2.moving-average-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	Volume
0	2016-11-02	778.200012	781.650024	763.450012	768.700012	768.700012	1872400
1	2016-11-03	767.250000	769.950012	759.030029	762.130005	762.130005	1943200
2	2016-11-04	750.659973	770.359985	750.560974	762.020020	762.020020	2134800
3	2016-11-07	774.500000	785.190002	772.549988	782.520020	782.520020	1585100
4	2016-11-08	783.400024	795.632996	780.190002	790.510010	790.510010	1350800

[3]:
```
short_window = int(0.025 * len(df))
long_window = int(0.05 * len(df))

signals = pd.DataFrame(index=df.index)
signals['signal'] = 0.0

signals['short_ma'] = df['Close'].rolling(window=short_window, min_periods=1,␣
↪center=False).mean()
signals['long_ma'] = df['Close'].rolling(window=long_window, min_periods=1,␣
↪center=False).mean()

signals['signal'][short_window:] = np.where(signals['short_ma'][short_window:]
                                            > signals['long_ma'][short_window:
↪], 1.0, 0.0)
```

```
signals['positions'] = signals['signal'].diff()

signals
```

[3]:

	signal	short_ma	long_ma	positions
0	0.0	768.700012	768.700012	NaN
1	0.0	765.415008	765.415008	0.0
2	0.0	764.283346	764.283346	0.0
3	0.0	768.842514	768.842514	0.0
4	0.0	773.176013	773.176013	0.0
5	0.0	775.198344	775.198344	0.0
6	1.0	774.175008	773.392866	1.0
7	1.0	772.823344	770.971260	0.0
8	1.0	768.500010	767.094456	0.0
9	0.0	764.495005	766.234009	-1.0
10	0.0	760.156667	766.074552	0.0
11	0.0	757.809998	766.504171	0.0
12	0.0	757.473327	765.824168	0.0
13	0.0	760.003326	766.413335	0.0
14	0.0	765.368327	766.934169	0.0
15	1.0	765.784993	765.139999	1.0
16	1.0	765.318329	762.737498	0.0
17	1.0	764.819997	761.314997	0.0
18	1.0	766.536672	762.005000	0.0
19	1.0	764.676666	762.339996	0.0
20	0.0	761.284993	763.326660	-1.0
21	0.0	759.536662	762.660828	0.0
22	0.0	759.676666	762.497498	0.0
23	0.0	758.154999	761.487498	0.0
24	0.0	758.213328	762.375000	0.0
25	0.0	761.276662	762.976664	0.0
26	1.0	768.171661	764.728327	1.0
27	1.0	774.633331	767.084997	0.0
28	1.0	780.229991	769.953328	0.0
29	1.0	786.556661	772.355830	0.0
..	…	…	…	…
222	0.0	924.373332	927.728338	0.0
223	0.0	924.943339	927.788340	0.0
224	0.0	925.056671	926.540003	0.0
225	1.0	926.700002	926.403335	1.0
226	1.0	930.480001	927.687500	0.0
227	1.0	933.466664	929.139999	0.0
228	1.0	937.909993	931.141662	0.0
229	1.0	942.033325	933.488332	0.0
230	1.0	948.169993	936.613332	0.0
231	1.0	952.639994	939.669998	0.0
232	1.0	956.885000	943.682500	0.0

```
233     1.0   961.783335  947.625000        0.0
234     1.0   964.765005  951.337499        0.0
235     1.0   967.986664  955.009995        0.0
236     1.0   973.230001  960.699997        0.0
237     1.0   979.255005  965.947500        0.0
238     1.0   982.541667  969.713333        0.0
239     1.0   984.726664  973.255000        0.0
240     1.0   987.256663  976.010834        0.0
241     1.0   990.625000  979.305832        0.0
242     1.0   989.825002  981.527502        0.0
243     1.0   989.886668  984.570836        0.0
244     1.0   986.348338  984.445002        0.0
245     0.0   982.771667  983.749166       -1.0
246     0.0   979.630005  983.443334        0.0
247     0.0   976.255005  983.440002        0.0
248     0.0   982.058339  985.941671        0.0
249     0.0   986.876668  988.381668        0.0
250     1.0   994.908335  990.628337        1.0
251     1.0  1004.068339  993.420003        0.0

[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(
```

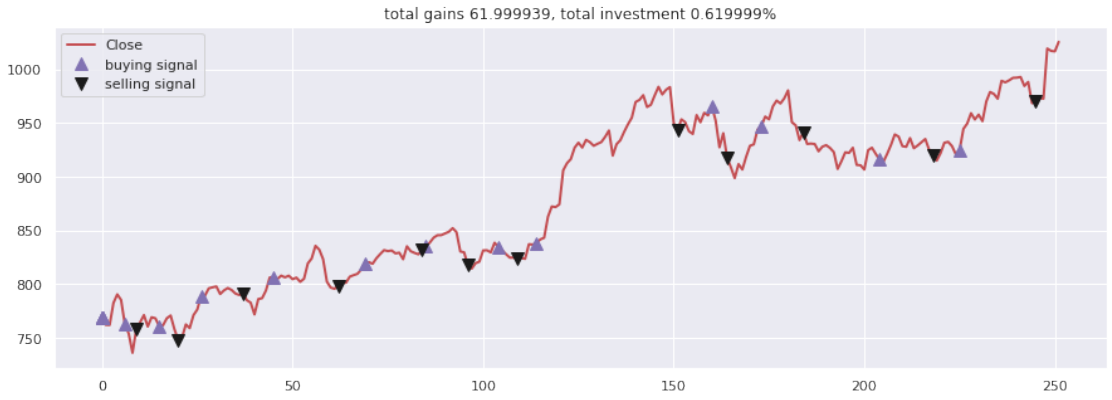
```
                'day %d: total balances %f, not enough money to buy a unit␣
↪price %f'
                % (i, initial_money, real_movement[i])
            )
        else:
            if shares > max_buy:
                buy_units = max_buy
            else:
                buy_units = shares
            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['positions'])
```

```
day 6: buy 1 units at price 762.559998, total balance 9237.440002
day 9, sell 1 units at price 758.489990, investment -0.533730 %, total balance
9995.929992,
day 15: buy 1 units at price 760.989990, total balance 9234.940002
day 20, sell 1 units at price 747.919983, investment -1.717501 %, total balance
9982.859985,
day 26: buy 1 units at price 789.289978, total balance 9193.570007
day 37, sell 1 units at price 791.549988, investment 0.286335 %, total balance
9985.119995,
day 45: buy 1 units at price 806.650024, total balance 9178.469971
day 62, sell 1 units at price 798.530029, investment -1.006632 %, total balance
9977.000000,
day 69: buy 1 units at price 819.239990, total balance 9157.760010
day 84, sell 1 units at price 831.909973, investment 1.546553 %, total balance
9989.669983,
day 85: buy 1 units at price 835.369995, total balance 9154.299988
day 96, sell 1 units at price 817.580017, investment -2.129593 %, total balance
9971.880005,
day 104: buy 1 units at price 834.570007, total balance 9137.309998
day 109, sell 1 units at price 823.349976, investment -1.344409 %, total balance
9960.659974,
day 114: buy 1 units at price 838.210022, total balance 9122.449952
day 151, sell 1 units at price 942.900024, investment 12.489710 %, total balance
10065.349976,
day 160: buy 1 units at price 965.590027, total balance 9099.759949
day 164, sell 1 units at price 917.789978, investment -4.950346 %, total balance
10017.549927,
day 173: buy 1 units at price 947.159973, total balance 9070.389954
day 184, sell 1 units at price 941.530029, investment -0.594403 %, total balance
10011.919983,
day 204: buy 1 units at price 915.890015, total balance 9096.029968
day 218, sell 1 units at price 920.289978, investment 0.480403 %, total balance
10016.319946,
day 225: buy 1 units at price 924.859985, total balance 9091.459961
day 245, sell 1 units at price 970.539978, investment 4.939125 %, total balance
10061.999939,
```

[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()
```



[ ]: