TS: FACTOR GENERATION, COMBINATION AND EVALUATION

A HANDBOOK

Xiwei Zhao

Investment Research Development Department,
Shenzhen Novel Research Investment Co., LTD

August 17, 2023

Part IV: Factor Selection

Part V: Factor Combination

Part I

Stock Data Loader

RETURN STOCK DATA WITH A DATAFRAME TYPE

- Three functions included enable downloading kinds of stock's data: domestic index, domestic industry and abroad index. The type of their returns is fixed.

Figure. sample of stock value dataframe

	date	open	close	high	low	volume	turnover	amplitude	quote change	change amount	turnover rate	symbol
0	2018-01-02	6263.15	6332.23	6332.61	6258.16	73032236	8.815399e+10	1.19	1.30	81.41	0.63	000905
1	2018-01-03	6331.72	6388.25	6391.98	6324.26	83936745	1.034547e+11	1.07	0.88	56.02	0.73	000905
2	2018-01-04	6380.27	6417.54	6418.26	6375.51	80543690	1.029297e+11	0.67	0.46	29.29	0.70	000905
3	2018-01-05	6414.77	6417.25	6435.85	6397.29	81581018	9.764230e+10	0.60	0.00	-0.29	0.71	000905
4	2018-01-08	6413.87	6446.18	6446.79	6393.88	89476928	1.063896e+11	0.82	0.45	28.93	0.77	000905
...	...	...	...	...	...	...	...	...	...	...	...	...
1308	2023-05-25	6000.22	5988.19	6028.64	5926.20	109922832	1.267692e+11	1.71	-0.29	-17.60	0.95	000905

Return calculators

- Calculate to get the training target for factors' design, prediction and back test.

Return	Calculation
Price	stock's close price
daily return rate	$\frac{stock's\ close[1]}{stock's\ close[0]} - 1$
Sharpe ratio	$\frac{(daily\ return\ rate - benchmark).mean()}{(daily\ return\ rate - benchmark).std()}$

Part II

Factor Generation

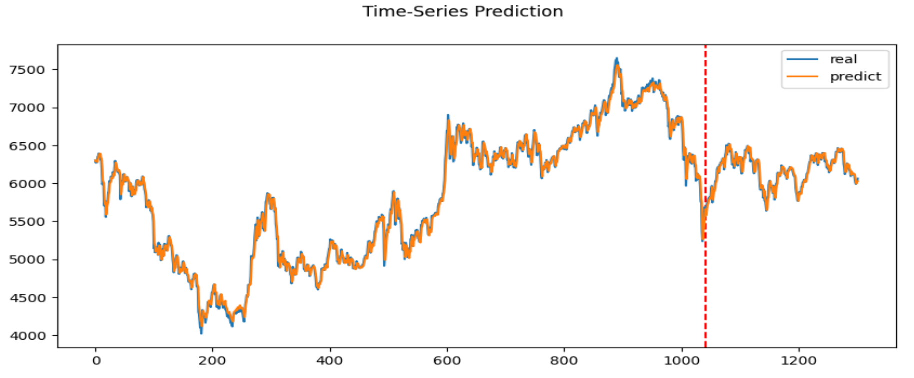
FACTOR GENERATOR

- Factors calculation methods are saved here, named as augur_xxxx.
- Users can decide whether to download all factors at once or calculate single factor to analyze.
  factorloader.all_main(code, start, end)
  factor = factorloader.single_main(code, start, end, factor_num = ’0001’)
- Return type is fixed: a DataFrame with two columns of date and factor value.
- An instruction file to classify factors is also saved.

