Machine Learning Application on Chinese Stock Market: Internship Report of High-frequency Trading

Haoyang Han
Master of Project Management Program
Department of Civil and Environmental Engineering
McCormick School of Engineering and Applied Science
Northwestern University
Evanston, Illinois

October 05, 2018

Proposed table of contents

Abstract

In this paper, author designed a series of experiments to gain familiarity of high frequency trading algorithms and procedures. While many machine learning and deep learning algorithms, such as linear model, decision tree, random forest, GBDT, Deep Neural Network and bi-directional LSTM (bi-LSTM) could gain excellent performance, an ensemble(stacking) model was designed with base models: LightGBM, XGBoost, Linear Regression (LR), k Nearest Neighbor(kNN) and Support Vector Machine (SVM) and second level model: Logistic Regression for final prediction (regression). Meanwhile, this paper also discussed the difference between several data pre-processing procedure. After comparison, we discovered that the combined standardization method of winsorization, batch-normalization and rolling-fitting could achieve best performance.

Index Terms: high frequency trading, machine learning, quantitative analysis, boosted tree models, alpha strategy.

1. Introduction

This paper is completed as a summer internship report in Alpha2Fund as a machine learning quantitative researcher. The full version of this report, sample codes, library and related papers could be found at https://github.com/HaoyangHan/2018SummerIntern. Using history data to predict the future value of each stock is the main mission for this summer internship. To gain success in quantitative industry, designing and implementing an efficient, accurate and robust algorithm for preprocessing, factor mining and trading system is the final target for every quant trader/researcher. Machine learning and its application, with higher accuracy for classification and regression, is gaining attraction by lots of quantitative researchers. In this paper, author gives a comprehensive introduction for how to design a complete trading algorithm from simple to complex which could pass trace-back analysis and earning profit in stock market, including what's the statistical pattern of original dataset, data manipulation process, data standardization method, different fitting scenario and model evaluation method. This paper finally showed a completed, complicated ensemble algorithm which combines several machine learning models together. Due to commercial confidential agreement, results which may result to leak were hidden. This paper also discusses future experiments could be done for algorithms performance improvement.

2. Background

Background

2.1 History of high frequency trading

2.2 Quantitative analysis application in high frequency trading

2.3 Machine learning application in high frequency trading

High frequency trading (HFT), which is also called algorithmic trading or ‘flash’ trading, is getting increasingly popular in Chinese stock trading market. Generally, it’s divided by stock trading and derivatives trading (Lattemann el al. 2012; Menkveld 2013). By developing algorithm for trading and metrics for evaluation, high frequency trading could get higher alpha factor which lead to higher revenue for hedge fund (Tortoriello, R. 2009; Leote el al. 2012). Quantitative analysis is using mathematical, statistical and economical knowledge to transfer real-world problem into mathematical problem then solving them. In high-frequency trading, the detailed quantitative analysis should be abstracting statistical patterns hidden in trading data into predictable model and applying those models into practice.

By designing and analyzing algorithms which enable computer to do regression, classification and clustering, machine learning has been growing rapidly in all fields of study, including data mining, natural language processing, computer vision, recommendation system, vision recognition and voice recognition (Goldberg et al., 1988; Collobert et al., 2008;). In high frequency trading, people use stocking trading data to feed machine learning algorithms for future stock price prediction. Deep learning method is the subset of machine learning. By combining multiple levels of regressor or classifier together in a single model, deep learning algorithms are trying to make more accurate classification or regression. In practice, researchers and engineering would develop deep neural network (DNN), convolutional neural network (CNN), recurrent neural network (RNN), generative adversarial nets (GAN), autoencoder (AE) and restricted Boltzmann machine (RBM) to solve real-world problems (Hinton et al., 2012; Girshick, 2015; Krizhevsky et al., 2012; Mikolov et al., 2010).

(Hyper-parameter) Since the dataset for training machine learning models are getting increasingly huge, many machine learning and deep learning algorithms would take long time to get trained from datasets. Meanwhile, hyperparameters, such as learning rate, iteration time and cost which should be considered before training. To improve efficiency and accuracy of model, this experiment used grid search to search for optimized hyperparameter to achieve the model's best time and accuracy performance.

Optimization method is pretty popular for learning algorithms. By selecting efficient optimizers, training process could be much faster, especially for training large scale datasets. In this paper, different optimizers, such as gradient descent, stochastic gradient descent (SGD), momentum, Nesterov, adagrad and adam are applied for maximum training speed.

3. Experiment, method and data resources

3.1 Computer environment and software version

Since different OS (such as CentOS, Mac OS, Windows, etc.) have totally different operation method, and different version of software (basically python and its packages). Thus, this list of software (and their versions) are designed by myself to make sure all my codes could work smoothly.

For Windows 10:

Putty (for SSH key remote connection);

Ubuntu (for SSH key connection and terminal command);

Jupyter notebook (for visualize scripts I wrote in local computer and small size data test);

Python 3.6.

For CentOS 7.0:

System packages: pip/brew/conda/yum/apt (necessary for python package management, install and upgrade);

Python Environment：

Anaconda (with python3.6)

Pytorch/Cudnn (GPU enabled)

Tensorflow 1.8.0 (GPU enabled)

Opencv3 (data processing for statistical visualization).

Theano

Keras (with Tensorflow)

Sklearn

Visualization packages(pyd3, matplotlib, seaborn).

3.2 Data resources

All original stock data for analysis comes from Chinese secondary stock market. By scrapping every minutes' stock information from Shenzhen Stock Exchange (SSE) trading history. Trading information including 1829 stocks' start value(S), end value(E), high value(H), low value(L), average value(A), and stock volume(V). 1829 points with 6 characteristics are recorded for each point (minute). Specially, we used around 40000 minutes in specific time zone. To further explore the stock rise/fall pattern, factor base is a very useful tools which contains the other quantitative researchers' effort. By transforming original data into clearer statistical distribution, factors are important tool for researchers to earn profit effectively. In this experiment, factors are a bunch of 2D arrays with shape of [40000, 1829]. We should attention that the prediction ability for factors is different, which is result of different factors' design methods, market's growth tendency and how robust our factors are to adjust the market's growth tendency. Started from combining small number of random-selected factors, the experiment gradually increased the total amount of factors used for developing a robust algorithm. The author also discussed different filter's efficiency for selecting factors. Those criteria are Information Correlation, Information Ratio, Sharpe Ratio, Mean Return, Pearson Correlation Coefficient, etc.

3.3 Data pre-processing pipeline

a.) Design Target Variable(y)

To successfully simulate the market growing tendency and grasp stocks with highest alpha, first issue we encounter is we need to define target variable(y) which have strong connection towards real stock pattern. In this experiment, we utilized mid-price for alpha calculation, and the designed relative alpha (alpha minus mean of alpha for each time point). The calculation formula is listed below:

$$Y(i+T,j) = \log(Price(i+T,j)/Price(i,j)) - avg(\log(Price(i+T)/Price(i))) \quad (1)$$

Here T means the timeslot we used to calculate T minute return. In experiment we chose 30 minutes, and we should notice that 30 columns should be deleted from both training dataset and testing dataset to avoid fitting and validating model with future information.

b.) Import factors from factor base.

Since there are huge number of factors in factor base, and total memory used for uploading factors are significant, researchers used binary file and float32 to save data. Relevant function in python should be:

```
x = np.memmap('/opt/hhyang/Data/factors/TSX[100001_1]',dtype = np.float32, shape = (40000,1829))        (2)
```

which used memmap function in numpy library for loading binary file. We should attention that each factor would consume 300M memory. Attention if factor used is too much (experimentally, larger than 150), remote computer would collapse.

c.) Reshape dataset:

After importing factors from factor base, researcher reshapes the 2D array into 1D array, after which we could combine all factors into a larger matrix (7316000 x number of factors). After reshaping, each factor could be considered as an independent variable for prediction.

