Project Report

Tao Bian

November 30, 2021

1 Summary

In this project, I build a simply alpha model and perform backtesting to validate its performance. The obtained model used a two-layers construction to reduce the turnover and boost the Sharpe ratio. The resulting portfolio maintains an average after fee Sharpe ratio above 2.9 over the whole backtesting period (including testing periods). Detailed performance summary is given at the end of this report.

2 Validate factors

Before constructing the model, I first inspect the raw factors provided in the *data_sets.zip* file by plotting the distribution of the stock counts in each factor across the backtesting period. It is each to see that factors from *data_set_1* to *data_set_7* are fundamental factors and are mainly updated at each quarter end. Factors *data_set_8* and *data_set_9* are technical factors and the stock counts match the counts of stocks in the trading universe. Factors *data_set_10* and *data_set_11* are probably some alternative data factors with less stocks compared with others.

Since the fundamental factors are not uniformly updated across the year, I filled in the missing data by applying forward fill with period of 80 days. This period covers almost four months which is longer than a quarter.

Based on the distribution of data, I separate the backtesting period into three sample periods. Pre-sample is from 2010 to 2012. In-sample is from 2012 to 2016. Post-sample is from 2016 to 2018. The reason is that from 2010 to 2012, the stock counts are small across these factors, and there is also a gap in 2011 which could distract our analysis. I also keep 2 years of data as out-of-sample to test the model performance.

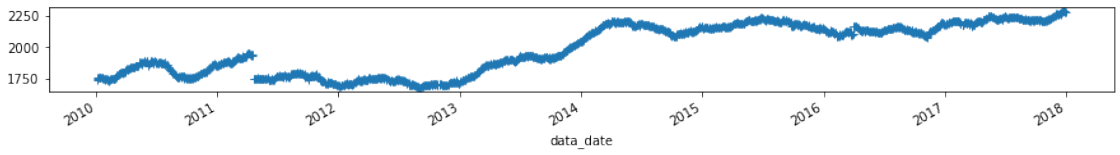

Figure 1: Trading universe.

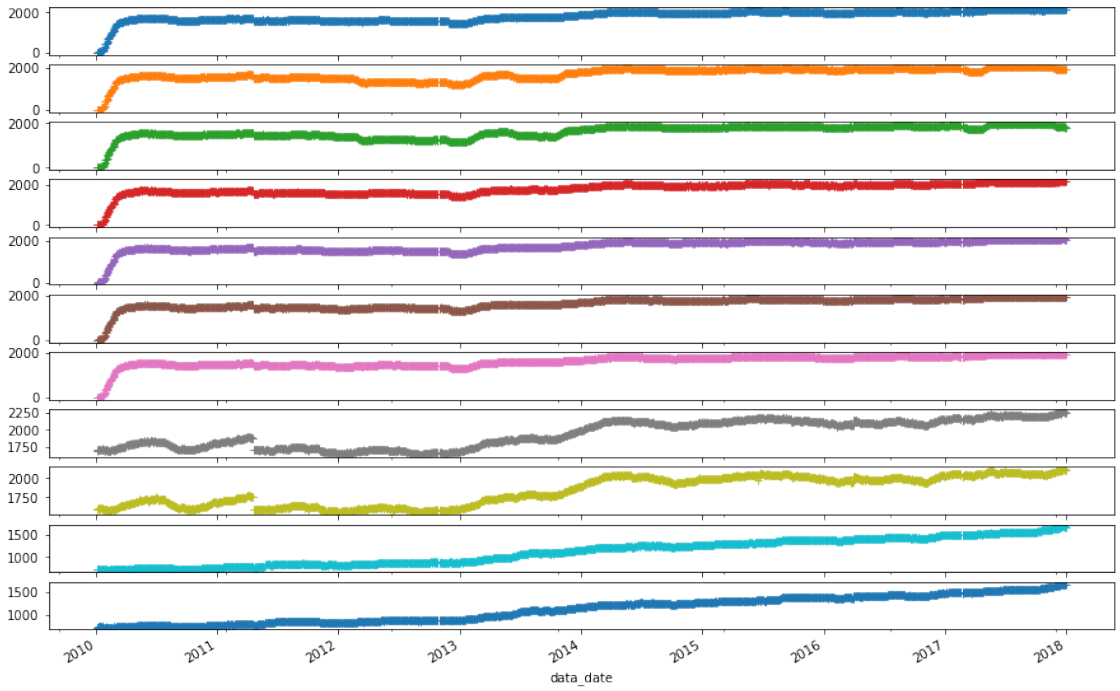

Figure 2: Stock counts in each factor across the backtesting period. Plots are ordered from *data_set_1* to *data_set_11* .

3 Backtest raw factors

For each raw factor, I also performed a simply backtesting to further investigate its properties. Give a factor, I first rank stocks on each day based on the factor score, and then groups these ranks into different percentile groups. For different factor, the counts of groups are different to ensure the stocks in each group are roughly the same. The number of groups should be at least 30 to draw statistically significant conclusions. For *data_set_1* to *data_set_7*, I use 30 groups; for *data_set_8* and *data_set_9*, I use 40 groups; for *data_set_10* and *data_set_11*, I use 30 groups. In particular, for a given factor, 9999 represents stocks that do not have scores under this factor but are still in the trading universe. I perform backtesting on each group. This is done in python using vectorization to speed up speed. It is assumed in the backtesting that the shares to be traded everyday are determined before market open using preclose price (yesterday's close price after adjustment).

I inspect the stock counts, turnover, mean return, volatility, and Sharpe ratio of each group during the in-sample period. Some of these plots are given below.