Models used I: LSTM

- Predict the stock's price and return the predicted daily return rate.
- Note that evaluation tool returns the prediction of price.
  factor = factorloader.single_main(code, start, end, factor_num = '0043', evaluation=True)

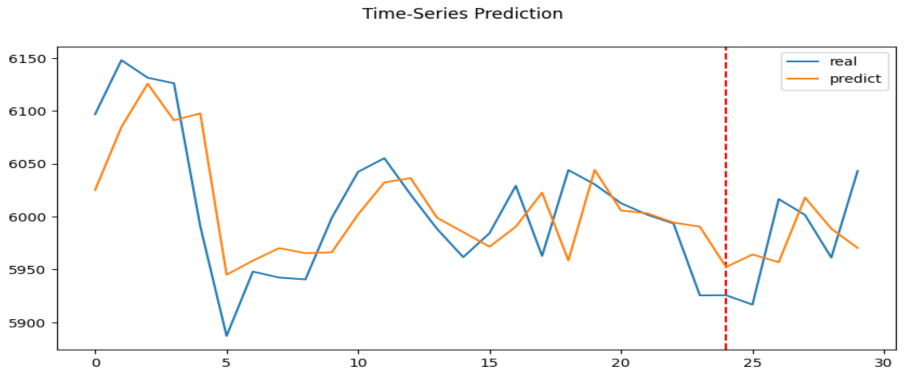
Figure. Prediction of stock's price in a long period



Models used I: LSTM

- Issue: Based on memory method, this model only fits the prediction of stable data.

Figure. Prediction of stock's price in a short period



Models used II: XGBoost

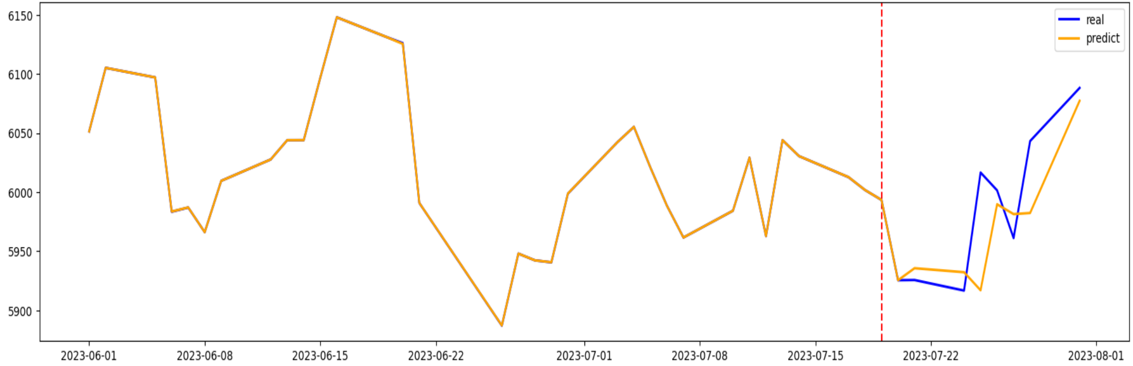
- Predict the stock's price and return the predicted daily return rate.
- Note that evaluation tool returns the prediction of price.
  factor = factorloader.single_main(code, start, end, factor_num = '0044', evaluation=True)

Figure. Prediction of stock's price in a long period

Models used II: XGBoost

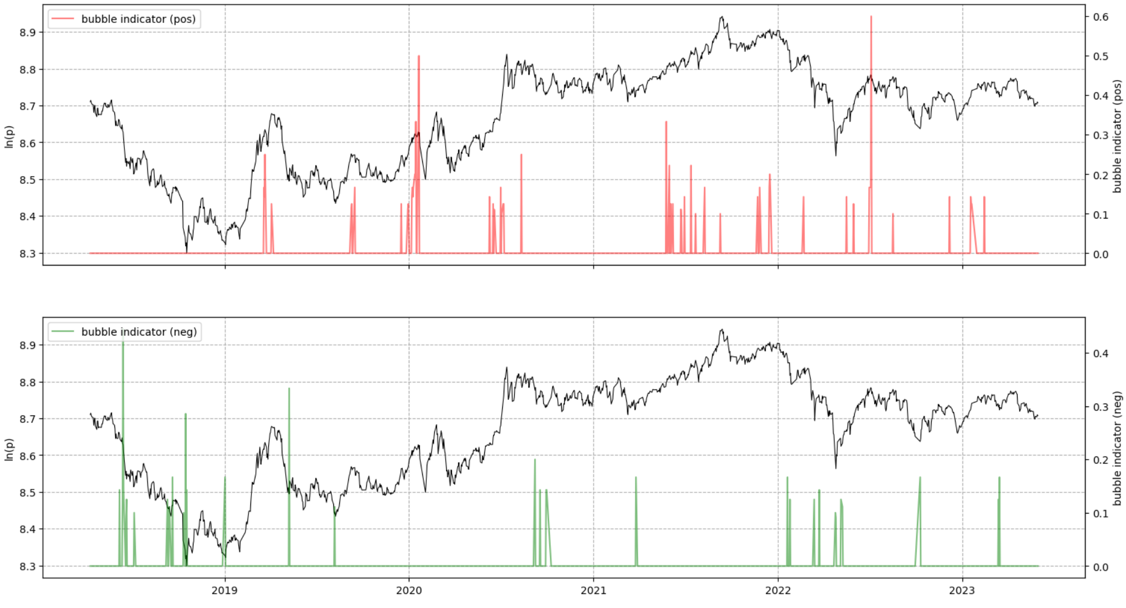
- Issue: in the test part, the model still needs the newest stock's price update, or the error will be large.

Figure. Prediction of stock's price in a short period



Models used III: LPPLS

Figure. Prediction of bubble burst in a long period



MODELS USED III: LPPLS

- Predict financial bubble burst time stamp.
  factor = factorloader.single_main(code, start, end, factor_num = ’0045’, evaluation=True)
- Issue: returns discrete data points, more work needed to design a factor based on these points.
- More information about the algorithm: Click here

Part III

Factor Evaluation

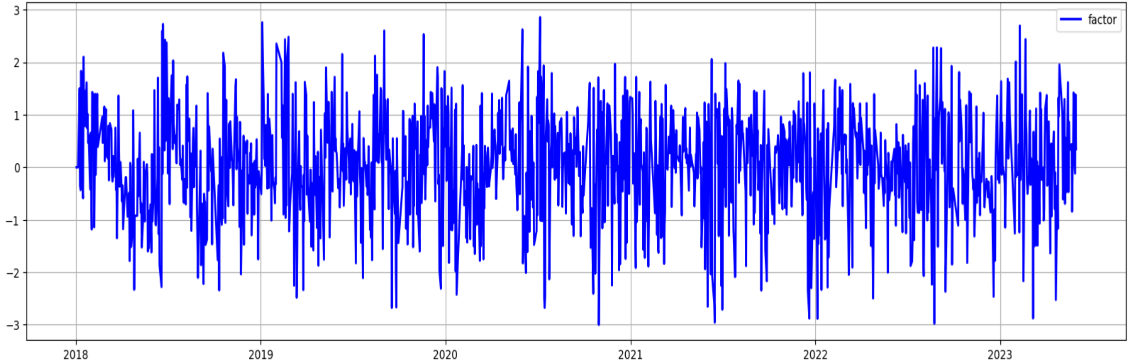
PRE-PROCESSING

- All functions are located in utils. Users can combine different pre-processing tools.
- Avoid using the future data.
  f = utils.normalize(factor['factor'])
  f = utils.standardize(f)
- **Note that all figures below in Part III are the analysis of factor *augur_0002*.**

Observe factor's distribution

- Plot the factor's distribution.

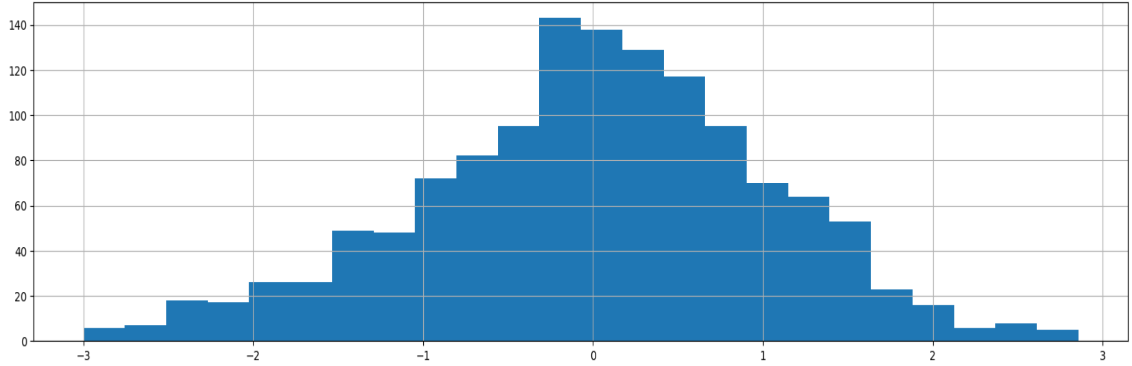
Figure. factor's distribution



Observe factor's distribution

- Plot the factor's distribution.

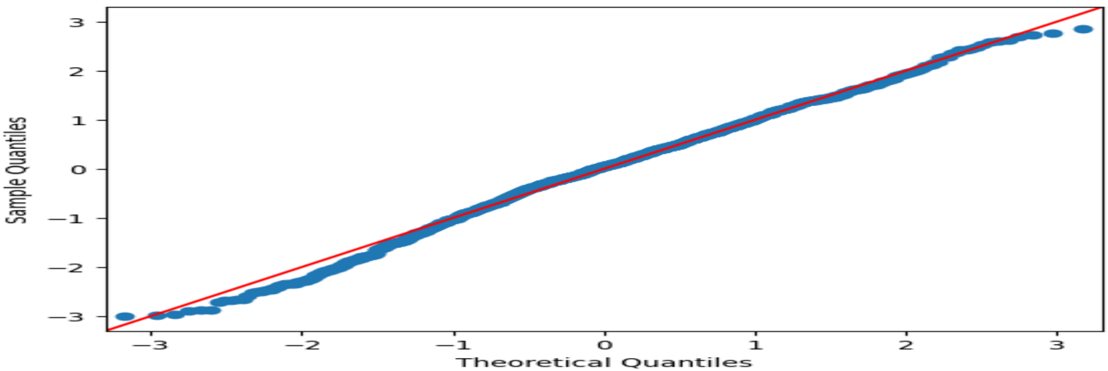
Figure. factor's distribution in a bar chart



Observe factor's distribution

- Compare the factor's distribution and the normal distribution.

Figure. a comparison with normal distribution



Closer to the red line is better, which means factor's distribution is similar to the normal distribution.

IC Test

- Calculate Spearman correlation value between stock's annual return rate and factor.
- Note that this test uses rolling window method.

Figure. IC test

- **The larger value is better**, which means the factor can track the stock's daily return rate effectively.

Granger causality test

- Calculate granger's causality value between stock's annual return rate and factor.
- Note that this test uses rolling window method.

Figure. Granger causality test

The larger value is better, which means the factor can track the stock's daily return rate effectively.

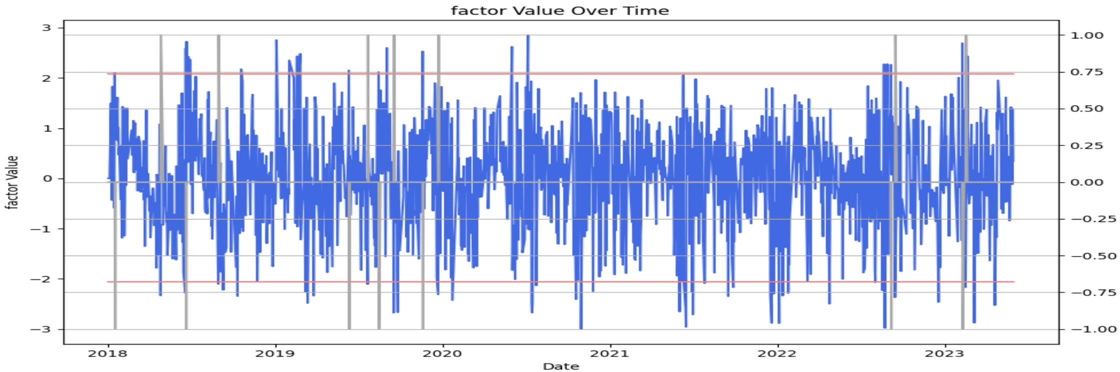
Factor back test

- Calculate two trading bounds automatically: users can determine bounds are constant or varying.
  bound = evaluate.cal_bound(factor, switch='constant')
- Back test main function: users can determine trading signal bounds direction, buy or sell.
  annual_ret = evaluate.backtest_main(data, factor, bound, upperbound = 'buy', plot=True)
- Note that **sell** signal value equals 1, and **buy** signal value equals -1.
- Trading assumptions:
  - Initial portfolio value is 1,000,000. So the first trading signal must be 'buy'.
  - Full position trade at each trading point.
- Plot: Red line is bound, gray line is calculated signal.

FACTOR BACK TEST

- Calculate trading signal (constant):

Figure. Trading signal



FACTOR BACK TEST

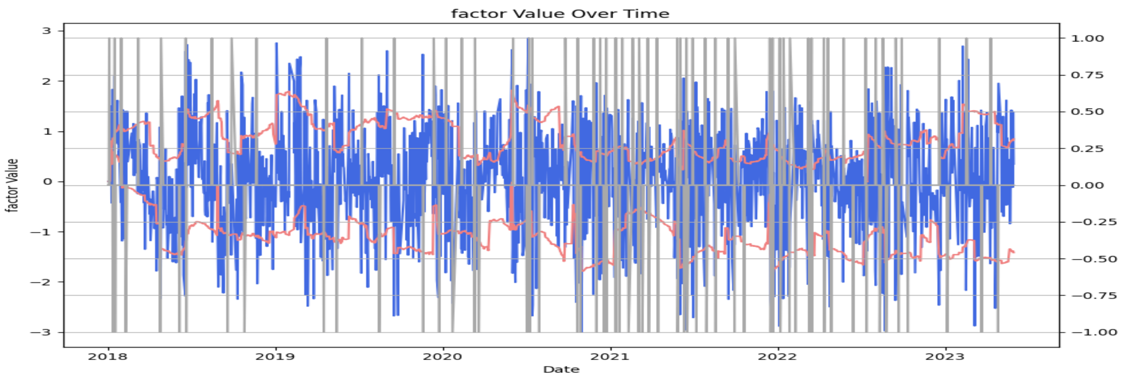
- Portfolio value and market value (constant bounds):

Figure. market value

FACTOR BACK TEST

- Calculate trading signal (varying):

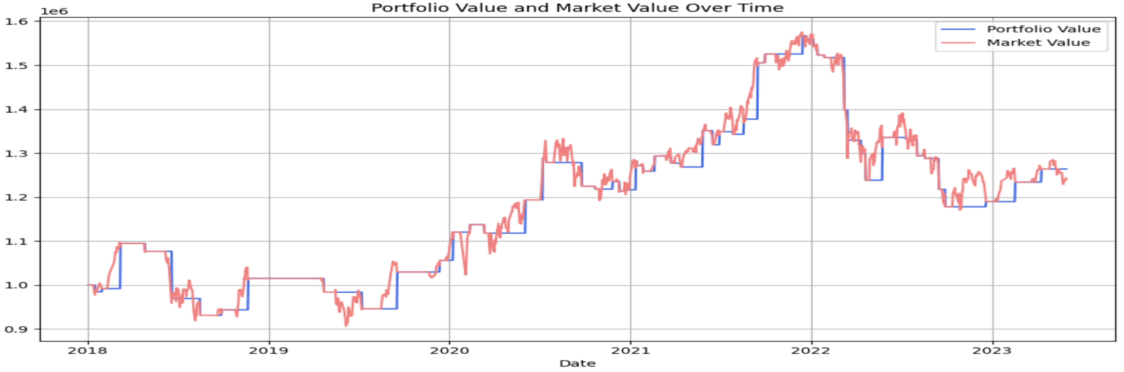
Figure. Trading signal



FACTOR BACK TEST

- Portfolio value and market value (varying bounds):

Figure. market value



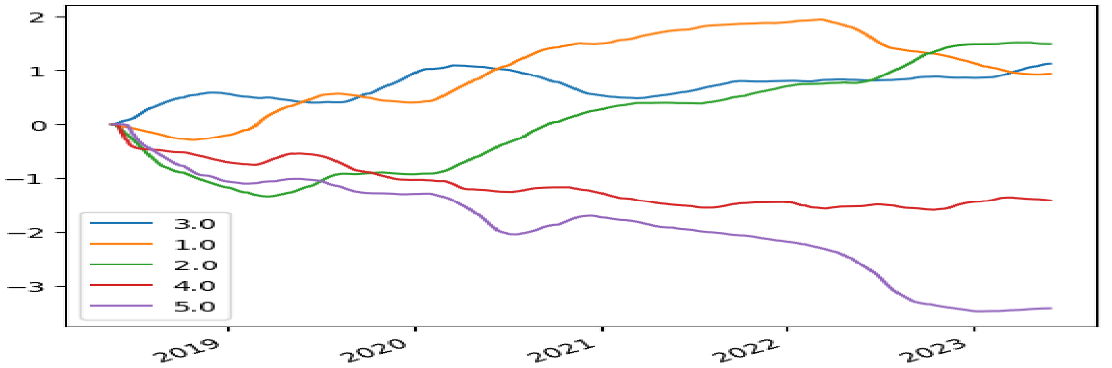
Factor layered test

- Divide factor value at each time point to different groups.
engine = evaluate.LayeredBacktestingEngine(data, factor, window=90, n_group=5, ret_period=1)
- Note that groups division part uses rolling window method to avoid using future data.
- Calculate the cumulative return rate at each group.
cret_group = engine.cal_roll_res_list()
- Using the stock's calculated daily return rate, computed in each group.

FACTOR LAYERED TEST

- Plot the cumulative return rate at each group.

Figure. Layered test result



- **The larger distinction among groups is better**, which means the factor is more effective to find the best trading time points.

Part IV

FACTOR SELECTION

Loading factors pool

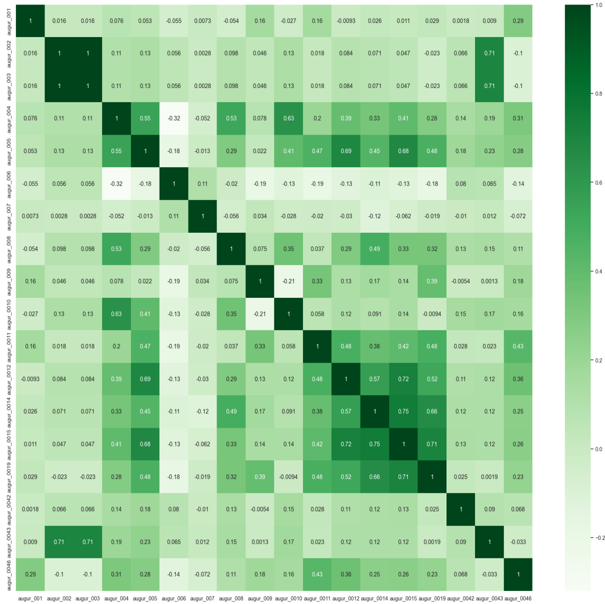
- Calculate all factors at one time, save them to a directory.
- Combine all factors to a DataFrame:

Figure. Factors pool

	augur_001	augur_002	augur_003	augur_004	augur_005	augur_006	augur_007	augur_008	augur_009	augur_0010	augur_0011
0	0.447214	0.145334	0.145480	1.150656	0.842910	-3.271404	0.000000	1.202537	0.795223	0.045925	0.798434
1	0.408248	0.080895	0.081031	1.132225	0.819249	-2.038458	0.000000	1.054304	0.587321	0.208255	1.249165
2	0.377964	0.083581	0.083716	1.003806	0.797493	-1.823935	0.000000	1.047264	0.313255	0.219667	1.470944
3	0.242185	0.567907	0.568069	0.924984	0.777400	-1.547834	0.000000	1.086238	0.246737	0.613225	1.171484
4	-2.087370	-1.040087	-1.040125	0.623622	0.284229	0.512716	0.000000	0.864493	0.857678	-0.279208	0.592403
...	...	...	...	...	...	...	...	...	...	...	...
832	0.729255	0.104708	0.104721	-0.685114	-1.235548	0.452780	0.258886	2.275926	-0.686059	0.221918	2.079226

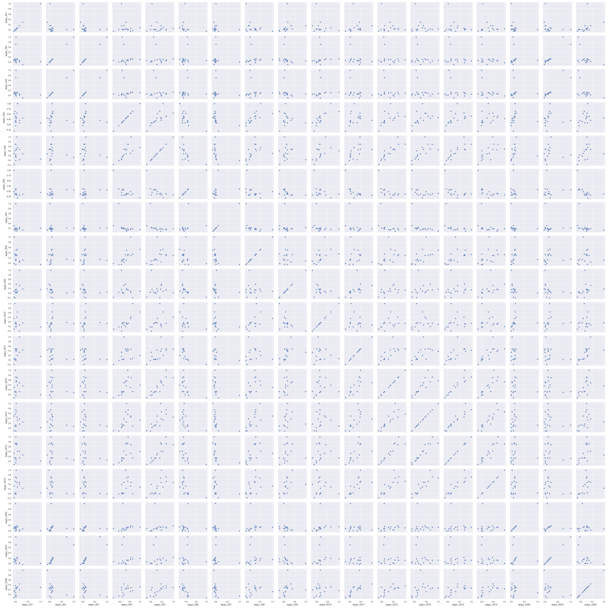
CHECK FACTORS MUTUAL CORRELATION

Figure. factors mutual correlation



CHECK FACTORS MUTUAL CORRELATION

Figure. factors scatter plot



Determine training target

- Use **cal_ret** function to compute the combination training target.
- Return a DataFrame with two columns:

Figure. Traing target

	date	ret
0	2020-02-21	-1052.883483
1	2020-02-24	-631.318707
2	2020-02-25	-507.752439
3	2020-02-26	-501.841853
4	2020-02-27	-530.608643
...	...	...
832	2023-07-25	-1701.815522
833	2023-07-26	-1702.088315

SELECT FACTORS

- Determine training size: a percentage. In the later part, we will only use these training data.
- Selection steps:
  1. Calculate Spearman correlation value between each factor and the training target. (a const, denoted as **ic**)
  2. Calculate the correlation value between the factor and other factors. (a const, denoted as **corr**)
  3. Using the formula below to calculate the selection value:

$$value = e^{-mean(\|corr\|)} \times (1 + ic)^2$$

    - Method: **The larger ic is better, the less mutual correlation value is better.**
  4. Rank all of the factors' selection value, choose the first 10 factors.

Part V

Factor Combination

Calculate combination weights

- Define the loss of the combination model as the mean squared error (MSE) between model outputs and true stock trend values:
$$L(w) = \frac{1}{nT}\sum_{t=1}^{T} ||z_t - y_t||^2$$
- To simplify the calculation of factor combination, we have:
$$L(w) = \frac{1}{n}(1 - 2\sum_{i=1}^{k} w_i \bar{\sigma}_y(f_i) + \sum_{i=1}^{k}\sum_{j=1}^{k} w_i w_j \bar{\sigma}(f_i(X), f_j(X)))$$
where:
$$\bar{\sigma}_y(f) = \bar{\sigma}(f(X), y)$$
$$\sigma(u_t, v_t) = \frac{\sum_{i=1}^{n}(u_{ti} - \bar{u}_t)(v_{ti} - \bar{v}_t)}{\sqrt{\sum_{i=1}^{n}(v_{ti} - \bar{v}_t)^2 \sum_{i=1}^{n}(v_{ti} - \bar{v}_t)^2}}$$
- For convenience, we denote the IC values between two sets of vectors averaged over all trading days as $\bar{\sigma}(u, v) = E_t[\sigma(u_t, v_t)]$.

Calculate combination weights

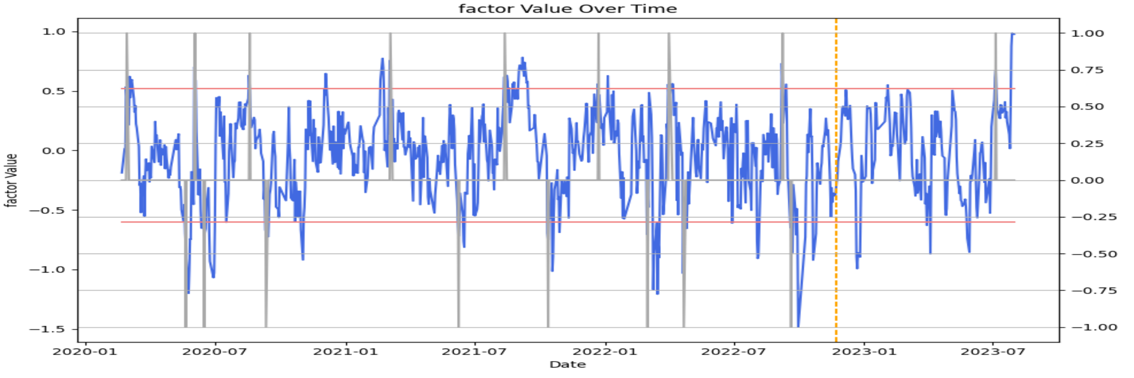
- Optimization method: **torch.optim.Adam**
- Optimization target: minimize the lost function $L(w)$.
- Return an array of weights.
- The factors we have chosen:

Figure. Traing target

	augur_009	augur_007	augur_008	augur_001	augur_0042	augur_0043	augur_0019	augur_0045	augur_0015	augur_006
0	0.795223	0.000000	1.202537	0.447214	0.712781	0.590481	0.491014	1.199383	0.984961	-3.271404
1	0.587321	0.000000	1.054304	0.408248	0.106915	0.483509	0.614854	1.151675	1.102556	-2.038458
2	0.313255	0.000000	1.047264	0.377964	0.634380	0.478431	0.517237	1.109293	1.316811	-1.823935
3	0.246737	0.000000	1.086238	0.242185	0.648014	0.258638	0.193605	1.071314	1.261900	-1.547834
4	0.857678	0.000000	0.864493	-2.087370	0.658354	-0.511247	0.219444	1.024295	1.142449	0.512716
...	...	...	...	...	...	...	...	...	...	...
832	-0.686059	0.258886	2.275926	0.729255	-1.336815	0.087743	1.260418	-1.165022	1.242818	0.452780

BACK TEST

Figure. Back test after factor combination



BACK TEST

Figure. Back test after factor combination

LAYERED TEST

Figure. Layered test after factor combination