1.turtle-agent

September 29, 2021

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

[2]:

```
df = pd.read_csv('../dataset/GOOG-year.csv')
df.head()
```

[2]:

```
         Date        Open        High         Low       Close   Adj Close  \
0  2016-11-02  778.200012  781.650024  763.450012  768.700012  768.700012
1  2016-11-03  767.250000  769.950012  759.030029  762.130005  762.130005
2  2016-11-04  750.659973  770.359985  750.560974  762.020020  762.020020
3  2016-11-07  774.500000  785.190002  772.549988  782.520020  782.520020
4  2016-11-08  783.400024  795.632996  780.190002  790.510010  790.510010

    Volume
0  1872400
1  1943200
2  2134800
3  1585100
4  1350800
```

[3]:

```
count = int(np.ceil(len(df) * 0.1))
signals = pd.DataFrame(index=df.index)
signals['signal'] = 0.0
signals['trend'] = df['Close']
signals['RollingMax'] = (signals.trend.shift(1).rolling(count).max())
signals['RollingMin'] = (signals.trend.shift(1).rolling(count).min())
signals.loc[signals['RollingMax'] < signals.trend, 'signal'] = -1
signals.loc[signals['RollingMin'] > signals.trend, 'signal'] = 1
signals
```

[3]:

```
     signal       trend  RollingMax  RollingMin
0       0.0  768.700012         NaN         NaN
1       0.0  762.130005         NaN         NaN
2       0.0  762.020020         NaN         NaN
```

```
3       0.0  782.520020         NaN         NaN
4       0.0  790.510010         NaN         NaN
5       0.0  785.309998         NaN         NaN
6       0.0  762.559998         NaN         NaN
7       0.0  754.020020         NaN         NaN
8       0.0  736.080017         NaN         NaN
9       0.0  758.489990         NaN         NaN
10      0.0  764.479980         NaN         NaN
11      0.0  771.229980         NaN         NaN
12      0.0  760.539978         NaN         NaN
13      0.0  769.200012         NaN         NaN
14      0.0  768.270020         NaN         NaN
15      0.0  760.989990         NaN         NaN
16      0.0  761.679993         NaN         NaN
17      0.0  768.239990         NaN         NaN
18      0.0  770.840027         NaN         NaN
19      0.0  758.039978         NaN         NaN
20      0.0  747.919983         NaN         NaN
21      0.0  750.500000         NaN         NaN
22      0.0  762.520020         NaN         NaN
23      0.0  759.109985         NaN         NaN
24      0.0  771.190002         NaN         NaN
25      0.0  776.419983         NaN         NaN
26      0.0  789.289978  790.510010  736.080017
27      0.0  789.270020  790.510010  736.080017
28     -1.0  796.099976  790.510010  736.080017
29     -1.0  797.070007  796.099976  736.080017
..      …         …          …          …
222     0.0  932.450012  939.330017  906.659973
223     0.0  928.530029  939.330017  906.659973
224     0.0  920.969971  939.330017  906.659973
225     0.0  924.859985  939.330017  906.659973
226    -1.0  944.489990  939.330017  906.659973
227    -1.0  949.500000  944.489990  913.809998
228    -1.0  959.109985  949.500000  913.809998
229     0.0  953.270020  959.109985  913.809998
230     0.0  957.789978  959.109985  913.809998
231     0.0  951.679993  959.109985  913.809998
232    -1.0  969.960022  959.109985  915.000000
233    -1.0  978.890015  969.960022  915.000000
234     0.0  977.000000  978.890015  915.000000
235     0.0  972.599976  978.890015  915.000000
236    -1.0  989.250000  978.890015  915.000000
237     0.0  987.830017  989.250000  915.000000
238    -1.0  989.679993  989.250000  915.000000
239    -1.0  992.000000  989.679993  915.000000
240    -1.0  992.179993  992.000000  915.000000
```

```
241    -1.0   992.809998   992.179993  915.000000
242     0.0   984.450012   992.809998  915.000000
243     0.0   988.200012   992.809998  915.000000
244     0.0   968.450012   992.809998  915.000000
245     0.0   970.539978   992.809998  915.000000
246     0.0   973.330017   992.809998  920.969971
247     0.0   972.559998   992.809998  920.969971
248    -1.0  1019.270020   992.809998  920.969971
249     0.0  1017.109985  1019.270020  920.969971
250     0.0  1016.640015  1019.270020  920.969971
251    -1.0  1025.500000  1019.270020  924.859985

[252 rows x 4 columns]
```