3.4 Data standardization method

Different

3.5 Model selection criteria

In this experiment, different evaluation criteria to evaluate the performance of model fitting. By splitting the training and testing dataset by designed method. After fitting model by training data, we predict y value of testing data then compare predicted value with true value by following metrics:

a. Mean Squared Error:

Mean Squared Error (MSE) considered the absolute error between true value and expected value into account and then measure the average value of squared errors. Attention that the lower this non-negative value is, the better performance would be (smaller error and higher predicting precision).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{3}$$

b. Root Mean Squared Error:

Root Mean Squared Error (RMSE or RMSD) is similar to MSE, which also measure the difference between predicted values and true values by measuring squared value of MSE. Since the RMSE have direct relation of model precision, it is usually listed and another metric for model performance evaluation. Equation for RMSE is calculated below:

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}. \tag{4}$$

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{T} (x_{1,t} - x_{2,t})^2}{T}}. \tag{5}$$

c. Information Correlation Coefficient:

Information Correlation Coefficient (IC), which is also called Pearson Correlation Coefficient, measure the linear covariance between two variables. Clearly that higher the absolute value of IC is, the higher correlation between two variables exist. Since those variables are predicted testing data and true testing data value, the value should be as close to 1 as possible. Attention that in realistic prediction, any value larger than 0 should have earning ability, but value greater than 0.1 should be robust and ideal. Here we list specific formula for calculating information correlation:

$$\mathrm{IC_i} = \rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{6}$$

$$\mathrm{IC_s} = \frac{\sum_{i=1}^{n} (IC_i)}{n} \tag{7}$$

$$\mathrm{IC} = \frac{\mathrm{mean}\ (IC_s)}{\mathrm{std}\ (IC_s)} \tag{8}$$

d. Information Ratio:

Information Ratio (IR) measures the adjusted return considering about market average risk. It’s defined by the deviation between active return and active risk, but in practice we will simply calculate the squared value of information correlation coefficient to get IR value. As a model evaluation metric in this series of experiment, a higher IR value would usually suggest a higher return rate for whole portfolio. Calculation equations for both regular method and practice method are listed below:

$$IR = \frac{E[R_p - R_b]}{\sigma} = \frac{\alpha}{\omega} = \frac{E[R_p - R_b]}{\sqrt{\mathrm{var}[R_p - R_b]}} \tag{9}$$

In which:

Rp: the portfolio return,

Rb: the benchmark return,

ω=σ: the standard deviation of the active return.

$$\mathrm{IR} = \sqrt{\mathrm{IC}} \tag{10}$$

e. Sharpe Ratio:

Sharpe ratio, which is also called reward-to-variability ratio, measures the performance of investment for specific portfolio. The higher the Sharpe Ratio is, the better performance would be. Here is the formula of calculating sharpe ratio:

$$S_a = \frac{E[R_a - R_b]}{\sigma_a} = \frac{E[R_a - R_b]}{\sqrt{\mathrm{var}[R_a - R_b]}} \tag{11}$$

3.6 Grid search for hyper-parameters

Since there are so many hyper-parameters in different ML models, such as punish efficient beta in lasso/ridge model, max_leafs/n_works/min_samples_split/max_depth in tree-based model and step_length, batch_size, iteration number, optimizer and dropout in neural networks, and all hyper-parameters could have positive/negative effect to model performance, we need to find the optimized value for each hyper-parameter(Lerman, P. M., 1980; Stoica, P., & Gershman, A. B. 1999). But the real pattern of stock market is not immutable, so we should use efficient algorithm to find optimized hyperparameters ones we decide which model to use. Grid search is the algorithms for finding the optimal value. After doing pre-experiment to get a rough range for individual hyper-parameter, grid search use same training dataset, same evaluation metric (i.e. Information Ratio, time consuming, etc.) and same computing machine to fit same model with different value to specific parameter.

3.7 Ideal outcome of the experiments:

Ideally, this experiment is designed to show a completed designing procedure to build predictive model from scratch, so the experiment output should be shown in chronologically order. In the following section, results of those experiments would be shown, including chart/graph comparison of different performances between multiple models, codes screenshots of explanation and statistical analysis result for individual factor. Results would be shown in following order:

a. Basic metrics comparison for different models with same data-preprocessing, standardization method and fitting method. For data-preprocessing, we combine each factor directly and replace all NaN value by 0. For standardization, we use [-1, 1] scale to normalize each timepoint and each factor. For fitting method, we use 30000 minutes for training model and 10000 minutes for testing. For baseline test, we compare following models: Linear Regression, Lasso Regression, Ridge Regression, Bayesian Regression, Kernel Ridge Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, AdaBoost, GBDT, Bagging Tree and Polynomial Regression. Here we compared time consuming, MSE, RMSE, IC, IR and Sharpe Value for each fitting process.
b. Improvement for data pre-processing pipeline. Following procedures are used to improve the performance (basically value of Information Ratio) for each machine learning models. First, we considered about collinearity existing in factor base, so we use Pearson correlation coefficient to measure the collinearity between factors. Results are shown via

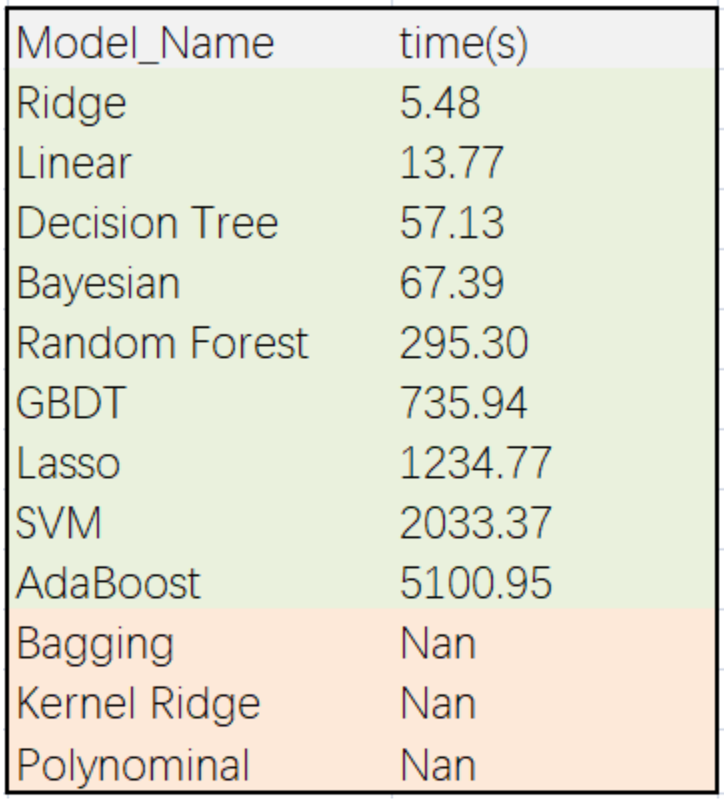
heatmap in seaborn package. Second, we delete several factors with low performance. Even through models have ability to maximize performance by reducing weight for those factors, it's still beneficial to delete those factors to make their weights to be 0. In this experiment we chose best K factors (K = 10, etc.) by evaluating their Information Ratio. Third, we changed fitting methodology by replacing 30000:10000 training/testing ratio to rolling based system, which means we used first 5000 minutes to predict future 5000 minutes' stock value, then use 5001-1000 minutes' value to predict the 10001-15000 minutes' value, until we meet the most recent observation. This strategy could significantly improve the model prediction accuracy and universality of model. Last, we make improvement to standardization method. By introducing a time window W (here we choose W = 500), we select data in this time window (1082*500) to consider the time series. Comparison of experiment results are listed in experiment results session.

c. Grid search result for tree-based model. After session A we know that tree-based model usually could achieve better performance, so we chose random forest, GBDT and XGBoost to do grid search for performance improving. Details are listed in result session.