Detailed results are given in the Jupyter notebook. Here are some observations:

- The last four factors have higher turnover rate, which is reasonable due to the nature of these factors.
- Most factors do not have the ability to generate short portfolio alone, since the mean returns in all those groups are positive. The only exceptions are *data_set_9* and *data_set_8* which produce two groups with negative return.
- The fundamental factors do not provide a strong separation, in the sense that several groups have similar return/Sharpe ratio and it is not easy to find the best group.

Given the above observations, our strategy is to build a two-layer model. The first layer uses fundamental factors to separate the whole trading universe into two groups. Then I use the last four high-frequency factors to build long and short models on these groups respectively. This method can bypass the shortcomings of fundamental factors listed above, but can still leverage the information in fundamental data to boost the performance of high-frequency models.

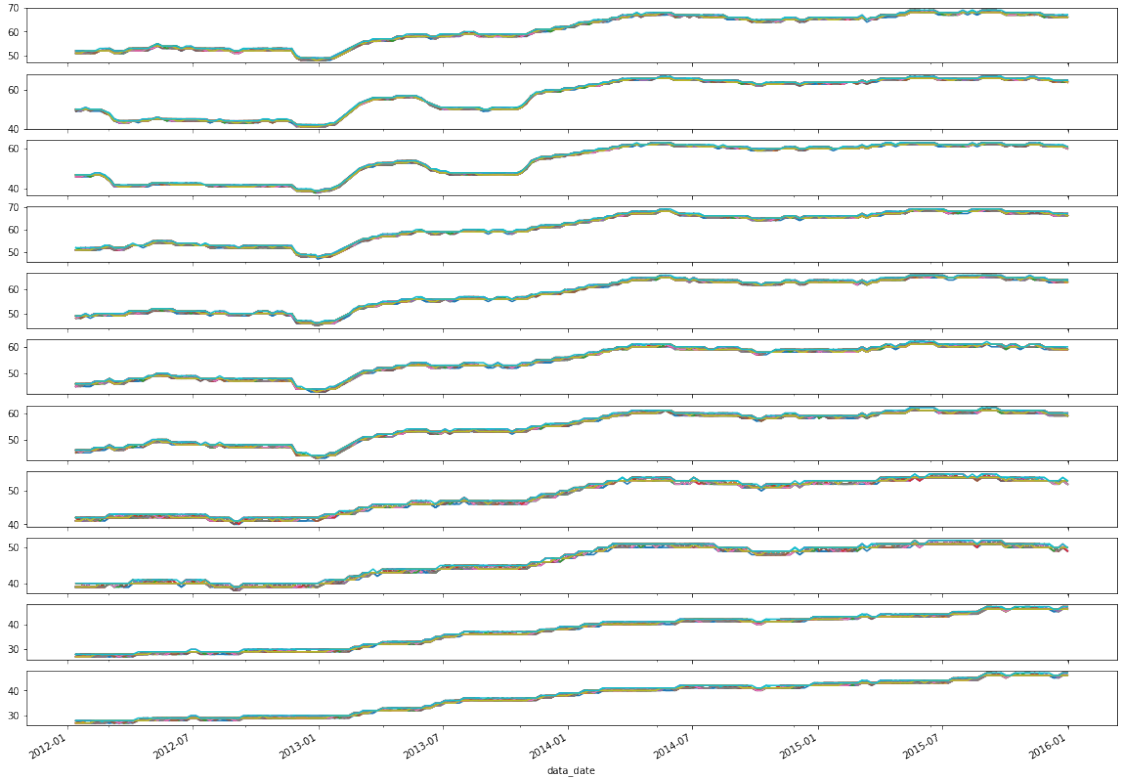

Figure 3: Stock counts in each group over the in-sample period. Plots are ordered from *data_set_1* to *data_set_11* .

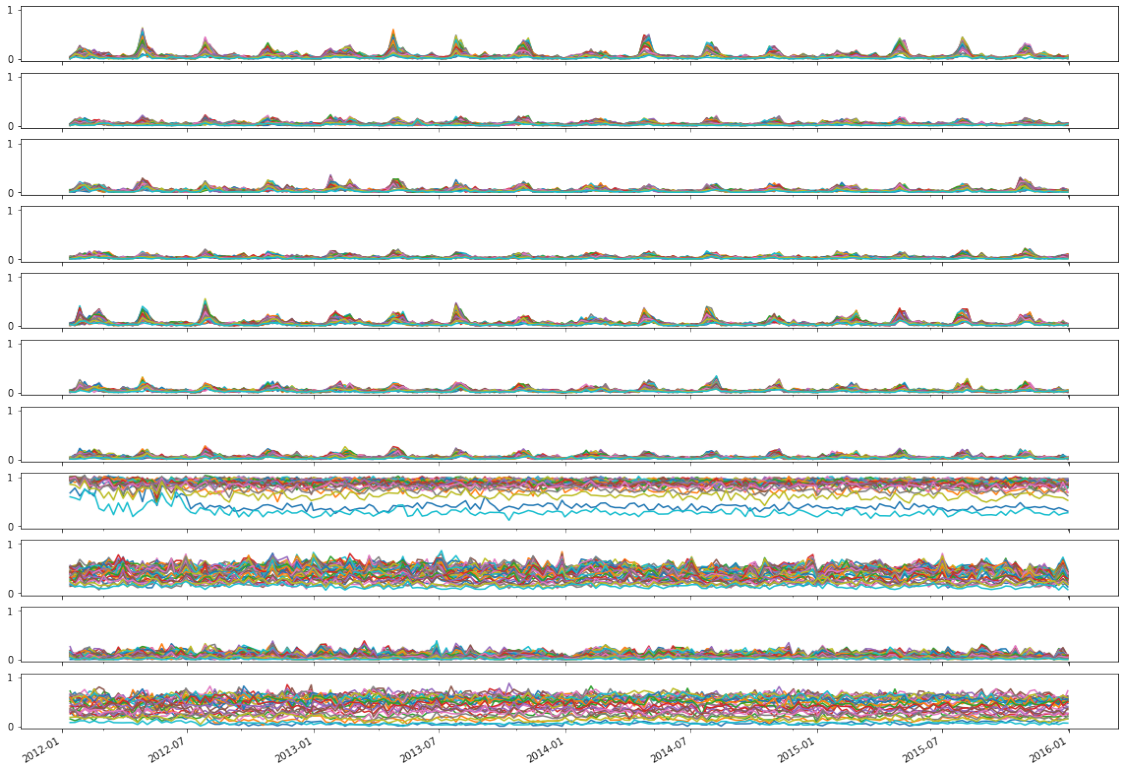

Figure 4: Daily turnovers in each group over the in-sample period. Plots are ordered from *data_set_1* to *data_set_11* .

Figure 5: Weekly return in each group over the in-sample period. Plots are ordered from *data_set_1* to *data_set_11* .

4 Combine factors

4.1 Layer 1

In the first layer, I redo the ranking and grouping process for the first seven fundamental factors. The difference here is to only use 6 groups in order to ensure a stable construction of the long and

short universe.

After performing the backtesting, I pick one group with high return from each factor to build the long universe by taking union. The rest part is used as short universe. The long universe and short universe have roughly the same number of stocks. See the figures below for the selected universes. Note that the number of stocks in fundamental factors is close to zero at the beginning, hence the long universe has a jump at the beginning.

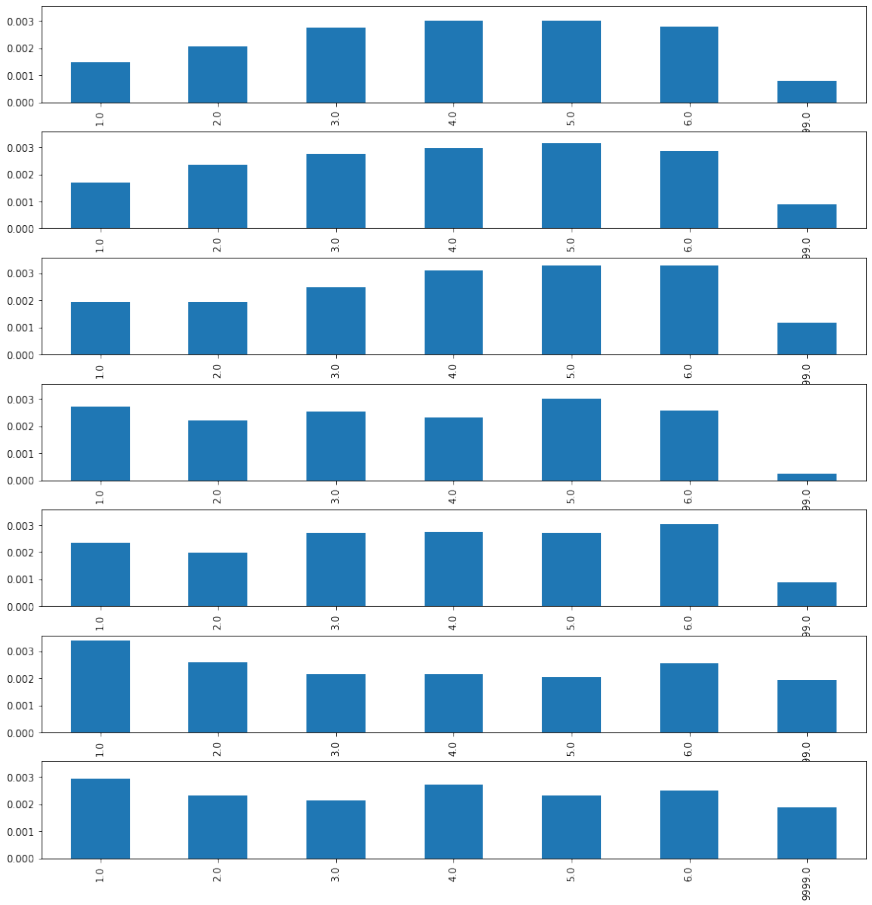

Figure 6: Weekly returns in each group over the in-sample period. Plots are ordered from *data_set_1* to *data_set_7* .

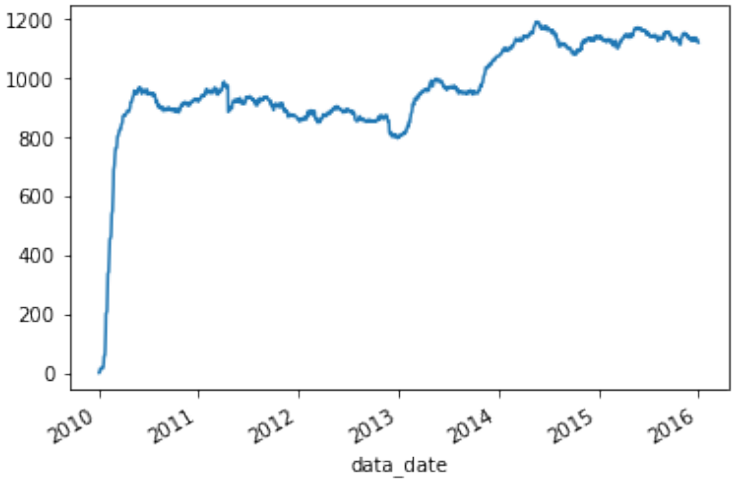

Figure 7: Long universe.

