| **Daily value at risk** | -0.801% |
| **Alpha** | 0.02 |
| **Beta** | 0.17 |

| Worst drawdown periods | Net drawdown in % | Peak date | Valley date | Recovery date | Duration |
|---|---|---|---|---|---|
| **0** | 15.62 | 2020-02-21 | 2020-03-19 | 2020-07-01 | 94 |
| **1** | 13.24 | 2021-11-09 | 2022-07-14 | NaT | NaN |
| **2** | 9.45 | 2015-04-28 | 2016-01-20 | 2016-06-29 | 307 |
| **3** | 7.09 | 2018-01-26 | 2018-11-23 | 2019-06-18 | 363 |
| **4** | 6.31 | 2016-08-18 | 2016-12-15 | 2017-07-28 | 247 |

| Stress Events | mean | min | max |
|---|---|---|---|
| **Oct14** | 0.02% | -0.30% | 0.43% |
| **Fall2015** | -0.06% | -1.16% | 0.75% |
| **New Normal** | 0.01% | -5.35% | 3.33% |

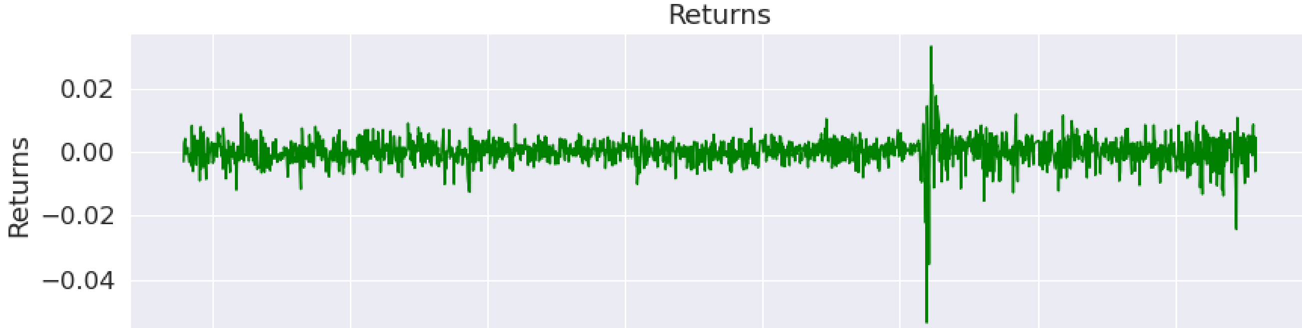


Cumulative returns volatility matched to benchmark

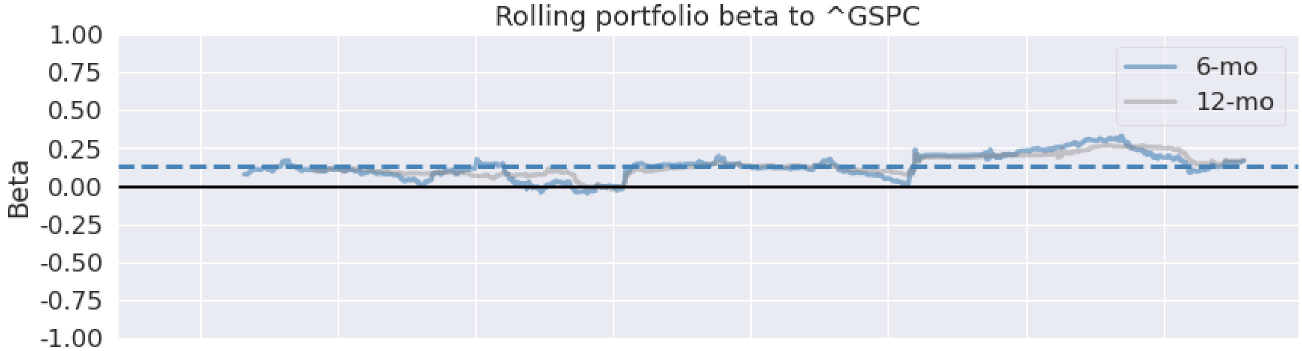



Cumulative returns on logarithmic scale

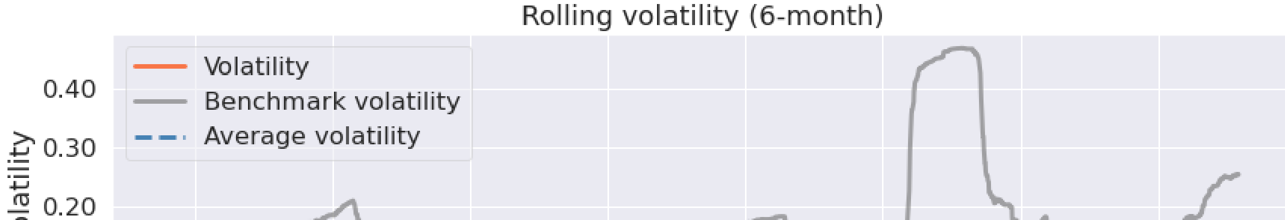

Returns

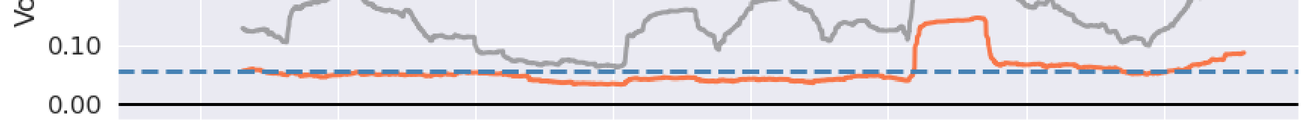



Rolling portfolio beta to ^GSPC

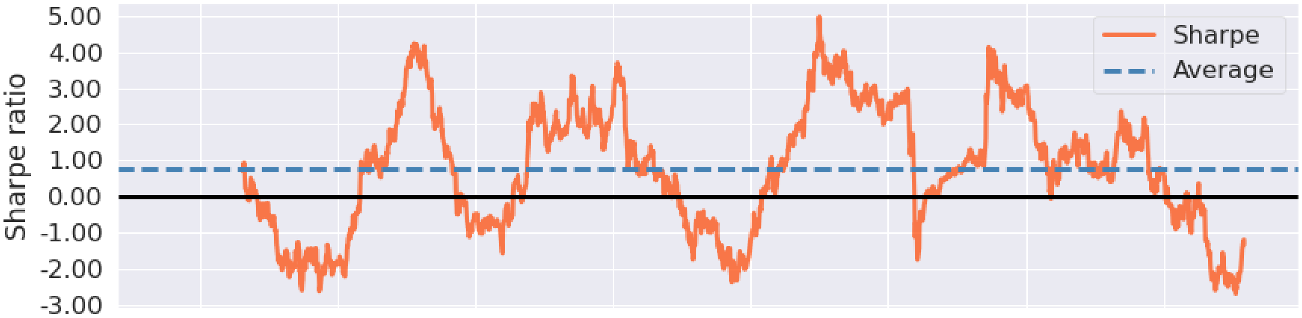



Rolling volatility (6-month)

