In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime
from dateutil.relativedelta import relativedelta,TH
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
```

```
Requirement already satisfied: opstrat in c:\users\91797\anaconda3\lib\site-packages
(0.1.7)
Requirement already satisfied: pandas in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (1.4.2)
Requirement already satisfied: yfinance in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (0.2.3)
Requirement already satisfied: numpy in c:\users\91797\anaconda3\lib\site-packages (f
rom opstrat) (1.21.5)
Requirement already satisfied: matplotlib in c:\users\91797\anaconda3\lib\site-packag
es (from opstrat) (3.5.1)
Requirement already satisfied: seaborn in c:\users\91797\anaconda3\lib\site-packages
(from opstrat) (0.11.2)
Requirement already satisfied: cycler>=0.10 in c:\users\91797\anaconda3\lib\site-pack
ages (from matplotlib->opstrat) (0.11.0)
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: fonttools>=4.22.0 in c:\users\91797\anaconda3\lib\site
-packages (from matplotlib->opstrat) (4.25.0)
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: 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: 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: appdirs>=1.4.4 in c:\users\91797\anaconda3\lib\site-pa
ckages (from yfinance->opstrat) (1.4.4)
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: requests>=2.26 in c:\users\91797\anaconda3\lib\site-pa
ckages (from yfinance->opstrat) (2.27.1)
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: beautifulsoup4>=4.11.1 in c:\users\91797\anaconda3\lib
\site-packages (from yfinance->opstrat) (4.11.1)
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: urllib3<1.27,>=1.21.1 in c:\users\91797\anaconda3\lib
\site-packages (from requests>=2.26->yfinance->opstrat) (1.26.9)
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: 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: certifi>=2017.4.17 in c:\users\91797\anaconda3\lib\sit
e-packages (from requests>=2.26->yfinance->opstrat) (2021.10.8)
```

In [2]:
```
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="7608666")
```

```
https://api.icicidirect.com/apiuser/login?api_key=your_api_key
```

In [3]:
```
n = breeze.get_historical_data_v2(interval="1day",
                            from_date= "2021-01-01T07:00:00.000Z",
                            to_date= "2022-01-01T07:00:00.000Z",
                            stock_code="SBIN",
                            exchange_code="NSE",
                            product_type="cash")
sbi =pd.DataFrame(n['Success'])
#sbi.index = sbi.datetime
#sbi.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume'}  , inplc
#sbi.rename( {'datetime' : 'date'} , inplace = True)
#sbi
```

In [4]:
```
n = breeze.get_historical_data(interval="1day",
                            from_date= "2018-01-01T07:00:00.000Z",
                            to_date= "2022-01-01T07:00:00.000Z",
                            stock_code="TCS",
                            exchange_code="NSE",
                            product_type="cash")
t =pd.DataFrame(n['Success'])
t.index = t.datetime
t.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' ,'product_ty
t.rename( {'datetime' : 'date'} , inplace = True)
#t['rsi14'] = ta.momentum.RSIIndicator(t['close'].astype(float).astype(int) , window =
#t['rsi28'] = ta.momentum.RSIIndicator(t['close'].astype(float).astype(int) , window =
t['rsi14'] = t.astype(float).astype(int).ta.rsi(length = 14)
t.dropna(inplace = True)
t['rsi28'] = t.astype(float).astype(int).ta.rsi(length = 28)
t.dropna(inplace = True)
t['close'] = t['close'].astype(float).astype(int)
t['high'] = t['high'].astype(float).astype(int)
t['low'] = t['low'].astype(float).astype(int)
t['open'] = t['open'].astype(float).astype(int)
t
```

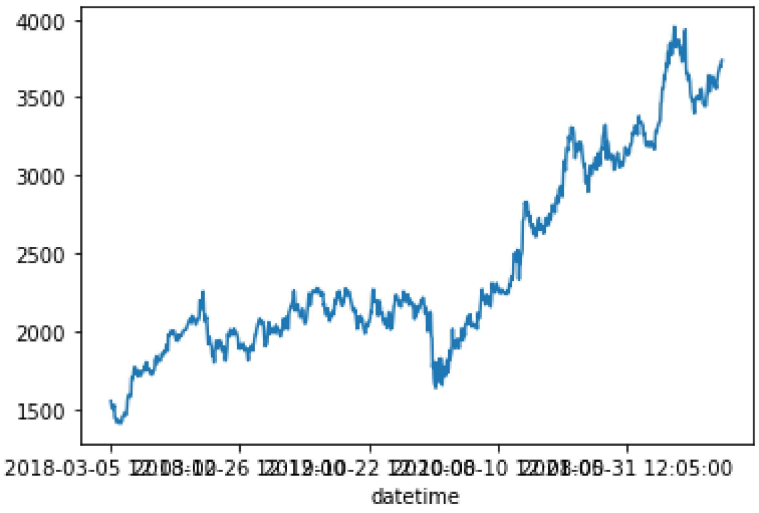
Out[4]:

datetime	open	high	low	close	rsi14	rsi28
2018-03-05 12:03:00	1524	1556	1516	1552	63.367715	57.869075
2018-03-06 12:03:00	1561	1563	1515	1521	55.045020	52.550691
2018-03-07 12:03:00	1535	1544	1509	1513	53.106559	51.289207
2018-03-08 12:03:00	1527	1527	1493	1501	50.248083	49.442946
2018-03-09 12:03:00	1502	1531	1488	1517	53.817423	51.840046
...	...	...	...	...	...	...
2021-12-27 12:12:00	3671	3700	3653	3696	63.355740	56.823792
2021-12-28 12:12:00	3710	3725	3693	3706	64.303143	57.340200
2021-12-29 12:12:00	3692	3719	3685	3694	62.224148	56.499257
2021-12-30 12:12:00	3681	3740	3680	3733	66.064282	58.548194
2021-12-31 12:12:00	3742	3760	3731	3738	66.533988	58.806161

949 rows × 6 columns

In [5]:
```
t.close.plot()
```

Out[5]:
```
<AxesSubplot:xlabel='datetime'>
```



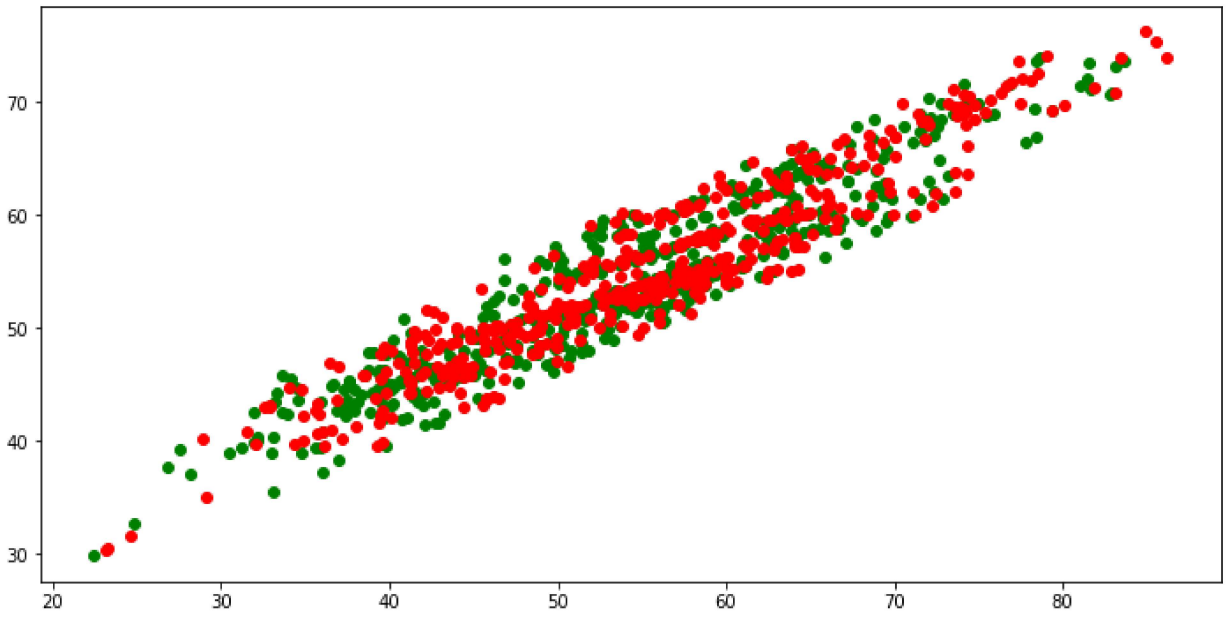
In [6]:
```
X= t[['rsi14', 'rsi28']]
Y= np.where(t['close'].shift(-1)> t['close']   ,1,-1)
split_percentage = 0.7
split = int(split_percentage*len(t))


X_train = X[:split]
Y_train = Y[:split]

X_test = X[split:]
Y_test = Y[split:]

knn = KNeighborsClassifier(n_neighbors=252)
```

```
# fit the model
knn.fit(X_train, Y_train)

# Accuracy Score
accuracy_train = accuracy_score(Y_train, knn.predict(X_train))
accuracy_test = accuracy_score(Y_test, knn.predict(X_test))

print ('Train_data Accuracy: %.2f' %accuracy_train)
print ('Test_data Accuracy: %.2f' %accuracy_test)
```

```
Train_data Accuracy: 0.53
Test_data Accuracy: 0.56
```

In [7]:

```
N =t[['rsi14', 'rsi28']]
N['label'] = Y
N['up rsi14'] = np.where(N['label'] == 1 , N.rsi14 , np.nan )
N['up rsi28'] = np.where(N['label'] == 1 , N.rsi28 , np.nan )
N['down rsi14'] = np.where(N['label'] == -1 , N.rsi14 , np.nan )
N['down rsi28'] = np.where(N['label'] == -1 , N.rsi28 , np.nan )
h =pd.DataFrame( N['up rsi14']   )
h['up rsi28'] = N['up rsi28']
h.dropna(inplace =  True)
h
n = pd.DataFrame( N['down rsi14']   )
n['down rsi28'] = N['down rsi28']
n.dropna(inplace =  True)
n
plt.figure(figsize=(12,6))
plt.scatter(h['up rsi14'] , h['up rsi28'] , c ="green")
plt.scatter(n['down rsi14'] , n['down rsi28'] , c ="red")
plt.show()
```



In [10]:

```
n = breeze.get_historical_data(interval="1day",
                               from_date= "2021-01-01T07:00:00.000Z",
                               to_date= "2023-02-28T07:00:00.000Z",
                               stock_code="TCS",
                               exchange_code="NSE",
                               product_type="cash")
m =pd.DataFrame(n['Success'])
```

```
m.index = m.datetime
m.drop( columns = {'datetime' , 'exchange_code' , 'stock_code' , 'volume' ,'product_ty
m.rename( {'datetime' : 'date'} , inplace = True)
m['close'] = t['close'].astype(float).astype(int)
m['high'] = t['high'].astype(float).astype(int)
m['low'] = t['low'].astype(float).astype(int)
m['open'] = t['open'].astype(float).astype(int)
#m['rsi14'] = ta.momentum.RSIIndicator(m['close'].astype(float).astype(int) , window =
#m['rsi28'] = ta.momentum.RSIIndicator(m['close'].astype(float).astype(int) , window =
m['rsi14'] = m.ta.rsi(length = 14)

m['rsi28'] = m.ta.rsi(length = 28)
m.dropna(inplace = True)