4. Experiment Result

4.1 Basic metrics comparison.

In this experiment, we use simplest standardization method by standardization without time window, simplest NaN value replacement by replacing them to 0, and simplest fitting method by training 30000 data points and testing on 10000 data points. Below we showed the result of fitting time. Attention that if fitting time is larger than 6000 seconds (100 minutes), we will ignore this model and report Nan value. Below the result showed in ascending order:

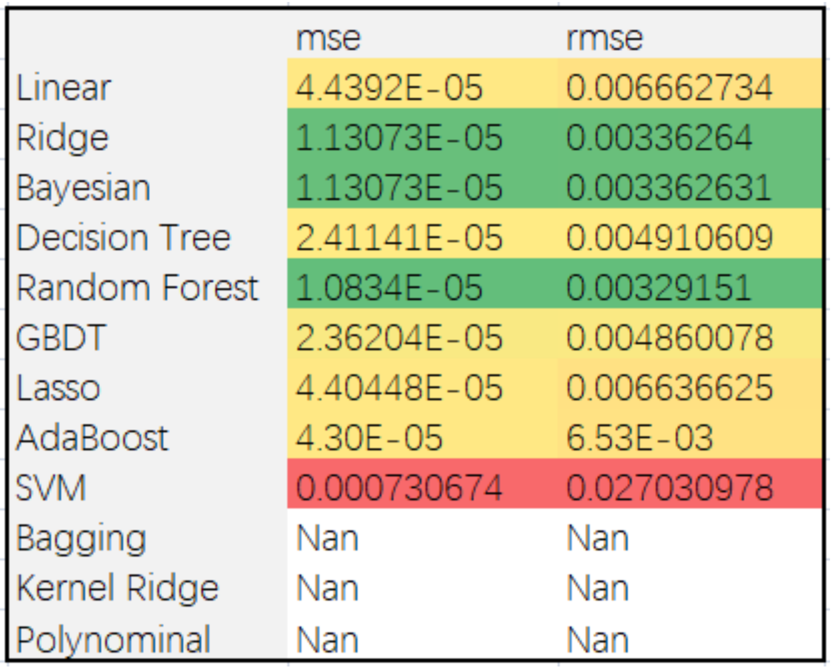


Model_Name	time(s)
Ridge	5.48
Linear	13.77
Decision Tree	57.13
Bayesian	67.39
Random Forest	295.30
GBDT	735.94
Lasso	1234.77
SVM	2033.37
AdaBoost	5100.95
Bagging	Nan
Kernel Ridge	Nan
Polynominal	Nan

Chart 1：Time Consuming Result for Machine Learning Models(Green: useable; Red: Error Reported)

Attention that even though the fitting time varies from each fitting iteration, it’s relevant position for time consuming would not change. We could see clearly that linear regression, ridge regression and decision tree has the most efficiency, while ensemble method (bagging, adaboost) and kernel ridge regression (KRR) have too much time complexity and cost too much computing recourses, so they are abandoned from experiment.

Since precision is one of the key indicators for model performance, we listed MSE and RMSE below for comparison. The result is listed below:

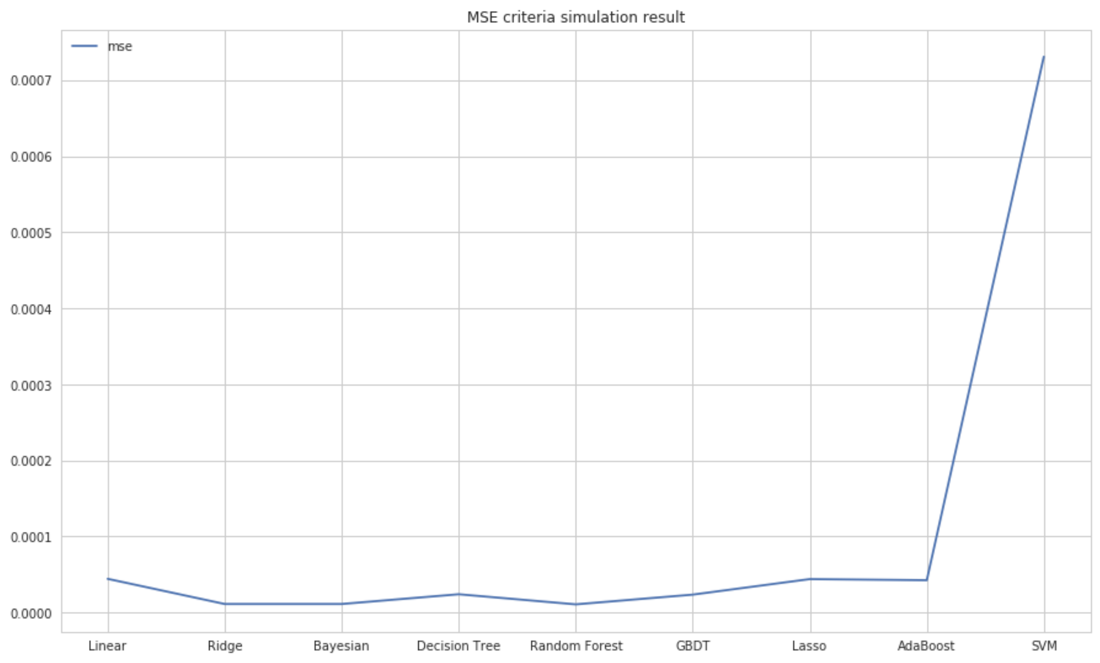


	mse	rmse
Linear	4.4392E-05	0.006662734
Ridge	1.13073E-05	0.00336264
Bayesian	1.13073E-05	0.003362631
Decision Tree	2.41141E-05	0.004910609
Random Forest	1.0834E-05	0.00329151
GBDT	2.36204E-05	0.004860078
Lasso	4.40448E-05	0.006636625
AdaBoost	4.30E-05	6.53E-03
SVM	0.000730674	0.027030978
Bagging	Nan	Nan
Kernel Ridge	Nan	Nan
Polynominal	Nan	Nan

Chart 2 ： MSE and RMSE result for machine learning models(Green: accurate; Red: inaccurate)

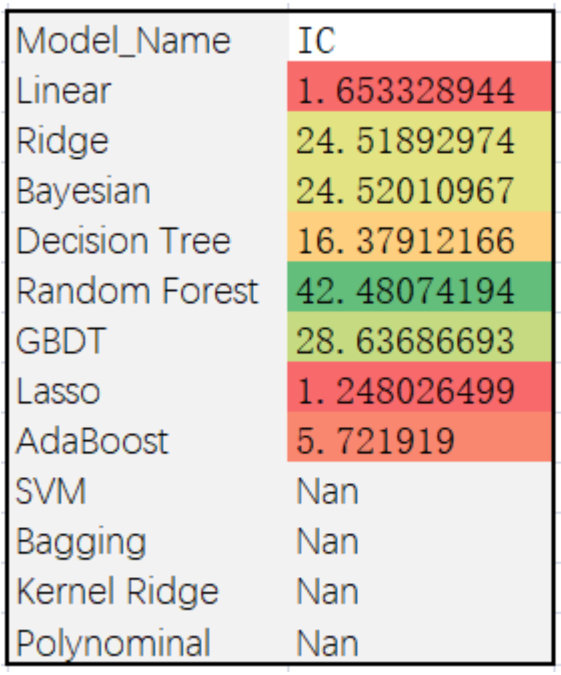
Here we used heatmap in Excel to show comparison result. Green background shows a better performance and red one shows a worse performance. Even through linear model could achieve good performance in lots of scenario, tree-based models still have the possibility of perform better: such as random forest in this case. Attention that since our target is predicting the best alpha, MSE and RMSE value consider about price make cause inaccuracy. In this experiment, random forest achieved best performance, followed by ridge regression (or Bayesian ridge regression- same theory with different fitting process), are the most accurate models according to comparison. Here support vector machine has worst performance, which also perform bad in time comparison and IC comparison. Thus, we didn’t select SVM for further comparison.

Below we made line plot for different models’ MSE values. This is more intuitive. Certainly that we should pay more attention to linear models and tree models.



Graph1 : MSE result for machine learning models.

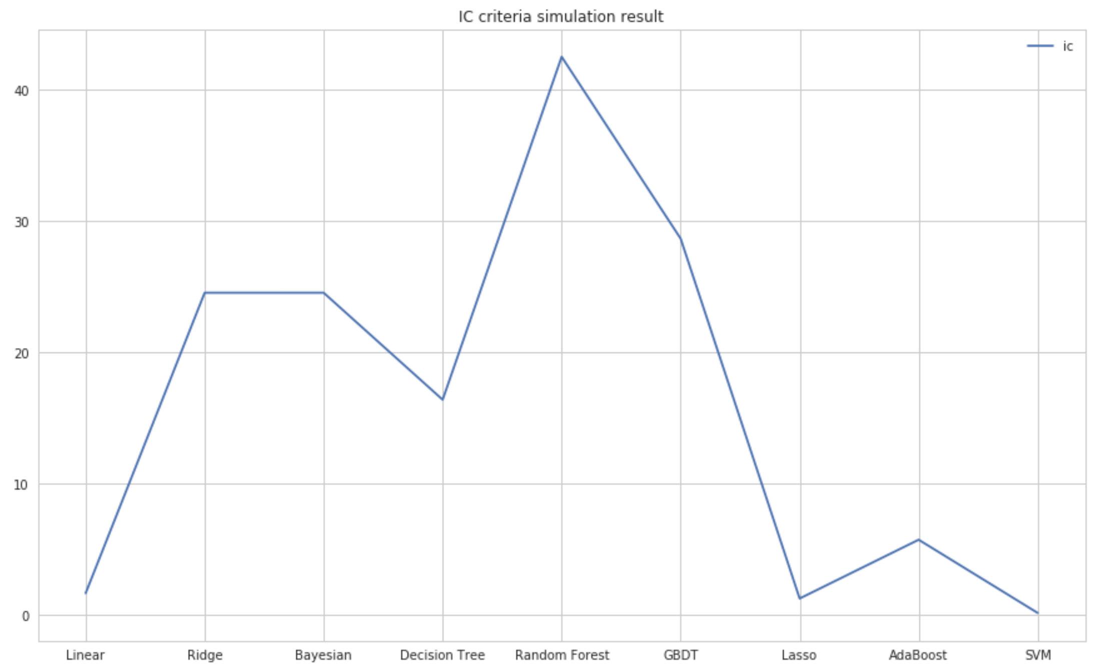
Information Correlation (IC) and Information Ratio (IR) have very direct correlation toward revenue we could earn for each portfolio. So, it's really important to see how strong the model's ability is to simulate best IC value. The comparison result is listed below. Random forest, without hesitation, achieved best performance, which is 90% higher compared to highest non-tree-based model (Ridge/BRR model). Followed by GBDT, we could make positive assumption that tree-based models should have better performance compared to non-tree-based models. Attention that while ridge regression (l2 regularization) have well performance, lasso regression (l1 regularization) performs extremely bad, even worse than normal linear regression. We should make a guess of whether sparse solution is not suitable for financial dataset prediction. Besides, we should understand that all the tree-based models have potential to improve. We didn't adjust hyperparameters to prevent overfitting and increase efficiency. Simply increase the number of CPU cores needed would reduce 95% of the time consuming.



Model_Name	IC
Linear	1. 653328944
Ridge	24. 51892974
Bayesian	24. 52010967
Decision Tree	16. 37912166
Random Forest	42. 48074194
GBDT	28. 63686693
Lasso	1. 248026499
AdaBoost	5. 721919
SVM	Nan
Bagging	Nan
Kernel Ridge	Nan
Polynominal	Nan

Chart 3：Information correlation result for machine learning models

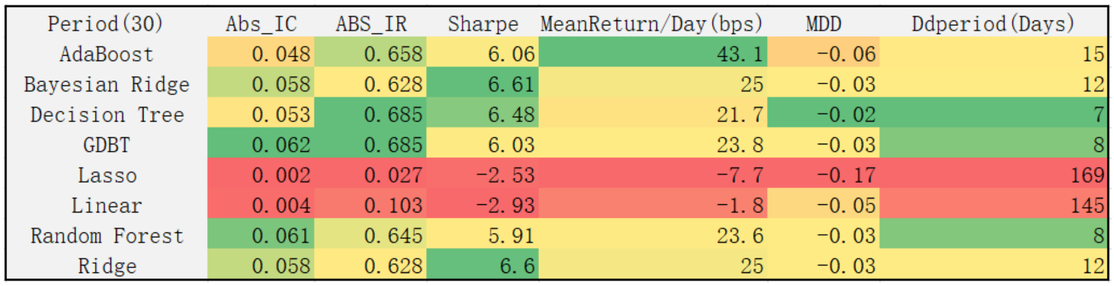
Below we made line plot for information correlation.



Graph 2：Line chart of Information Correlation for machine learning models

Above criteria is designed and computed by experiment designer, while there already exist a standard statistical analysis script to calculate those indicators. Simply by putting predicted value (which is predicted by given model), IC, IR, Sharpe value, mean return, maximum drawdown (MDD) and DD period would be calculated automatically. Mean return measure the total amount we could earn by standard value for this portfolio, so higher the MR is, better portfolio should be. MDD and DD period measure the maximum drawback a portfolio could be and how long does it exist, so higher MDD and lower DD period could minimize the loss portfolios could have. Detailed analysis for statistical analysis result is listed in section 4.2.

Below is the statistical analysis result for combined factors. Red highlights show bad performance for specific evaluation metric and green show good performance. We could see clearly that GBDT, decision tree, Adaboost and random forest perform great overall and Ridge (Bayesian Ridge) is the best model in linear models. Meanwhile, linear regression and lasso regression perform worst, which is the same result compared to original analysis.



Period(30)	Abs_IC	ABS_IR	Sharpe	MeanReturn/Day(bps)	MDD	Ddperiod(Days)
AdaBoost	0.048	0.658	6.06	43.1	-0.06	15
Bayesian Ridge	0.058	0.628	6.61	25	-0.03	12
Decision Tree	0.053	0.685	6.48	21.7	-0.02	7
GDBT	0.062	0.685	6.03	23.8	-0.03	8
Lasso	0.002	0.027	-2.53	-7.7	-0.17	169
Linear	0.004	0.103	-2.93	-1.8	-0.05	145
Random Forest	0.061	0.645	5.91	23.6	-0.03	8
Ridge	0.058	0.628	6.6	25	-0.03	12

Chart 4. Statistical Analysis result for combined factors.

After basic metrics comparison, some conclusions could be drawn for building further algorithms:

a. Tree based models usually have better performance;

b. Regularization method is useful in many cases;

c. Normalization, nan-value processing, collinearity filtering and re-structure fitting methodology could improve final result.

So, in next session we will apply those conclusions in future experiments to see what kind of improvement could models achieve after those adjustments.

4.2 Improving model performance after feature engineering

Before we start to see results for feature engineering (normalization, nan-value processing, collinearity filtering and re-structure fitting methodology), let's explain details of statistical analysis result (Chart 5,6 and 7). First part of the analysis have a concentration on looking at the structure of a factor itself. In those analysis result, factors are divided by 10 quantiles and ret_x means x minutes return.

Chart 5 is a very classical bad-performance factor. Its alpha factor has negative revenue return, because left upper chart is completely mess with no prediction ability; right upper chart shows inverse prediction as well. Meanwhile, cumulative return chart shows similar cumulative results for return of 10, 20 and 30 minutes, which ideally should be ret30 = 3*ret10 and ret20 = 2*ret10. Thus, this factor should not be considered in any scenario.

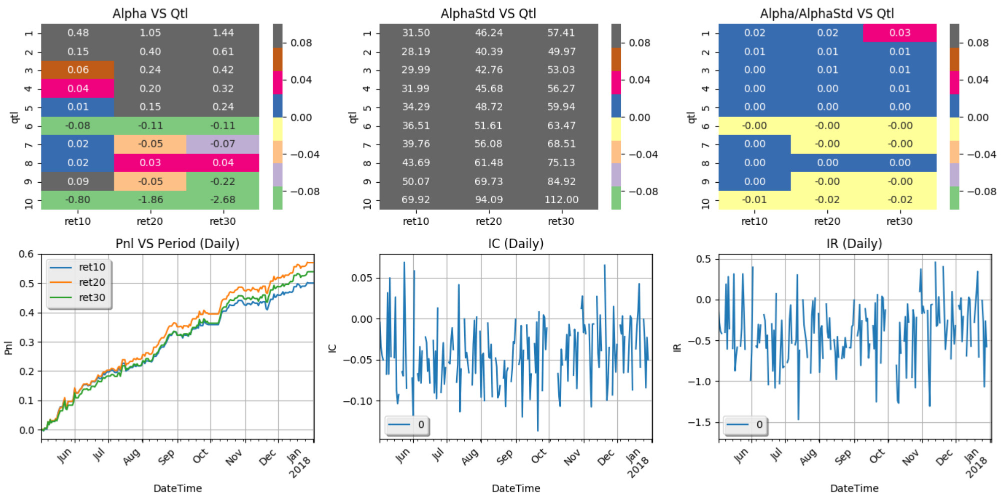


Chart 5：Factor 43-1 Analysis result(example of bad performance factor)

Chart 6 is a very classical normal-performance factor. Its alpha factor has small positive revenue return. Even though left upper chart is completely mess with no prediction ability and right upper chart shows inverse prediction as well, cumulative return chart shows positive feedback for return of 10, 20 and 30 minutes. This factor should have at least some prediction ability in some time points. Thus, this factor could be select for model fitting- if it's information correlation is high enough.

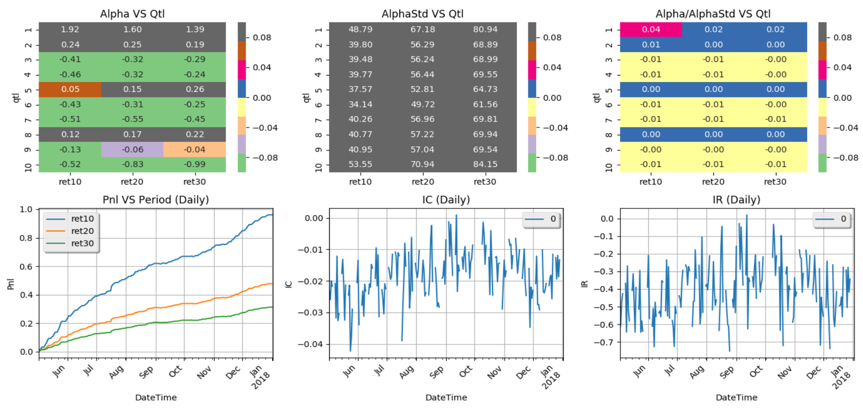


Chart 6：Factor 02-1 Analysis result(example of normal performance factor)

Chart 7 shows a really perfect factor (factor 41-1). A very regular and ordered alpha chart, structured relevant alpha chart and perfect return to time chart indicate that this factor must forecast the market accurately. Absolutely every model should pick up this factor without hesitation.

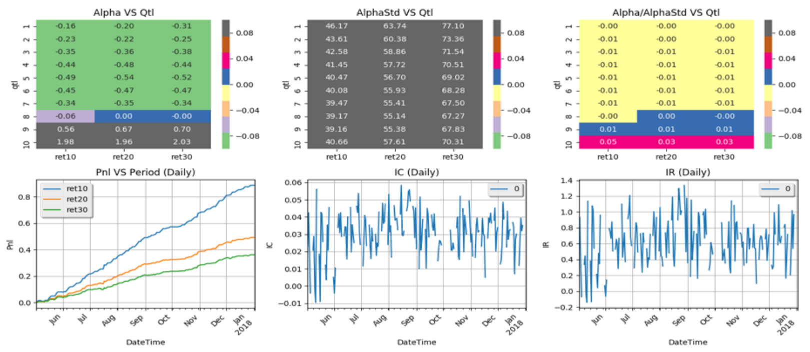
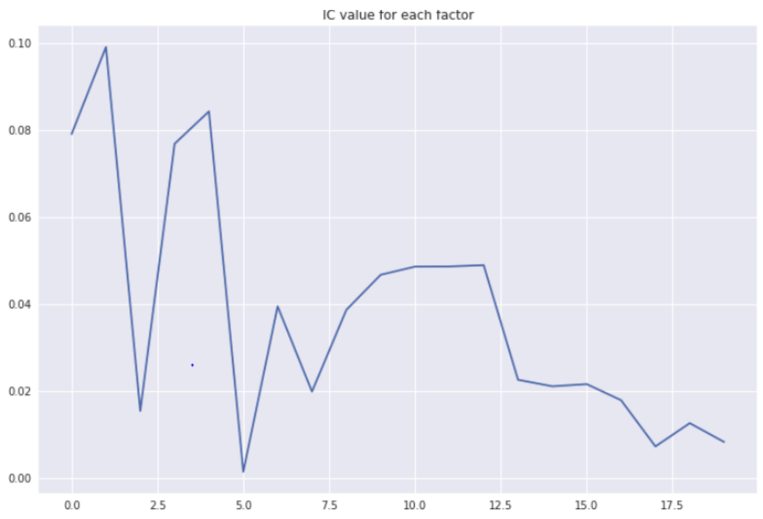


Chart 7：Factor 41-1 Analysis result(example of good performance factor)

Below the Information Correlation value is shown in a line chart version.

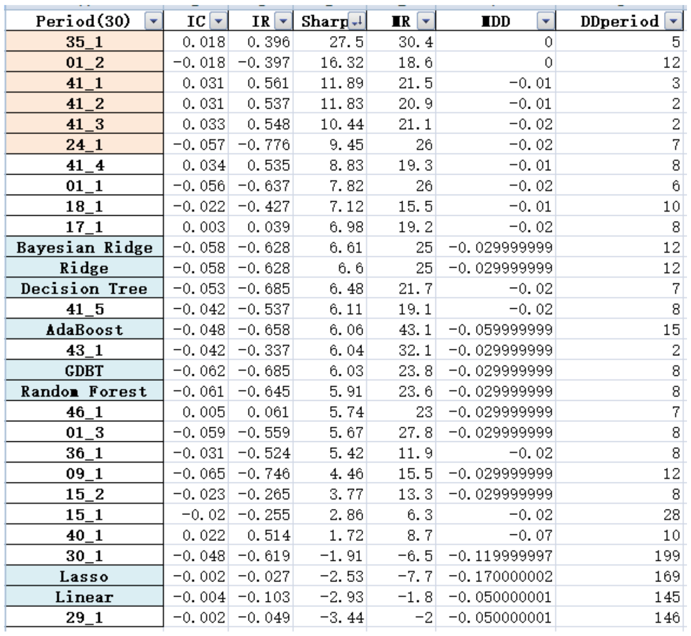


Graph 3：Line chart of Information Correlation for sample factor base factors.

Below is the result of the second part analysis, which has been shown on prior analysis. But factors themselves could be analysis just like results of the models. If the performance of specific models cannot beat against the best factor, certainly that this model could not be qualified for real-word trading. Here we will see how was the model improved during pre-processing, standardization, winsorization, model engineering, and changing fitting method.

4.2.1 Initial Fitting Result

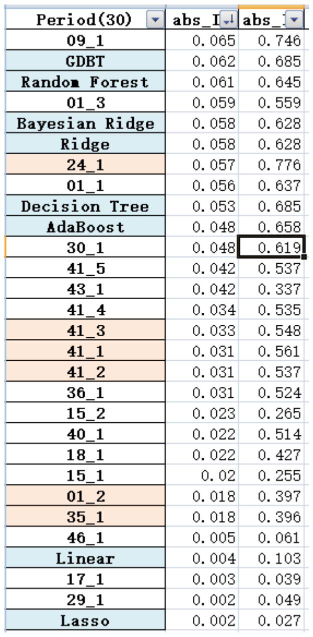
Below shows the initial result of fitting model and 21 factors full analysis. The results are shown in descending order ordered by Sharpe ratio. Attention that the other metrics, especially IC, IR, are also important for model evaluation. In this scenario, we could see that similar to before result, linear regression and lasso regression performs really bad, so we will treat them as a baseline of model comparison. We could see that ridge models performs better than tree-based models currently.



Period(30)	IC	IR	Sharp	AR	MDD	DDperiod
35_1	0.018	0.396	27.5	30.4	0	5
01_2	-0.018	-0.397	16.32	18.6	0	12
41_1	0.031	0.561	11.89	21.5	-0.01	3
41_2	0.031	0.537	11.83	20.9	-0.01	2
41_3	0.033	0.548	10.44	21.1	-0.02	2
24_1	-0.057	-0.776	9.45	26	-0.02	7
41_4	0.034	0.535	8.83	19.3	-0.01	8
01_1	-0.056	-0.637	7.82	26	-0.02	6
18_1	-0.022	-0.427	7.12	15.5	-0.01	10
17_1	0.003	0.039	6.98	19.2	-0.02	8
Bayesian Ridge	-0.058	-0.628	6.61	25	-0.029999999	12
Ridge	-0.058	-0.628	6.6	25	-0.029999999	12
Decision Tree	-0.053	-0.685	6.48	21.7	-0.02	7
41_5	-0.042	-0.537	6.11	19.1	-0.02	8
AdaBoost	-0.048	-0.658	6.06	43.1	-0.059999999	15
43_1	-0.042	-0.337	6.04	32.1	-0.029999999	2
GDBT	-0.062	-0.685	6.03	23.8	-0.029999999	8
Random Forest	-0.061	-0.645	5.91	23.6	-0.029999999	8
46_1	0.005	0.061	5.74	23	-0.029999999	7
01_3	-0.059	-0.559	5.67	27.8	-0.029999999	8
36_1	-0.031	-0.524	5.42	11.9	-0.02	8
09_1	-0.065	-0.746	4.46	15.5	-0.029999999	12
15_2	-0.023	-0.265	3.77	13.3	-0.029999999	8
15_1	-0.02	-0.255	2.86	6.3	-0.02	28
40_1	0.022	0.514	1.72	8.7	-0.07	10
30_1	-0.048	-0.619	-1.91	-6.5	-0.119999997	199
Lasso	-0.002	-0.027	-2.53	-7.7	-0.170000002	169
Linear	-0.004	-0.103	-2.93	-1.8	-0.050000001	145
29_1	-0.002	-0.049	-3.44	-2	-0.050000001	146

Chart 8 : Absolute IC/IR analysis result for Sample alpha factors, without factor adjustment.(21 factors)

Below chart shows exactly same result but ordered by Information Ratio in descending order. Clearly that Tree based model works better much better on Information Ratio, but still worse than several strong factors (such as 09-1, 01-3 and 01-1). Still Lasso and linear regression perform worst among models.

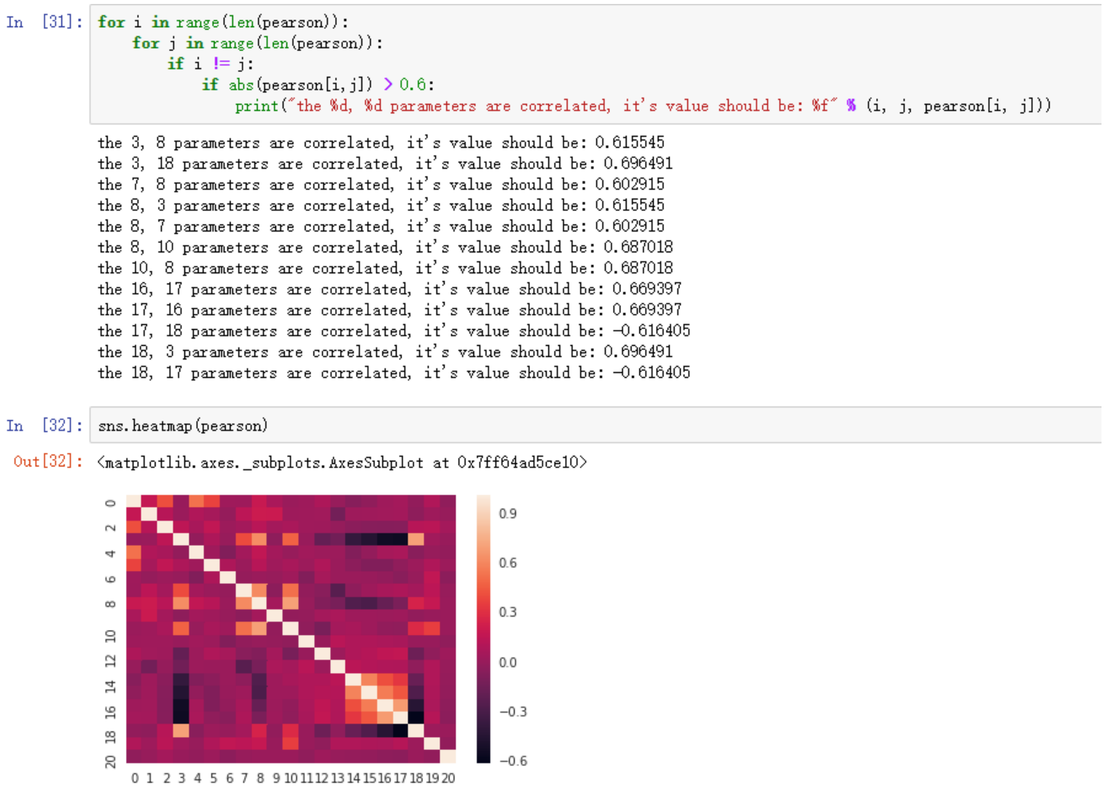


Period(30)	abs_I	abs_
09_1	0.065	0.746
GDBT	0.062	0.685
Random Forest	0.061	0.645
01_3	0.059	0.559
Bayesian Ridge	0.058	0.628
Ridge	0.058	0.628
24_1	0.057	0.776
01_1	0.056	0.637
Decision Tree	0.053	0.685
AdaBoost	0.048	0.658
30_1	0.048	0.619
41_5	0.042	0.537
43_1	0.042	0.337
41_4	0.034	0.535
41_3	0.033	0.548
41_1	0.031	0.561
41_2	0.031	0.537
36_1	0.031	0.524
15_2	0.023	0.265
40_1	0.022	0.514
18_1	0.022	0.427
15_1	0.02	0.255
01_2	0.018	0.397
35_1	0.018	0.396
46_1	0.005	0.061
Linear	0.004	0.103
17_1	0.003	0.039
29_1	0.002	0.049
Lasso	0.002	0.027

Chart 9 : Absolute IC/IR analysis result for Sample alpha factors, with factor adjustment.(21 factors)

We applied following methods to improve model performance, which are basically improving IC, IR and Sharpe ratio, especially Information Ratio:

a. Using Pearson Correlation Coefficient to check the collinearity of factors. If the Pearson value is higher than 0.6 we should say these two factors are highly correlated and at least one of these factors should be removed from fitting. Below screenshot shows the heatmap result of factor base. We could see that black blocks (which show correlation higher than 0.6), which is pair (3, 8), (3, 18), (7, 8), (8, 10), (16, 17), (17, 18) are highly correlated. Thus, we removed factor No. 3, 8, 17 from factor base.



```
In [31]: for i in range(len(pearson)):
             for j in range(len(pearson)):
                 if i != j:
                     if abs(pearson[i,j]) > 0.6:
                         print("the %d, %d parameters are correlated, it's value should be: %f" % (i, j, pearson[i, j]))

the 3, 8 parameters are correlated, it's value should be: 0.615545
the 3, 18 parameters are correlated, it's value should be: 0.696491
the 7, 8 parameters are correlated, it's value should be: 0.602915
the 8, 3 parameters are correlated, it's value should be: 0.615545
the 8, 7 parameters are correlated, it's value should be: 0.602915
the 8, 10 parameters are correlated, it's value should be: 0.687018
the 10, 8 parameters are correlated, it's value should be: 0.687018
the 16, 17 parameters are correlated, it's value should be: 0.669397
the 17, 16 parameters are correlated, it's value should be: 0.669397
the 17, 18 parameters are correlated, it's value should be: -0.616405
the 18, 3 parameters are correlated, it's value should be: 0.696491
the 18, 17 parameters are correlated, it's value should be: -0.616405

In [32]: sns.heatmap(pearson)
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff64ad5ce10>
```

Screenshot 1. Pearson Correlation Results and Heatmap Visualization.

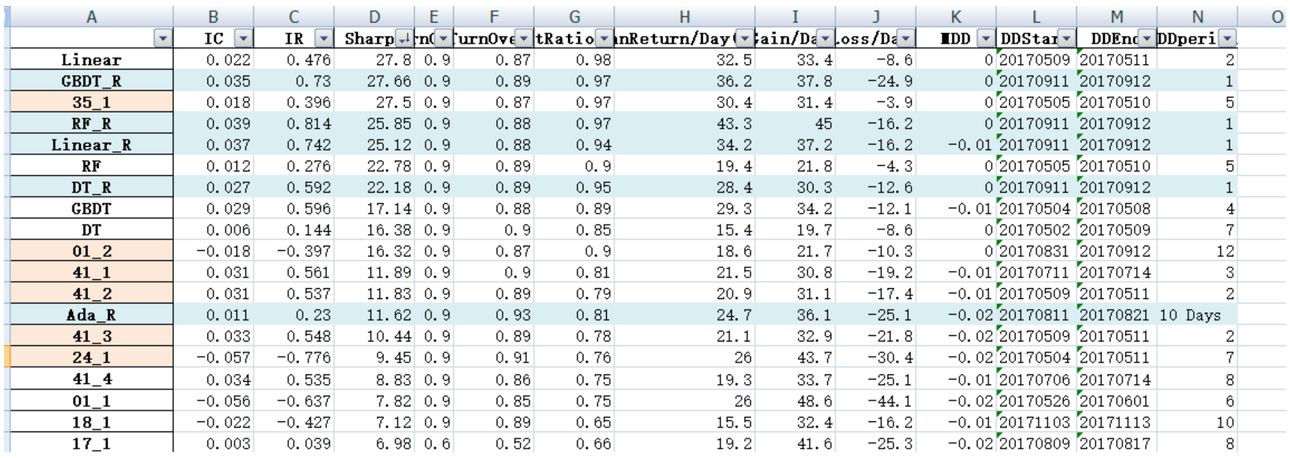
b. We improved standardization from rescaling by maximum and minimum value to 95% percentile and 5% percentile value, then transfer all value larger than 5% percentile or smaller than 95% percentile to 5% and 95% percentile value. This processing avoid impact of abnormal value, which is called winsorization.

c. Improving training process from 5000: 5000 rolling to 5000: 1000 rolling, which improved prediction accuracy.

d. Select highest 10 IR value factors for result enhancement.

e. If a factor has too much nan-value (higher than 1299900, which is 2.93% of total data points for one factor), then we will abolish this factor because those nan value should have more specified statistical explanation, and we shouldn't rudely replace then all by zero.

After feature engineering process, the result of full-factor analysis could be shown below. It's shown in descending order for Sharpe ratio, but clearly, we could see all models' information ratio perform better compare to original result shown in Chart 8 and 9. Specially, random forest, improved linear regression and GBDT perform also great. We will discuss about those three models in later part.



A	B	C	D	E	F	G	H	I	J	K	L	M	N
	IC	IR	Sharp	rn(	turnOve	tRatio	nReturn/Day(	ain/Da	oss/Da	MDD	DDStar	DDEnd	DDperi
Linear	0.022	0.476	27.8	0.9	0.87	0.98	32.5	33.4	-8.6	0	20170509	20170511	2
GBDT_R	0.035	0.73	27.66	0.9	0.89	0.97	36.2	37.8	-24.9	0	20170911	20170912	1
35_1	0.018	0.396	27.5	0.9	0.87	0.97	30.4	31.4	-3.9	0	20170505	20170510	5
RF_R	0.039	0.814	25.85	0.9	0.88	0.97	43.3	45	-16.2	0	20170911	20170912	1
Linear_R	0.037	0.742	25.12	0.9	0.88	0.94	34.2	37.2	-16.2	-0.01	20170911	20170912	1
RF	0.012	0.276	22.78	0.9	0.89	0.9	19.4	21.8	-4.3	0	20170505	20170510	5
DT_R	0.027	0.592	22.18	0.9	0.89	0.95	28.4	30.3	-12.6	0	20170911	20170912	1
GBDT	0.029	0.596	17.14	0.9	0.88	0.89	29.3	34.2	-12.1	-0.01	20170504	20170508	4
DT	0.006	0.144	16.38	0.9	0.9	0.85	15.4	19.7	-8.6	0	20170502	20170509	7
01_2	-0.018	-0.397	16.32	0.9	0.87	0.9	18.6	21.7	-10.3	0	20170831	20170912	12
41_1	0.031	0.561	11.89	0.9	0.9	0.81	21.5	30.8	-19.2	-0.01	20170711	20170714	3
41_2	0.031	0.537	11.83	0.9	0.89	0.79	20.9	31.1	-17.4	-0.01	20170509	20170511	2
Ada_R	0.011	0.23	11.62	0.9	0.93	0.81	24.7	36.1	-25.1	-0.02	20170811	20170821	10 Days
41_3	0.033	0.548	10.44	0.9	0.89	0.78	21.1	32.9	-21.8	-0.02	20170509	20170511	2
24_1	-0.057	-0.776	9.45	0.9	0.91	0.76	26	43.7	-30.4	-0.02	20170504	20170511	7
41_4	0.034	0.535	8.83	0.9	0.86	0.75	19.3	33.7	-25.1	-0.01	20170706	20170714	8
01_1	-0.056	-0.637	7.82	0.9	0.85	0.75	26	48.6	-44.1	-0.02	20170526	20170601	6
18_1	-0.022	-0.427	7.12	0.9	0.89	0.65	15.5	32.4	-16.2	-0.01	20171103	20171113	10
17_1	0.003	0.039	6.98	0.6	0.52	0.66	19.2	41.6	-25.3	-0.02	20170809	20170817	8

Chart 10：All metric analysis result for Sample alpha factors, with factor adjustment, feature engineering and parameter adjustment (21 factors).

After feature engineering process, the result of full-factor analysis could be shown below. It's shown in descending order for Sharpe ratio, but clearly, we could see all models' information ratio perform better compare to original result shown in Chart 8 and 9. Specially, random forest, improved linear regression and GBDT perform also great. We will discuss

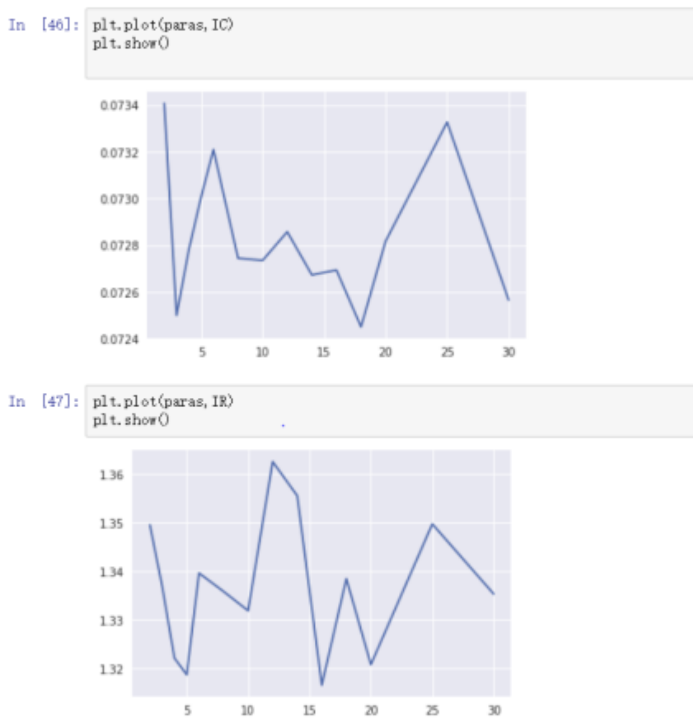
about those three models in later part. Attention that random forest has the best performance, generally speaking.

4.3 Grid search for random forest's hyper-parameter

Since we already found out that random forest is the best model, we are trying to find the optimized hyper parameters for maximizing its performance. By simply fix the other hyperparameters and increasing one specific hyperparameter gradually, we fit the model each time and gain feedback (in this case, information correlation). In this example, we will just discuss some hyperparameters of random forest such as min_samples_split and max_dept, but in real-word experiment we performed lots of other models' major hyperparameters as well, such as GBDT, XGBoost, LightGBM and Deep Neural Network.

4.3.1 min_samples_split:

The result of grid search for min_samples_split is shown below (as a plotting chart). By changing this parameter from 1 to 30, a very fluctuated, randomly moving plotting result could be observed. From the chart we should say that this parameter is not sensitive to grid search (or we should say this parameter is not essential towards the whole model, which could also be shown by scale of plot chart). The range of model fitting result would just be 0.01, which is just less than 2% of the total value. So, this parameter is not considerable for grid search.



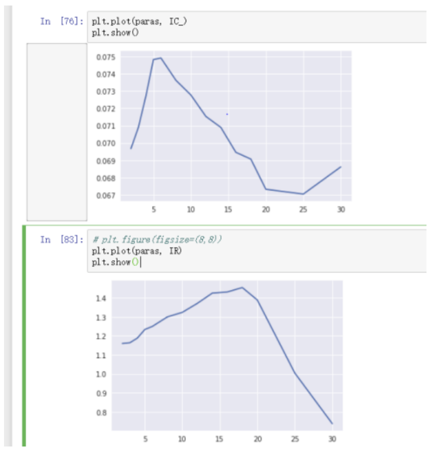
```
In [46]: plt.plot(paras, IC)
         plt.show()
```

```
In [47]: plt.plot(paras, IR)
         plt.show()
```

Screenshot 2. Grid Search Result (IC, IR) for hyperparameter: min_sample_split in GBDT.

4.3.2 max_depth:

max_depth is another tested hyperparameter, and below is the result of grid search for this parameter. It's really easy to see an obvious tendency for both Information Correlation and Information Ratio, but the tendency is certainly different. Here we measure the IR value so we should choose 20 (which is the maximum value).



```
In [76]: plt.plot(paras, IC_)
         plt.show()
```

```
In [83]: # plt.figure(figsize=(8,8))
         plt.plot(paras, IR)
         plt.show()
```

Screenshot 3. Grid Search Result for hyperparameter: min_sample_split in GBDT.

Since the max_depth hyperparameter is sensitive, adjusting this parameter is really necessary. In our experiment, IR jumped from 1.36 to 1.47 after adjustment, which is a significant amount of improvement.

After all these baseline test, transformations and improvements toward model and model fitting method, we successfully selected tree-based model among lots of machine learning models and successfully improved the performance of models. Even though there are lots of further improvement achieved during our experiment (the content shown above should be around 30% of the total content), we couldn’t do further discussion due to pages constrain and avoid commercial leak. But I’ll discuss what future step could we do in future experiments.

5. Conclusions

As it’s discussed in previous session, the final target for this capstone report is to design a robust strategy for automatic trading, and our experiment shows how to define the target variable for training, how to import and describe dataset, how to deal with nan value, how to do standardization, how to do collinearity test, how to fit the model, how to evaluate the model and how to improve the performance of model step by step. Generally speaking tree-based model, especially random forest performs better among machine learning algorithms, so tree-based models are selected as a base model in future ensemble methods.

After all those steps, the next thing we should do is to evaluate the model in simulated-trading system to see how much profit we can actually earn. This is the most important part and lots of experiments were designed for this part. Then we need to use more advanced tree-based algorithms (such as GBDT, XGBoost, LightGBM and CatBoost) to see whether better model could bring better results (both for time and for information ratio). Actually, in experiment they really did better compared to random forest. Finally, we considered about the robustness of our algorithm. As it's shown that linear regression usually performs worst among models, amazingly linear regression has the highest Sharpe Ratio. This indicate that linear regression is not robust enough and may change significantly due to small input changes. So we try to make our algorithms more stable by considering several good-performing factors together. In this session we used stacking method for model ensemble.

Even though we tried to find an approximately optimized solution, the market is changing every day and the algorithm could loss it's predicting ability anytime. Thus, a quantitative researcher should really develop multiple strategies and test time together at same time, to make sure we choose best model for specific time.

6. References

Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167). ACM.

Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95-99.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Lattemann, C., Loos, P., Gomolka, J., Burghof, H. P., Breuer, A., Gomber, P., ... & Zajonz, R. (2012). High Frequency Trading. Business & Information Systems Engineering, 4(2), 93-108.

Leote, R. A. U. L., Lu, X., & Moulin, P. (2012). Demystifying equity risk-based strategies: A simple alpha plus beta description. Journal of Portfolio Management, 38, 56-70.

Menkveld, A. J. (2013). High frequency trading and the new market makers. Journal of Financial Markets, 16(4), 712-740.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association.

Tortoriello, R. (2009). Quantitative strategies for achieving alpha. McGraw Hill.

Quantitative equity portfolio management; modern techniques and applications. (2007). Reference and Research Book News, 22(3), Reference and Research Book News, Aug 2007, Vol.22(3).

Jensen, M. C., Black, F., & Scholes, M. S. (1972). The capital asset pricing model: Some empirical tests.

Lerman, P. M. (1980). Fitting segmented regression models by grid search. Applied Statistics, 77-84.

Stoica, P., & Gershman, A. B. (1999). Maximum-likelihood DOA estimation by data-supported grid search. IEEE Signal Processing Letters, 6(10), 273-275.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural networks, 12(1), 145-151.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Sra, S., Nowozin, S., & Wright, S. J. (Eds.). (2012). Optimization for machine learning. Mit Press.