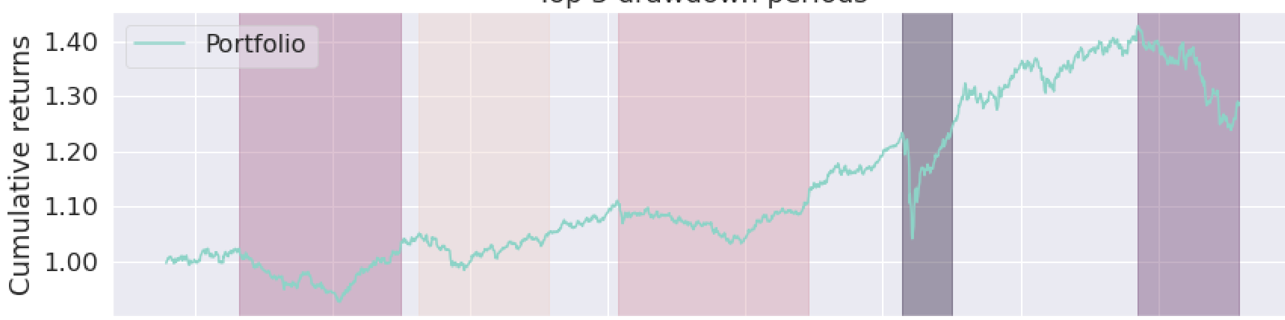




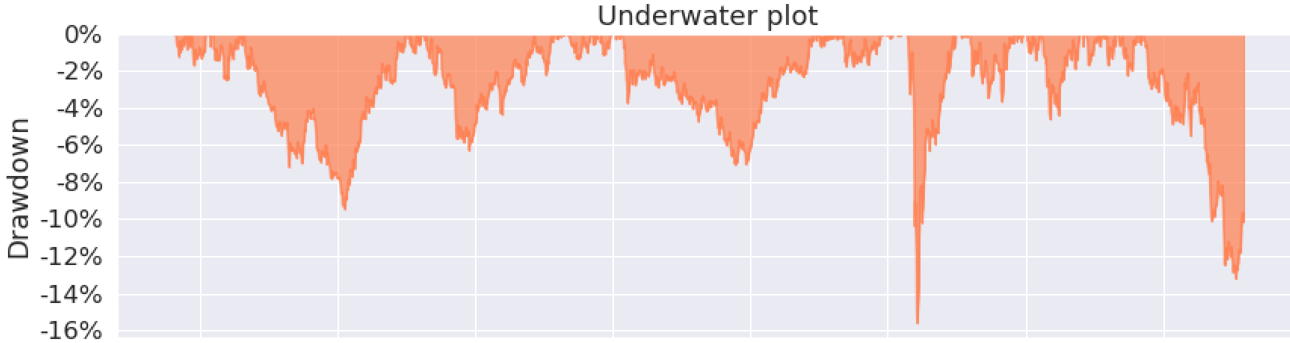

Rolling Sharpe ratio (6-month)



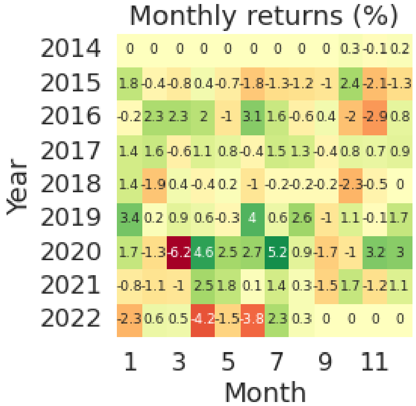

Top 5 drawdown periods



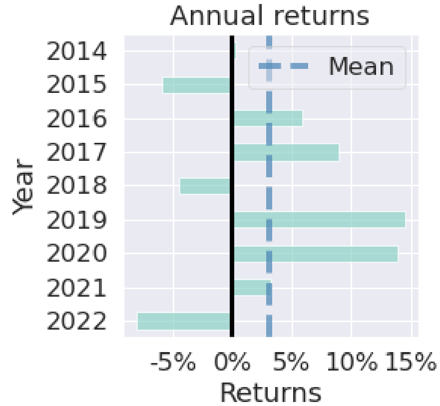

Underwater plot



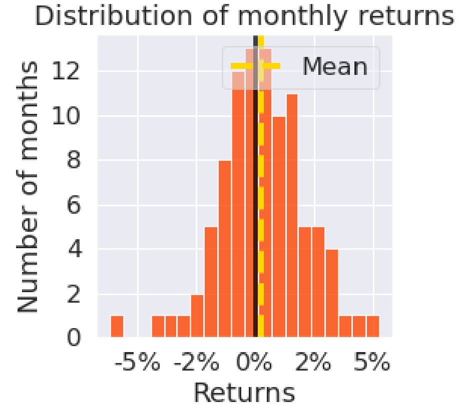

Monthly returns (%)



