Lasso Verification

Data Reading

```python
data = pd.read_csv('data/fret.csv')
data = data.set_index(pd.to_datetime(data['date'],format='%Y-%m-%d %H:%M'))
data.drop(['date'],axis = 1,inplace = True)
data.drop(['R2'],axis = 1,inplace = True)
data.head()
```

]:

date	mkt	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability	fequality	fprofitability	fvalue
2006-12-18	0.021884	0.001301	0.000062	-0.000825	0.001971	0.000675	-0.001601	-0.002369	0.001764	-0.002787
2006-12-19	0.004089	0.000886	0.000118	0.001074	-0.003136	-0.001030	0.000809	0.000508	-0.002583	0.001104
2006-12-20	0.010546	-0.002768	-0.001299	0.001113	-0.003718	0.000540	-0.002922	-0.000120	0.003587	-0.001841
2006-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Drop R2 columns and make date as index

Data Reversing

```python
time_length = int(str(data.index[-1]-data.index[0])[:4])/365
non_industry_columns = ['mkt', 'fsize', 'fvolatility', 'fliquidity', 'fmomentum', 'fquality',
        'fevariability', 'fequality', 'fprofitability', 'fvalue', 'fgrowth',
        'fsentiment', 'fdivyld']
data_reversed = data.copy()
data_reversed[['fsize', 'fliquidity','fmomentum','fevariability']] =\
         -data_reversed[['fsize', 'fliquidity','fmomentum','fevariability']].copy()
## reverse the data
data_reversed_without_mkt = data_reversed.copy()
data_reversed_without_mkt.drop(columns=['mkt'],inplace = True)
data_reversed
```

Reverse the data

date	mkt	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability	fequality	fprofitability	fvalue	...
2006-12-18	0.021884	-0.001301	0.000062	0.000825	-0.001971	0.000675	0.001601	-0.002369	0.001764	-0.002787	...
2006-12-19	0.004089	-0.000886	0.000118	-0.001074	0.003136	-0.001030	-0.000809	0.000508	-0.002583	0.001104	...
2006-12-20	0.010546	0.002768	-0.001299	-0.001113	0.003718	0.000540	0.002922	-0.000120	0.003587	-0.001841	...
2006-12-21	-0.013952	0.000833	-0.001085	0.001847	0.007468	-0.000096	0.000816	-0.000073	0.001526	-0.000756	...
2006-12-22	-0.010079	0.003985	-0.002734	0.003572	0.001342	-0.001037	0.000791	-0.000797	-0.003769	-0.000567	...
...	...	...	...	...	...	...	...	...	...	...	...

Drop the marketing

]:

date	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability	fequality	fprofitability	fvalue	fgrowth	...
2006-12-18	-0.001301	0.000062	0.000825	-0.001971	0.000675	0.001601	-0.002369	0.001764	-0.002787	0.000103	...
2006-12-19	-0.000886	0.000118	-0.001074	0.003136	-0.001030	-0.000809	0.000508	-0.002583	0.001104	0.001177	...
2006-12-20	0.002768	-0.001299	-0.001113	0.003718	0.000540	0.002922	-0.000120	0.003587	-0.001841	0.001391	...
2006-12-21	0.000833	-0.001085	0.001847	0.007468	-0.000096	0.000816	-0.000073	0.001526	-0.000756	0.001173	...
2006-12-22	0.003985	-0.002734	0.003572	0.001342	-0.001037	0.000791	-0.000797	-0.003769	-0.000567	-0.002774	...
...	...	...	...	...	...	...	...	...	...	...	...

Feature Making

```python
momentum_list = [[-1,-1], [-2,-2], [-3,-3], [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]]
## 第一位为回看起始 第二位为会看终止 【-6, -21】表示从过去第六天看到过去弟21天
factor_list = data_reversed.columns
def momentum_cal(input_data,input_list):
    data_momentum = input_data.shift(-input_list[0])
    data_momentum = data_momentum.rolling(-(input_list[1]-input_list[0])+1).sum()
    return data_momentum
data_momentum_list = []
for num in range(len(momentum_list)):
    data_momentum_list.append('data_momentum'+str(num))
    names['data_momentum'+str(num)] = momentum_cal(data_reversed,momentum_list[num])
data_momentum0
```

The original one:

date	mkt	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability	fequality	fprofitability	fvalue	...
2006-12-18	0.021884	0.001301	0.000062	-0.000825	0.001971	0.000675	-0.001601	-0.002369	0.001764	-0.002787	...
2006-12-19	0.004089	0.000886	0.000118	0.001074	-0.003136	-0.001030	0.000809	0.000508	-0.002583	0.001104	...

Past 1 day data

Move the past day data to the current index

date	mkt	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability	fequality	fprofitability	fvalue	...	S'
2006-12-18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2006-12-19	0.021884	-0.001301	0.000062	0.000825	-0.001971	0.000675	0.001601	-0.002369	0.001764	-0.002787	...	-
2006-12-20	0.004089	-0.000886	0.000118	-0.001074	0.003136	-0.001030	-0.000809	0.000508	-0.002583	0.001104	...	-
2006-	0.010546	0.002768	-0.001299	-0.001113	0.003718	0.000540	0.002922	-0.000120	0.003587	-0.001841	...	

The original one

date	mkt	
2006-12-18	0.021884	0
2006-12-19	0.004089	0
2006-12-20	0.010546	-0
2006-12-21	-0.013952	-0
2006-12-22	-0.010079	-0

Past [-4,5] day data

date	mkt	fsize	fvolatility	fliqu
2006-12-22	NaN	NaN	NaN	
2006-12-25	0.025973	-0.002187	0.000180	-0.00
2006-12-26	0.014635	0.001882	-0.001181	-0.00
2006-	-0.003406	0.003601	-0.002384	0.00

0.021884+0.004089 = 0.025973

So as the code has the same structure, the rest are all correct.

Feature appending

```
for factor in factor_list:
    names[factor] = pd.DataFrame()
    for num in range(len(momentum_list)):
        names[factor] = pd.concat([names[factor],names['data_momentum'+str(num)][factor]\
                                   .rename(str(momentum_list[num][0])+'_'+factor)],axis =1,join='outer')
    names[factor].dropna(how='any',inplace = True)
mkt
```

The same date of different feature matrixes:

```
data_momentum5.iloc[251:]
```

date	mkt	fsize
2007-12-28	-0.158375	0.007380
2008-01-02	-0.129464	0.002506

```
data_momentum6.iloc[251:]
```

date	mkt	fsize
2007-12-28	0.301685	-0.003930
2008-01-02	0.355772	-0.000545

```
data_momentum7.iloc[251:]
```

date	mkt	fsize
2007-12-28	NaN	NaN
2008-01-02	0.803041	0.083786

The concated feature matrix:

```
    names[factor].dropna(how='any',inplace = True)
mkt
```

5 6 7

	-1_mkt	-2_mkt	-3_mkt	-4_mkt	-6_mkt	-22_mkt	-65_mkt	-127_mkt
2008-01-02	-0.006879	0.016640	0.013014	0.024584	0.082445	-0.129464	0.355772	0.803041
2008-01-03	0.017101	-0.006879	0.016640	0.017030	0.122929	-0.138432	0.374626	0.751355

Label Generating

The return of each day

date	mkt
2006-12-18	0.021884
2006-12-19	0.004089
2006-12-20	0.010546
2006-12-21	-0.013952
2006-12-22	-0.010079
2006-12-25	0.017438

```
0.004089+0.010546-0.013952-0.010079+0.017438
```

0.008041999999999997

The next week return for the current day index

```
y_train_1day = data_reversed.shift(-1)
y_train_1week = data_reversed.rolling(5).sum().shift(-5)
y_train_1month = data_reversed.rolling(21).sum().shift(-21)
y_train_1week.head()
```

date	mkt	fsize	fvolatility	fliquidity	fmomentum	fquality	fevariability
2006-12-18	0.008042	0.003383	-0.002038	0.004150	0.014159	-0.001333	0.002685
2006-12-19	0.000023	0.003119	-0.002684	0.008013	0.005454	-0.003121	0.002330
2006-12-20	0.006381	-0.003093	0.002396	0.010332	0.001075	-0.002940	-0.000740

So this part is also correct

Model fitting

```
def linear_pred_day(y_train_all, factor_all, all_index, day_lag):
    result = pd.DataFrame(columns =factor_list, index = all_index)
    # initial the model
    model_LinearRegression = linear_model.Lasso(alpha=1e-6, fit_intercept=False, max_iter=5000)
    # loop over all the fators
    for factor in factor_all:
        for date in range(750+day_lag, len(names[factor])):
            # to aviod using the future data, we need to start the train data including X and Y from the past
            # as the Y is using the next d/w/m return, the train should start from a a/w/m ago
            # so here need to dvide the day_lag
            x_train = names[factor].iloc[date-750-day_lag:date-day_lag]
            y_train = y_train_all.loc[x_train.index, factor]
            # drop the nan if Y has nan drop that line
            y_train.dropna(inplace = True)
            x_train = x_train.loc[y_train.index]
            x_test = names[factor].iloc[date]
            # normalizing X
            train = np.array((x_train-x_train.mean())/x_train.std())
            test = np.array((x_test-x_test.mean())/x_test.std()).reshape(1, -1)
```

```
            # normalizing X
            train = np.array((x_train-x_train.mean())/x_train.std())
            test = np.array((x_test-x_test.mean())/x_test.std()).reshape(1, -1)

            # fit the model and make predictions
            model_LinearRegression.fit(train, y_train)
            y_pred = model_LinearRegression.predict(test)
#             save the data with corresponding test input date
            result.loc[names[factor].iloc[date].name, factor] = y_pred[0]
        print(model_LinearRegression.coef_, factor)
    result.dropna(how = 'all', inplace = True)
    return result
```

Quintile Making

As I just use the code before, I just put a test example here

```
test_data
```

	a	b	c	d	e	f	g	h	i	j
0	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11

```
label_data
```

	a	b	c	d	e	f	g	h	i	j
0	10	9	8	7	6	5	4	3	2	1
1	11	10	9	8	7	6	5	4	3	2

1 mean the next day

```
quintile_res = quintile_nth_day_return(test_data, label_data.dropna(), 1)
quintile_res
```

	quintile_1	quintile_2	quintile_3	quintile_4	quintile_5	the_hedge_portfolio
0	5.5	4.5	3.5	2.5	1.5	-4.0
1	NaN	NaN	NaN	NaN	NaN	NaN

LASSO RESULT

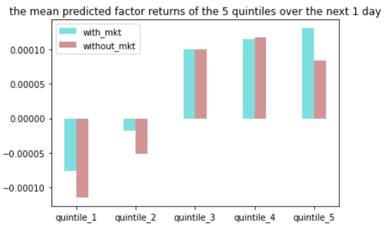
Code example of quitile spliting

Data reversed is the data loaded from fret.csv, the same as the training set

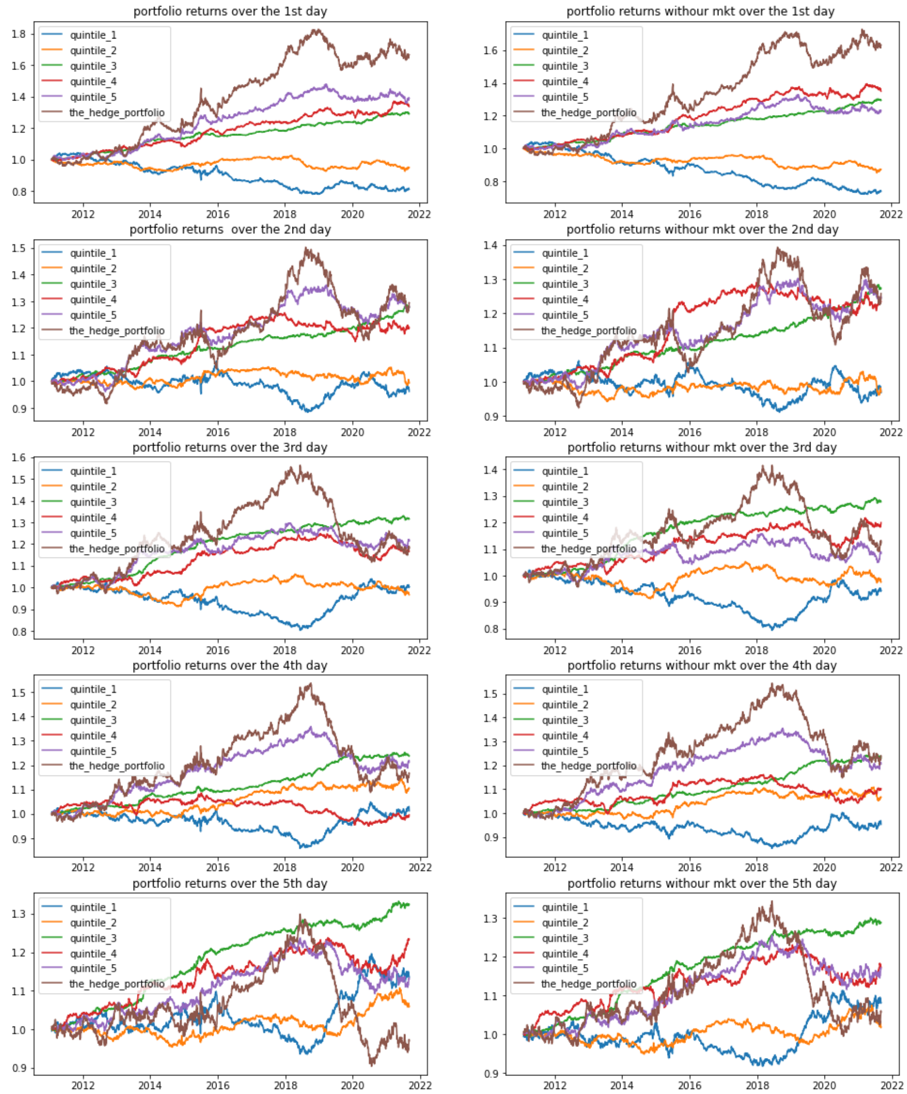
```
quintile_1st_day_res = quintile_nth_day_return(data_reversed, result_month.dropna(), 1)
quintile_2nd_day_res = quintile_nth_day_return(data_reversed, result_month.dropna(), 2)
quintile_3rd_day_res = quintile_nth_day_return(data_reversed, result_month.dropna(), 3)
quintile_4th_day_res = quintile_nth_day_return(data_reversed, result_month.dropna(), 4)
quintile_5th_day_res = quintile_nth_day_return(data_reversed, result_month.dropna(), 5)
```

```
quintile_1st_day_res_without_mkt = quintile_nth_day_return(data_reversed_without_mkt, result_month_without_mkt.dropna(), 1)
quintile_2nd_day_res_without_mkt = quintile_nth_day_return(data_reversed_without_mkt, result_month_without_mkt.dropna(), 2)
quintile_3rd_day_res_without_mkt = quintile_nth_day_return(data_reversed_without_mkt, result_month_without_mkt.dropna(), 3)
quintile_4th_day_res_without_mkt = quintile_nth_day_return(data_reversed_without_mkt, result_month_without_mkt.dropna(), 4)
quintile_5th_day_res_without_mkt = quintile_nth_day_return(data_reversed_without_mkt, result_month_without_mkt.dropna(), 5)
```

The Model is predicting the next day return with next day return as label in Lasso regression
The input is the **single factor** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]
The model is updated **each day** before next day prediction alpha = 1e-6
For the quintile split I just use the original data as the return.
The reason of week, month predcition performs better is that that has the return of past days in trainning so the label is more smooth and more predictable.



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	0.04925	0.067791	0.763207
2nd_the_hedge_portfolio	0.024575	0.068441	0.398867
3rd_the_hedge_portfolio	0.015818	0.066193	0.276846
4th_the_hedge_portfolio	0.014646	0.067167	0.256158
5th_the_hedge_portfolio	-0.002881	0.066788	-0.011031
1st_the_hedge_portfolio_without_mkt	0.04711	0.06452	0.766011
2nd_the_hedge_portfolio_without_mkt	0.021818	0.064429	0.376573
3rd_the_hedge_portfolio_without_mkt	0.011138	0.062289	0.213957
4th_the_hedge_portfolio_without_mkt	0.020185	0.063504	0.355329
5th_the_hedge_portfolio_without_mkt	0.005231	0.063493	0.116259



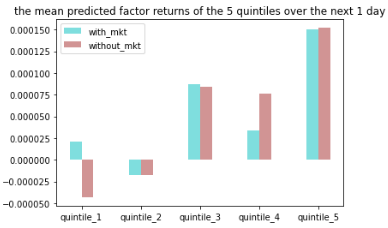
The Model is predicting the <u>next day</u> return with <u>next week return as label</u> in Lasso regression

The input is the **single factor** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]

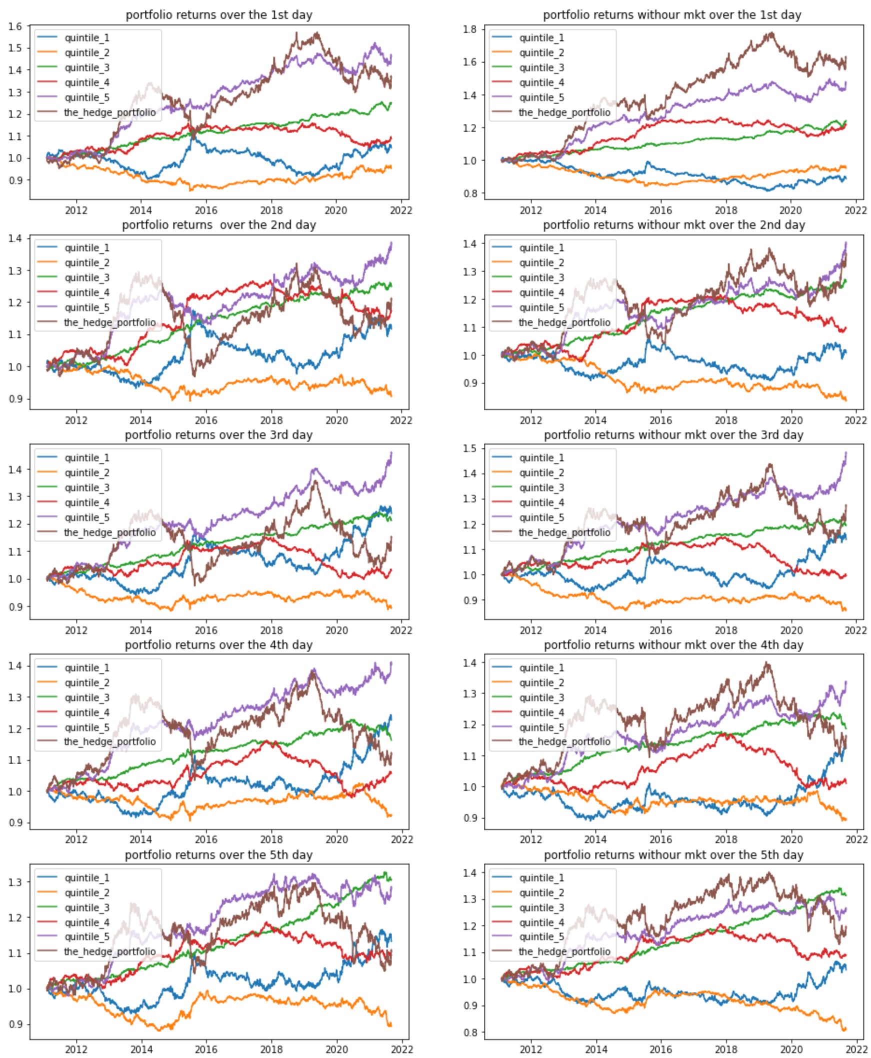
The model is updated **<u>each day</u>** before next day prediction alpha = 1e-6

For the quintile split I just use the original data as the return.

<u>The reason of week, month predcition performs better is that that has the return of past days in trainning so the label is more smooth and more predictable.</u>



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	0.029721	0.064525	0.498204
2nd_the_hedge_portfolio	0.017955	0.064513	0.315308
3rd_the_hedge_portfolio	0.013231	0.064645	0.240987
4th_the_hedge_portfolio	0.010435	0.064101	0.198287
5th_the_hedge_portfolio	0.009302	0.0646	0.179441
1st_the_hedge_portfolio_without_mkt	0.046669	0.062736	0.777731
2nd_the_hedge_portfolio_without_mkt	0.029224	0.062772	0.502262
3rd_the_hedge_portfolio_without_mkt	0.022907	0.062699	0.402053
4th_the_hedge_portfolio_without_mkt	0.013954	0.062815	0.257844
5th_the_hedge_portfolio_without_mkt	0.016599	0.0635	0.297889



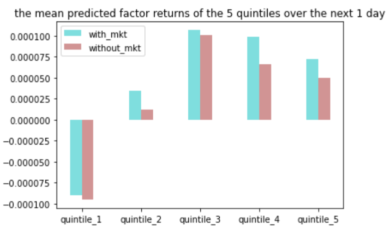
The Model is predicting the <ins>next day</ins> return with <ins>next month return as label</ins> in Lasso regression

The input is the **single factor** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]

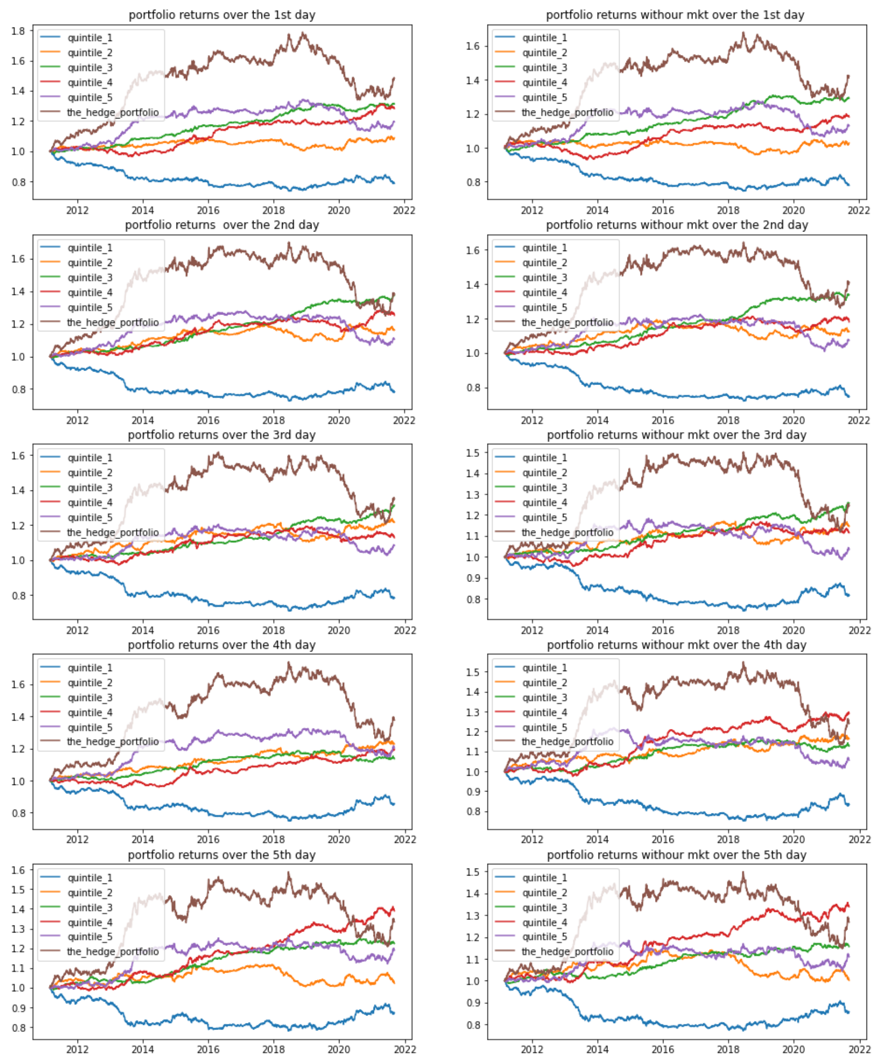
The model is updated <ins>**each day**</ins> before next day prediction alpha = 1e-6

For the quintile split I just use the original data as the return.

<ins>The reason of week, month predcition performs better is that that has the return of past days in trainning so the label is more smooth and more predictable.</ins>



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	0.038062	0.06536	0.619686
2nd_the_hedge_portfolio	0.031261	0.064397	0.522966
3rd_the_hedge_portfolio	0.028875	0.06384	0.489755
4th_the_hedge_portfolio	0.03076	0.064209	0.516646
5th_the_hedge_portfolio	0.027766	0.064493	0.468399
1st_the_hedge_portfolio_without_mkt	0.033993	0.063056	0.576029
2nd_the_hedge_portfolio_without_mkt	0.033281	0.062821	0.566459
3rd_the_hedge_portfolio_without_mkt	0.021053	0.062417	0.373947
4th_the_hedge_portfolio_without_mkt	0.020708	0.062442	0.368308
5th_the_hedge_portfolio_without_mkt	0.022812	0.062712	0.400756



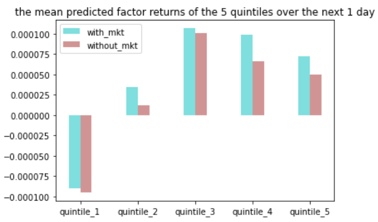
The Model is predicting the next day return with next day return as label in Lasso regression

The input is the **all the factors** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]

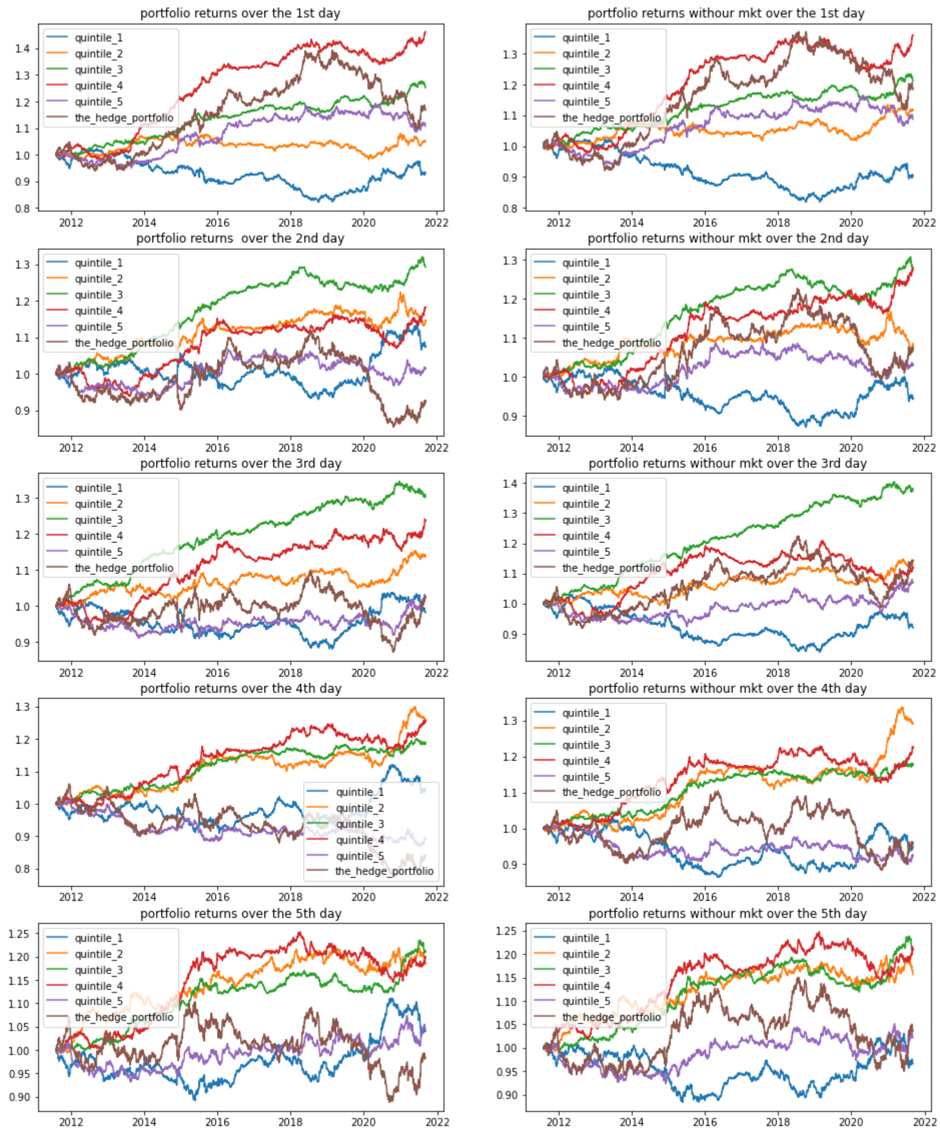
The model is updated **each day** before next day prediction alpha = 1e-4

For the quintile split I just use the original data as the return.

The cumulative factor returns over the next 1 day 1 week and 1 month, in histogram



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	0.01541	0.060975	0.288346
2nd_the_hedge_portfolio	-0.007888	0.061189	-0.102387
3rd_the_hedge_portfolio	0.002377	0.060842	0.070538
4th_the_hedge_portfolio	-0.017149	0.061774	-0.256952
5th_the_hedge_portfolio	-0.00155	0.062355	0.00561
1st_the_hedge_portfolio_without_mkt	0.016832	0.05893	0.320685
2nd_the_hedge_portfolio_without_mkt	0.006725	0.058192	0.147488
3rd_the_hedge_portfolio_without_mkt	0.013256	0.058451	0.260806
4th_the_hedge_portfolio_without_mkt	-0.004187	0.059525	-0.042688
5th_the_hedge_portfolio_without_mkt	0.003687	0.060004	0.09305



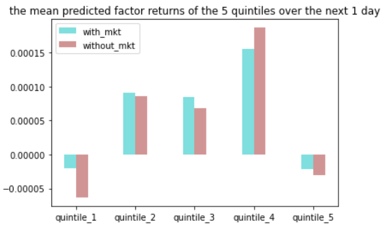
The Model is predicting the <u>next day</u> return with <u>next week return as label</u> in Lasso regression

The input is the **all the factors** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]

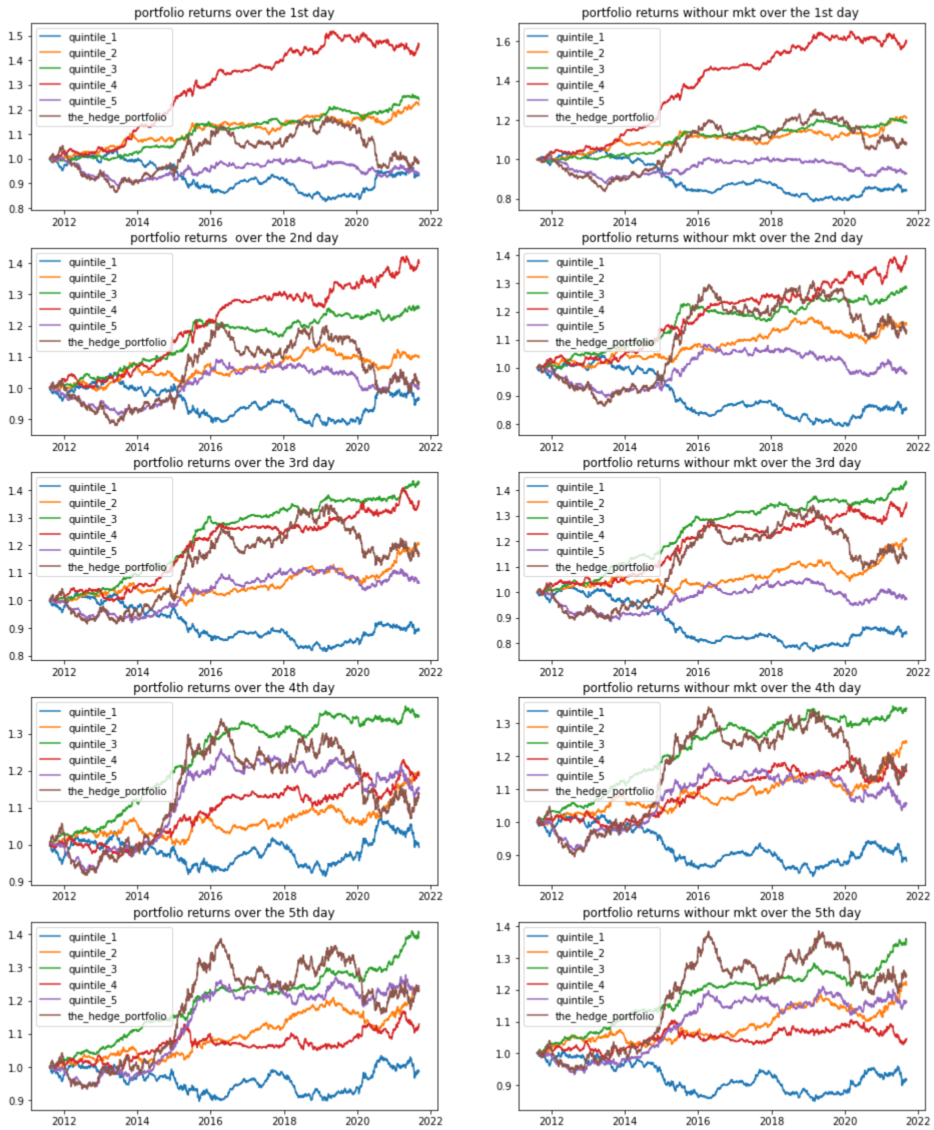
The model is updated **<u>each day</u>** before next day prediction alpha = 1e-4

For the quintile split I just use the original data as the return.

The cumulative factor returns over the next 1 day 1 week and 1 month, in histogram



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	-0.001072	0.06064	0.012144
2nd_the_hedge_portfolio	0.001687	0.060352	0.058897
3rd_the_hedge_portfolio	0.015638	0.060771	0.292845
4th_the_hedge_portfolio	0.012826	0.060251	0.247568
5th_the_hedge_portfolio	0.02076	0.060798	0.377867
1st_the_hedge_portfolio_without_mkt	0.008114	0.058012	0.172224
2nd_the_hedge_portfolio_without_mkt	0.012291	0.057723	0.246358
3rd_the_hedge_portfolio_without_mkt	0.013018	0.05887	0.255263
4th_the_hedge_portfolio_without_mkt	0.015931	0.058871	0.305427
5th_the_hedge_portfolio_without_mkt	0.022025	0.058389	0.412812



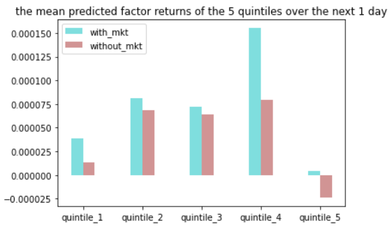
The Model is predicting the next day return with next month return as label in Lasso regression

The input is the **all the factors** return over day -1, -2, -3, [-4,-5],[-6,-21],[-22,-64],[-65, -126],[-127,-252]

The model is updated **each day** before next day prediction alpha = 1e-4

For the quintile split I just use the original data as the return.

The cumulative factor returns over the next 1 day 1 week and 1 month, in histogram



	annualized_return	annualized_volatility	sharpe_ratio
1st_the_hedge_portfolio	-0.009429	0.05812	-0.138602
2nd_the_hedge_portfolio	-0.01109	0.059341	-0.163554
3rd_the_hedge_portfolio	-0.009853	0.059394	-0.141727
4th_the_hedge_portfolio	0.002933	0.059092	0.080537
5th_the_hedge_portfolio	0.008596	0.059028	0.178671
1st_the_hedge_portfolio_without_mkt	-0.007823	0.055474	-0.117896
2nd_the_hedge_portfolio_without_mkt	-0.013006	0.0571	-0.207181
3rd_the_hedge_portfolio_without_mkt	-0.014598	0.057417	-0.23466
4th_the_hedge_portfolio_without_mkt	-0.004887	0.057243	-0.059394
5th_the_hedge_portfolio_without_mkt	0.000711	0.056897	0.041302