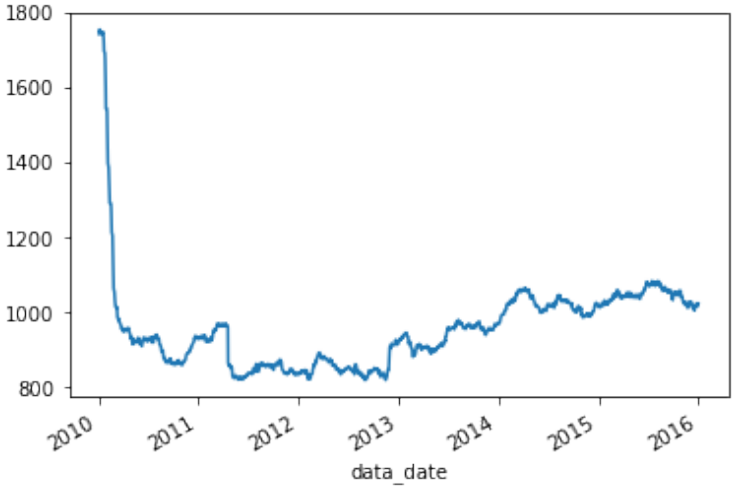

Figure 8: Short universe.

4.2 Layer 2

After constructing the long universe and the short universe. I constraint the last four factors over these universes and redo the backtesting. Since the universe is cut by half, the number of groups is also cut by half. Hence, I use 20 groups for *data_set_8* and *data_set_9*, and 30 groups for *data_set_10* and *data_set_11*. I also compared the obtained backtesting results with previous results which use whole trading universe. I can see the model performance is indeed improved. The turnover is also reduced. I have included some plots below. For details, see the Jupyter notebook.

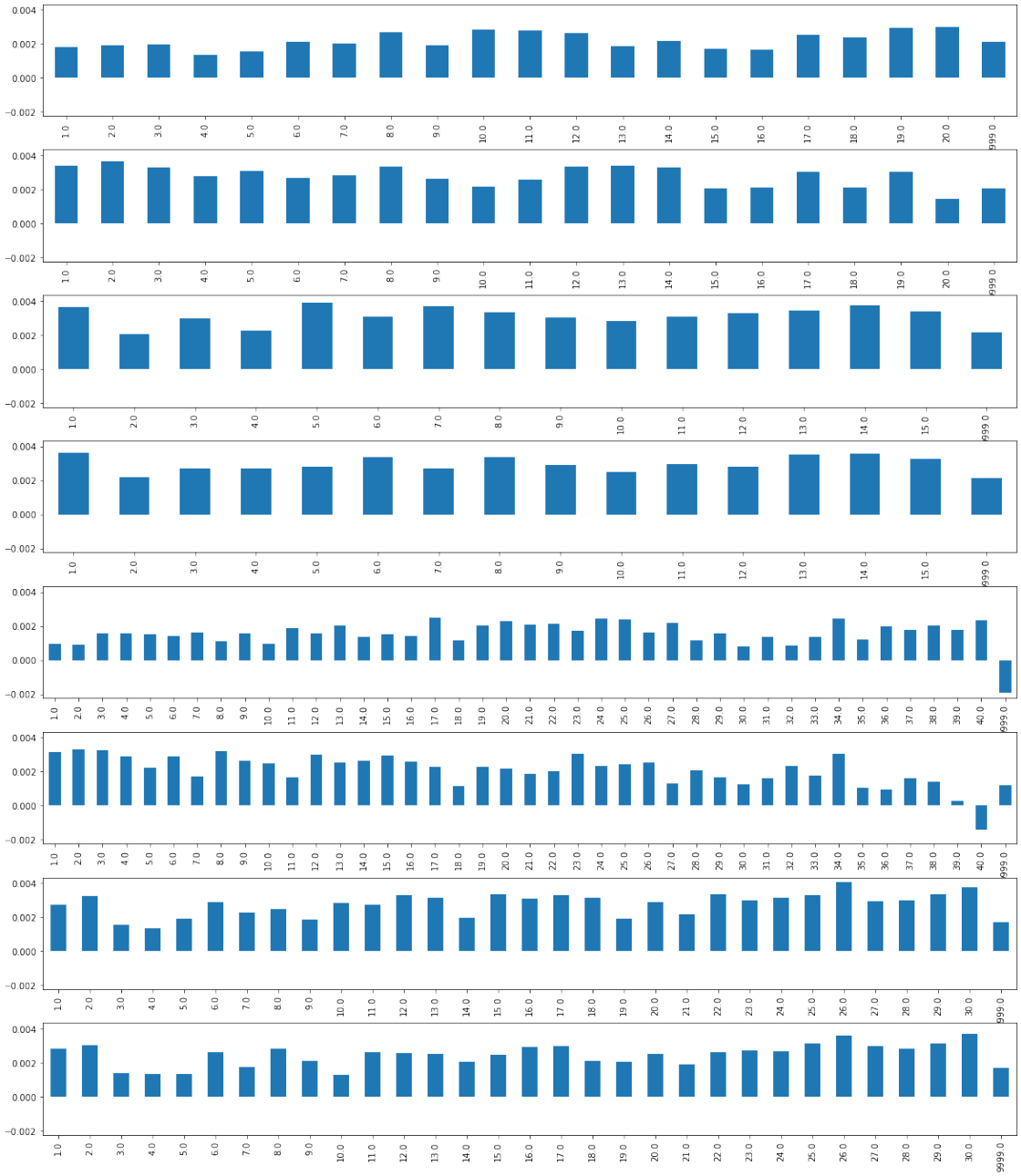

Figure 9: Weekly returns in each group over the in-sample period. Plots are ordered from *data_set_8* to *data_set_11* . The first four plots are in the long universe. The last four plots are in the whole trading universe.

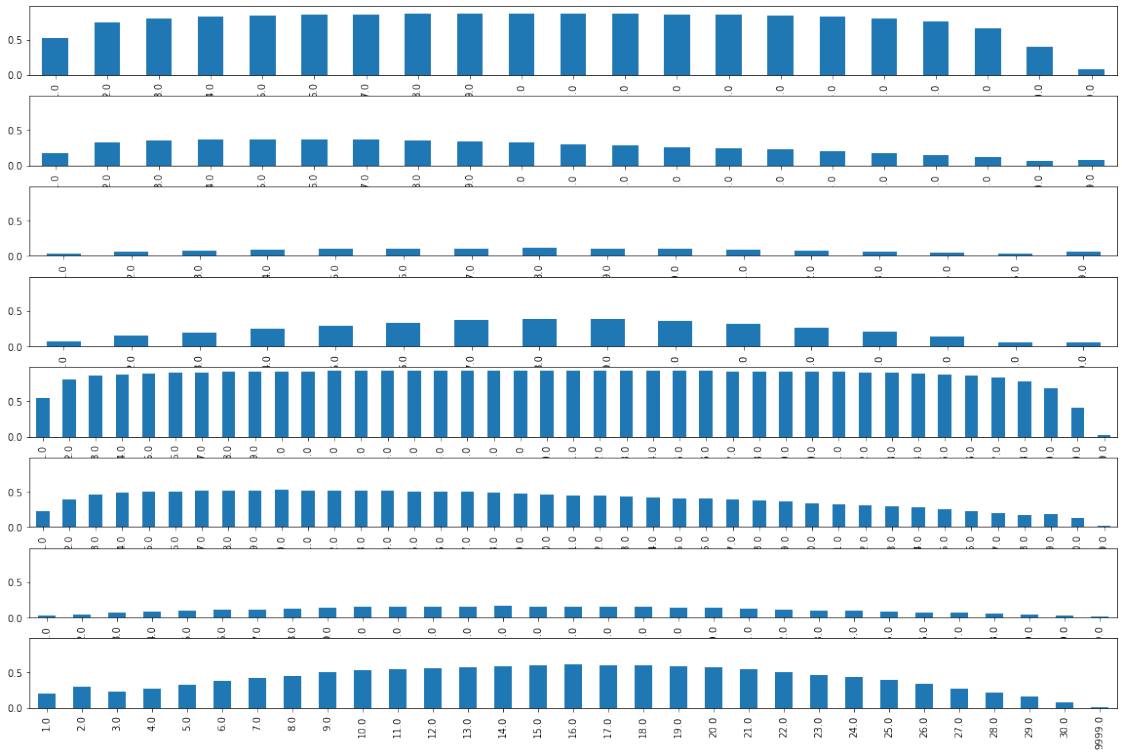

Figure 10: Average daily turnovers in each group over the in-sample period. Plots are ordered from *data_set_8* to *data_set_11* . The first four plots are in the long universe. The last four plots are in the whole trading universe.

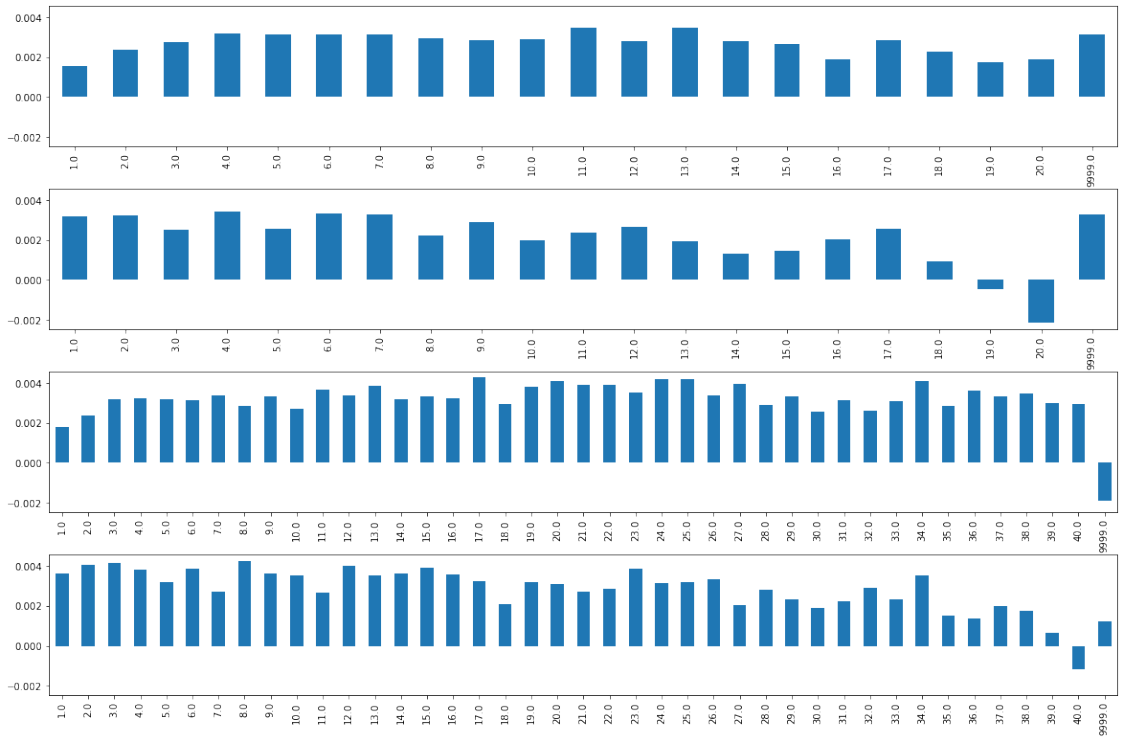

Figure 11: Weekly returns in each group over the in-sample period. Plots are ordered from *data_set_8* to *data_set_9* . The first four plots are in the short universe. The last four plots are in the whole trading universe.

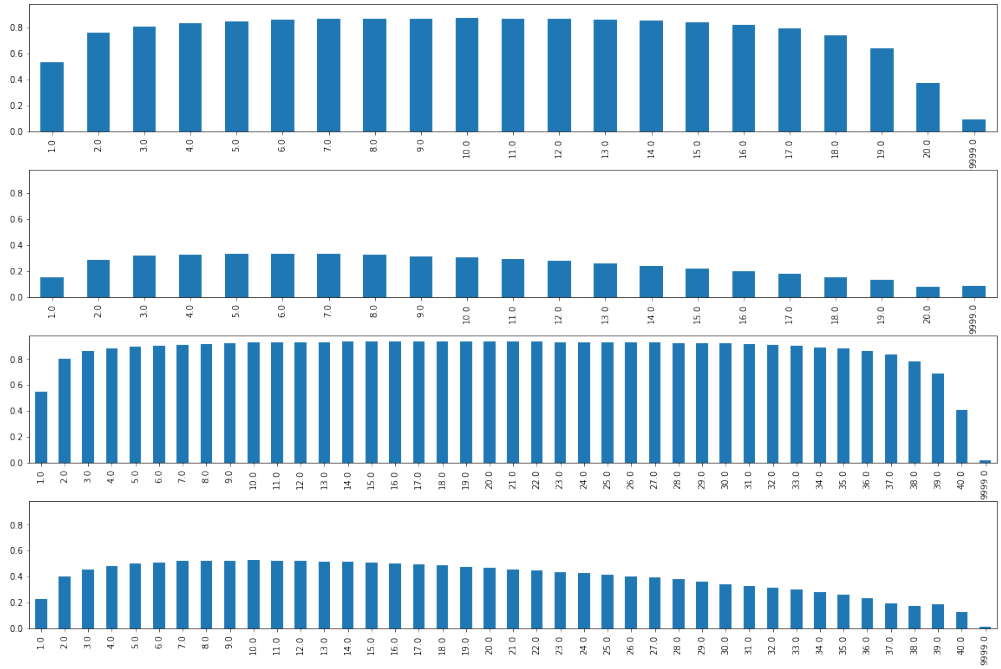

Figure 12: Average daily turnovers in each group over the in-sample period. Plots are ordered from *data_set_8* to *data_set_9* . The first four plots are in the short universe. The last four plots are in the whole trading universe.

4.3 Select long-short models

Finally I select four models two long and two short) with good performance in return and Sharpe ratio to build our model. Note that the return of group 9999 in *data_set_8* is no longer negative under short universe. Hence I also add the original one from whole trading universe back into our final portfolio construction.

L1. Long group 1 in *data_set_10*, under long universe;

L2. Long group 1 in *data_set_11*, under long universe;

S1. Short group 9999 in *data_set_8*, under whole trading universe;

S2. Short group 20 in *data_set_9*, under short universe.

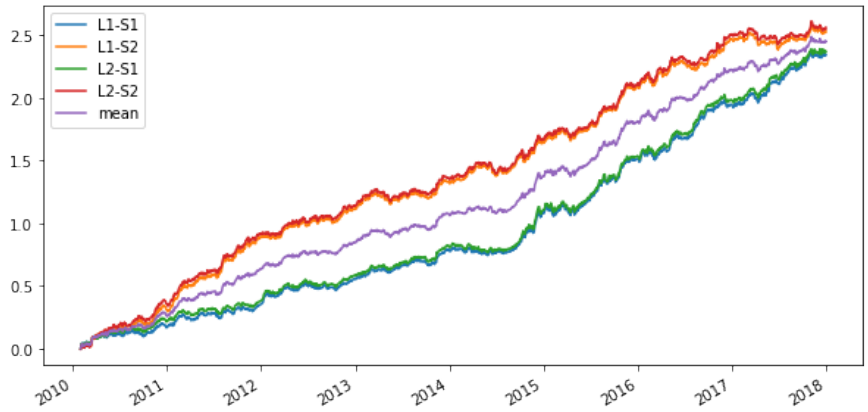

Figure 13: NAVs of four long-short models. X-Y means long X and short Y. "mean" is the average of all four models.

5 Backtest the final model

Next I perform a more accurate backtesting. This backtesting model also allows us to control turnover and perform mean-variance optimization (MVO) to further enhance the models. I tried to control the turnover and conduct MVO at stock level, and unfortunately they did not provide improvements. Probably this is because the turnover is not high even without control, and it is too noisy to perform MVO at stock level due to estimation errors and noise in the data. It is still possible to boot the model performance by fine tuning the parameters but this may lead to over fitting.

For simplicity, I weighted the four models selected above equally. The parameters I selected for each long or short model are given below. I used equal weights with simple backtesting method without turnover control. The target total position of each model is 0.5. Trading costs are selected as 0.0001 for both buy and sell actions.

```
config = {
    'target': 0.5,
    'weighting_method': 'EW', #'MVO', #'EW',
    'backtesting_method': 'vanilla', #'turnover', #'vanilla', #'turnover',
    'begin_date': '2010-01-01',
    'end_date': '2018-01-01',
    'trading_cost': (-cost_buy, -cost_sell),
}
```

The backtesting results with detailed summary are given below. Note that the trading cost does not have a large impact on the model performance.

Performance scores over the whole backtesting period is given below. The suffix "_cost" means the score is calculated with trading cost included.

Sharpe	Sharpe_cost	Sortino	Sortino_cost	pnl	pnl_cost
2.970892	2.921986	4.738831	4.65958	0.326355	0.320998

volatility	volatility_cost	pos_days	pos_days_cost
0.109178	0.109172	0.586687	0.586687

I also summarize the performance scores by year below.

data_date	Sharpe	Sharpe_cost	Sortino	Sortino_cost	pnl	pnl_cost
2010	2.839873	2.783407	5.273227	5.153001	0.303743	0.297617
2011	4.308907	4.254465	7.454608	7.360199	0.410726	0.405551
2012	2.320390	2.262657	3.297650	3.215063	0.205160	0.200111
2013	2.386734	2.334757	4.668421	4.565583	0.233050	0.228013
2014	3.074348	3.030376	4.550455	4.485163	0.346863	0.341938
2015	3.500512	3.461190	5.371074	5.310705	0.473001	0.467726
2016	3.479294	3.426774	5.648549	5.562491	0.399575	0.393579
2017	1.966238	1.922354	3.073093	3.003931	0.237406	0.232135

data_date	volatility	volatility_cost	pos_days	pos_days_cost
2010	0.106252	0.106207	0.579365	0.579365
2011	0.094856	0.094853	0.607143	0.607143
2012	0.087554	0.087557	0.580000	0.580000
2013	0.096806	0.096804	0.539683	0.539683
2014	0.112174	0.112177	0.615079	0.615079
2015	0.134552	0.134557	0.634921	0.634921
2016	0.114269	0.114271	0.591270	0.591270
2017	0.119724	0.119715	0.545817	0.545817

Plots on the portfolio performance are given below.

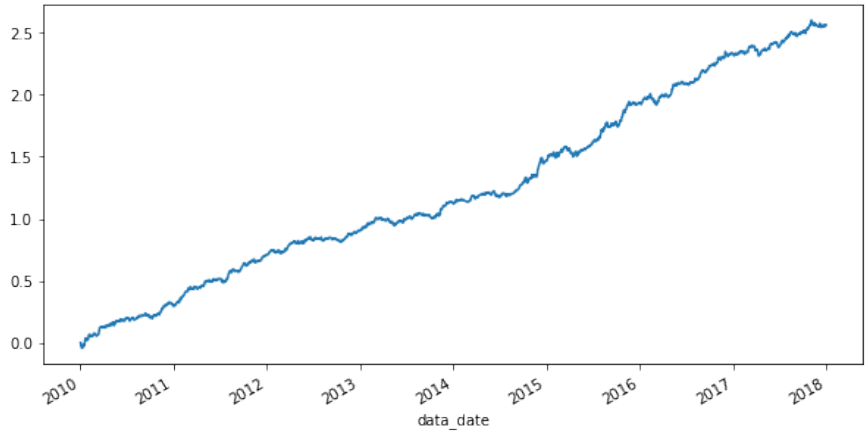

Figure 14: NAV with trading cost using arithmetic daily portfolio return.

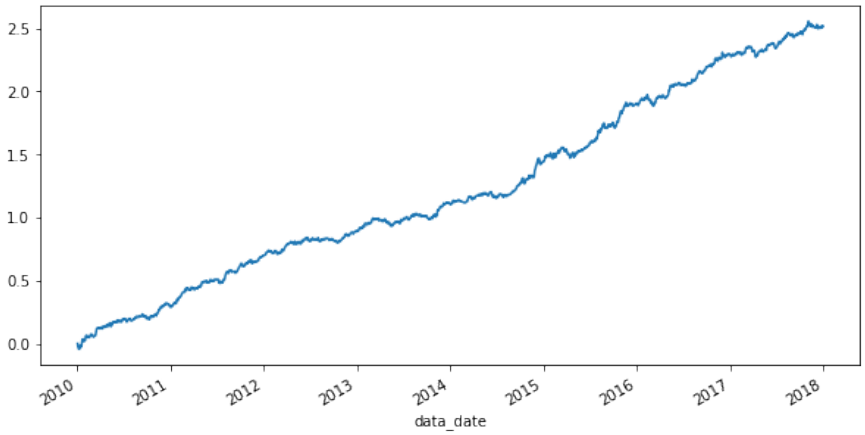

Figure 15: NAV with trading cost using geometric return.

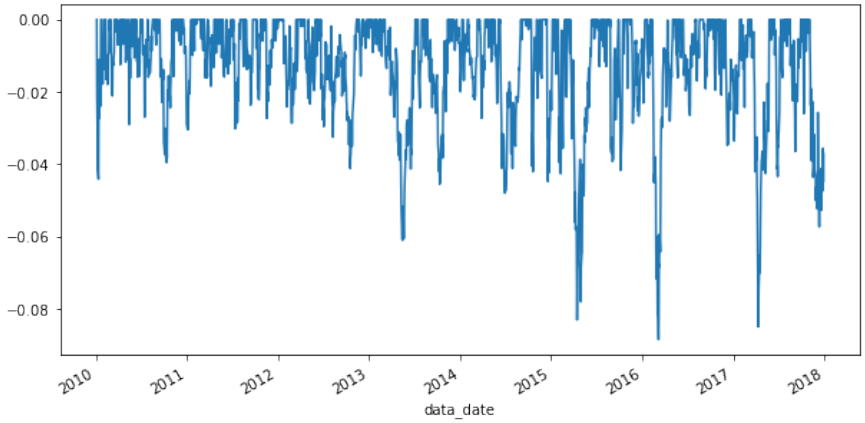

Figure 16: Drawdown with trading cost.

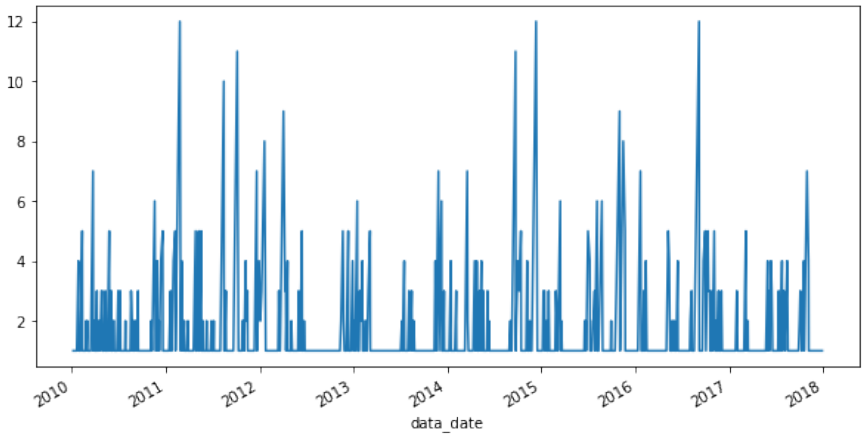

Figure 17: Drawdown period with trading cost.

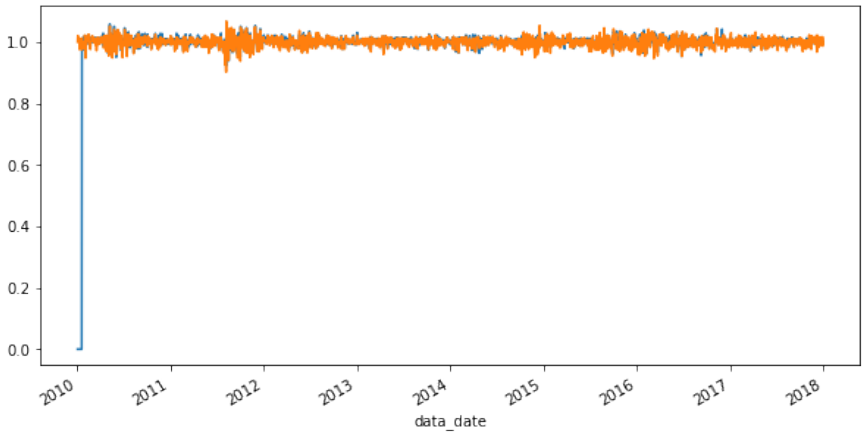

Figure 18: End of day positions. Blue is long side, red is short side.

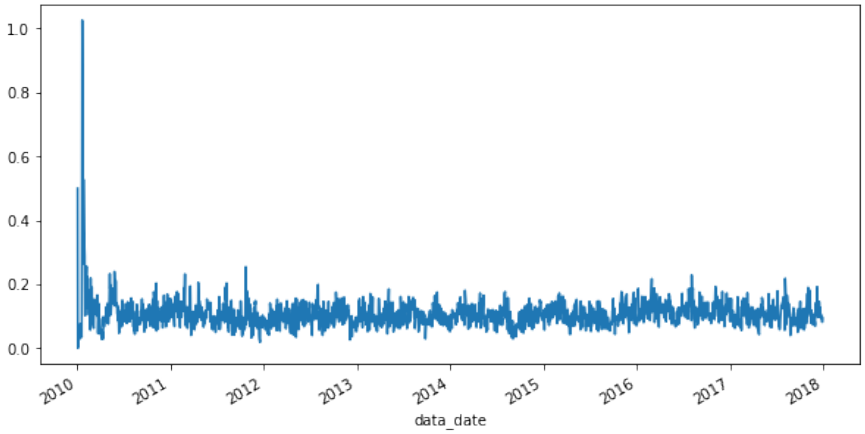

Figure 19: Daily turnover rate.

6 Discussions

The constructed model retains a high Sharpe ratio even during the testing period from 2016-01-01 to 2018-01-01. Nevertheless, I still find some large drawdowns during the backtesting, especially after 2015. Since the portfolios generated by the factors constructed above all have the same drawdown at same period, it is not easy to remove the drawdown by selecting more subportfolios from above analysis into our final portfolio. Another issue is that most of the factors provided are not good at generating short models. In fact, it seems that factor S2 stopped working after 2016, and the number of stocks in S1 are decreasing.

It is possible to reduce the drawdown and volatility by applying risk models or including industry group information in the model. However, I have tried to control the risk by making the produced alpha factors orthogonal to the five risk factors provided in the file *risk_factors.csv* and did not find any meaningful improvement. I also tried to enforce industry neutral in the MVO by controlling the weights among different industry groups. This also did not give any improvement. Another possible solution is to use the whole sample instead of the in-sample period when building the alpha model. However this may lead to overfitting and hence I did not pursue this approach.

One potential improvement is to construct alpha models for each industry and then combine them

together. In this way it is possible to generate factors with uncorrelated drawdowns and hence mitigate them through diversification. This may also generate more short models that can be used to construct the long-short portfolio.