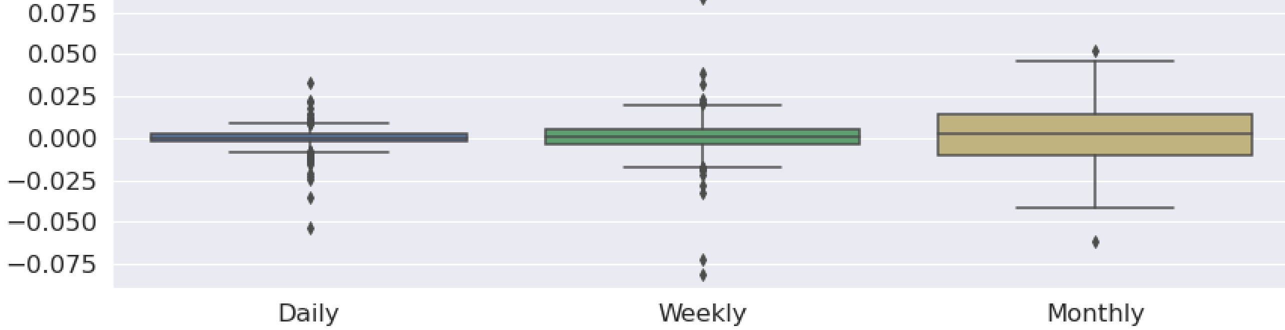

Annual returns

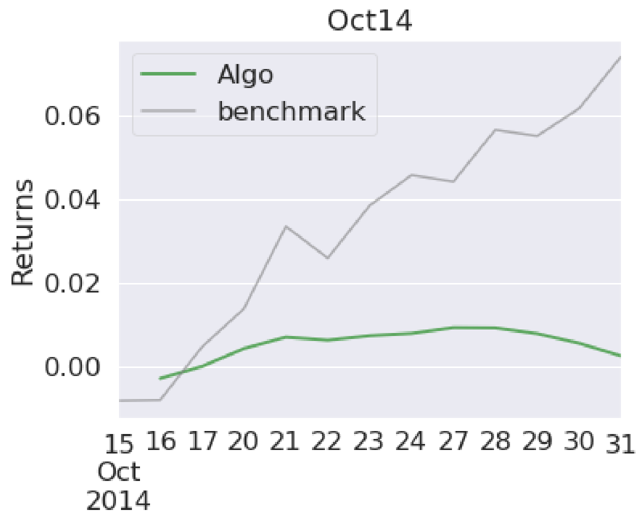


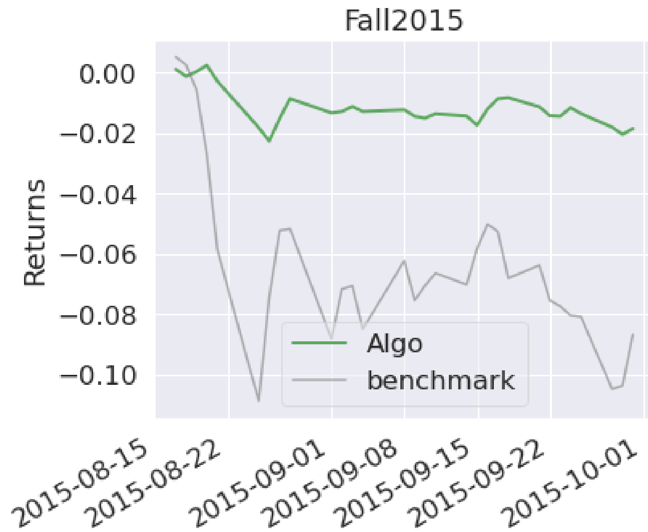

Distribution of monthly returns

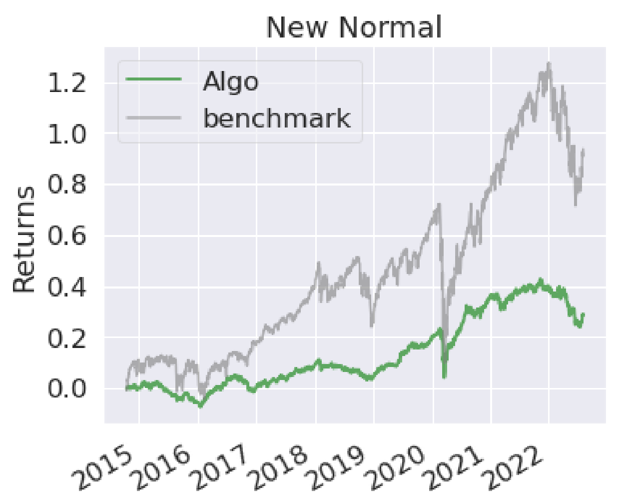



Return quantiles


0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
Daily
Weekly
Monthly


Oct14
Algo
benchmark
Returns
0.06
0.04
0.02
0.00
15 16 17 20 21 22 23 24 27 28 29 30 31
Oct
2014


Fall2015
Returns
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
Algo
benchmark
2015-08-15
2015-08-22
2015-09-01
2015-09-08
2015-09-15
2015-09-22
2015-10-01


New Normal
Algo
benchmark
Returns
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2015 2016 2017 2018 2019 2020 2021 2022

```
In [ ]:
```