#m.dropna(inplace = True)

m
```

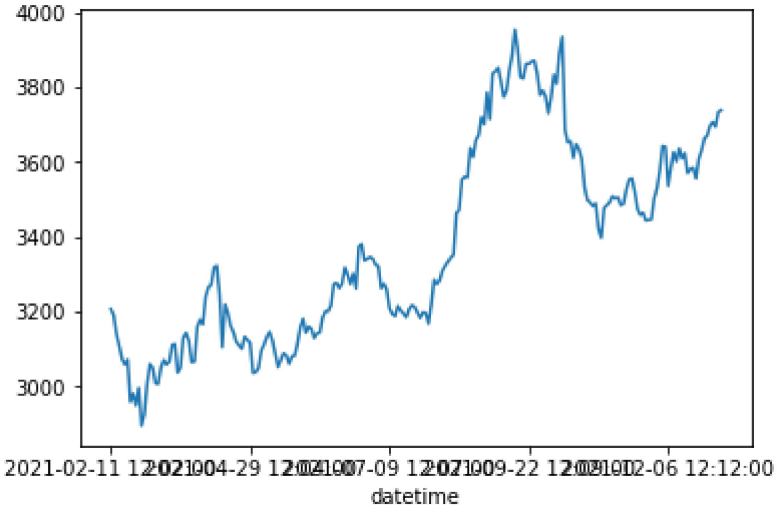
Out[10]:

datetime	open	high	low	close	rsi14	rsi28
2021-02-11 12:02:00	3215.0	3219.0	3185.0	3206.0	55.067286	58.184801
2021-02-12 12:02:00	3184.0	3245.0	3182.0	3190.0	53.014711	56.805429
2021-02-15 12:02:00	3209.0	3209.0	3131.0	3139.0	47.000941	52.677418
2021-02-16 12:02:00	3150.0	3167.0	3100.0	3108.0	43.752121	50.370089
2021-02-17 12:02:00	3105.0	3132.0	3045.0	3073.0	40.360071	47.912902
...	...	...	...	...	...	...
2021-12-27 12:12:00	3671.0	3700.0	3653.0	3696.0	63.355740	56.823079
2021-12-28 12:12:00	3710.0	3725.0	3693.0	3706.0	64.303143	57.339567
2021-12-29 12:12:00	3692.0	3719.0	3685.0	3694.0	62.224148	56.498518
2021-12-30 12:12:00	3681.0	3740.0	3680.0	3733.0	66.064282	58.547758
2021-12-31 12:12:00	3742.0	3760.0	3731.0	3738.0	66.533988	58.805761

220 rows × 6 columns

In [11]:
```
rsi= m[['rsi14', 'rsi28']]
p = knn.predict(rsi)
m.close.plot()
```

Out[11]:
```
<AxesSubplot:xlabel='datetime'>
```



In [12]:
```
for i in range (len(m)):
    print(p[i])
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
-1
-1
1
1
-1
-1
-1
-1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1
1
1
1
1
-1
-1
1
1
1
1
1
1
-1
-1
-1
1
-1
-1
-1
-1
-1
-1
1
-1
-1
-1
-1
1
1
-1
1
-1
-1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
-1
1
-1
1
```

```
1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
1
1
1
1
-1
1
1
1
1
1
-1
1
-1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
-1
1
-1
1
1
1
1
1
1
1
1
1
-1
-1
1
1
1
-1
-1
```

In [13]:
```
l = [-1]
for i in range (len(m)):
    if p[i] == p[i-1] :
        l.append(5)
    if p[i] != p[i-1] :
        l.append(p[i])
b = np.array(l)
b = b.astype(float)
b[b == 5] = np.nan
b
buy = np.where(b == (-1) , np.nan , b)
for i in range (len(m)) :
    if buy[i] == 1 :
        buy[i] = m['close'][i]


sell = np.where(b==1 , np.nan , b)
for i in range (len(m)) :
    if sell[i] == -1 :
```

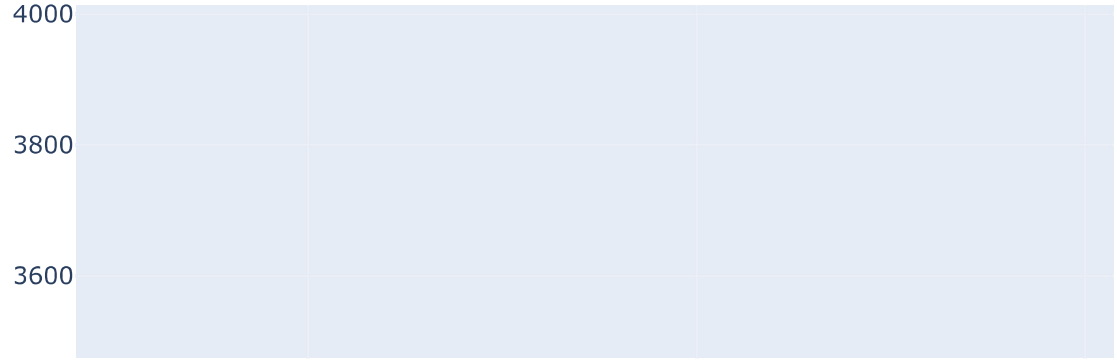
```
        sell[i] = m['close'][i]
sell
```

Out[13]:

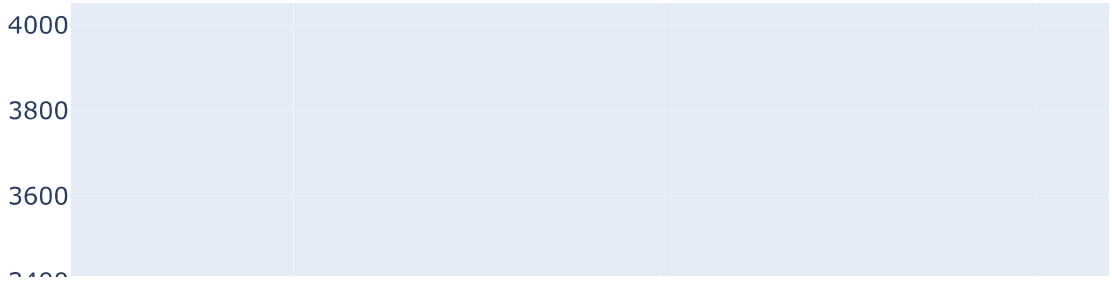
```
array([3206.,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan, 3165.,   nan,   nan,
         nan, 3317.,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan, 3180.,   nan,   nan,
         nan,   nan,   nan,   nan,   nan, 3200.,   nan,   nan,   nan,
       3276.,   nan,   nan,   nan,   nan,   nan,   nan, 3261.,   nan,
         nan,   nan,   nan,   nan, 3341.,   nan, 3321.,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
       3273.,   nan, 3309.,   nan,   nan, 3344.,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan, 3779.,   nan,   nan,   nan,   nan,   nan, 3810.,   nan,
       3935.,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,   nan,
         nan,   nan,   nan,   nan,   nan, 3601.,   nan, 3609.,   nan,
         nan,   nan,   nan,   nan,   nan,   nan,   nan, 3670.,   nan,
         nan,   nan,   nan, 3738.,   nan])
```

In [14]:

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
fig =px.line( x= m.index , y= m.close  )
fig.add_scatter (x = m.index , y= buy , mode = "markers" ,marker = dict(size = 5, colo
fig.add_scatter (x = m.index , y= sell , mode = "markers" ,marker = dict(size = 5, co
```



In [16]:
```
fig = go.Figure()
fig.add_trace(go.Candlestick(x = m.index ,
                             open = m.open,
                             high = m.high ,
                             low = m.low ,
                             close = m.close , name = 'market data'))
fig.add_scatter (x = m.index , y= buy , mode = "markers" ,marker = dict(size = 5, colo
fig.add_scatter (x = m.index , y= sell , mode = "markers" ,marker = dict(size = 5, co
fig.show()
```



In [ ]: