Tree Strategy HW1 Q1, Q2

January 21, 2024

0.1 Imports

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import accuracy_score, confusion_matrix
pd.set_option('use_inf_as_na', True)
from collections import Counter
```

```
/var/folders/sp/wlr6xm29791 8vx6kjh2z1dk00000gn/T/ipykernel_58828/2166773765.py:6
: FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  pd.set_option('use_inf_as_na', True)
```

0.2 Loading the Data Set (you need to put in the file where you have stored the data)

[2]:
```
raw_data = pd.read_pickle('dataset.pkl')
```

[4]:
```
raw_data = raw_data.drop([x  for x in raw_data.columns if 'fqtr' in x],axis=1)
```

0.3 Restricting to Companies with Market Cap > 1 Billion

[5]:
```
data = raw_data[raw_data['market_cap'] > 1000.0]
```

0.4 The Total Number of Companies w/ Market Cap > 1 Billion that appear during our time horizon

[6]:
```
len(data.index.get_level_values(1).unique())
```

[6]: 4076

0.5 Filling in Missing Values

[8]:
```
data = data.copy()
data.replace([np.inf,-np.inf],np.nan,inplace=True)
data = data.fillna(method='ffill')
```

```
/var/folders/sp/wlr6xm29791 8vx6kjh2z1dk00000gn/T/ipykernel_58828/970161762.py:3:
FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  data = data.fillna(method='ffill')
```

[9]:
```
data = data.fillna(0)
```

[10]:
```
data['pred_rel_return']
```

[10]:
```
date        ticker
2000-02-09  CSCO     -0.025923
            ROP       0.066175
2000-02-10  CMOS      0.241345
2000-02-11  DELL      0.306035
2000-02-15  VAL       0.043852
                        …
2018-12-21  NKE      -0.100100
            SAFM     -0.100100
            SCHL     -0.100100
            WBA      -0.100100
2018-12-24  KMX      -0.100100
Name: pred_rel_return, Length: 111468, dtype: float64
```

0.6 HW Question 1

Inserting a column in the dataset where entries are 1 if stock outperforms SPY in the earnings period, and -1 otherwise:

[11]:
```
# function to return appropriate values based on performance
def f_1(x):
    if x > 0:
        return 1
    else:
        return -1
```

[12]:
```
# apply the function to the column of relative returns
data = data.copy()
data['rel_performance_1'] = data['pred_rel_return'].apply(f_1)
```

This is the column of labels next to the original relative returns:

[13]:
```
data[['pred_rel_return', 'rel_performance_1']]
```

[13]:

```
                         pred_rel_return  rel_performance_1
date       ticker
2000-02-09 CSCO                -0.025923                 -1
           ROP                  0.066175                  1
2000-02-10 CMOS                 0.241345                  1
2000-02-11 DELL                 0.306035                  1
2000-02-15 VAL                  0.043852                  1
…                                    …                 …
2018-12-21 NKE                 -0.100100                 -1
           SAFM                -0.100100                 -1
           SCHL                -0.100100                 -1
           WBA                 -0.100100                 -1
2018-12-24 KMX                 -0.100100                 -1

[111468 rows x 2 columns]
```

Thus we can observe that the labels for the stocks whose relative returns are postive (i.e. indiciating it outperformed the SPY) have label 1 (e.g CMOS has label 1).
Otherwise if they are negative or zero, the label is -1 (e.g. NKE has label -1).

0.7 HW Question 2

Inserting a column in the dataset where entries are:
- 2 if the stock return is more than 5% higher than the SPY return
- 1 if it is between 1% and 5% higher than the SPY return
- 0 if it is between -1% and 1% relative to the SPY return
- -1 if it is between -1% and -5% relative to the SPY return

[14]:

```python
# function to return appropriate values based on performance as detailed above
def f_2(x):
    if x > 0.05:
        return 2
    elif x > 0.01:
        return 1
    elif x > -0.01:
        return 0
    elif x > -0.05:
        return -1
```

[15]:

```python
# apply the function to the column of relative returns
data = data.copy()
data['rel_performance_2'] = data['pred_rel_return'].apply(f_2)
```

This is the column of labels next to the original relative returns:

[16]:

```python
data[['pred_rel_return', 'rel_performance_2']]
```

[16]:
```
                       pred_rel_return  rel_performance_2
date       ticker
2000-02-09 CSCO              -0.025923               -1.0
           ROP                0.066175                2.0
2000-02-10 CMOS               0.241345                2.0
2000-02-11 DELL               0.306035                2.0
2000-02-15 VAL                0.043852                1.0
…                                  …                 …
2018-12-21 NKE               -0.100100                NaN
           SAFM              -0.100100                NaN
           SCHL              -0.100100                NaN
           WBA               -0.100100                NaN
2018-12-24 KMX               -0.100100                NaN

[111468 rows x 2 columns]
```

Thus we can observe that the labels were applied correctly:
- CMOS had a relative return of 0.24135 (i.e. ~ 24.1% higher than the SPY) so its label is 2
- VAL had a relative return of 0.043852 (i.e. ~ 4.3% higher than the SPY) so its label is 1
- CSCO had a relative return of -0.025923 (i.e. ~ -2.5% relative to the SPY) so its label is -1
- NKE had a relative return of -0.100100 (i.e. ~ -10% relative to the SPY) so its label is NaN (as we have not assigned a label for those stocks with relative performance less than 5% compared to the SPY.

[17]:
```
# Find those stocks whose relative performance 2 label is 0
data[['pred_rel_return', 'rel_performance_2']].loc[data['rel_performance_2'] ==
 ↪0]
```

[17]:
```
                       pred_rel_return  rel_performance_2
date       ticker
2000-02-24 MDT                0.005511                0.0
           ORTL               0.005511                0.0
2000-03-08 DDS               -0.004425                0.0
2000-04-13 DJ                -0.009937                0.0
2000-04-14 CYN               -0.003053                0.0
…                                  …                 …
2018-09-20 CPRT              -0.004948                0.0
2018-09-27 KMX               -0.008830                0.0
2018-10-01 MTN                0.004993                0.0
2018-10-02 CALM               0.004993                0.0
           KMG                0.004993                0.0

[7647 rows x 2 columns]
```

It remains to check that if we can observe that the label 0 was applied correctly:
- MDT had a relative return of 0.005511 (i.e. ~ 0.55% higher than the SPY) so its label is 0 as this is between -1% and 1%

Thus all labels have been applied appropriately.