TERA HW#2

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
from sklearn.linear_model import Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor as RT
from sklearn.ensemble import RandomForestRegressor as RF
from sklearn.svm import SVR
import multiprocess as mp
```

§ Question 1

寻找与InnerCode==3的股票的时间日期完全一致的股票。按照InnerCode从小到大的规则，汇报前十的结果。

[2]:
```
# Load Data
dat = pd.read_csv('play_data.csv')

# InnerCode 3 is the reference code
reference_code_trading_days = dat[dat["InnerCode"] == 3]["TradingDay"]

# Filter the DataFrame for codes with the same trading days
matching_codes = dat[dat["TradingDay"].
 ↪isin(reference_code_trading_days)]["InnerCode"].unique()
top10 = matching_codes[:10]
# Preserve the top 10 stocks by codes
print(top10)
```

```
[ 3  6 14 17 20 23 26 28 31 34]
```

§ Question 2

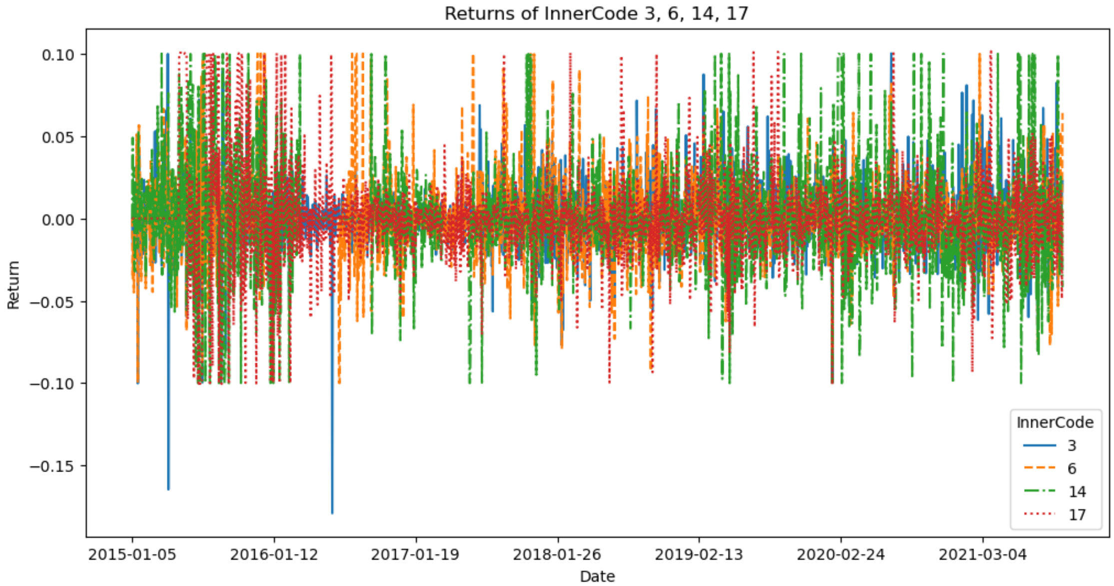
针对这十只股票，根据ClosePrice，求出回报率（Return）。缺失值做补0处理。将InnerCode == 3, 6, 14, 17这4只股票的Return，画在同一个图里，纵轴为时间。

[3]:
```
# update the data frame by selected 20 inner codes
dat_selected = dat[dat["InnerCode"].isin(matching_codes[:20])].copy()

# Calculate the return rate, add it as a new column, and replace the first␣
↪missing observation with 0
dat_selected.loc[:, "Return"] = dat_selected.groupby("InnerCode")["ClosePrice"].
↪pct_change()
dat_selected.loc[:, "Return"] = dat_selected.groupby("InnerCode")["Return"].
↪fillna(0)

# Filter the DataFrame for InnerCodes 3, 6, 14, and 17
selected_codes = [3, 6, 14, 17]
dat_filtered = dat_selected[dat_selected["InnerCode"].isin(selected_codes)]
dat_pivot = dat_filtered.pivot(index="TradingDay", columns="InnerCode",␣
↪values="Return")

# Plotting
styles = ['-', '--', '-.', ':']
dat_pivot.plot(figsize=(12, 6), style=styles)
plt.title("Returns of InnerCode 3, 6, 14, 17")
plt.xlabel("Date")
plt.ylabel("Return")
plt.legend(title="InnerCode")
plt.show()
```



§ Question 3

对InnerCode为3的股票的Return使用线性OLS建模，解释变量如下：Constant, OpenPrice, HighPrice, LowPrice, ClosePrice, log(TurnoverVolume), log(TurnoverValue), AvgPrice, log(TotalMV), log(NegotiableMV)。所有解释变量需要滞后一期。汇报系数估计值，相应的标准误，模型的R-square和Ajusted R-square。简述和第一个作业里类似的题目比，有哪些显著不同，并解释。

[4]:
```
IN = dat_selected["InnerCode"] == 3
y = dat_selected[IN]["Return"]
x = dat_selected[IN].iloc[:,3:-2]
x.iloc[:,[4,5,7,8]] = np.log(x.iloc[:,[4,5,7,8]])
x = x.shift(1)
x = x.iloc[1:-1,:]
y = y.iloc[1:-1]
x = sm.add_constant(x)
result = sm.OLS(y,x).fit()
print(result.summary())
```

OLS Regression Results

Dep. Variable:	Return	R-squared:	0.014
Model:	OLS	Adj. R-squared:	0.009
Method:	Least Squares	F-statistic:	2.578
Date:	Sun, 15 Oct 2023	Prob (F-statistic):	0.00600
Time:	22:05:25	Log-Likelihood:	3930.0
No. Observations:	1640	AIC:	-7840.
Df Residuals:	1630	BIC:	-7786.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0882	0.176	-0.501	0.617	-0.434	0.257
OpenPrice	-0.0092	0.004	-2.339	0.019	-0.017	-0.001
HighPrice	0.0008	0.008	0.102	0.919	-0.014	0.016
LowPrice	0.0092	0.008	1.221	0.222	-0.006	0.024
ClosePrice	-0.0184	0.005	-3.741	0.000	-0.028	-0.009
TurnoverVolume	0.0172	0.017	1.003	0.316	-0.016	0.051
TurnoverValue	-0.0153	0.016	-0.935	0.350	-0.047	0.017
AvgPrice	0.0184	0.013	1.409	0.159	-0.007	0.044
TotalMV	-0.0049	0.017	-0.284	0.777	-0.039	0.029
NegotiableMV	0.0082	0.011	0.718	0.473	-0.014	0.030

Omnibus:	247.459	Durbin-Watson:	1.946
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2836.010
Skew:	-0.291	Prob(JB):	0.00
Kurtosis:	9.416	Cond. No.	1.79e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.79e+04. This might indicate that there are strong multicollinearity or other numerical problems.

§ Question 4

以题目3中的响应变量和解释变量类别为数据，分别考虑题目1中所寻找到的十只股票，重复课后习题一中的滚动窗口实验，窗口长度设为1000，并汇报不同算法在MSFE下的表现。

[5]:
```
%%time

import warnings

# Suppress RuntimeWarning
warnings.filterwarnings("ignore", category=RuntimeWarning)

# define main execution function
def forecast_fun(data):
    import numpy as np
    import pandas as pd
    import statsmodels.api as sm
    from sklearn.linear_model import Lasso, Ridge
    from sklearn.tree import DecisionTreeRegressor as RT
    from sklearn.ensemble import RandomForestRegressor as RF
    from sklearn.svm import SVR

    # ready data
    yt, xt, ye, xe = data
    # ols
    ols = sm.OLS(yt,xt).fit()
    f1 = ols.predict(xe)
    # lasso
    lasso_model = Lasso()
    lasso_model.fit(xt, yt)
    f2 = lasso_model.predict(xe)
    # ridge
    ridge_model = Ridge(alpha=0.1)
    ridge_model.fit(xt, yt)
    f3 = ridge_model.predict(xe)
    # regression tree
    tree_model = RT()
```

```
    tree_model.fit(xt, yt)
    f4 = tree_model.predict(xe)
    # random forest regression
    forest_model = RF(max_features=3)
    forest_model.fit(xt, yt)
    f5 = forest_model.predict(xe)
    # SVR (Linear Kernel)
    linear_svr_model = SVR(kernel='linear')
    linear_svr_model.fit(xt, yt)
    f6 = linear_svr_model.predict(xe)
    # SVR (Gaussian Kernel)
    gaussian_svr_model = SVR(kernel='rbf')
    gaussian_svr_model.fit(xt, yt)
    f7 = gaussian_svr_model.predict(xe)
    # merge results and deliver output
    f = np.hstack((f1.values,f2,f3,f4,f5,f6,f7))
    ERRt = f-np.full(f.size,ye)
    return ERRt

# Main execution
if __name__ == '__main__':
    # start worker pool
    num_cores = mp.cpu_count()
    pool = mp.Pool(processes=num_cores)
    # Generate data partitions
    WL = 1000
    T = y.size
    Result = []
    for i in top10:
        IN = dat_selected["InnerCode"] == i
        y = dat_selected[IN]["Return"]
        x = dat_selected[IN].iloc[:,3:-2]
        x.iloc[:,[4,5,7,8]] = np.log(x.iloc[:,[4,5,7,8]])
        x = x.shift(1)
        x = x.iloc[1:-1,:]
        y = y.iloc[1:-1]
```

```
        x.replace([np.inf, -np.inf, np.nan], 0, inplace=True)
        y.replace([np.inf, -np.inf, np.nan], 0, inplace=True)
        x = sm.add_constant(x)
        datafull = []
        for t in range(T-WL):
            INt = np.arange(t,t+WL)
            INe = [t+WL]
            yt = y.iloc[INt]
            xt = x.iloc[INt,:]
            ye = y.iloc[INe]
            xe = x.iloc[INe,:]
            datafull.append([yt,xt,ye,xe])
        # parallel estimation
        results = pool.map(forecast_fun, datafull)
        ERR = np.array(results)
        Result.append(np.mean(ERR ** 2,axis=0))

    pool.close()
    pool.join()
```

```
CPU times: total: 10 s
Wall time: 2min 16s
```

[6]:

```
# print evaluation results
VN = ['LM','Lasso','Ridge','RT','RF','SVR-L','SVR-G']
R = pd.DataFrame(Result)
R.columns = VN
tmp = pd.DataFrame({'InnerCode': top10})
R = pd.concat([tmp, R], axis=1)

# Function to highlight the minimum value in each row
def highlight_min(s):
    is_min = s == s.min()
    return ['background-color: blue' if v else '' for v in is_min]

# Apply the style
styled_R = R.style.apply(highlight_min, axis=1, subset=VN)
```

```python
# Display the styled DataFrame
styled_R
```

[6]:

	InnerCode	LM	Lasso	Ridge	RT	RF	SVR-L	SVR-G
0	3	0.000531	0.000520	0.000529	0.001075	0.000610	0.000806	0.000607
1	6	0.000425	0.000413	0.000414	0.000881	0.000462	0.000421	0.000421
2	14	0.001480	0.001318	0.001348	0.002578	0.001488	0.001318	0.001318
3	17	0.000607	0.000583	0.000610	0.001510	0.000706	0.000586	0.000587
4	20	0.000521	0.000523	0.000522	0.001078	0.000605	0.000522	0.000523
5	23	0.001165	0.001153	0.001156	0.002333	0.001328	0.001150	0.001144
6	26	0.000322	0.000320	0.000322	0.000867	0.000386	0.000323	0.000321
7	28	0.001483	0.001361	0.001408	0.002516	0.001519	0.001489	0.001345
8	31	0.000629	0.000609	0.000621	0.001324	0.000667	0.000609	0.000608
9	34	0.070105	0.001028	0.001069	0.002510	0.001197	0.001027	0.001027

§ Question 5

以题目4中的滚动窗口实验为基础，考虑以下交易策略：

a) 相同时间窗口下，预测下一期所有股票的回报率，并选取预测回报率最高的股票。
b) 保留该选取股票的实际回报率，并滚动到下一期。
c) 重复a),b)步骤只到数据最后，统计所有算法选取下的平均实际回报率。

[7]:

```
%%time

# Main execution
if __name__ == '__main__':
    # start worker pool
    num_cores = mp.cpu_count()
    pool = mp.Pool(processes=num_cores)
    # Generate data partitions
    WL = 1000
    T = y.size
    data_stock = []
    YE = []
    for t in range(T-WL):
        INt = np.arange(t,t+WL)
        INe = [t+WL]
        for i in top10:
            IN = dat_selected["InnerCode"] == i
            y = dat_selected[IN]["Return"]
            x = dat_selected[IN].iloc[:,3:-2]
            x.iloc[:,[4,5,7,8]] = np.log(x.iloc[:,[4,5,7,8]])
            x = x.shift(1)
            x = x.iloc[1:-1,:]
            y = y.iloc[1:-1]
            x.replace([np.inf, -np.inf, np.nan], 0, inplace=True)
            y.replace([np.inf, -np.inf, np.nan], 0, inplace=True)
            x = sm.add_constant(x)
            yt = y.iloc[INt]
            xt = x.iloc[INt,:]
            ye = y.iloc[INe]
            xe = x.iloc[INe,:]
            data_stock.append([yt,xt,ye,xe])
            YE.append(ye)

    # parallel estimation
    results = pool.map(forecast_fun, data_stock)
```

```
    pool.close()
    pool.join()
```

```
CPU times: total: 24.4 s
Wall time: 2min 39s
```

[8]:
```
# calculate final result
Result = []
YE_array = np.array(YE)
for t in range(T-WL):
    tmp = pd.DataFrame(results[10*t:10*(t+1)])
    Yt = YE_array[10*t:10*(t+1)]
    max_index = tmp.idxmax(axis=0)
    Result.append(Yt[max_index.values])

# print trading results
VN = ['LM','Lasso','Ridge','RT','RF','SVR-L','SVR-G']
a = np.mean(Result,axis=0)
a_reshaped = a.reshape(1, -1)
R = pd.DataFrame(a_reshaped, columns = VN)
R
```

[8]:

	LM	Lasso	Ridge	RT	RF	SVR-L	SVR-G
0	-0.031226	-0.03254	-0.031722	-0.017613	-0.027407	-0.031485	-0.032265

§ Question 6

想出一种交易策略，击败题目5中的所有方法。

[9]:
```
# simply aveaging
SA = []
tmp = np.array(YE)
for t in range(T-WL):
    SA.append(np.mean(tmp[10*t:10*(t+1)]))
print("Simple Averaging across All Stocks: ", np.mean(SA))
```

```
Simple Averaging across All Stocks:  0.00048709064159909875
```