[4]:

```python
def buy_stock(
    real_movement,
    signal,
    initial_money = 10000,
    max_buy = 1,
    max_sell = 1,
):
    """
    real_movement = actual movement in the real world
    delay = how much interval you want to delay to change our decision from buy␣
 ↪to sell, vice versa
    initial_state = 1 is buy, 0 is sell
    initial_money = 1000, ignore what kind of currency
    max_buy = max quantity for share to buy
    max_sell = max quantity for share to sell
    """
    starting_money = initial_money
    states_sell = []
    states_buy = []
    current_inventory = 0

    def buy(i, initial_money, current_inventory):
        shares = initial_money // real_movement[i]
        if shares < 1:
            print(
                'day %d: total balances %f, not enough money to buy a unit␣
 ↪price %f'
                % (i, initial_money, real_movement[i])
            )
        else:
            if shares > max_buy:
                buy_units = max_buy
            else:
                buy_units = shares
            initial_money -= buy_units * real_movement[i]
            current_inventory += buy_units
            print(
                'day %d: buy %d units at price %f, total balance %f'
                % (i, buy_units, buy_units * real_movement[i], initial_money)
            )
            states_buy.append(0)
        return initial_money, current_inventory

    for i in range(real_movement.shape[0] - int(0.025 * len(df))):
        state = signal[i]
        if state == 1:
            initial_money, current_inventory = buy(
                i, initial_money, current_inventory
            )
            states_buy.append(i)
        elif state == -1:
            if current_inventory == 0:
                    print('day %d: cannot sell anything, inventory 0' % (i))
            else:
                if current_inventory > max_sell:
                    sell_units = max_sell
                else:
                    sell_units = current_inventory
                current_inventory -= sell_units
                total_sell = sell_units * real_movement[i]
                initial_money += total_sell
                try:
                    invest = (
                        (real_movement[i] - real_movement[states_buy[-1]])
                        / real_movement[states_buy[-1]]
                    ) * 100
                except:
                    invest = 0
                print(
                    'day %d, sell %d units at price %f, investment %f %%, total balance %f,'
                    % (i, sell_units, total_sell, invest, initial_money)
                )
            states_sell.append(i)

    invest = ((initial_money - starting_money) / starting_money) * 100
    total_gains = initial_money - starting_money
    return states_buy, states_sell, total_gains, invest
```

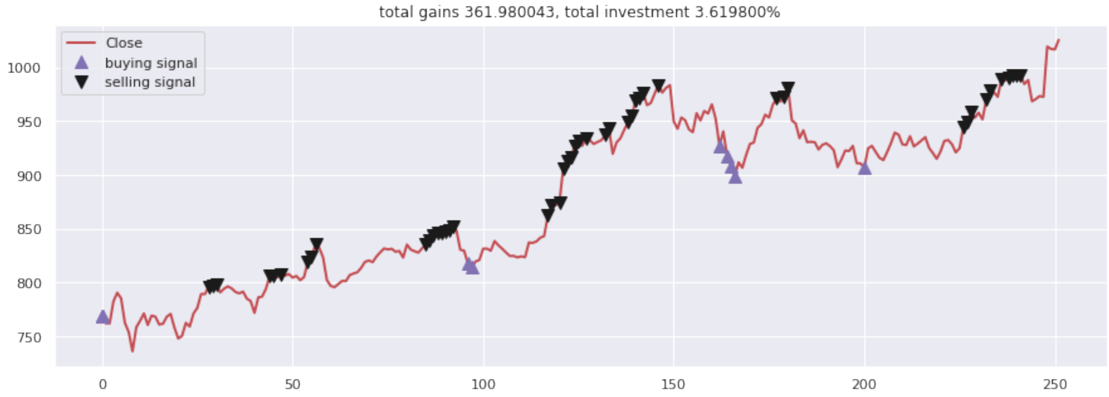
[5]:

```
states_buy, states_sell, total_gains, invest = buy_stock(df.Close,␣
 ↪signals['signal'])
```

```
day 28: cannot sell anything, inventory 0
day 29: cannot sell anything, inventory 0
day 30: cannot sell anything, inventory 0
day 44: cannot sell anything, inventory 0
day 45: cannot sell anything, inventory 0
day 47: cannot sell anything, inventory 0
day 54: cannot sell anything, inventory 0
day 55: cannot sell anything, inventory 0
day 56: cannot sell anything, inventory 0
day 85: cannot sell anything, inventory 0
day 86: cannot sell anything, inventory 0
day 87: cannot sell anything, inventory 0
day 88: cannot sell anything, inventory 0
day 89: cannot sell anything, inventory 0
day 90: cannot sell anything, inventory 0
day 91: cannot sell anything, inventory 0
day 92: cannot sell anything, inventory 0
day 96: buy 1 units at price 817.580017, total balance 9182.419983
day 97: buy 1 units at price 814.429993, total balance 8367.989990
day 117, sell 1 units at price 862.760010, investment 5.934214 %, total balance
9230.750000,
day 118, sell 1 units at price 872.299988, investment 7.105582 %, total balance
10103.049988,
day 120: cannot sell anything, inventory 0
day 121: cannot sell anything, inventory 0
day 122: cannot sell anything, inventory 0
day 123: cannot sell anything, inventory 0
day 124: cannot sell anything, inventory 0
day 125: cannot sell anything, inventory 0
day 127: cannot sell anything, inventory 0
day 132: cannot sell anything, inventory 0
day 133: cannot sell anything, inventory 0
day 138: cannot sell anything, inventory 0
day 139: cannot sell anything, inventory 0
day 140: cannot sell anything, inventory 0
day 141: cannot sell anything, inventory 0
day 142: cannot sell anything, inventory 0
day 146: cannot sell anything, inventory 0
day 162: buy 1 units at price 927.330017, total balance 9175.719971
day 164: buy 1 units at price 917.789978, total balance 8257.929993
day 165: buy 1 units at price 908.729980, total balance 7349.200013
day 166: buy 1 units at price 898.700012, total balance 6450.500001
day 177, sell 1 units at price 970.890015, investment 8.032714 %, total balance
7421.390016,
day 179, sell 1 units at price 972.919983, investment 8.258592 %, total balance
8394.309999,
day 180, sell 1 units at price 980.340027, investment 9.084234 %, total balance
9374.650026,
day 200: buy 1 units at price 906.659973, total balance 8467.990053
day 226, sell 1 units at price 944.489990, investment 4.172459 %, total balance
9412.480043,
day 227, sell 1 units at price 949.500000, investment 4.725038 %, total balance
10361.980043,
day 228: cannot sell anything, inventory 0
day 232: cannot sell anything, inventory 0
day 233: cannot sell anything, inventory 0
day 236: cannot sell anything, inventory 0
day 238: cannot sell anything, inventory 0
day 239: cannot sell anything, inventory 0
day 240: cannot sell anything, inventory 0
day 241: cannot sell anything, inventory 0
```

[6]:

```
close = df['Close']
fig = plt.figure(figsize = (15,5))
plt.plot(close, color='r', lw=2.)
plt.plot(close, '^', markersize=10, color='m', label = 'buying signal', markevery = states_buy)
plt.plot(close, 'v', markersize=10, color='k', label = 'selling signal', markevery = states_sell)
plt.title('total gains %f, total investment %f%%'%(total_gains, invest))
plt.legend()
plt.show()
```



[ ]: