In [1]:

```
import pandas as pd
import numpy as np
from datetime import datetime
from dateutil.relativedelta import relativedelta,TH , TU
import warnings
warnings.filterwarnings('ignore')
!pip install opstrat
import opstrat as op
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import ta
import pandas_ta as ta
from mpl_toolkits import mplot3d
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score


from breeze_connect import BreezeConnect

# Initialize SDK
breeze = BreezeConnect(api_key="f65&396497Vq0161W2d18ngV35%5755@")

# Obtain your session key from https://api.icicidirect.com/apiuser/login?api_key=YOUR_
# Incase your api-key has special characters(like +,=,!) then encode the api key befor
import urllib
print("https://api.icicidirect.com/apiuser/login?api_key="+urllib.parse.quote_plus("yo

# Generate Session
breeze.generate_session(api_secret="5l88!0708cJ04~74!X1V71a90j4~392+",
                        session_token="13527037")
```

```
Requirement already satisfied: opstrat in c:\users\91797\anaconda3\lib\site-packages
(0.1.7)
Requirement already satisfied: seaborn in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (0.11.2)
Requirement already satisfied: yfinance in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (0.2.3)
Requirement already satisfied: pandas in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (1.4.2)
Requirement already satisfied: matplotlib in c:\users\91797\anaconda3\lib\site-packag
es (from opstrat) (3.5.1)
Requirement already satisfied: numpy in c:\users\91797\anaconda3\lib\site-packages (f
rom opstrat) (1.21.5)
Requirement already satisfied: cycler>=0.10 in c:\users\91797\anaconda3\lib\site-pack
ages (from matplotlib->opstrat) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\91797\anaconda3\lib\site-
packages (from matplotlib->opstrat) (3.0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91797\anaconda3\lib\site
-packages (from matplotlib->opstrat) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91797\anaconda3\lib\site
-packages (from matplotlib->opstrat) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\91797\anaconda3\lib\site-p
ackages (from matplotlib->opstrat) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91797\anaconda3\lib\s
ite-packages (from matplotlib->opstrat) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91797\anaconda3\lib\site-pac
kages (from matplotlib->opstrat) (9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\91797\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib->opstrat) (1.16.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\91797\anaconda3\lib\site-pack
ages (from pandas->opstrat) (2022.7)
Requirement already satisfied: scipy>=1.0 in c:\users\91797\anaconda3\lib\site-packag
es (from seaborn->opstrat) (1.7.3)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\91797\anaconda3\lib\site-pa
ckages (from yfinance->opstrat) (1.4.4)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\91797\anaconda3\lib
\site-packages (from yfinance->opstrat) (4.11.1)
Requirement already satisfied: html5lib>=1.1 in c:\users\91797\anaconda3\lib\site-pac
kages (from yfinance->opstrat) (1.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\91797\anaconda3\lib\site
-packages (from yfinance->opstrat) (2.3.4)
Requirement already satisfied: lxml>=4.9.1 in c:\users\91797\anaconda3\lib\site-packa
ges (from yfinance->opstrat) (4.9.2)
Requirement already satisfied: cryptography>=3.3.2 in c:\users\91797\anaconda3\lib\si
te-packages (from yfinance->opstrat) (3.4.8)
Requirement already satisfied: requests>=2.26 in c:\users\91797\anaconda3\lib\site-pa
ckages (from yfinance->opstrat) (2.27.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\91797\anaconda3\lib\si
te-packages (from yfinance->opstrat) (0.0.11)
Requirement already satisfied: soupsieve>1.2 in c:\users\91797\anaconda3\lib\site-pac
kages (from beautifulsoup4>=4.11.1->yfinance->opstrat) (2.3.1)
Requirement already satisfied: cffi>=1.12 in c:\users\91797\anaconda3\lib\site-packag
es (from cryptography>=3.3.2->yfinance->opstrat) (1.15.0)
Requirement already satisfied: pycparser in c:\users\91797\anaconda3\lib\site-package
s (from cffi>=1.12->cryptography>=3.3.2->yfinance->opstrat) (2.21)
Requirement already satisfied: webencodings in c:\users\91797\anaconda3\lib\site-pack
ages (from html5lib>=1.1->yfinance->opstrat) (0.5.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\91797\anaconda3\lib\sit
e-packages (from requests>=2.26->yfinance->opstrat) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\91797\anaconda3
\lib\site-packages (from requests>=2.26->yfinance->opstrat) (2.0.4)
```

```
Requirement already satisfied: idna<4,>=2.5 in c:\users\91797\anaconda3\lib\site-pack
ages (from requests>=2.26->yfinance->opstrat) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\91797\anaconda3\lib
\site-packages (from requests>=2.26->yfinance->opstrat) (1.26.9)
https://api.icicidirect.com/apiuser/login?api_key=your_api_key
```

In [3]:
```
n = breeze.get_historical_data_v2(interval="1day",
                            from_date= "2023-01-01T07:00:00.000Z",
                            to_date= "2023-06-30T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m =pd.DataFrame(n['Success'])
m.index = m.datetime
m.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplace
m.rename( {'datetime' : 'date'} , inplace = True)
m['Expiry'] = 0
for i in range (len(m)) :
    m['Expiry'].iloc[i] = pd.to_datetime(m.index[i]).date() + relativedelta( weekday=
    #m['weekday'] = pd.to_datetime(m['Expiry'][i]).weekday()
m
```

Out[3]:

datetime	close	high	low	open	Expiry
2023-01-02 00:00:00	18197.45	18215.15	18086.50	18131.70	2023-01-05
2023-01-03 00:00:00	18232.55	18251.95	18149.80	18163.20	2023-01-05
2023-01-04 00:00:00	18042.95	18243.00	18020.60	18230.65	2023-01-05
2023-01-05 00:00:00	17992.15	18120.30	17892.60	18101.95	2023-01-05
2023-01-06 00:00:00	17859.45	18047.40	17795.55	18008.05	2023-01-12
...	...	...	...	...	...
2023-06-16 00:00:00	18826.00	18863.95	18712.70	18723.30	2023-06-22
2023-06-19 00:00:00	18754.20	18879.00	18719.30	18873.30	2023-06-22
2023-06-20 00:00:00	18833.70	18838.35	18661.15	18752.35	2023-06-22
2023-06-21 00:00:00	18862.10	18875.65	18794.90	18849.40	2023-06-22
2023-06-22 00:00:00	18779.35	18886.05	18760.60	18853.60	2023-06-22

117 rows × 5 columns

In [4]:
```
n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-02-15T07:00:00.000Z",
                            to_date= "2023-02-28T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m1 =pd.DataFrame(n['Success'])
m1.index = m1.datetime
m1.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m1.rename( {'datetime' : 'date'} , inplace = True)
#m1 = m1[2:]
```

```
n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-03-20T07:00:00.000Z",
                            to_date= "2023-04-10T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m2 =pd.DataFrame(n['Success'])
m2.index = m2.datetime
m2.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m2.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
```

```
mm = pd.concat([m1,m2])
mm['ema'] = mm['close'].ewm(com=5).mean()
mm['ema'].astype(float).astype(int)
mm
```

Out[4]:

datetime	close	high	low	open	ema
2023-02-15 09:05:00	17896.60	17896.60	17896.60	17896.60	17896.600000
2023-02-15 09:10:00	17896.60	17896.60	17896.60	17896.60	17896.600000
2023-02-15 09:15:00	17890.80	17900.30	17878.00	17896.60	17894.305495
2023-02-15 09:20:00	17880.70	17892.05	17865.65	17891.50	17889.925782
2023-02-15 09:25:00	17871.60	17883.20	17867.20	17881.85	17884.819308
...	...	...	...	...	...
2023-04-06 15:05:00	17601.75	17606.60	17598.55	17605.35	17597.752679
2023-04-06 15:10:00	17591.35	17602.90	17590.70	17600.80	17596.685566
2023-04-06 15:15:00	17592.55	17601.95	17591.00	17591.90	17595.996305
2023-04-06 15:20:00	17605.70	17608.35	17590.65	17591.95	17597.613588
2023-04-06 15:25:00	17602.00	17606.30	17597.30	17605.60	17598.344656

1615 rows × 5 columns

In [5]:

```
m1
```

Out[5]:

datetime	close	high	low	open
2023-02-15 09:05:00	17896.60	17896.60	17896.60	17896.60
2023-02-15 09:10:00	17896.60	17896.60	17896.60	17896.60
2023-02-15 09:15:00	17890.80	17900.30	17878.00	17896.60
2023-02-15 09:20:00	17880.70	17892.05	17865.65	17891.50
2023-02-15 09:25:00	17871.60	17883.20	17867.20	17881.85
...	...	...	...	...
2023-02-27 15:05:00	17388.55	17392.05	17367.80	17370.45
2023-02-27 15:10:00	17402.50	17405.30	17388.70	17388.70
2023-02-27 15:15:00	17396.35	17404.85	17388.55	17401.75
2023-02-27 15:20:00	17408.05	17409.65	17392.45	17396.10
2023-02-27 15:25:00	17406.35	17411.05	17399.30	17408.05

692 rows × 4 columns

In [6]:
```
m2
```

Out[6]:

datetime	close	high	low	open
2023-03-20 09:05:00	17066.60	17066.60	17066.60	17066.60
2023-03-20 09:15:00	16983.20	17057.55	16959.30	17043.90
2023-03-20 09:20:00	16968.60	16983.35	16934.65	16983.35
2023-03-20 09:25:00	16957.95	16985.40	16956.20	16969.25
2023-03-20 09:30:00	16942.85	16962.65	16932.95	16957.15
...	...	...	...	...
2023-04-06 15:05:00	17601.75	17606.60	17598.55	17605.35
2023-04-06 15:10:00	17591.35	17602.90	17590.70	17600.80
2023-04-06 15:15:00	17592.55	17601.95	17591.00	17591.90
2023-04-06 15:20:00	17605.70	17608.35	17590.65	17591.95
2023-04-06 15:25:00	17602.00	17606.30	17597.30	17605.60

923 rows × 4 columns

In [12]:
```
s =  "2023-03-23T07:00:00.000Z"
ss =  "2023-03-24T07:00:00.000Z"
ni = breeze.get_historical_data_v2(interval="5minute",
                                   from_date= s ,
                                   to_date = ss  ,
```

```
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
nifty_intraday =pd.DataFrame(ni['Success'])
nifty_intraday.index = nifty_intraday.datetime
nifty_intraday.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume'
nifty_intraday.rename( {'datetime' : 'date'} , inplace = True)
nifty_intraday['10 ema'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].ema
#nifty_intraday['supb'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].supb
#nifty_intraday['sups'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].sups

nifty_intraday =nifty_intraday[2:]

#print(nifty_intraday)









strike_multiple = 50
nifty_intraday['10 ema'] = round(nifty_intraday['10 ema'])
nifty_intraday['ATM STRIKE'] = strike_multiple*( round(nifty_intraday['10 ema'] / stri
nifty_intraday['Switch'] = 0
nifty_intraday['Switch'].iloc[0] = 1
for i in range (len(nifty_intraday)) :
    if nifty_intraday['ATM STRIKE'].iloc[i] != nifty_intraday['ATM STRIKE'].iloc[i-1]
        nifty_intraday['Switch'].iloc[i] = 1
    elif nifty_intraday['ATM STRIKE'].iloc[i] != nifty_intraday['ATM STRIKE'].iloc[i-1
        nifty_intraday['Switch'].iloc[i] = 0




#nifty_intraday['ATM STRIKE'].value_counts()
#nifty_intraday['Switch'].value_counts(
nifty_intraday['ATM STRIKE'] = nifty_intraday['ATM STRIKE'].astype(float).astype(int)
nifty_intraday['10 ema'] = nifty_intraday['10 ema'].astype(float).astype(int)
nifty_intraday

e = m[m.index == (s[0:10] + ' ' + '00:00:00') ]
Expiry =str(e.Expiry[0])
p =nifty_intraday[nifty_intraday['Switch'] == 1]
#nifty_intraday
#for i in range (len(p)) :
#    p.iloc[i]['datetime'][11:19]
```

In [13]:

```
import math
buy_price = 0
exit_price = 0
trade = 0
cols = ['Entry Time' , 'Exit Time' , 'Entry Price' , 'Exit Price' , 'Points' , 'profit
new = 1
d = []
for i in range (len(nifty_intraday)) :
    if new == 1  :
        w = breeze.get_historical_data_v2(interval="5minute",
```

```
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="call",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i] )
        w =pd.DataFrame(w['Success'])
        w.index = pd.to_datetime(w.index)
        w =w[ ['close' , 'datetime'] ]
        w.reset_index(drop=True, inplace=True)
        for j in range (len(w)) :
            w['datetime'].iloc[j] = w['datetime'].iloc[j][11:16]


        ww = breeze.get_historical_data_v2(interval="5minute",
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="put",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i] )
        ww =pd.DataFrame(ww['Success'])
        ww.index = pd.to_datetime(ww.index)
        ww =ww[ ['close' , 'datetime'] ]
        ww.reset_index(drop=True, inplace=True)
        for j1 in range (len(ww)) :
            ww['datetime'].iloc[j1] = ww['datetime'].iloc[j1][11:16]

        ww['close'] = ww['close'] + w['close']
        buy_price = ww['close'].iloc[i]
        buy_time = ww['datetime'].iloc[i]
        new = 0


    if trade == 1  :
        w = breeze.get_historical_data_v2(interval="5minute",
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="call",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i] )
        w =pd.DataFrame(w['Success'])
        w.index = pd.to_datetime(w.index)
        w =w[ ['close' , 'datetime'] ]
        w.reset_index(drop=True, inplace=True)
        for j2 in range (len(w)) :
            w['datetime'].iloc[j2] = w['datetime'].iloc[j2][11:16]


        ww = breeze.get_historical_data_v2(interval="5minute",
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
```

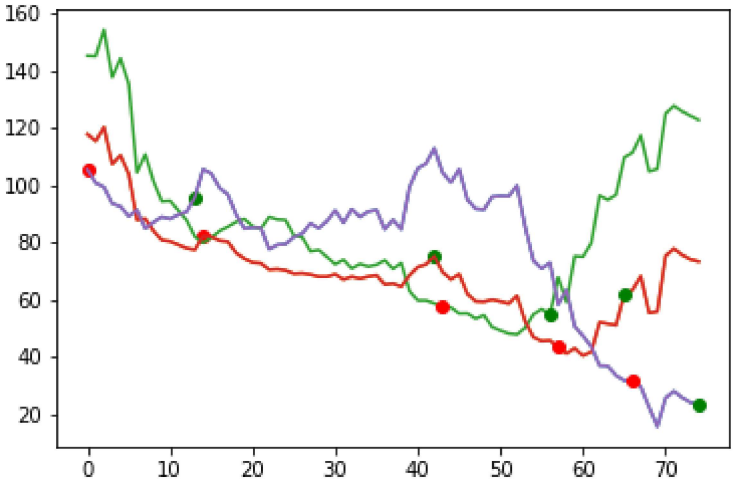
```
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="put",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i] )
        ww =pd.DataFrame(ww['Success'])
        ww.index = pd.to_datetime(ww.index)
        ww =ww[ ['close' , 'datetime'] ]
        ww.reset_index(drop=True, inplace=True)
        for j3 in range (len(ww)) :
            ww['datetime'].iloc[j3] = ww['datetime'].iloc[j3][11:16]

        ww['close'] = ww['close'] + w['close']
        buy_price = ww['close'].iloc[i]
        buy_time = ww['datetime'].iloc[i]
        trade = 0
        #print(buy_price)
        #print(buy_time)
        #print(i)



    if (nifty_intraday['Switch'].iloc[i] == 1) and (i != 0)  :
        exit_price = ww['close'].iloc[i]
        exit_time = ww['datetime'].iloc[i]
        points = buy_price - exit_price
        plt.plot(ww['close'])
        plt.plot( ww[ ww['datetime'] == buy_time ]['close'] , 'ro')
        plt.plot( ww[ ww['datetime'] == exit_time ]['close'] , 'go')
        notional_value = nifty_intraday['ATM STRIKE'].iloc[i] * 50
        span = notional_value * 0.1
        exposure = notional_value * 0.06
        margin_ =  span + exposure
        margin = margin_ + margin_
        q = math.floor( 500000/(margin*0.3) )
        #qq = q/ 50
        #qqq = math.floor(qq)*50
        profit = q*points*50
        #print( q )
        d.append([buy_time , exit_time , buy_price , exit_price , points , profit])
        straddle = pd.DataFrame(d , columns = cols)
        print(straddle)
        #trade = 0
        trade = 1

    if  i == 74  :
        exit_price = ww['close'].iloc[i]
        exit_time = ww['datetime'].iloc[i]
        points = buy_price - exit_price
        notional_value = nifty_intraday['ATM STRIKE'].iloc[i] * 50
        span = notional_value * 0.1
        exposure = notional_value * 0.06
        margin_ =  span + exposure
        margin = margin_ + margin_
        q1 = math.floor( 500000/(margin*0.3) )
        #qq1 = q1/ 50
        #qqq1 = math.floor(qq1)*50
        profit = q1*points*50
        ww['close'].plot()
        plt.plot( ww[ ww['datetime'] == buy_time ]['close'] , 'ro')
```

```
        plt.plot( ww[ ww['datetime'] == exit_time ]['close'] , 'go')
        d.append([buy_time , exit_time , buy_price , exit_price , points , profit])
        straddle = pd.DataFrame(d , columns = cols)
        print(straddle)
        #trade = 0
        new = 1



#print(ww)
#plt.plot(ww['close'])
summ = 0
summm = 0
for i in range(len(straddle)) :
    summ = summ + straddle['Points'].iloc[i]
    summm = summm + straddle['profit'].iloc[i]
d.append([0 , 0 , 0 , 0 , summ , summm])
straddle = pd.DataFrame(d , columns = cols)
straddle
```

```
  Entry Time Exit Time  Entry Price  Exit Price  Points  profit
0      09:15     10:20       105.35        95.7    9.65  2895.0
  Entry Time Exit Time  Entry Price  Exit Price  Points  profit
0      09:15     10:20       105.35       95.70    9.65  2895.0
1      10:25     12:45        82.20       75.15    7.05  2115.0
  Entry Time Exit Time  Entry Price  Exit Price  Points  profit
0      09:15     10:20       105.35       95.70    9.65  2895.0
1      10:25     12:45        82.20       75.15    7.05  2115.0
2      12:50     13:55        57.85       54.85    3.00   900.0
  Entry Time Exit Time  Entry Price  Exit Price  Points  profit
0      09:15     10:20       105.35       95.70    9.65  2895.0
1      10:25     12:45        82.20       75.15    7.05  2115.0
2      12:50     13:55        57.85       54.85    3.00   900.0
3      14:00     14:40        43.25       62.00  -18.75 -5625.0
  Entry Time Exit Time  Entry Price  Exit Price  Points  profit
0      09:15     10:20       105.35       95.70    9.65  2895.0
1      10:25     12:45        82.20       75.15    7.05  2115.0
2      12:50     13:55        57.85       54.85    3.00   900.0
3      14:00     14:40        43.25       62.00  -18.75 -5625.0
4      14:45     15:25        31.75       23.25    8.50  2550.0
```

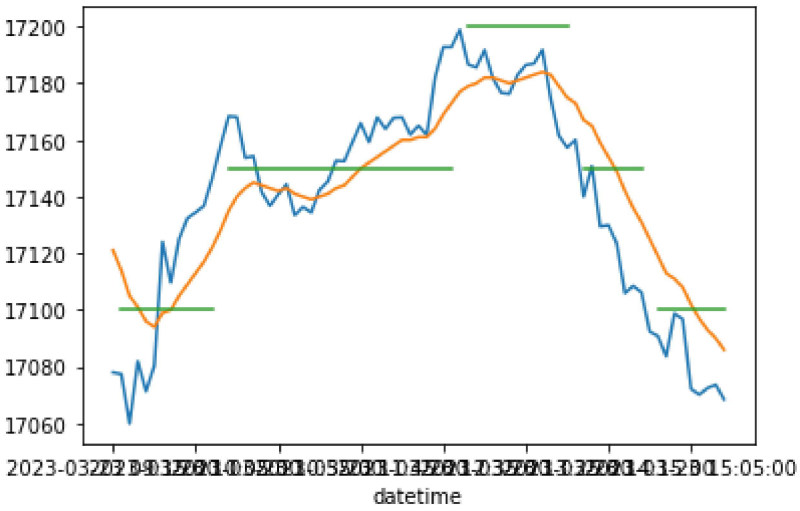
Out[13]:

	Entry Time	Exit Time	Entry Price	Exit Price	Points	profit
0	09:15	10:20	105.35	95.70	9.65	2895.0
1	10:25	12:45	82.20	75.15	7.05	2115.0
2	12:50	13:55	57.85	54.85	3.00	900.0
3	14:00	14:40	43.25	62.00	-18.75	-5625.0
4	14:45	15:25	31.75	23.25	8.50	2550.0
5	0	0	0.00	0.00	9.45	2835.0

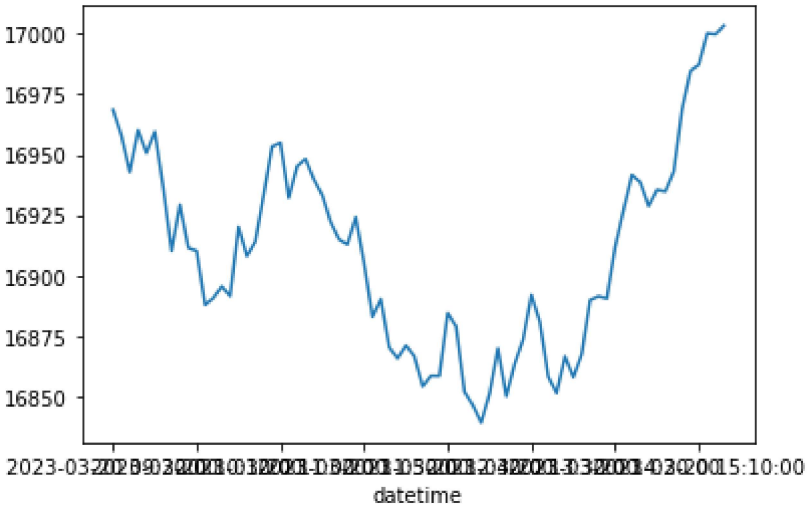


In [14]:
```
nifty_intraday.close.plot()
nifty_intraday['10 ema'].plot()
nifty_intraday['ATM STRIKE'] = np.where(nifty_intraday['Switch'] == 1 , np.nan , nifty
nifty_intraday['ATM STRIKE'].plot()
```

Out[14]: <AxesSubplot:xlabel='datetime'>



In [52]:
```
try :
    nifty_intraday.close.plot()
except  AttributeError or IntCastingNaNError:
    print(9)
```



In [ ]:
```
ik =nifty_intraday[nifty_intraday['Switch'] == 1]
ik
```

In [ ]:
```
com = []
for i in range (len(ik)) :
    if i != ( len(ik)-1 ) :
        com.append( nifty_intraday[ik.index[i] : ik.index[i +1] ] )
    if i == ( len(ik)-1 ) :
        com.append( nifty_intraday[ik.index[i] : ik.index[i ][0 : 11] + '15:25:00' ] )

com[0]
```

In [ ]:
```
nifty_intraday.close.plot()
nifty_intraday['10 ema'].plot()
```

In [ ]:
```

```

In [ ]:
```
s[0:10] + ' 09:15:00'
```

In [ ]:
```
#CONVERTING 5 MINUTE DATA TO 15 MINUTE



aa = 0

b15 = []
for i in range (len(m1)):
    aa = aa + 1
    if aa == 1 :
        b15.append(m1[i:i+1])
        pass

    if aa == 3:
        aa = 0
        pass
```

```
m1 =pd.concat(b15)
```

In [ ]:
```
import math
math.floor(253)
```

In [ ]:
```
k = 274
kk = k / 25
```

In [ ]:
```
math.floor(kk)*25
```

In [ ]:
```
254/25
```

In [ ]:
```
n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-05-01T07:00:00.000Z",
                            to_date= "2023-05-25T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m3 =pd.DataFrame(n['Success'])
m3.index = m3.datetime
m3.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m3.rename( {'datetime' : 'date'} , inplace = True)
m3
```

In [ ]:
```
w = breeze.get_historical_data_v2(interval="5minute",
                            from_date=   "2023-04-20T07:00:00.000Z"  ,
                            to_date= "2023-04-21T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= '2023-04-20 00:00:00',
                            right="call",
                            strike_price = 17650)
t =pd.DataFrame(w['Success'])
t.index = pd.to_datetime(t.index)
t =t[ ['close' , 'datetime'] ]
t.reset_index(drop=True, inplace=True)

for i in range (len(t)) :
    t['datetime'].iloc[i] = t['datetime'].iloc[i][11:16]
t
```

In [ ]:
```
Expiry
```

In [ ]:
```

```

In [29]:
```
n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-01-01T07:00:00.000Z",
                            to_date= "2023-01-20T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m1 =pd.DataFrame(n['Success'])
m1.index = m1.datetime
m1.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
```

```
m1.rename( {'datetime' : 'date'} , inplace = True)
#m1 = m1[2:]
print(m1)




n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-01-20T07:00:00.000Z",
                            to_date= "2023-02-10T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m2 =pd.DataFrame(n['Success'])
m2.index = m2.datetime
m2.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m2.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
print(m2)

n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-02-10T07:00:00.000Z",
                            to_date= "2023-03-19T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m3 =pd.DataFrame(n['Success'])
m3.index = m3.datetime
m3.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m3.rename( {'datetime' : 'date'} , inplace = True)
#m1 = m1[2:]
print(m3)

n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-03-19T07:00:00.000Z",
                            to_date= "2023-04-07T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m4 =pd.DataFrame(n['Success'])
m4.index = m4.datetime
m4.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m4.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
print(m4)

n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-04-08T07:00:00.000Z",
                            to_date= "2023-04-29T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m5 =pd.DataFrame(n['Success'])
m5.index = m5.datetime
m5.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m5.rename( {'datetime' : 'date'} , inplace = True)
#m1 = m1[2:]
print(m5)
```

```
n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-05-01T07:00:00.000Z",
                            to_date= "2023-05-27T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m6 =pd.DataFrame(n['Success'])
m6.index = m6.datetime
m6.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m6.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
print(m6)

n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-05-28T07:00:00.000Z",
                            to_date= "2023-06-15T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m7 =pd.DataFrame(n['Success'])
m7.index = m7.datetime
m7.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m7.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
print(m7)

n = breeze.get_historical_data_v2(interval="5minute",
                            from_date= "2023-06-16T07:00:00.000Z",
                            to_date= "2023-06-20T07:00:00.000Z",
                            stock_code="NIFTY",
                            exchange_code="NSE",
                            product_type="cash")
m8 =pd.DataFrame(n['Success'])
m8.index = m8.datetime
m8.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' }  , inplac
m8.rename( {'datetime' : 'date'} , inplace = True)
#m2 = m2[2:]
print(m8)




mm = pd.concat([m1,m2,m3,m4,m5,m6,m7,m8])
mm['ema'] = mm['close'].ewm(com=5).mean()
mm['ema'].astype(float).astype(int)
mm['Date'] = 0
for i in range (len(mm)) :
    mm['Date'].iloc[i] = mm.index[i][0:10]
df2 = mm['Date'].drop_duplicates()
np.array(df2)
```

```
                        close      high       low      open
datetime
2023-01-02 14:30:00  18158.75  18158.75  18143.80  18147.50
2023-01-02 14:35:00  18155.90  18159.15  18145.85  18158.95
2023-01-02 14:40:00  18156.70  18164.05  18150.75  18155.60
2023-01-02 14:45:00  18190.20  18196.65  18153.80  18156.05
2023-01-02 14:50:00  18195.75  18206.25  18188.50  18189.65
...                       ...       ...       ...       ...
2023-01-19 15:05:00  18120.05  18120.05  18114.10  18118.50
2023-01-19 15:10:00  18103.05  18121.60  18101.30  18119.95
2023-01-19 15:15:00  18103.75  18106.95  18100.00  18102.90
2023-01-19 15:20:00  18099.15  18108.35  18098.40  18103.40
2023-01-19 15:25:00  18089.05  18099.80  18084.95  18099.45

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-01-20 15:05:00  18030.95  18037.70  18020.05  18035.30
2023-01-20 15:10:00  18029.35  18032.90  18024.75  18031.15
2023-01-20 15:15:00  18025.15  18032.75  18024.85  18029.45
2023-01-20 15:20:00  18025.15  18028.70  18016.20  18025.65
2023-01-20 15:25:00  18027.30  18027.35  18019.50  18025.25
...                       ...       ...       ...       ...
2023-02-09 15:05:00  17893.35  17893.65  17878.45  17881.00
2023-02-09 15:10:00  17893.55  17899.20  17891.50  17895.10
2023-02-09 15:15:00  17893.30  17899.75  17891.95  17893.10
2023-02-09 15:20:00  17900.40  17901.10  17890.90  17893.35
2023-02-09 15:25:00  17897.10  17904.80  17895.10  17900.85

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-02-28 09:15:00  17412.80  17414.75  17368.80  17383.25
2023-02-28 09:20:00  17403.70  17427.50  17380.15  17413.10
2023-02-28 09:25:00  17407.50  17413.30  17398.90  17403.20
2023-02-28 09:30:00  17432.50  17438.30  17400.60  17406.95
2023-02-28 09:35:00  17434.05  17440.20  17422.30  17432.75
...                       ...       ...       ...       ...
2023-03-17 15:05:00  17119.60  17120.45  17077.75  17088.50
2023-03-17 15:10:00  17113.50  17124.95  17101.50  17122.05
2023-03-17 15:15:00  17091.50  17114.15  17089.55  17112.95
2023-03-17 15:20:00  17100.00  17100.00  17082.90  17091.55
2023-03-17 15:25:00  17123.70  17125.90  17091.55  17099.15

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-03-20 09:05:00  17066.60  17066.60  17066.60  17066.60
2023-03-20 09:15:00  16983.20  17057.55  16959.30  17043.90
2023-03-20 09:20:00  16968.60  16983.35  16934.65  16983.35
2023-03-20 09:25:00  16957.95  16985.40  16956.20  16969.25
2023-03-20 09:30:00  16942.85  16962.65  16932.95  16957.15
...                       ...       ...       ...       ...
2023-04-06 15:05:00  17601.75  17606.60  17598.55  17605.35
2023-04-06 15:10:00  17591.35  17602.90  17590.70  17600.80
2023-04-06 15:15:00  17592.55  17601.95  17591.00  17591.90
2023-04-06 15:20:00  17605.70  17608.35  17590.65  17591.95
2023-04-06 15:25:00  17602.00  17606.30  17597.30  17605.60

[923 rows x 4 columns]
```

```
                        close      high       low      open
datetime
2023-04-11 09:10:00  17704.80  17704.80  17704.80  17704.80
2023-04-11 09:15:00  17680.20  17704.80  17669.00  17704.80
2023-04-11 09:20:00  17704.35  17705.40  17683.85  17685.60
2023-04-11 09:25:00  17697.45  17707.15  17694.40  17703.80
2023-04-11 09:30:00  17707.40  17711.30  17697.55  17698.15
...                       ...       ...       ...       ...
2023-04-28 15:05:00  18060.30  18060.30  18045.45  18047.80
2023-04-28 15:10:00  18064.05  18070.75  18058.75  18059.55
2023-04-28 15:15:00  18083.90  18084.10  18060.50  18063.70
2023-04-28 15:20:00  18084.05  18088.80  18083.45  18083.85
2023-04-28 15:25:00  18049.55  18083.85  18046.35  18083.80

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-05-10 09:05:00  18313.60  18313.60  18313.60  18313.60
2023-05-10 09:10:00  18313.60  18313.60  18313.60  18313.60
2023-05-10 09:15:00  18313.35  18321.50  18292.70  18307.55
2023-05-10 09:20:00  18309.65  18322.30  18305.50  18315.65
2023-05-10 09:25:00  18305.60  18314.75  18301.10  18310.05
...                       ...       ...       ...       ...
2023-05-26 15:05:00  18503.90  18507.05  18499.90  18504.05
2023-05-26 15:10:00  18505.00  18507.20  18499.60  18503.70
2023-05-26 15:15:00  18498.00  18505.70  18496.40  18505.10
2023-05-26 15:20:00  18495.70  18499.45  18491.85  18498.75
2023-05-26 15:25:00  18491.90  18503.10  18491.90  18496.00

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-05-29 09:10:00  18619.15  18619.15  18619.15  18619.15
2023-05-29 09:15:00  18622.60  18640.80  18596.75  18619.15
2023-05-29 09:20:00  18615.95  18635.25  18611.05  18622.45
2023-05-29 09:25:00  18629.00  18640.35  18614.90  18616.40
2023-05-29 09:30:00  18613.30  18632.80  18611.65  18629.15
...                       ...       ...       ...       ...
2023-06-14 15:05:00  18755.20  18758.05  18753.15  18756.45
2023-06-14 15:10:00  18758.40  18759.65  18754.70  18755.20
2023-06-14 15:15:00  18757.00  18760.70  18754.30  18758.55
2023-06-14 15:20:00  18750.65  18757.20  18749.25  18756.75
2023-06-14 15:25:00  18749.65  18752.30  18745.10  18750.95

[1000 rows x 4 columns]
                        close      high       low      open
datetime
2023-06-16 09:05:00  18723.30  18723.30  18723.30  18723.30
2023-06-16 09:10:00  18723.30  18723.30  18723.30  18723.30
2023-06-16 09:15:00  18744.75  18750.75  18712.70  18723.30
2023-06-16 09:20:00  18744.95  18747.60  18734.95  18746.10
2023-06-16 09:25:00  18752.05  18754.70  18744.55  18745.50
...                       ...       ...       ...       ...
2023-06-19 15:05:00  18755.00  18755.00  18746.90  18750.95
2023-06-19 15:10:00  18760.50  18763.80  18751.75  18754.45
2023-06-19 15:15:00  18755.90  18763.65  18753.40  18761.10
2023-06-19 15:20:00  18754.20  18758.70  18747.15  18755.90
2023-06-19 15:25:00  18754.20  18755.85  18746.85  18755.10

[154 rows x 4 columns]
```

Out[29]:
```
array(['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05',
       '2023-01-06', '2023-01-09', '2023-01-10', '2023-01-11',
       '2023-01-12', '2023-01-13', '2023-01-16', '2023-01-17',
       '2023-01-18', '2023-01-19', '2023-01-20', '2023-01-23',
       '2023-01-24', '2023-01-25', '2023-01-27', '2023-01-30',
       '2023-01-31', '2023-02-01', '2023-02-02', '2023-02-03',
       '2023-02-06', '2023-02-07', '2023-02-08', '2023-02-09',
       '2023-02-28', '2023-03-01', '2023-03-02', '2023-03-03',
       '2023-03-06', '2023-03-08', '2023-03-09', '2023-03-10',
       '2023-03-13', '2023-03-14', '2023-03-15', '2023-03-16',
       '2023-03-17', '2023-03-20', '2023-03-21', '2023-03-22',
       '2023-03-23', '2023-03-24', '2023-03-27', '2023-03-28',
       '2023-03-29', '2023-03-31', '2023-04-03', '2023-04-05',
       '2023-04-06', '2023-04-11', '2023-04-12', '2023-04-13',
       '2023-04-17', '2023-04-18', '2023-04-19', '2023-04-20',
       '2023-04-21', '2023-04-24', '2023-04-25', '2023-04-26',
       '2023-04-27', '2023-04-28', '2023-05-10', '2023-05-11',
       '2023-05-12', '2023-05-15', '2023-05-16', '2023-05-17',
       '2023-05-18', '2023-05-19', '2023-05-22', '2023-05-23',
       '2023-05-24', '2023-05-25', '2023-05-26', '2023-05-29',
       '2023-05-30', '2023-05-31', '2023-06-01', '2023-06-02',
       '2023-06-05', '2023-06-06', '2023-06-07', '2023-06-08',
       '2023-06-09', '2023-06-12', '2023-06-13', '2023-06-14',
       '2023-06-16', '2023-06-19'], dtype=object)
```

In [30]:
```
cum_profit = []
for iq in range (len(df2)) :
    try :
        s = df2[iq][0:10] + 'T07:00:00.000Z'
        ss = df2[iq+1][0:10] + 'T07:00:00.000Z'
        ni = breeze.get_historical_data_v2(interval="5minute",
                           from_date= s ,
                           to_date = ss  ,
                           stock_code="NIFTY",
                           exchange_code="NSE",
                           product_type="cash")
        nifty_intraday =pd.DataFrame(ni['Success'])
        nifty_intraday.index = nifty_intraday.datetime
        nifty_intraday.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' ,
        nifty_intraday.rename( {'datetime' : 'date'} , inplace = True)
        nifty_intraday['10 ema'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].e
        #nifty_intraday['supb'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].su
        #nifty_intraday['sups'] = mm[s[0:10] + ' 09:15:00' : s[0:10] + ' 15:25:00'].su

        nifty_intraday =nifty_intraday[2:]

        #print(nifty_intraday)

        strike_multiple = 50
        nifty_intraday['10 ema'] = round(nifty_intraday['10 ema'])
        nifty_intraday['ATM STRIKE'] = strike_multiple*( round(nifty_intraday['10 ema'
        nifty_intraday['Switch'] = 0
        nifty_intraday['Switch'].iloc[0] = 1
        for i in range (len(nifty_intraday)) :
            if nifty_intraday['ATM STRIKE'].iloc[i] != nifty_intraday['ATM STRIKE'].il
                nifty_intraday['Switch'].iloc[i] = 1
            elif nifty_intraday['ATM STRIKE'].iloc[i] != nifty_intraday['ATM STRIKE'],
                nifty_intraday['Switch'].iloc[i] = 0
```

```
#nifty_intraday['ATM STRIKE'].value_counts()
#nifty_intraday['Switch'].value_counts(
nifty_intraday.dropna(inplace = True)
nifty_intraday['ATM STRIKE'] = nifty_intraday['ATM STRIKE'].astype(float).asty
nifty_intraday['10 ema'] = nifty_intraday['10 ema'].astype(float).astype(int)
nifty_intraday

e = m[m.index == (s[0:10] + ' ' + '00:00:00') ]
Expiry =str(e.Expiry[0])
p =nifty_intraday[nifty_intraday['Switch'] == 1]
#nifty_intraday
#for i in range (len(p)) :
#    p.iloc[i]['datetime'][11:19]



import math
buy_price = 0
exit_price = 0
trade = 0
cols = ['Entry Time' , 'Exit Time' , 'Entry Price' , 'Exit Price' , 'Points' ,
new = 1
d = []
for i in range (len(nifty_intraday)) :
    if new == 1  :
        w = breeze.get_historical_data_v2(interval="5minute",
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="call",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i
        w =pd.DataFrame(w['Success'])
        w.index = pd.to_datetime(w.index)
        w =w[ ['close' , 'datetime'] ]
        w.reset_index(drop=True, inplace=True)
        for j in range (len(w)) :
            w['datetime'].iloc[j] = w['datetime'].iloc[j][11:16]


        ww = breeze.get_historical_data_v2(interval="5minute",
                            from_date= s ,
                            to_date= ss,
                            stock_code="NIFTY",
                            exchange_code="NFO",
                            product_type="options",
                            expiry_date= Expiry,
                            right="put",
                            strike_price = nifty_intraday['ATM STRIKE'].iloc[i
        ww =pd.DataFrame(ww['Success'])
        ww.index = pd.to_datetime(ww.index)
        ww =ww[ ['close' , 'datetime'] ]
        ww.reset_index(drop=True, inplace=True)
        for j1 in range (len(ww)) :
            ww['datetime'].iloc[j1] = ww['datetime'].iloc[j1][11:16]

        ww['close'] = ww['close'] + w['close']
```

```
            buy_price = ww['close'].iloc[i]
            buy_time = ww['datetime'].iloc[i]
            new = 0


        if trade == 1  :
            w = breeze.get_historical_data_v2(interval="5minute",
                                from_date= s ,
                                to_date= ss,
                                stock_code="NIFTY",
                                exchange_code="NFO",
                                product_type="options",
                                expiry_date= Expiry,
                                right="call",
                                strike_price = nifty_intraday['ATM STRIKE'].iloc[i
            w =pd.DataFrame(w['Success'])
            w.index = pd.to_datetime(w.index)
            w =w[ ['close' , 'datetime'] ]
            w.reset_index(drop=True, inplace=True)
            for j2 in range (len(w)) :
                w['datetime'].iloc[j2] = w['datetime'].iloc[j2][11:16]


            ww = breeze.get_historical_data_v2(interval="5minute",
                                from_date= s ,
                                to_date= ss,
                                stock_code="NIFTY",
                                exchange_code="NFO",
                                product_type="options",
                                expiry_date= Expiry,
                                right="put",
                                strike_price = nifty_intraday['ATM STRIKE'].iloc[i
            ww =pd.DataFrame(ww['Success'])
            ww.index = pd.to_datetime(ww.index)
            ww =ww[ ['close' , 'datetime'] ]
            ww.reset_index(drop=True, inplace=True)
            for j3 in range (len(ww)) :
                ww['datetime'].iloc[j3] = ww['datetime'].iloc[j3][11:16]

            ww['close'] = ww['close'] + w['close']
            buy_price = ww['close'].iloc[i]
            buy_time = ww['datetime'].iloc[i]
            trade = 0
            #print(buy_price)
            #print(buy_time)
            #print(i)



        if (nifty_intraday['Switch'].iloc[i] == 1) and (i != 0)  :
            exit_price = ww['close'].iloc[i]
            exit_time = ww['datetime'].iloc[i]
            points = buy_price - exit_price
            #plt.plot(ww['close'])
            #plt.plot( ww[ ww['datetime'] == buy_time ]['close'] , 'ro')
            #plt.plot( ww[ ww['datetime'] == exit_time ]['close'] , 'go')
            notional_value = nifty_intraday['ATM STRIKE'].iloc[i] * 50
            span = notional_value * 0.1
            exposure = notional_value * 0.06
            margin_ =  span + exposure
```

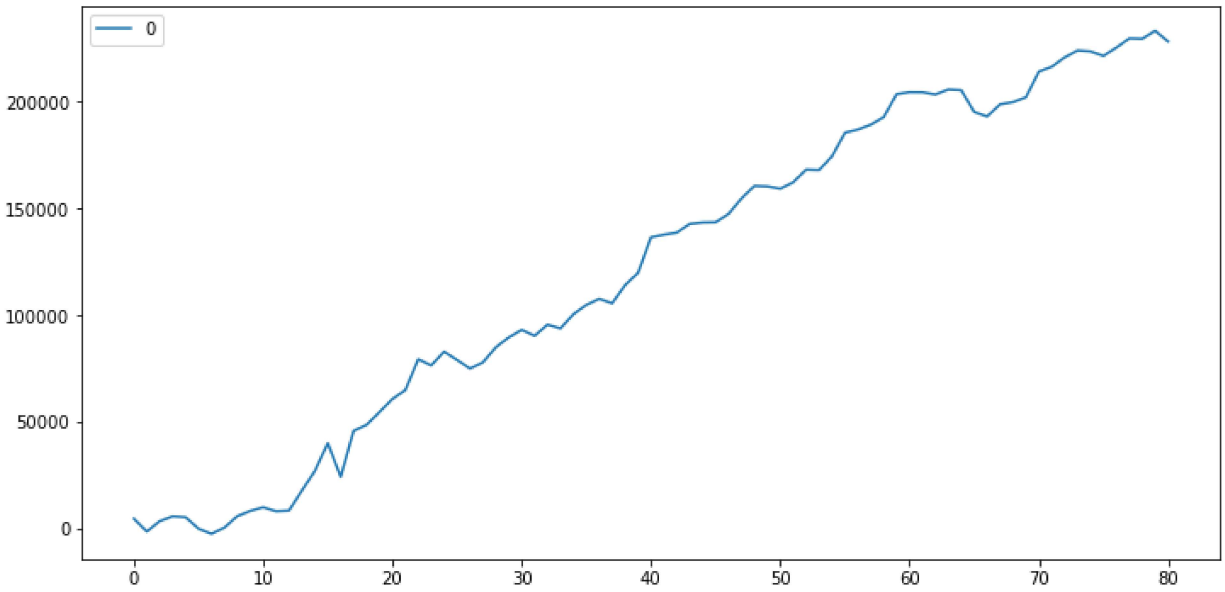
```
                margin = margin_ + margin_
                q = math.floor( 500000/(margin*0.3) )
                #qq = q/ 50
                #qqq = math.floor(qq)*50
                profit = q*points*50
                #print( q )
                d.append([buy_time , exit_time , buy_price , exit_price , points , pro
                straddle = pd.DataFrame(d , columns = cols)
                #print(straddle)
                #trade = 0
                trade = 1

            if  i == 74  :
                exit_price = ww['close'].iloc[i]
                exit_time = ww['datetime'].iloc[i]
                points = buy_price - exit_price
                notional_value = nifty_intraday['ATM STRIKE'].iloc[i] * 50
                span = notional_value * 0.1
                exposure = notional_value * 0.06
                margin_ =  span + exposure
                margin = margin_ + margin_
                q1 = math.floor( 500000/(margin*0.3) )
                #qq1 = q1/ 50
                #qqq1 = math.floor(qq1)*50
                profit = q1*points*50
                #ww['close'].plot()
                #plt.plot( ww[ ww['datetime'] == buy_time ]['close'] , 'ro')
                #plt.plot( ww[ ww['datetime'] == exit_time ]['close'] , 'go')
                d.append([buy_time , exit_time , buy_price , exit_price , points , pro
                straddle = pd.DataFrame(d , columns = cols)
                #print(straddle)
                #trade = 0
                new = 1



        #print(ww)
        #plt.plot(ww['close'])
        summ = 0
        summm = 0
        for i in range(len(straddle)) :
            summ = summ + straddle['Points'].iloc[i]
            summm = summm + straddle['profit'].iloc[i]
        d.append([0 , 0 , 0 , 0 , summ , summm])
        straddle = pd.DataFrame(d , columns = cols)
        cum_profit.append(summm)
        print(summm)
    except  (AttributeError)  :
        cum_profit.append(0)
        print(0)
    except (KeyError) :
        print(0)
    except(IndexError) :
        print(0)
    except(ValueError):
        print(0)
```

0
4437.499999999993
-6062.500000000001
4862.500000000005
2275.0000000000055
-400.00000000001955
-5399.999999999998
-2399.9999999999986
2875.0000000000073
5399.999999999998
2437.5
1725.0000000000086
-1862.499999999997
312.5000000000035
0
0
0
0
9437.50000000003
9162.499999999967
13037.499999999982
-15812.499999999993
21612.49999999999
2800.000000000011
6012.50000000001
6137.499999999995
4187.500000000007
14462.499999999998
-2922.5000000000127
0
6477.500000000004
-3900.000000000005
-4025.0000000000127
2874.9999999999995
7124.999999999998
4625.0
0
0
3524.999999999991
-2760.0000000000055
5235.000000000015
-1799.999999999991
6690.000000000004
4364.9999999999945
2834.999999999999
0
0
0
0
-2085.0000000000055
8547.500000000002
5712.499999999998
16762.5
1199.999999999999
962.4999999999985
4099.999999999998
637.4999999999749
99.99999999998727
3912.499999999998
7262.500000000003

```
5875.0
-224.9999999999907
-1074.9999999999955
2999.9999999999964
5950.000000000002
-212.4999999999912
6450.000000000007
11075.0
1499.999999999993
2312.5
3512.4999999999886
10787.500000000004
912.5000000000023
-37.49999999999426
-1025.0000000000055
2337.5000000000055
-312.49999999999636
-10250.000000000007
-2087.4999999999986
5687.500000000007
1037.5000000000123
2250.0000000000036
12075.000000000002
2287.5000000000086
4400.0000000000055
3225.0000000000045
-437.5
-2087.499999999998
3899.999999999998
4287.500000000002
-162.500000000005
3750.0
-5037.4999999999945
0
```

In [42]:
```
pd.DataFrame(cum_profit).cumsum().plot(figsize = (12,6))
```

Out[42]:
```
<AxesSubplot:>
```



In [33]:
```
(228357/500000)*100
```

Out[33]: 45.6714

In [41]:
```
t =pd.DataFrame(cum_profit)
t[t[0] == 0]
```

Out[41]:

	0

In [ ]:
```
#cum_profit = [10 ,3 , 12 , 12 , 6 ]
```

In [ ]:
```
tpp = []
tpp.append(cum_profit)
tpp
```

In [ ]: