In [151]:

```
import importlib

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import datetime

from model.Portfolio import Portfolio
from model.Optimizer import Optimizer
plt.rcParams["figure.figsize"] = 10, 15
```

In [153]:

```
names = ["C38U", "ND8U", "V01", "AGS", "N2IU"]

p = Portfolio()

# Set risk-free investment as 2%, approximately SSB's returns
p.rf = 0.02

# Add all assets
for name in names:
    p.addAsset(f"data/{name}.csv", name)

# Convert non SGD assets to SGD
p.addExchangeRate("data/forex/SGDEUR.csv", "EUR", True)
p.addExchangeRate("data/forex/USDSGD.csv", "USD", False)
```

In [154]:

```
currentWeight = [20, 20, 20, 10, 10]

normalisedWeight = np.array(currentWeight)/np.sum(currentWeight)
normalisedWeight
```

Out[154]:

```
array([0.25 , 0.25 , 0.25 , 0.125, 0.125])
```

In [155]:

```
currentResult, currentBtPlot = p.backtest(normalisedWeight)
```

In [156]:

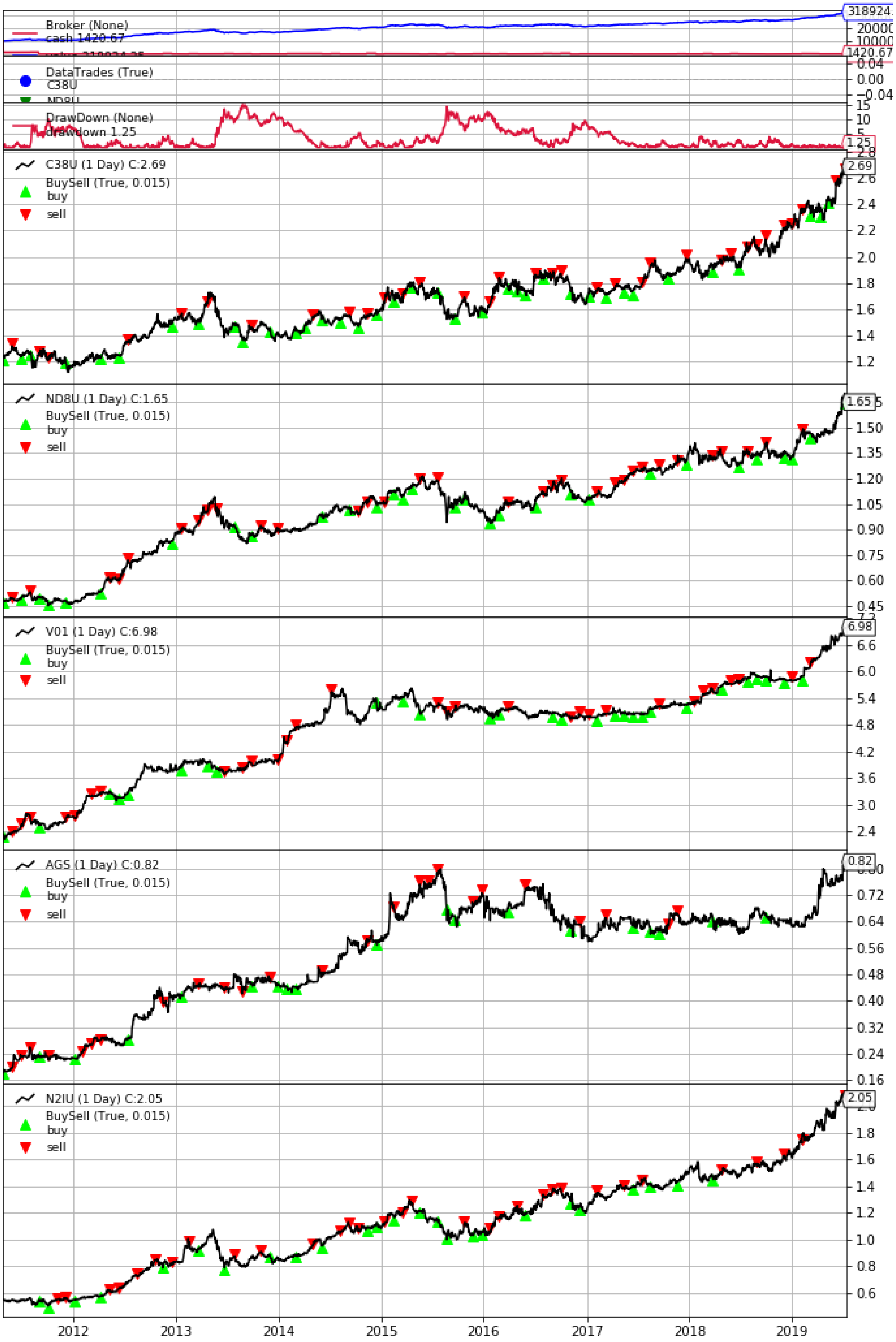
```
currentResult
```

Out[156]:

```
{'dateStart': Timestamp('2011-04-27 00:00:00'),
 'dateEnd': Timestamp('2019-07-11 00:00:00'),
 'days': 2997,
 'valueStart': 100000.0,
 'valueEnd': 318924.251603,
 'sharpe': 0.7644076593017374,
 'drawdown': 1.2546853565725675,
 'drawdownPeriod': 4,
 'moneydown': 4052.3400000000256,
 'maxDrawdown': 14.999717943917124,
 'maxDrawdownPeriod': 348,
 'maxMoneydown': 32956.984265999956,
 'averageReturns': 0.14847382650990446,
 'standardDeviation': 0.16806977918989102,
 'positiveYears': 9,
 'negativeYears': 0,
 'noChangeYears': 0,
 'bestYearReturns': 0.5600920127392031,
 'worstYearReturns': 0.013689248195242865}
```

In [157]:

```
currentBtPlot()
```



Out[157]:

```
[[<Figure size 720x1080 with 8 Axes>]]
```

Optimisation

Next, we will attempt to optimise the portfolio without introducing look ahead bias (using time-series k-folds). We will incrementally train the data over longer range of data and getting the average optimised weights for the portfolio.

In [158]:

```
o = Optimizer(p)
optimisedWeight, tests = o.kfoldTs(10)
```

In [159]:

```
optimisedWeight
```

Out[159]:

```
[0.05032977909314753,
 0.16045394863378196,
 0.4681740547205534,
 0.1324297762682293,
 0.18861244128428814]
```

In [160]:

```
tests
```

Out[160]:

```
{'sharpeRaw': [49.35365158234682,
  2.843689266140626,
  22.58767056576129,
  20.8540535465062,
  -1.0681781160922585,
  4.67773479036433,
  -5.5939905992103105,
  32.17720171943942,
  10.542912255920768,
  47.67866492721997],
 'sharpeAvg': 18.40534099383969,
 'sharpeStd': 18.642675424195932,
 'weightsRaw': [array([0.        , 0.02128336, 0.58788416, 0.16982415,
0.22100833]),
  array([0.05437949, 0.16370714, 0.40014823, 0.10450526, 0.27725987]),
  array([0.04365283, 0.20925269, 0.44670763, 0.14908186, 0.151305  ]),
  array([0.04483019, 0.1775817 , 0.49778886, 0.12319309, 0.15660615]),
  array([0.06955451, 0.18669239, 0.43215576, 0.12904634, 0.182551  ]),
  array([0.06458136, 0.16831826, 0.45313533, 0.14273817, 0.17122687]),
  array([0.07059734, 0.1558231 , 0.44239662, 0.15512288, 0.17606007]),
  array([0.04788712, 0.17076563, 0.46945662, 0.12390259, 0.18798804]),
  array([0.05071822, 0.18461817, 0.46066529, 0.11713571, 0.1868626 ]),
  array([0.05709673, 0.16649703, 0.49140204, 0.10974771, 0.1752564
8])],
 'weightsStd': 0    0.019130
 1    0.048531
 2    0.048043
 3    0.019986
 4    0.034645
 dtype: float64}
```

In [161]:

```
optimisedResult, optimisedBtPlot = p.backtest(optimisedWeight)
```

In [162]:

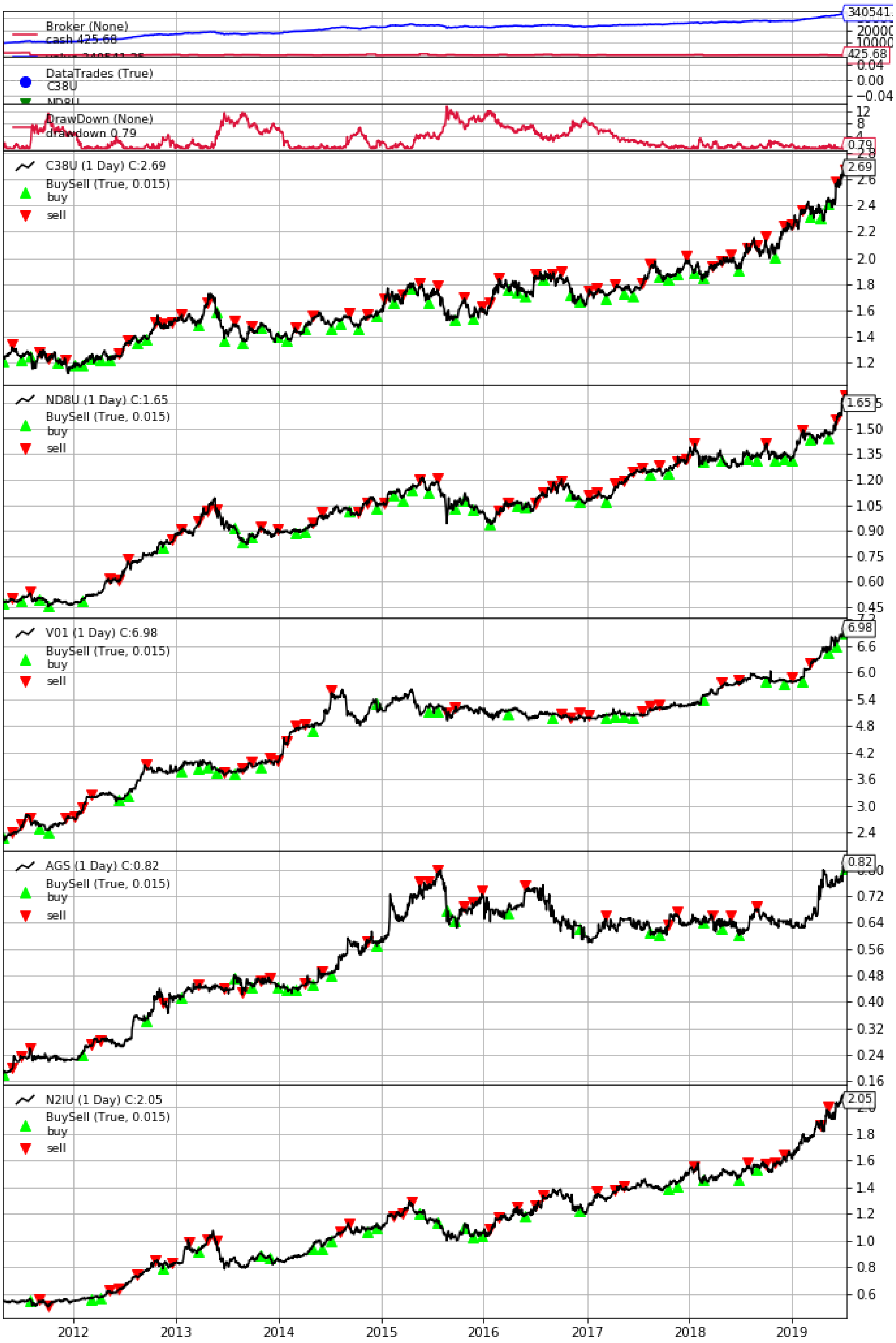
```
optimisedResult
```

Out[162]:

```
{'dateStart': Timestamp('2011-04-27 00:00:00'),
 'dateEnd': Timestamp('2019-07-11 00:00:00'),
 'days': 2997,
 'valueStart': 100000.0,
 'valueEnd': 340541.24781499995,
 'sharpe': 0.8382892911140901,
 'drawdown': 0.7863788353313458,
 'drawdownPeriod': 5,
 'moneydown': 2699.170000000042,
 'maxDrawdown': 13.56217422430473,
 'maxDrawdownPeriod': 630,
 'maxMoneydown': 34113.10347500001,
 'averageReturns': 0.15614904091944531,
 'standardDeviation': 0.1624129550056671,
 'positiveYears': 9,
 'negativeYears': 0,
 'noChangeYears': 0,
 'bestYearReturns': 0.5425357071342343,
 'worstYearReturns': 0.005555763975754857}
```

In [163]:

```
optimisedBtPlot()
```



Out[163]:

```
[[<Figure size 720x1080 with 8 Axes>]]
```

In [115]:

```
dict(zip(names, np.array(optimisedWeight)*100))
```

Out[115]:

```
{'C38U': 5.032977909314753,
 'ND8U': 16.045394863378196,
 'V01': 46.81740547205534,
 'AGS': 13.24297762682293,
 'N2IU': 18.861244128428815}
```

In [ ]:

```

```