Machine Learning Application in Stock Return Prediction

1. Abstract

The returns of CSI 300 index on unsystematic risk is important and relative predictable, which reflects stock price changes in Chinese stock market, and the basic approach in index prediction lies in finding the effective alpha factors. This project aims to construct an index time series forecasting model with multi alpha factors through using several machine learning algorithms. Our 112 alpha factors are all from the transaction data through deriving, changing and synthesizing the information like price, turnover and so on. Based on daily frequency trade data from China in the recent 10 years, we establish a transactional alpha model using supervised learning models. We compare the ability to predict index returns of Linear Regression models, Tree models and classic Neural Networks. Considering the effect of time series, we set the time windows as 21 work days, which means that there are 2352 (21days*112alphas) features for models without time series prediction ability, and 112 features for models with time series prediction ability.

The results show that the effectiveness of Neural Networks is better Linear Regression models and Tree models. Specifically, ARD Regression, LSTM (Long Short-Term Memory) algorithm and Extra Tree algorithm perform best in Linear Regression models, Tree models and Neural Networks separately. Nevertheless, feature selection is hardly helpful for improving the effectiveness of model prediction ability.

Furthermore, we construct a stock portfolio based on LSTM with setting loss function as MSE by using 000300.SH stocks. With completing the portfolio reallocation every 21 trading days, our long portfolio choosing the 30 stocks with highest predicted daily return, while short portfolio choosing the 30 stocks with the lowest predicted daily return. The annual return in the back test period (2017/12/5-2019/10/31) of long portfolio, short portfolio and long minus short portfolio are 0.60%, -47.61% and 63.29% respectively.

2. Introduction and related work

There have been many attempts to predict the stock future price. Many of them have achieved some results. The returns of stock could be divided into beta returns and alpha returns. The former class moves with the market fluctuation while the latter one could gain more stable expected return once the recognize the pattern of it. As a result, we use alpha returns in our project. We find a series of effective alpha factors and put these multi factors into many prediction methods to compare different methods' predicting ability.

Sahoo and Charlapally (2015) predicted the stock's future value using auto regression [1]. It can catch the linear relationship between input and output value. Gharehchopogh et, al (2013) also implement Linear Regression for predicting the behavior of S&P index and achieved a good performance [2]. Besides Linear Regression algorithm, Dutta et, al (2012) used Logistic Regression and various financial ratios as independent variables to investigate indicators that significantly affect the performance of stocks actively traded on the Indian stock market and it had a very high accuracy on classify "good" and "poor" stock [3]. Tree methods have also been applied to financial prediction task. Nair et, al (2010) presented the design and performance evaluation of a hybrid decision tree-rough set based system for predicting the next days' trend in the Bombay Stock Exchange [4].

Besides the tradition methods, deep learning is a hot tool in recent years to predict the stock. Kakushadze (2016) used daily Sharpe ratio, turnover and cents-per share to calculate 101 real-life quantitative trading alphas for building alpha return models [5].

Selvin et, al (2017) applied CNN, RNN and LSTM for stock price prediction and found that CNN identified as the best model because of irregular and aperiodic changes often occurs in the stock market [6]. Due to the limitations of RNN in stock prediction, Zhang et, al (2018) proposed a new model combining CNN and RNN, where the predicted mean square error is 30% lower than the RNN model [7]. The application of neural network in stock prediction has been developing for several years, and models based on high-frequency data perform well in previews researches.

3. Dataset and Features

We downloaded daily trade data between 01/01/ 2014 and 11/27/2019 from Wind Database and select significant and easily observed trading information as the input, including opening price, highest price, lowest price, closing price, previous closing price, average price, transaction volume and turnover rate. Based on the stocks alpha factors construction methods provided by GuoTaiJunAn's research report and the applicability of single index in our project, we selected and calculated 112 alpha factors through deriving, changing and synthesizing the input data. Following are the definition of these alpha factors:

Factor	Computational Expressions
Alpha1	*(-1* DELTA((((CLOSE -LOW) -(HIGH -CLOSE)) / (HIGH -LOW)), 1))*
Alpha2	*SUM((CLOSE=DELAY(CLOSE,1)?0:CLOSE-(CLOSE>DELAY(CLOSE,1)?MIN(LOW,DELAY(CLOSE,1)):MAX(HIGH,DELAY(CLOSE,1)))),6)*
Alpha3	*((((SUM(CLOSE, 8) / 8) + STD(CLOSE, 8)) < (SUM(CLOSE, 2) / 2)) ? (-1 * 1) : (((SUM(CLOSE, 2) / 2) < ((SUM(CLOSE, 8) / 8) -STD(CLOSE, 8))) ? 1 : (((1 < (VOLUME / MEAN(VOLUME,20))) || ((VOLUME / MEAN(VOLUME,20)) == 1)) ? 1 : (-1 * 1))))*
Alpha4	*(-1 * TSMAX(CORR(TSRANK(VOLUME, 5), TSRANK(HIGH, 5), 5), 3))*
Alpha5	*SMA(((HIGH+LOW)/2-(DELAY(HIGH,1)+DELAY(LOW,1))/2)*(HIGH-LOW)/VOLUME,7,2)*
Alpha6	*SUM(((CLOSE-LOW)-(HIGH-CLOSE))./(HIGH-LOW).*VOLUME,6)*
Alpha7	*(((HIGH * LOW)^0.5) -VWAP)*
Alpha8	*CLOSE-DELAY(CLOSE,5)*
Alpha9	*OPEN/DELAY(CLOSE,1)-1*
Alpha10	*CLOSE/DELAY(CLOSE,5)*
Alpha11	*(CLOSE<DELAY(CLOSE,5)?(CLOSE-DELAY(CLOSE,5))/DELAY(CLOSE,5):(CLOSE=DELAY(CLOSE,5)?0:(CLOSE-DELAY(CLOSE,5))/CLOSE))*
Alpha12	*(CLOSE-DELAY(CLOSE,6))/DELAY(CLOSE,6)*100*
Alpha13	*REGBETA(MEAN(CLOSE,6),SEQUENCE(6))*
Alpha14	*SMEAN(((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6)-DELAY((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6),3)),12,1)*
Alpha15	*SMA((CLOSE>DELAY(CLOSE,1)?STD(CLOSE:20),0),20,1)/(SMA((CLOSE>DELAY(CLOSE,1)?STD(CLOSE,20):0),20,1)+SMA((CLOSE<=DELAY(CLOSE,1)?STD(CLOSE,20):0),20,1))*100*
Alpha16	*SMA(CLOSE-DELAY(CLOSE,5),5,1)*

Alpha17	*((-1 * RANK((DELTA(CLOSE, 7) * (1 -RANK(DECAYLINEAR((VOLUME / MEAN(VOLUME,20)), 9)))))) * (1 + RANK(SUM(RET, 250))))*
Alpha18	*3*SMA((CLOSE-TSMIN(LOW,9))/(TSMAX(HIGH,9)-TSMIN(LOW,9))*100,3,1)-2*SMA(SMA((CLOSE-TSMIN(LOW,9))/(MAX(HIGH,9)-TSMAX(LOW,9))*100,3,1),3,1)*
Alpha19	*(CLOSE-DELAY(CLOSE,6))/DELAY(CLOSE,6)*VOLUME*
Alpha20	*(CLOSE-MEAN(CLOSE,12))/MEAN(CLOSE,12)*100*
Alpha21	*MEAN(CLOSE,12)/CLOSE*
Alpha22	*(MIN(RANK(DECAYLINEAR(DELTA(OPEN, 1), 15)), RANK(DECAYLINEAR(CORR((VOLUME), ((OPEN * 0.65) + (OPEN *0.35)), 17),7))) * -1)*
Alpha23	*(-1 * RANK(((SUM(OPEN, 5) * SUM(RET, 5)) -DELAY((SUM(OPEN, 5) * SUM(RET, 5)), 10))))*
Alpha24	*(((SUM(HIGH, 20) / 20) < HIGH) ? (-1 * DELTA(HIGH, 2)) : 0)*
Alpha25	*SUM((CLOSE>DELAY(CLOSE,1)?VOLUME:0),26)/SUM((CLOSE<=DELAY(CLOSE,1)?VOLUME:0), 26)*100*
Alpha26	*((-1 * RANK(STD(HIGH, 10))) * CORR(HIGH, VOLUME, 10))*
Alpha27	*SUM((CLOSE>DELAY(CLOSE,1)?VOLUME:(CLOSE<DELAY(CLOSE,1)?-VOLUME:0)),6)*
Alpha28	*(TSRANK(DECAYLINEAR(CORR(((LOW)), MEAN(VOLUME,10), 7), 6),4) + TSRANK(DECAYLINEAR(DELTA((VWAP), 3), 10), 15))*
Alpha29	*(MEAN(CLOSE,3)+MEAN(CLOSE,6)+MEAN(CLOSE,12)+MEAN(CLOSE,24))/(4*CLOSE)*
Alpha30	*SMA((TSMAX(HIGH,6)-CLOSE)/(TSMAX(HIGH,6)-TSMIN(LOW,6))*100,9,1)*
Alpha31	*(-1*((RANK(((SIGN((CLOSE -DELAY(CLOSE,1))) + SIGN((DELAY(CLOSE, 1) -DELAY(CLOSE, 2)))) + SIGN((DELAY(CLOSE, 2) -DELAY(CLOSE, 3)))))) * SUM(VOLUME, 5)) / SUM(VOLUME, 20))*
Alpha32	*SUM(((HIGH+LOW)>=(DELAY(HIGH,1)+DELAY(LOW,1))?0:MAX(ABS(HIGH-DELAY(HIGH,1)),ABS(LOW-DELAY(LOW,1)))),12)/(SUM(((HIGH+LOW)>=(DELAY(HIGH,1)+DELAY(LOW,1))?0:MAX(ABS(HIGH-DELAY(HIGH,1)),ABS(LOW-DELAY(LOW,1)))),12)+SUM(((HIGH+LOW)<=(DELAY(HIGH,1)+DELAY(LOW,1))?0:MAX(ABS(HIGH-DELAY(HIGH,1)),ABS(LOW-DELAY(LOW,1)))),12))*
Alpha33	*SUM(MAX(0,HIGH-DELAY((HIGH+LOW+CLOSE)/3,1)),26)/SUM(MAX(0,DELAY((HIGH+LOW+CLOSE)/3,1)-L),26)*100*
Alpha34	*COUNT(CLOSE>DELAY(CLOSE,1),12)/12*100*
Alpha35	*SMA((CLOSE-TSMIN(LOW,9))/(TSMAX(HIGH,9)-TSMIN(LOW,9))*100,3,1)*
Alpha36	*COUNT(CLOSE>DELAY(CLOSE,1),20)/20*100*
Alpha37	*SUM((CLOSE=DELAY(CLOSE,1)?0:CLOSE-(CLOSE>DELAY(CLOSE,1)?MIN(LOW,DELAY(CLOSE,1)):MAX(HIGH,DELAY(CLOSE,1)))),20)*
Alpha38	*SUM(((CLOSE-LOW)-(HIGH-CLOSE))./(HIGH-LOW).*VOLUME,20)*
Alpha39	*SMA(MAX(CLOSE-DELAY(CLOSE,1),0),6,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),6,1)*100*
Alpha40	*MEAN(CLOSE,6)/CLOSE*
Alpha41	*(CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6)*100*
Alpha42	*SMA(MAX(CLOSE-DELAY(CLOSE,1),0),24,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),24,1)*100*
Alpha43	*SMA(((HIGH+LOW)/2-(DELAY(HIGH,1)+DELAY(LOW,1))/2)*(HIGH-LOW)/VOLUME,15,2)*
Alpha44	*STD(AMOUNT,6)*
Alpha45	*(CLOSE-MEAN(CLOSE,24))/MEAN(CLOSE,24)*100*
Alpha46	*SMA((TSMAX(HIGH,6)-CLOSE)/(TSMAX(HIGH,6)-TSMIN(LOW,6))*100,15,1)*

Alpha47	*STD(ABS((CLOSE/DELAY(CLOSE,1)-1))/VOLUME,20)/MEAN(ABS((CLOSE/DELAY(CLOSE,1)-1))/VOLUME,20)*
Alpha48	*((HIGH+LOW+CLOSE)/3-MA((HIGH+LOW+CLOSE)/3,12))/(0.015*MEAN(ABS(CLOSE-MEAN((HIGH+LOW+CLOSE)/3,12)),12))*
Alpha49	*SMA(MAX(CLOSE-DELAY(CLOSE,1),0),12,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),12,1)*100*
Alpha50	*(VOLUME-DELAY(VOLUME,5))/DELAY(VOLUME,5)*100*
Alpha51	*SMA(VOLUME,21,2)*
Alpha52	*SMA((TSMAX(HIGH,6)-CLOSE)/(TSMAX(HIGH,6)-TSMIN(LOW,6))*100,20,1)*
Alpha53	*(-1 * RANK(COVIANCE(RANK(HIGH), RANK(VOLUME), 5)))*
Alpha54	*SUM((CLOSE>DELAY(CLOSE,1)?VOLUME:(CLOSE<DELAY(CLOSE,1)?-VOLUME:0)),20)*
Alpha55	*(TSRANK((VOLUME / MEAN(VOLUME,20)), 20) * TSRANK((-1 * DELTA(CLOSE, 7)), 8))*
Alpha56	*((0.25 < (((DELAY(CLOSE, 20) -DELAY(CLOSE, 10)) / 10) -((DELAY(CLOSE, 10) -CLOSE) / 10))) ? (-1 * 1) : (((((DELAY(CLOSE, 20) -DELAY(CLOSE, 10)) / 10) -((DELAY(CLOSE, 10) -CLOSE) / 10)) < 0) ? 1 : ((-1 * 1) * (CLOSE -DELAY(CLOSE, 1)))))*
Alpha57	*((RANK(DECAYLINEAR(DELTA(VWAP, 4), 7)) + TSRANK(DECAYLINEAR(((((LOW * 0.9) + (LOW * 0.1)) -VWAP) / (OPEN -((HIGH + LOW) / 2))), 11), 7)) * -1)*
Alpha58	*2*(SMA(CLOSE,13,2)-SMA(CLOSE,27,2)-SMA(SMA(CLOSE,13,2)-SMA(CLOSE,27,2),10,2))*
Alpha59	*SUM((OPEN>=DELAY(OPEN,1)?0:MAX((OPEN-LOW),(OPEN-DELAY(OPEN,1)))),20)*
Alpha60	*SUM((CLOSE>DELAY(CLOSE,1)?VOLUME:(CLOSE<DELAY(CLOSE,1)?-VOLUME:0)),30)*
Alpha61	*STD(AMOUNT,20)*
Alpha62	*SMA(SMA((CLOSE-TSMIN(LOW,9))/(TSMAX(HIGH,9)-TSMIN(LOW,9))*100,3,1),3,1)*
Alpha63	*STD(VOLUME,10)*
Alpha64	*((((DELTA((SUM(CLOSE, 100) / 100), 100) / DELAY(CLOSE, 100)) < 0.05) || ((DELTA((SUM(CLOSE, 100) / 100), 100) / DELAY(CLOSE, 100)) == 0.05)) ? (-1 * (CLOSE -TSMIN(CLOSE, 100))) : (-1 * DELTA(CLOSE, 3)))*
Alpha65	*STD(VOLUME,20)*
Alpha66	*SMA(MAX(VOLUME-DELAY(VOLUME,1),0),6,1)/SMA(ABS(VOLUME-DELAY(VOLUME,1)),6,1)*100*
Alpha67	*(-1 * (DELTA(CORR(HIGH, VOLUME, 5), 5) * RANK(STD(CLOSE, 20))))*
Alpha68	*CLOSE-DELAY(CLOSE,20)*
Alpha69	*SMA(HIGH-LOW,10,2)/SMA(SMA(HIGH-LOW,10,2),10,2)*
Alpha70	*SMA(VOL*((CLOSE-LOW)-(HIGH-CLOSE))/(HIGH-LOW),11,2)-SMA(VOL*((CLOSE-LOW)-(HIGH-CLOSE))/(HIGH-LOW),4,2)*
Alpha71	*(SUM((CLOSE-DELAY(CLOSE,1)>0?CLOSE-DELAY(CLOSE,1):0),12)-SUM((CLOSE-DELAY(CLOSE,1)<0?ABS(CLOSE-DELAY(CLOSE,1)):0),12))/(SUM((CLOSE-DELAY(CLOSE,1)>0?CLOSE-DELAY(CLOSE,1):0),12)+SUM((CLOSE-DELAY(CLOSE,1)<0?ABS(CLOSE-DELAY(CLOSE,1)):0),12))*100*
Alpha72	*(-1 * ((RANK((SUM(DELAY(CLOSE, 5), 20) / 20)) * CORR(CLOSE, VOLUME, 2)) * RANK(CORR(SUM(CLOSE, 5), SUM(CLOSE, 20), 2))))*
Alpha73	*((RANK(DELAY(((HIGH -LOW) / (SUM(CLOSE, 5) / 5)), 2)) * RANK(RANK(VOLUME))) / (((HIGH -LOW) / (SUM(CLOSE, 5) / 5)) / (VWAP -CLOSE)))*
Alpha74	*REGBETA(CLOSE,SEQUENCE,20)*
Alpha75	*((TSRANK(VOLUME, 32) * (1 -TSRANK(((CLOSE + HIGH) -LOW), 16))) * (1 -TSRANK(RET,32)))*
Alpha76	*SUM(HIGH-OPEN,20)/SUM(OPEN-LOW,20)*100*
Alpha77	*(SMA(SMA(SMA(LOG(CLOSE),13,2),13,2),13,2)-*

	DELAY(SMA(SMA(SMA(LOG(CLOSE),13,2),13,2),13,2),1))/DELAY(SMA(SMA(SMA(LOG(CLOSE),13,2),13,2),13,2),1)
Alpha78	*(CLOSE -VWAP) / DECAYLINEAR(RANK(TSMAX(CLOSE, 30)),2)*
Alpha79	*(CLOSE+HIGH+LOW)/3*
Alpha80	*SUM((CLOSE-DELAY(CLOSE,1)<0?ABS(CLOSE-DELAY(CLOSE,1)):0),12)*
Alpha81	*(CLOSE-DELAY(CLOSE,12))/DELAY(CLOSE,12)*VOLUME*
Alpha82	*SMA(DELAY(CLOSE/DELAY(CLOSE,20),1),20,1)*
Alpha83	*((-1 * RANK(DELTA(RET, 3))) * CORR(OPEN, VOLUME, 10))*
Alpha84	*(-1 * CORR(OPEN, VOLUME, 10))*
Alpha85	*SUMIF(ABS(CLOSE/DELAY(CLOSE,1)-1)/AMOUNT,20,CLOSE<DELAY(CLOSE,1))/COUNT(CLOSE<DELAY(CLOSE,1),20)*
Alpha86	*(MEAN(VOLUME,9)-MEAN(VOLUME,26))/MEAN(VOLUME,12)*100*
Alpha87	*(CLOSE+HIGH+LOW)/3*VOLUME*
Alpha88	*SMA(MEAN(DELAY(SMA(DELAY(CLOSE/DELAY(CLOSE,9),1),9,1),1),12)-MEAN(DELAY(SMA(DELAY(CLOSE/DELAY(CLOSE,9),1),9,1),1),26),9,1)*
Alpha89	*(MEAN(CLOSE,3)+MEAN(CLOSE,6)+MEAN(CLOSE,12)+MEAN(CLOSE,24))/4*
Alpha90	*SMA(VOLUME,13,2)-SMA(VOLUME,27,2)-SMA(SMA(VOLUME,13,2)-SMA(VOLUME,27,2),10,2)*
Alpha91	*((HIGH-SMA(CLOSE,15,2))-(LOW-SMA(CLOSE,15,2)))/CLOSE*
Alpha92	*((CLOSE-SUM(MIN(LOW,DELAY(CLOSE,1)),6))/SUM(MAX(HGIH,DELAY(CLOSE,1))-MIN(LOW,DELAY(CLOSE,1)),6)*12*24+(CLOSE-SUM(MIN(LOW,DELAY(CLOSE,1)),12))/SUM(MAX(HGIH,DELAY(CLOSE,1))-MIN(LOW,DELAY(CLOSE,1)),12)*6*24+(CLOSE-SUM(MIN(LOW,DELAY(CLOSE,1)),24))/SUM(MAX(HGIH,DELAY(CLOSE,1))-MIN(LOW,DELAY(CLOSE,1)),24)*6*24)*100/(6*12+6*24+12*24)*
Alpha93	*SMA((CLOSE<=DELAY(CLOSE,1)?STD(CLOSE,20):0),20,1)*
Alpha94	*MEAN(MAX(MAX((HIGH-LOW),ABS(DELAY(CLOSE,1)-HIGH)),ABS(DELAY(CLOSE,1)-LOW)),12)*
Alpha95	*(SMA(MAX(CLOSE-DELAY(CLOSE,1),0),12,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),12,1)*100-MIN(SMA(MAX(CLOSE-DELAY(CLOSE,1),0),12,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),12,1)*100,12))/(MAX(SMA(MAX(CLOSE-DELAY(CLOSE,1),0),12,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),12,1)*100,12)-MIN(SMA(MAX(CLOSE-DELAY(CLOSE,1),0),12,1)/SMA(ABS(CLOSE-DELAY(CLOSE,1)),12,1)*100,12))*
Alpha96	*SMA((((CLOSE>DELAY(CLOSE,1))?1/(CLOSE-DELAY(CLOSE,1)):1)-MIN((((CLOSE>DELAY(CLOSE,1))?1/(CLOSE-DELAY(CLOSE,1)):1),12))/(HIGH-LOW)*100,13,2)*
Alpha97	*SUM((CLOSE-DELAY(CLOSE,1)>0?CLOSE-DELAY(CLOSE,1):0),12)*
Alpha98	*(-1*VOLUME/MEAN(VOLUME,20))*
Alpha99	*SMA(MEAN(DELAY(SMA(CLOSE-DELAY(CLOSE,1),9,1),1),12)-MEAN(DELAY(SMA(CLOSE-DELAY(CLOSE,1),9,1),1),26),10,1)*
Alpha100	*((((RANK((1 / CLOSE)) * VOLUME) / MEAN(VOLUME,20)) * ((HIGH * RANK((HIGH -CLOSE))) / (SUM(HIGH, 5) / 5))) -RANK((VWAP -DELAY(VWAP, 5))))*
Alpha101	*((-1 * ((LOW -CLOSE) * (OPEN^5))) / ((CLOSE -HIGH) * (CLOSE^5)))*
Alpha102	*MEAN(ABS(SUM((LD>0 & LD>HD)?LD:0,14)*100/SUM(TR,14)-SUM((HD>0 & HD>LD)?HD:0,14)*100/SUM(TR,14))/(SUM((LD>0 & LD>HD)?LD:0,14)*100/SUM(TR,14)+SUM((HD>0 & HD>LD)?HD:0,14)*100/SUM(TR,14))*100,6)*

Alpha103	*3*SMA(CLOSE,13,2)-* *2*SMA(SMA(CLOSE,13,2),13,2)+SMA(SMA(SMA(LOG(CLOSE),13,2),13,2),13,2);*
Alpha104	*SMA((CLOSE>DELAY(CLOSE,1)?STD(CLOSE,20):0),20,1)*
Alpha105	*CORR(RANK(((CLOSE -TSMIN(LOW, 12)) / (TSMAX(HIGH, 12) -TSMIN(LOW,12)))),* *RANK(VOLUME), 6)*
Alpha106	*(CLOSE-DELAY(CLOSE,1))/DELAY(CLOSE,1)*VOLUME*
Alpha107	*((MEAN(VOLUME,20) < VOLUME) ? ((-1 * TSRANK(ABS(DELTA(CLOSE, 7)), 60)) ** *SIGN(DELTA(CLOSE, 7)) : (-1 * VOLUME)))*
Alpha108	*(MEAN(ABS(SUM((LD>0 & LD>HD)?LD:0,14)*100/SUM(TR,14)-SUM((HD>0 &* *HD>LD)?HD:0,14)*100/SUM(TR,14))/(SUM((LD>0 &* *LD>HD)?LD:0,14)*100/SUM(TR,14)+SUM((HD>0 &* *HD>LD)?HD:0,14)*100/SUM(TR,14))*100,6)+DELAY(MEAN(ABS(SUM((LD>0 &* *LD>HD)?LD:0,14)*100/SUM(TR,14)-SUM((HD>0 &* *HD>LD)?HD:0,14)*100/SUM(TR,14))/(SUM((LD>0 &* *LD>HD)?LD:0,14)*100/SUM(TR,14)+SUM((HD>0 &* *HD>LD)?HD:0,14)*100/SUM(TR,14))*100,6),6))/2*
Alpha109	*SUM((OPEN<=DELAY(OPEN,1)?0:MAX((HIGH-OPEN),(OPEN-DELAY(OPEN,1)))),20)*
Alpha110	*((HIGH-LOW–SMA(HIGH-LOW,11,2))/SMA(HIGH-LOW,11,2))*100*
Alpha111	*MEAN(ABS(CLOSE-MEAN(CLOSE,6)),6)*
Alpha112	*((CORR(MEAN(VOLUME,20), LOW, 5) + ((HIGH + LOW) / 2)) -CLOSE)*

Because of some alpha factors request using previous data, we finally input data between 2015/01/05 and 2019/11/27 in our later models. In terms of Linear Regression models and Tree models that are not able to solve the effect of time series, we process the initial inputs as a flat time series inputs, in which case each work day have its 112 alpha factors and previous 20 work days' alpha factors as its features. As for Deep Learning Models that have the time series prediction ability, the inputs are each work day's 112 alpha factors. The predict target is the return for the next trading day:

return = (closing price next trading day-today's closing price) / today's closing price

4. Methods

We choose many traditional predicting methods, including Linear Regression, Ridge Regression, Orthogonal Matching Pursuit, Bayesian Ridge Regression, Decision Tree Regression, Random Forest Regression, AdaBoost Regression, Extra Tree Regression, GBDT, XGBoost, lightBGM and many deep learning methods like MLP, LSTM and CNN.

4.1 Linear Regression Models

- **Linear Regression**

For this regression problem, we assume the target value and factors has linear relationship, like this:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

Mathematically Linear Regression solves a problem of the form:

$$\min_w ||Xw - y||_2^2$$

- **Ridge Regression**

Ridge Regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

- **Orthogonal Matching Pursuit**

Orthogonal Matching Pursuit implements the OMP algorithm for approximating the fit of a linear model with constraints imposed on the number of non-zero coefficients (i.e. the pseudo-norm).

Orthogonal Matching Pursuit can approximate the optimum solution vector with a fixed number of non-zero elements:

$$\underset{\gamma}{arg\ min} ||y - X\gamma||_2^2 \ subject\ to\ ||\gamma||_0 \leq n_{nonzero_{coefs}}$$

- **Bayesian Ridge Regression**

Bayesian Regression techniques can be used to include regularization parameters in the estimation procedure: the regularization parameter is not set in a hard sense but tuned to the data at hand.

To obtain a fully probabilistic model, the output y is assumed to be Gaussian distributed around Xw:

$$\mathrm{P}(\mathrm{y}|\mathrm{X}, \mathrm{w}, \alpha) = \mathrm{N}(\mathrm{y}|\mathrm{Xw}, \alpha)$$

Bayesian ridge estimates a probabilistic model of the regression problem as described above. The prior for the coefficient is given by a spherical Gaussian:

$$\mathrm{P}(\mathrm{w}|\lambda) = \mathrm{N}(\mathrm{w}|0, \lambda^{-1}\mathbf{I}_{\mathrm{p}})$$

- **ARD Regression**

ARD regression is very similar to Bayesian Ridge Regression, but can lead to sparser coefficients. ARD Regression poses a different prior over , by dropping the assumption of the Gaussian being spherical.

This means each coefficient w_i is drawn from a Gaussian distribution, centered on zero and with a precision λ_i:

$$\mathrm{P}(\mathrm{w}|\lambda) = \mathrm{N}(\mathrm{w}|0, A^{-1}) \text{ with } \mathrm{diag}(\mathrm{A}) = \lambda = \{\lambda_1, \ldots, \lambda_{\mathrm{p}}\}$$

- **Theil-Sen Regression**

The Theil–Sen estimator is an unbiased estimator of the true slope in simple Linear Regression. The estimation of the model is done by calculating the slopes and intercepts of a subpopulation of all possible combinations of p subsample points.

4.2 Tree Models:

- **Decision Tree**

Decision Tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

- **Random Forest**

A random Forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **AdaBoost**

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

- **Extra Tree**

Extra Tree implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **GBDT, XGBoost and LightBGM**

Gradient Boosting Decision Tree is a popular machine learning algorithm.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

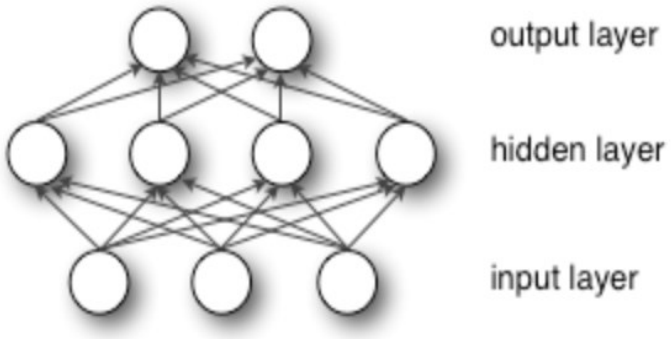
We choose three effective implementations: GBDT, XGBoost and lightBGM.

4.3 Deep Learning methods

- **Multilayer Perceptron**

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to refer to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation).

An MLP with a single -hidden-layer can be represented graphically as follows:



MLP trains using Stochastic Gradient Descent, Adam, or L-BFGS. Stochastic Gradient Descent (SGD) updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$$

In our project, we use a 3-layer MLP. Here are the main code:

```
model = Sequential()
model.add(Dense(500, input_dim= X_train.shape[1]))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(250))
model.add(Activation('relu'))
model.add(Dense(1))
model.add(Activation('linear'))
model.compile(optimizer='adam',
              loss='mae')
# fit network
history = model.fit(X_train,
          y_train,
          epochs=40,
          batch_size = 72,
          verbose=2,
          validation_data=(X_test, y_test))
score = model.evaluate(X_test, y_test, verbose=2)
```

- **LSTM**

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory.

We use a LSTM network with 50 neurons in one hidden layer. Here is part of our code:

```
# design network
model = models.Sequential()
model.add(layers.LSTM(50, input_shape=(X_train.shape[1], X_train.
   shape[2])))
model.add(layers.Dense(1))
model.compile(loss='mae', optimizer='adam')
# fit network
history = model.fit(X_train, y_train,
epochs=40, batch_size=72,
validation_data=(X_test, y_test), verbose=2, shuffle=False)
score = model.evaluate(X_test, y_test, verbose=2)
```

- **CNN**

Convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. This include time-series data, which can be thought of as a 1D and image data, which can be thought of as a 2D grid of pixels.[8]

As the same, we use a 3-layer-CNN network to predict the value. Here is the main architecture code:

```
model = Sequential((Convolution1D(input_shape=(window_size, nb_input_series),
kernel_size=filter_length, activation="relu", filters=nb_filter),

MaxPooling1D(), # Downsample the output of convolution by 2X.
#Convolution1D(nb_filter=nb_filter, filter_length=filter_length, activation='relu'),
Convolution1D(kernel_size=filter_length, activation="relu", filters=nb_filter),
MaxPooling1D(),
Flatten(),
Dense(nb_outputs, activation='linear'), # For binary classification, change the activation to 'sigmoid'
))
opt = Adam(lr=0.001)
#model.compile(loss='mean_squared_error', optimizer=opt, metrics=['mae']
model.compile(loss='mse', optimizer='adam')
# Fitting the RNN to the Training set
history =model.fit(X_train, y_train,
epochs=40, batch_size=72, validation_data=(X_test, y_test),
verbose=2, shuffle=False)
score = model.evaluate(X_test, y_test, verbose=2)
```

For all the three models, we use both MAE and MSE as their loss functions to evaluate the predictive effect and set Adam as the optimizer. In this project the epoch of three models are set to be 40, just for general model test rather than a precise model test.

5. Results and Discussion

Based on the dataset and features, we established and completed the returns prediction using different models and algorithms. In order to evaluate the effectiveness of models, we also calculated the related indicators including MSE (mean squared error), RMSE (root mean squared error), MAPE (mean absolute percentage error), and R squared which could measure the performance of our algorithms both in sample and out of sample.

$$\text{MSE} = \frac{1}{M}\sum_{m=1}^{M}(y_m - \hat{y}_m)^2$$

$$\text{RMSE} = \sqrt[2]{\frac{1}{M}\sum_{m=1}^{M}(y_m - \hat{y}_m)^2}$$

$$\text{MAPE} = \sum_{m=1}^{M}\left|\frac{(y_m - \hat{y}_m)}{y_m}\right| \times \frac{100}{M}$$

$$R^2 = \frac{SS_{mean} - SS_{fit}}{SS_{mean}}$$

$$SS_{mean} = \sum_{m=1}^{M} (y_m - \overline{y_m})^2$$

$$SS_{fit} = \sum_{m=1}^{M} (y_m - \hat{y}_m)^2$$

5.1 Model Performance

- **Linear Regression Models**

Linear Regression Models perform well on index returns prediction on the whole, whose RMSE almost ranges from 0.010 to 0.018. ARD Regression performs best in Linear Regression Models, followed by Ridge Regression, Theil-Sen Regression, Linear Regression and Orthogonal Matching Pursuit regression. When it comes to MAPE Ridge Regression got a relatively good performance while other algorithms all perform worse than linear regression since if MAPE is smaller, the model is better also, ARD model shows its power with the positive R2 while other models all get negative out of sample R2.

Table1 Linear Regression Models' Performances for out-of-sample test

Models	MSE	RMSE	MAPE	R^2
Linear Reg	2.70E-04	0.016429	676.358623	-0.634284
Ridge Reg	2.10E-04	0.014486	567.914601	-0.270555
Orthogonal Matching Pursuit	3.04E-04	0.017427	1257.426224	-0.838862
Bayesian Ridge Reg	3.30E-04	0.018178	1295.151346	-1.000719
ARD Reg	1.12E-04	0.010579	1019.717486	0.322437
Theil-Sen Reg	2.52E-04	0.015866	1190.395527	-0.524240

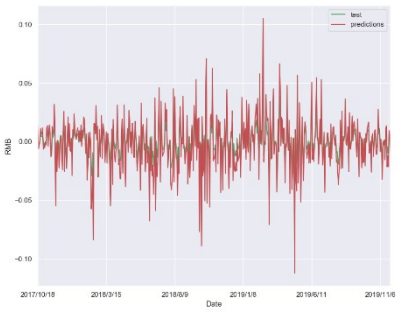
In the following figures, the green lines represent the actual value while the red lines represent prediction in test samples:

Figure 1 Linear Regression Models' Performances for out-of-sample test

Linear Reg	Ridge Reg
	
Orthogonal Matching Pursuit	**Bayesian Ridge Reg**

ARD Reg

Theil-Sen Reg



- **Tree Models**

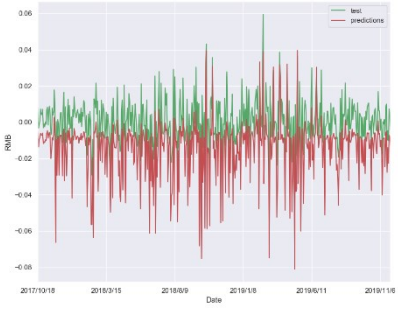
Tree Models perform very similar to each other with Extra Tree model performs best followed by XGBoost and GBDT. However, all tree models got a negative R2 except for Extra Tree model.

Table 2 Tree Models' Performances for out-of-sample test

Models	MSE	RMSE	MAPE	R^2
Decision Tree	0.000309	0.017570	976.164090	-0.869076
Random Forest	0.000338	0.018380	1033.370251	-1.045473
AdaBoost	0.000486	0.022048	1268.186591	-1.943283
Extra Tree	0.000103	0.010166	924.726673	0.374289
GBDT	0.000247	0.015727	951.759966	-0.497548
XGBoost	0.000167	0.012910	897.770139	-0.009141
lightBGM	0.000360	0.018985	1098.510281	-1.182203

Figure 2 Tree Models' Performances for out-of-sample test

Decision Tree | **Random Forest**

	
AdaBoost	**Extra Tree**
	
GBDT	**XGBoost**
	
lightBGM	
	

The green lines represent the actual value while the red lines represent prediction in test samples. Compared with the generalized linear models, tree models could actually catch the up and down of the real value but could not fit the real value as good as linear models.

- **Deep Learning Models**

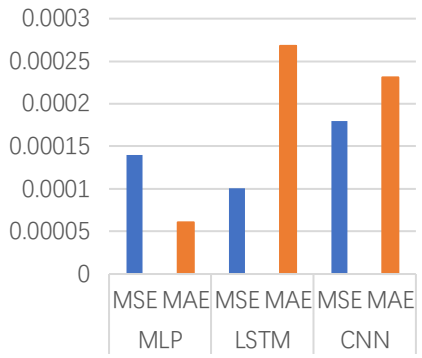
In Our project, we applied MLP, LSTM and CNN to index returns prediction. From the out of sample results of the indicators for deep learning models, MLP model with MAE loss function performs best followed by LSTM models with MSE loss function. Generally speaking, all the linear models, tree models and deep learning models' prediction power is close to each other.

Table 3 Deep Learning Models' Performances for out-of-sample test

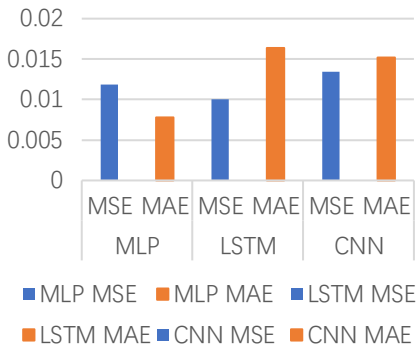
Models	MSE	RMSE	MAPE	R^2
MLP_MSE	0.00014	0.011847	652.3769	0.123663
MLP_MAE	6.07E-05	0.007789	307.167	0.621158
LSTM_MSE	0.000101	0.010054	407.1982	0.368832
LSTM_MAE	0.000268	0.016372	1242.465	-0.67363
CNN_MSE	0.00018	0.013415	236.8308	-0.12367
CNN_MAE	0.000231	0.015191	689.5703	-0.44104

Figures 3 Deep Learning Models' Performances for out-of-sample test

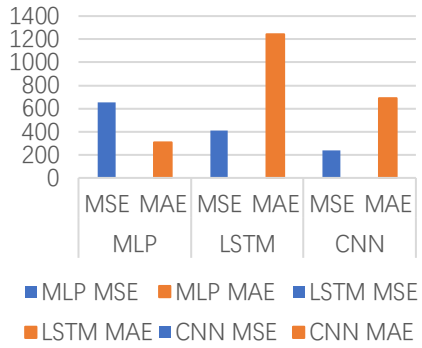
MSE



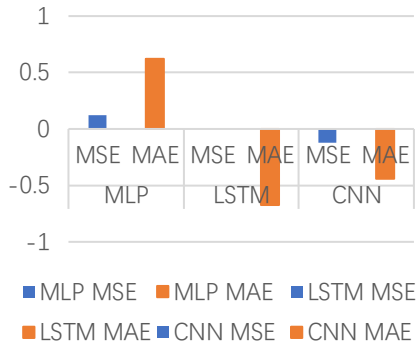
RMSE



MAPE

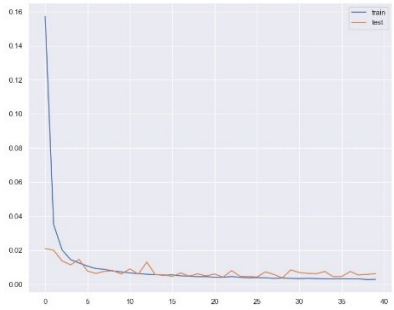
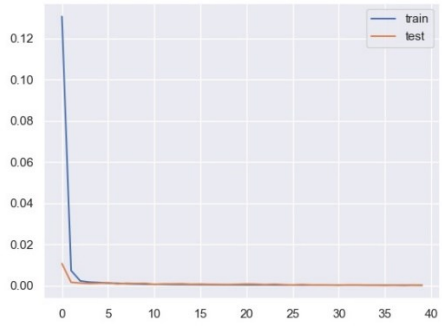
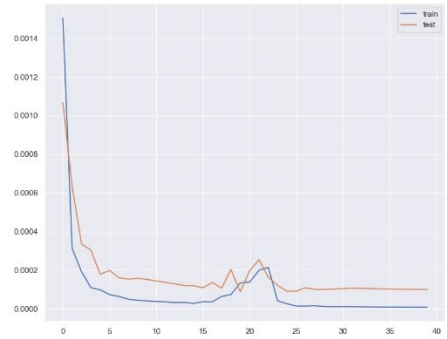
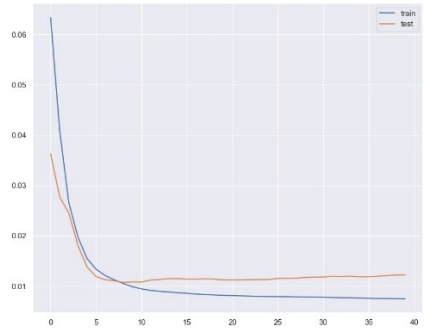
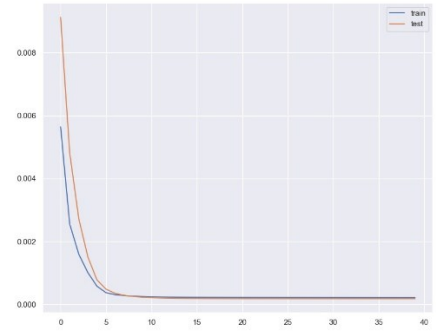


R^2



The graphs below show the loss history for the deep learning models with the blue line indicates the performance in train sample and the red line in test samples. MLP and CNN got a quick decrease in loss value while LSTM performs worst, especially for the circumstance of MAE loss function.

Figure 4 Loss History for Deep Learning Models

MLP (loss function = MAE)	MLP (loss function = MSE)
	
LSTM (loss function = MAE)	**LSTM (loss function = MSE)**
	
CNN (loss function = MAE)	**CNN (loss function = MSE)**
	

From the out of sample results of the indicators for deep learning models, except CNN, other models could get a good fit. The results might be that the input of our data is just 2-D data, which could not be fitted into CNN well.

Figure 5 Deep Learning Models' Performances for out-of-sample test

MLP (loss function = MAE)	MLP (loss function = MSE)

	
LSTM (loss function = MAE)	**LSTM (loss function = MSE)**
	
CNN (loss function = MAE)	**CNN (loss function = MSE)**
	

Append 1 Models Performance in-sample

- **Feature Importance**

The table below shows the top 17 most important features in our linear and tree models. We count the number that there are selected and listed in the count. For example, 4 days lag for alpha 178 was selected by 8 models and rank the top.

Table 4 Top 17 Important Features in Linear Regression and Tree models

Rank	Factor	Explanation	Count
1	alpha178t-4	4 days lag for alpha178: ((CLOSE-DELAY(CLOSE,1))/DELAY(CLOSE,1)*VOLUME)	8

2	alpha22t-18	18 days lag for alpha 22: (SMEAN(((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6)-DELAY((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6),3)),12,1))	7
3	alpha152t-4	4 days lag for alpha 152: (SMA(MEAN(DELAY(SMA(DELAY(CLOSE/DELAY(CLOSE,9),1),9,1),1),12)-MEAN(DELAY(SMA(DELAY(CLOSE/DELAY(CLOSE,9),1),9,1),1),26),9,1))	6
4	alpha178t-5	5 days lag for alpha 178: ((CLOSE-DELAY(CLOSE,1))/DELAY(CLOSE,1)*VOLUME)	6
5	alpha126t-2	2 days lag for alpha 126: ((CLOSE+HIGH+LOW)/3)	5
6	alpha78t	alpha 78: (((HIGH+LOW+CLOSE)/3-MA((HIGH+LOW+CLOSE)/3,12))/(0.015*MEAN(ABS(CLOSE-MEAN((HIGH+LOW+CLOSE)/3,12)),12)))	5
7	alpha153t-1	1 day lag for alpha 153: ((MEAN(CLOSE,3)+MEAN(CLOSE,6)+MEAN(CLOSE,12)+MEAN(CLOSE,24))/4)	5
8	alpha22t-6	6 days lag for alpha 22: (SMEAN(((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6)-DELAY((CLOSE-MEAN(CLOSE,6))/MEAN(CLOSE,6),3)),12,1))	5
9	alpha170t-10	10 days lag for alpha 170: (((((RANK((1 / CLOSE)) * VOLUME) / MEAN(VOLUME,20)) * ((HIGH * RANK((HIGH -CLOSE))) / (SUM(HIGH, 5) / 5))) - RANK((VWAP -DELAY(VWAP, 5)))))	5
10	alpha114t-3	3 days lag for alpha 114: (((RANK(DELAY(((HIGH -LOW) / (SUM(CLOSE, 5) / 5)), 2)) * RANK(RANK(VOLUME))) / (((HIGH -LOW) / (SUM(CLOSE, 5) / 5)) / (VWAP - CLOSE))))	4
11	alpha78t-13	13 days lag for alpha 78: (((HIGH+LOW+CLOSE)/3-MA((HIGH+LOW+CLOSE)/3,12))/(0.015*MEAN(ABS(CLOSE-MEAN((HIGH+LOW+CLOSE)/3,12)),12)))	4
12	alpha38t-19	19 days lag for alpha 38: ((((SUM(HIGH, 20) / 20) < HIGH) ? (-1 * DELTA(HIGH, 2)) : 0))	4
13	alpha171t	alpha 171: (((-1 * ((LOW -CLOSE) * (OPEN^5))) / ((CLOSE - HIGH) * (CLOSE^5))))	4
14	alpha191t-3	3 days lag for alpha 191: (((CORR(MEAN(VOLUME,20), LOW, 5) + ((HIGH + LOW) / 2)) -CLOSE))	4
15	alpha38t-4	4 days lag for alpha 38: ((((SUM(HIGH, 20) / 20) < HIGH) ? (-1 * DELTA(HIGH, 2)) : 0))	3
16	alpha52t-16	16 days lag for alpha 52: (SMA((TSMAX(HIGH,6)-CLOSE)/(TSMAX(HIGH,6)-TSMIN(LOW,6))*100,20,1))	3

17	alpha169t-10	10 days lag for alpha 169: (SMA(MEAN(DELAY(SMA(CLOSE-DELAY(CLOSE,1),9,1),1),12)-MEAN(DELAY(SMA(CLOSE-DELAY(CLOSE,1),9,1),1),26),10,1))	3

Append 2 Feature Importance

5.2 Model Performance after Feature Selection

Considering the high-dimensional inputs for Linear Regression and Decision Tree models, we introduce the feature engineering for comparing if features dimensionality deduction improve the models' performance. In the previous step, we found that GBDT performs best in Decision Tree Models. Therefore, we use RFE (Recursive Feature Elimination) with setting estimator as GBDT and target features number as 300 to select important features for further test. GBDT was used to complete each round of training, after which several features that have lower weight coefficients were eliminated, and then the next round of training start based on the new feature set.

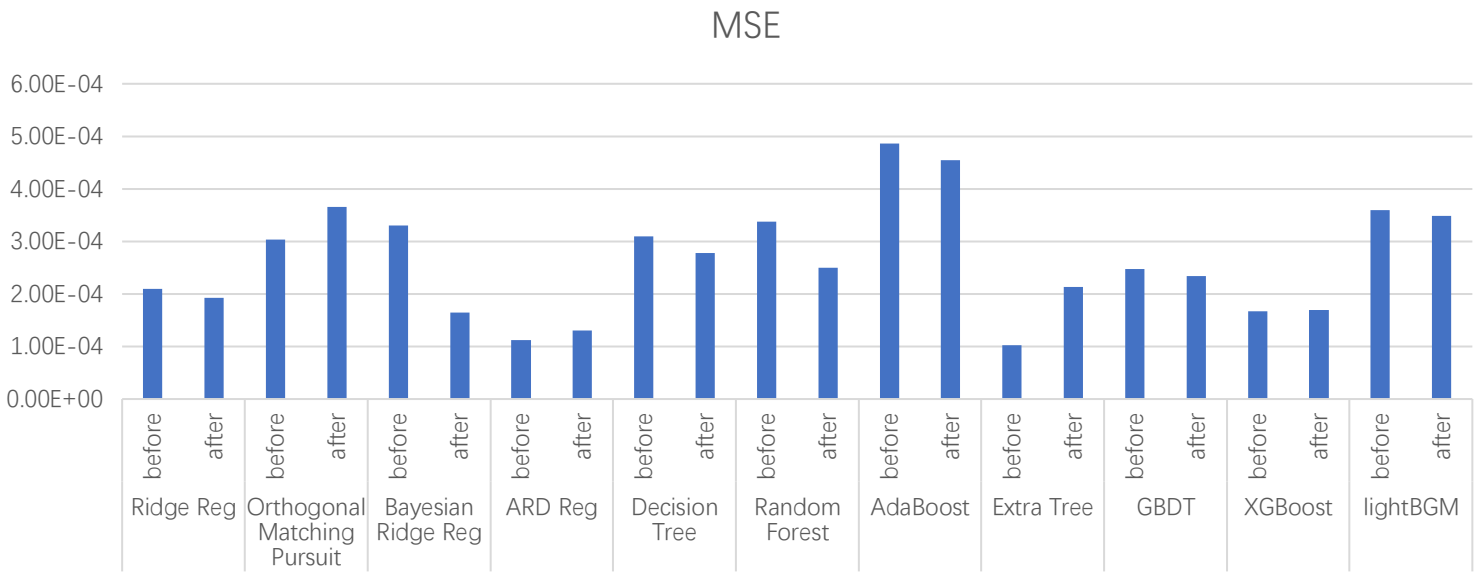
The table below shows the out of sample results before and after feature selection. If the indicator is 1, it means the model performs better after feature selection; vise versa. We could see that feature selection actually work for Bayesian Ridge Regression, Decision Tree, Random Forest, AdaBoost, GBDT and lightBGM.

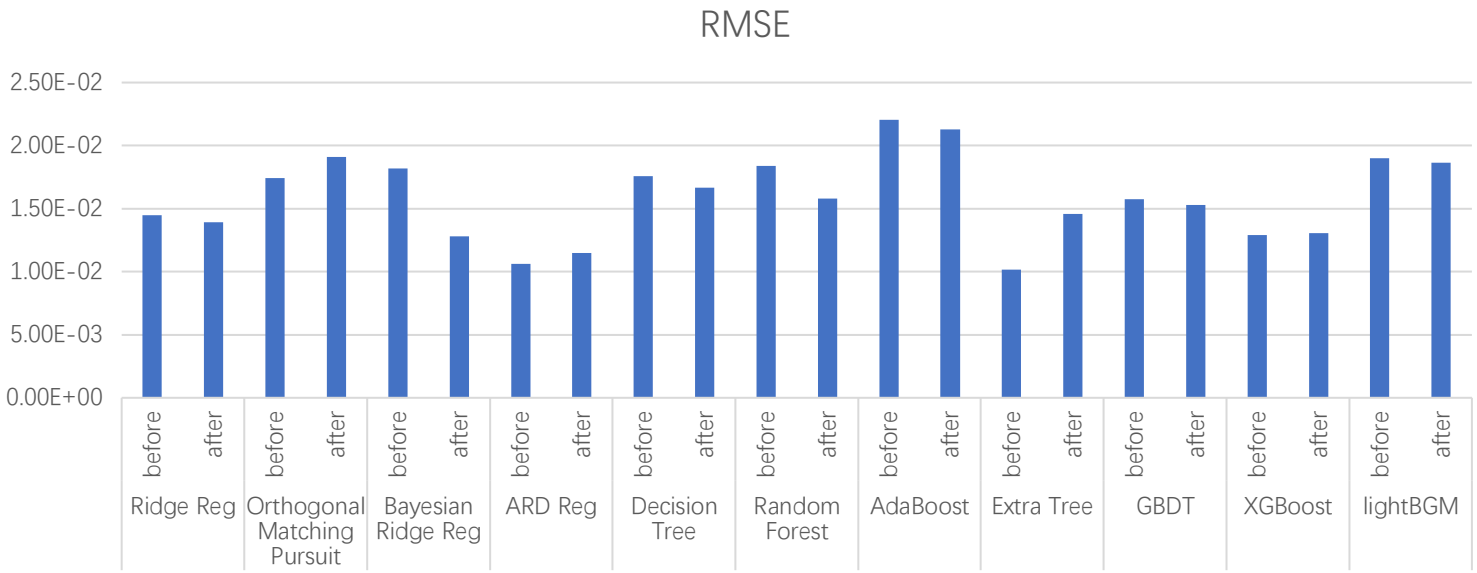
Table 5 Models' Performances before and after Feature Selection

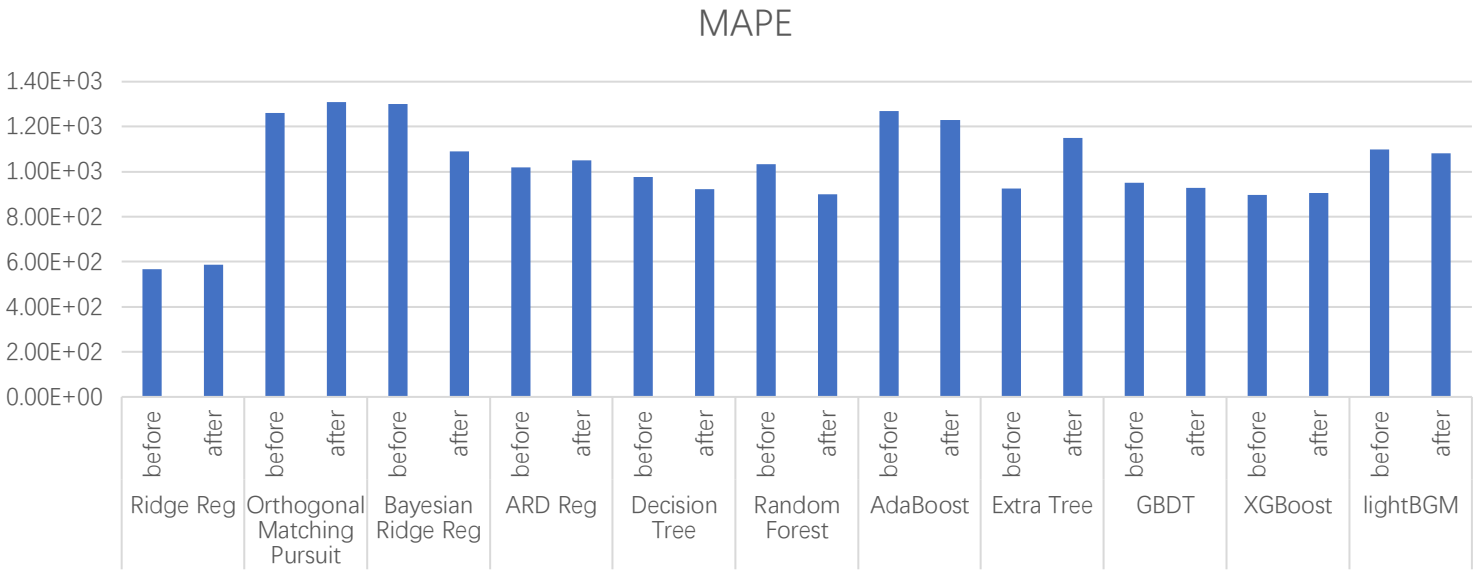
Models	Feature Selection	MSE	RMSE	MAPE	R^2
Linear Reg	before	2.70E-04	1.64E-02	6.76E+02	-6.34E-01
	after	1.51E+06	1.23E+03	6.77E+07	-9.13E+09
	fs work or not	0	0	0	0
Ridge Reg	before	2.10E-04	1.45E-02	5.68E+02	-2.71E-01
	after	1.93E-04	1.39E-02	5.87E+02	-1.70E-01
	fs work or not	1	1	0	1
Orthogonal Matching Pursuit	before	3.04E-04	1.74E-02	1.26E+03	-8.39E-01
	after	3.66E-04	1.91E-02	1.31E+03	-1.22E+00
	fs work or not	0	0	0	0
Bayesian Ridge Reg	before	3.30E-04	1.82E-02	1.30E+03	-1.00E+00
	after	1.65E-04	1.28E-02	1.09E+03	4.00E-03
	fs work or not	1	1	1	1
ARD Reg	before	1.12E-04	1.06E-02	1.02E+03	3.22E-01
	after	1.31E-04	1.15E-02	1.05E+03	2.04E-01
	fs work or not	0	0	0	0
Theil-Sen Reg	before	2.52E-04	1.59E-02	1.19E+03	-5.24E-01
	after	1.67E+07	4.09E+03	2.09E+08	-1.01E+11
	fs work or not	0	0	0	0
Decision Tree	before	0.000309	0.017570	976.164090	-0.869076
	after	0.000278	0.016666	922.902488	-0.681709

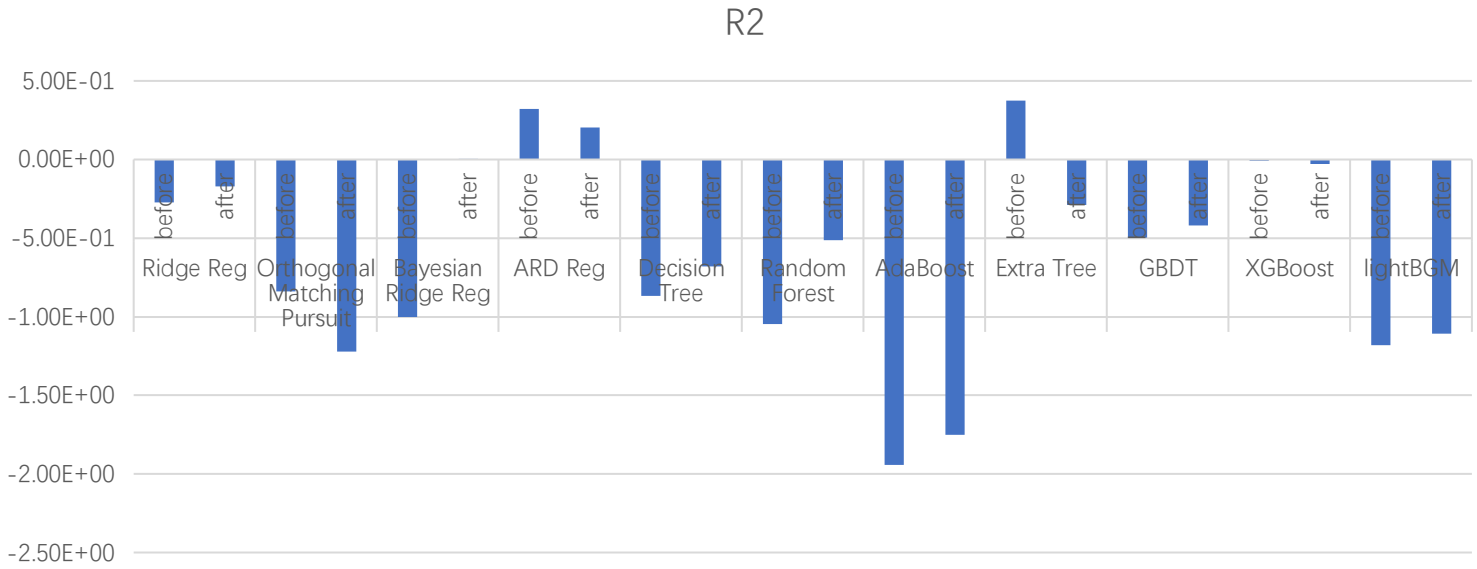
	fs work or not	1	1	1	1
Random Forest	before	0.000338	0.018380	1033.370251	-1.045473
	after	0.000250	0.015804	899.164820	-0.512276
	fs work or not	1	1	1	1
AdaBoost	before	0.000486	0.022048	1268.186591	-1.943283
	after	0.000454	0.021309	1228.838424	-1.749294
	fs work or not	1	1	1	1
Extra Tree	before	0.000103	0.010166	924.726673	0.374289
	after	0.000213	0.014592	1150.695472	-0.289279
	fs work or not	0	0	0	0
GBDT	before	0.000247	0.015727	951.759966	-0.497548
	after	0.000234	0.015310	928.636759	-0.419198
	fs work or not	1	1	1	1
XGBoost	before	0.000167	0.012910	897.770139	-0.009141
	after	0.000170	0.013043	905.164444	-0.029992
	fs work or not	0	0	0	0
lightBGM	before	0.000360	0.018985	1098.510281	-1.182203
	after	0.000348	0.018660	1080.704125	-1.108317
	fs work or not	1	1	1	1

The results show that the feature engineering could improve the MSE, RMSE and MAPE except for Orthogonal Matching Regression, ADR regression, extra tree model, and Decision tree. Meanwhile, R^2 is generally improved except for Orthogonal Matching pursuit model, ARD regression, extra tree model and XGBoost with R^2 chaging from positive value to negative for extra tree model.The results show that the feature engineering could improve the MSE and RMSE except for Orthogonal Matching Regression, ADR regression, extra tree model, and Decision tree.




RMSE
2.50E-02
2.00E-02
1.50E-02
1.00E-02
5.00E-03
0.00E+00
before
after
Ridge Reg
Orthogonal Matching Pursuit
Bayesian Ridge Reg
ARD Reg
Decision Tree
Random Forest
AdaBoost
Extra Tree
GBDT
XGBoost
lightBGM


MAPE
1.40E+03
1.20E+03
1.00E+03
8.00E+02
6.00E+02
4.00E+02
2.00E+02
0.00E+00
before
after
Ridge Reg
Orthogonal Matching Pursuit
Bayesian Ridge Reg
ARD Reg
Decision Tree
Random Forest
AdaBoost
Extra Tree
GBDT
XGBoost
lightBGM


R2
5.00E-01
0.00E+00
-5.00E-01
-1.00E+00
-1.50E+00
-2.00E+00
-2.50E+00
before
after
Ridge Reg
Orthogonal Matching Pursuit
Bayesian Ridge Reg
ARD Reg
Decision Tree
Random Forest
AdaBoost
Extra Tree
GBDT
XGBoost
lightBGM

5.3 Stock Selection and Portfolio Construction

- **Dataset and Methods**

In the portfolio construction part, we gathered the daily prediction of the components of the sh300 index. The back test time starts at 2017/12/5 and ends at 2019/10/31. Using LSTM with setting loss function as MSE, we obtained the stock returns prediction in the back test period. Considering the limited time of training models for 300 stocks, we just trained twice for stock returns prediction. In each round of prediction, the training samples cover 692 work days and the test part cover 250 work days, in which case we obtained prediction returns in 500 work days for each stock. We adjust the stocks we invested every 21 trading days according to their prediction of the return for the next trading day. In this case, we suppose that returns would remain for 21 trading days and did not take the volatility into consideration. We long 30 stocks with the highest predicted daily return and short 30 stocks with the lowest predicted daily return and construct long portfolio, short portfolio and long minus short portfolio and then calculated the net asset value in this back test.

In addition, we calculated the annual return of the portfolios, checked the t-statistic for our prediction singles, calculated volatility, maximum drawdown(it is the maximum observed loss from a peak to a trough of a portfolio, before a new peak is attained), accumulated return, sharpe ratio(it is the average return earned in excess of the risk-free rate per unit of volatility or total risk), reward to value at risk(it is the average return earned to the value at risk with the denominator measures the extreme risk of the portfolio), and reward to conditional value at risk(it is the average return earned to the value at risk with the denominator measures the relative extreme risk of the portfolio)

$$\text{Sharpe Ratio}=\frac{\sum_{i=1}^{n}(R_i-R_f)}{\sigma_{R_i}}$$

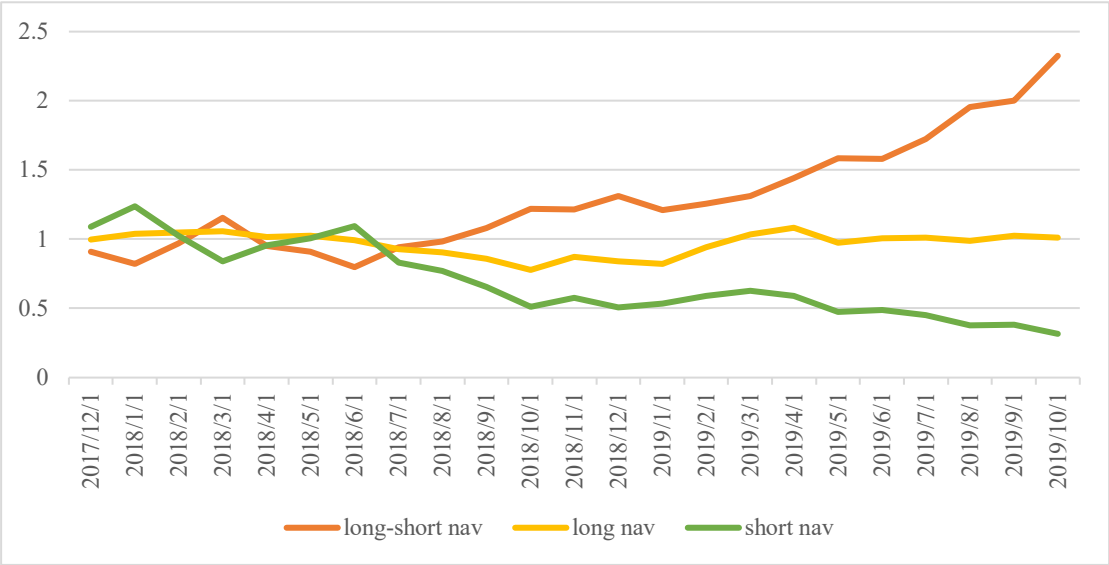
when calculating Sharpe ratio, we suppose that the risk free rate is 3% and remain stable during the whole period.

$$\text{maximum drawdown } = \frac{trough\ valuw - peak\ value}{peak\ value}$$

- **Results**

Figure ? shows the net asset value for the long portfolio, short portfolio and long minus short portfolio. The long portfolio got a horizontal performance during the back test time period while the short portfolio performs relatively well which makes the net asset value of the long-minus-short portfolio earns a good increase, especially after 2018/7/1.

Figure 6 Net Asset Value for Portfolios



The table below shows the annual return, results for t-statistic check, volatility of the return series, accumulated return, Sharpe ratio, Reward to VaR and Reward to CVaR for the three portfolios. All of them would earn a positive return in the market since the minus of the short leg is what we actually got in the practice. The t statistic show that long minus short portfolio would get a more stable return series. However, all the of three portfolios get a relatively low Sharpe ratio, which blames to their high volatility, or high total risk. The positive value of Reward to VaR and Reward to CVaR for short portfolio reveals that it would possess more long tail risk.

Table 6 Portfolio Performances

Indicators	**long**	**short**	**long-short**
annual-return	0.60%	-47.61%	63.29%
t-statistic	0.1754	-1.4897	1.9512
volatility	95.63%	201.57%	162.49%
Max Drawdown	26.37%	74.47%	30.76%
Accumulated return	1.011	0.3161	2.3251
Sharpe-ratio	0.0251	-0.2511	0.371
Reward_to_VaR	-0.0157	0.2954	-0.2253
Reward_to_CVaR	-0.5391	0.8338	-1.1853

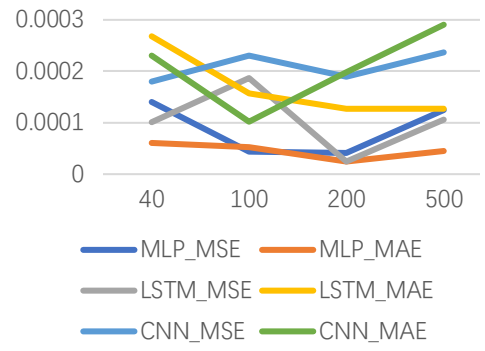
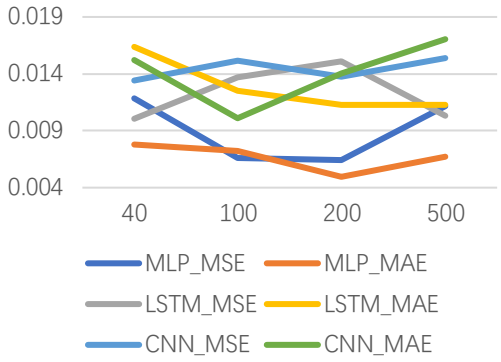
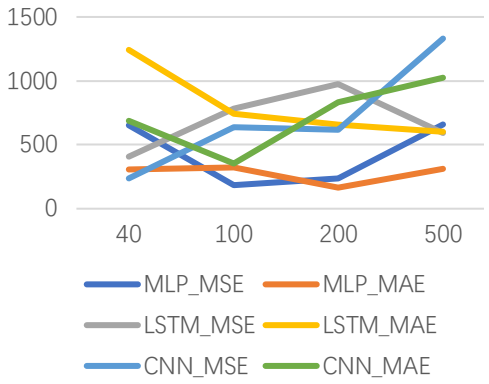
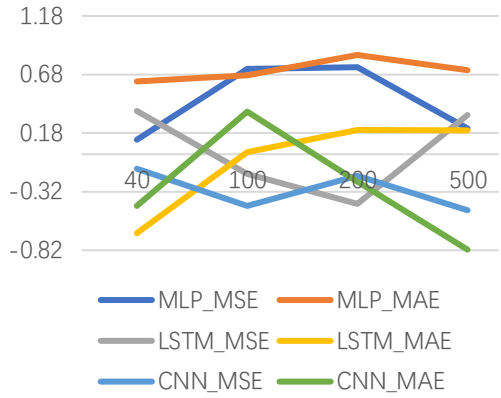
Append 1 Net Asset Value for Portfolios

5.4 Parameter Tuning

- **The Epoch Number**

The Epoch number is a parameter that defines the number of times the learning algorithm works in the entire training dataset. A Epoch means that each sample in the training dataset has a chance to update its internal model parameters. We test different Epoch number and the results show below:

Figures 7 Deep Learning Models' Performances with Epochs Tuning

MSE	RMSE
	
MAPE	**R^2**
	

While training, not the more epochs, the better. It may pass the minimal loss point or occur overfitting. In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". In regression analysis, overfitting occurs frequently. If the model is overfitting, it may show a bad performance on the samples besides the training dataset.

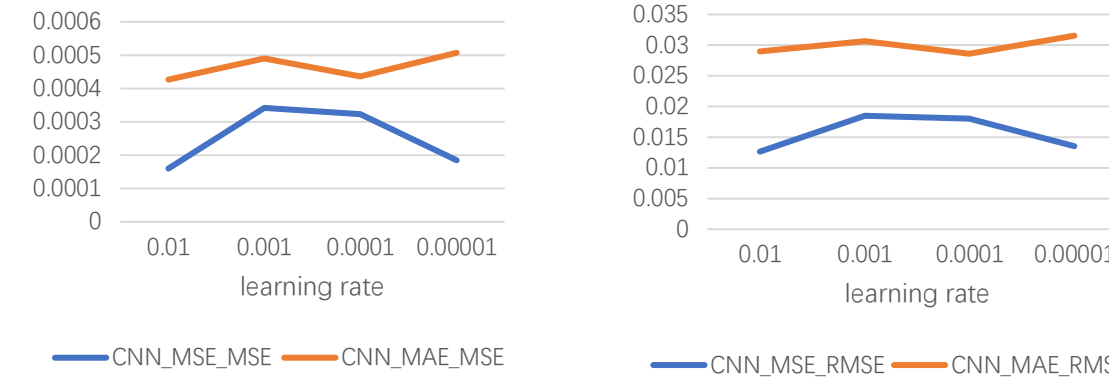
From the results, 200 epochs has the best performance. The indicators increase instead while continuing growing epochs. Therefore, we can see 500 epochs' result is worse than 200 epochs' result.

- **Learning Rate for CNN**

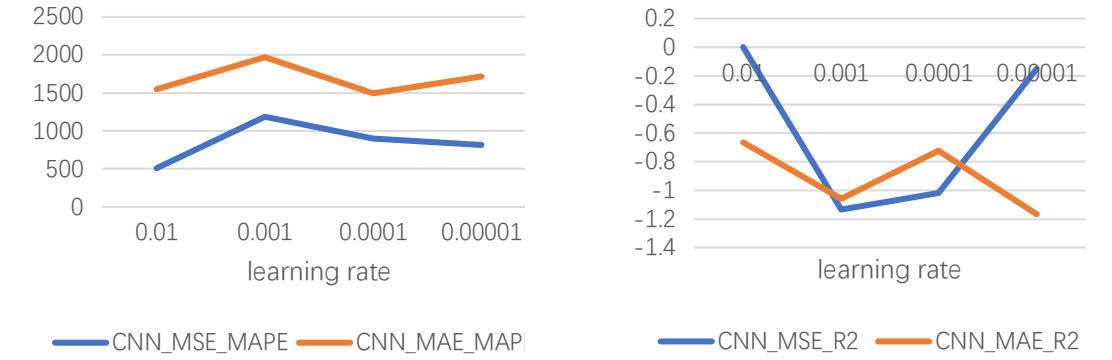
We have parameter tuning for the CNN models. We tried different learning rate and here are the result:

Figures 8 Deep Learning Models' Performances with Learning Rate Tuning

MSE	RMSE



MAPE **R^2**



From the result, we can see that different learning rate has different impact to CNN model. Learning rate influences to what extent newly acquired information overrides old information, therefore, it metaphorically represents the speed at which a machine learning model "learns". In order to achieve faster convergence, prevent oscillations and getting stuck in undesirable local minimal the learning rate is often varied during training either in accordance to a learning rate schedule or by using an adaptive learning rate.[9] A good learning rate should use less time to get the minimal loss.

We use four learning rate with different magnitude to train the CNN models with 60 epochs. The time they cost did not have much difference. As a result, looking above all the four indicators, setting learning rate as 0.001 has the better performance.

6. Conclusion and Future Work

In our project, we test 13 models in index returns prediction. On the whole, Neural Networks performs best in index prediction, MLP and LSTM performs better with RMSE ranging from 0.007 to 0.03 no matter the loss function setting as MAE or MSE. Linear Regression models and Tree Models' RMSE almost ranges from 0.01 to 0.02. As for The stock portfolios based on LSTM_MSE models perform pretty well with good annual return. Take the limit research period into consideration, we didn't apply all models into stock returns prediction and portfolio construction but just used LSTM with setting loss function as MSE. In addition, there are still some methods to optimize the prediction ability of models in future work, like parameters turning, longer time window setting, introducing economic fundamentals data and so on. And

comparing effectiveness of models between Chinese market and American market is also an interesting research direction.

Reference:

[1] Dr. P.K.Sahoo, Mr. Krishna charlapally ,"Stock Price Prediction Using Regression Analysis", International Journal of Scientific & Engineering Research, Volume 6, Issue 3, March-2015 ISSN 2229-5518

[2] Farhad soleimanian gharehchopogh, tahmineh haddadi bonab and seyyed reza khaze ,"a linear regression approach to prediction of stock market trading volume: a case study", international journal of managing value and supply chains (ijmvsc) vol.4, no. 3, september 2013

[3] Dutta A, Bandopadhyay G, Sengupta S. Prediction of stock performance in indian stock market using logistic regression[J]. International Journal of Business and Information, 2012, 7(1).

[4] Nair B B, Mohandas V P, Sakthivel N R. A decision tree—rough set hybrid system for stock market trend prediction[J]. International Journal of Computer Applications, 2010, 6(9): 1-6.

[5] Kakushadze,Z.(2016).101formulaicalphas.Papers,2016(84),7281.

[6] Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE.

[7] Zhang, R., Yuan, Z., & Shao, X. (2018). A New Combined CNN-RNN Model for Sector Stock Price Analysis. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). IEEE Computer Society.

[8] Selvin S, Vinayakumar R, Gopalakrishnan E A, et al. Stock price prediction using LSTM, RNN and CNN-sliding window model[C]//2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017: 1643-1647.

[9] Suki Lau (29 July 2017). "Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning".

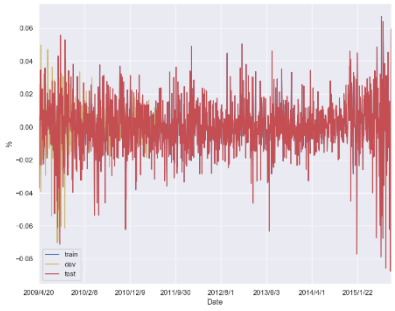
Append 1 Models Performance in-sample

Table 1 Linear Regression Models' Performances for in-sample

Models	MSE	RMSE	MAPE	R^2
Linear Reg	2.42E-32	1.55E-16	2.82E-11	1
Ridge Reg	1.02E-06	0.001011403	70.59449856	0.996203481
Orthogonal Matching Pursuit	1.20E-06	0.001096328	2656.362149	0.995539153
Bayesian Ridge Reg	2.33E-09	4.82E-05	2657.931741	0.999991364
ARD Reg	4.82E-08	0.000219622	2653.309142	0.999820986
Theil-Sen Reg	2.67E-32	1.63E-16	2657.889915	1

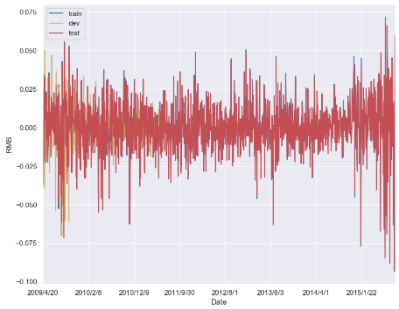
Figure 1 Linear Regression Models' Performances for in-sample

Linear Reg	Ridge Reg



Orthogonal Matching Pursuit

Bayesian Ridge Reg



ARD Reg

Theil-Sen Reg

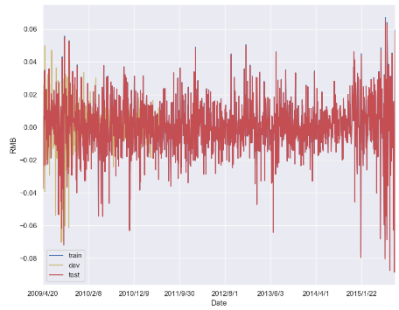


Table 2 Tree Models' Performances for in-sample

Models	MSE	RMSE	MAPE	R^2
Decision Tree	4.35E-08	0.00020856	2656.824367	0.999838564
Random Forest	2.29E-06	0.001513501	2643.47568	0.99149837
AdaBoost	1.42E-05	0.00377305	2994.590808	0.947164978
Extra Tree	6.80E-09	8.25E-05	2657.238908	0.999974744
GBDT	9.37E-07	0.000967797	2658.887273	0.996523793
XGBoost	1.56E-06	0.001248679	2652.158025	0.994213198
lightBGM	5.48E-07	0.00074051	2654.346023	0.997964842

Figure 2 Tree Models' Performances for in-sample

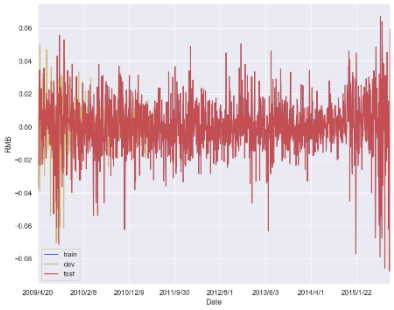
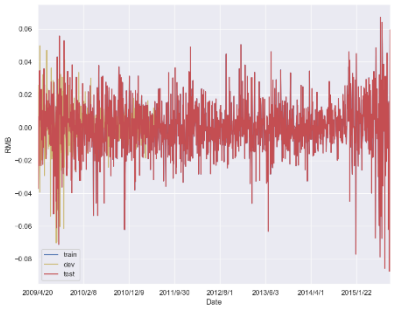
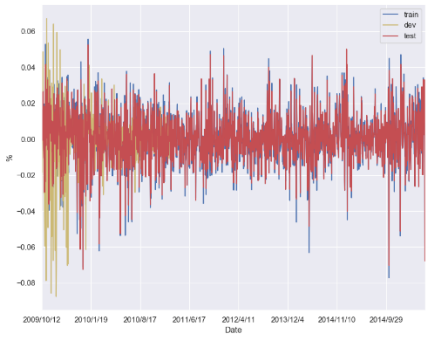
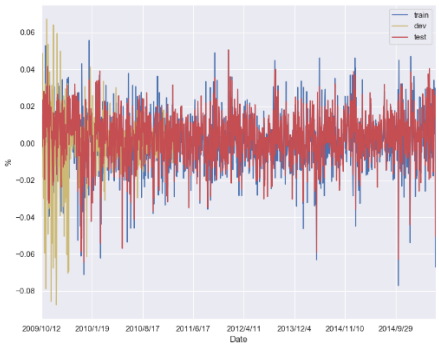
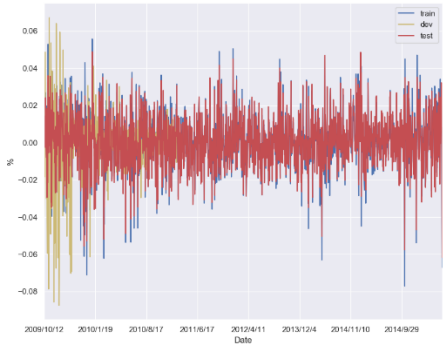
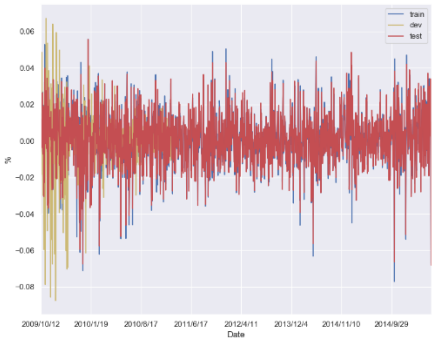
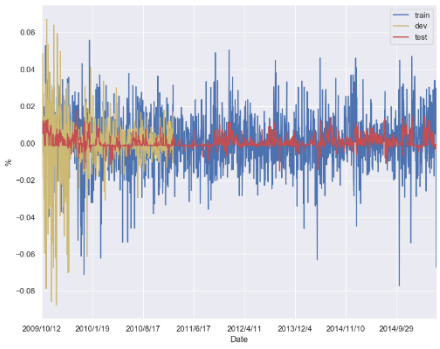
Decision Tree	Random Forest
	
AdaBoost	**Extra Tree**
	
GBDT	**XGBoost**
	
lightBGM	
	

Table 3 Deep Learning Models' Performances for in-sample

Models	MSE	RMSE	MAPE	R^2

MLP_MSE	4.12E-05	0.006422	1188.64	0.816291
MLP_MAE	5.78E-06	0.002404	377.8054	0.974262
LSTM_MSE	8.18E-06	0.00286	274.095	0.963571
LSTM_MAE	1.11E-05	0.003336	1045.729	0.950445
CNN_MSE	0.000213	0.014604	1304.764	0.050086
CNN_MAE	0.000124	0.011142	2244.598	0.447077

Figure 3 Deep Learning Models' Performances for in-sample

MLP (loss function = MAE)	**MLP (loss function = MSE)**
	
LSTM (loss function = MAE)	**LSTM (loss function = MSE)**
	
CNN (loss function = MAE)	**CNN (loss function = MSE)**
	

Append 2 Feature Importance

	Linear Reg	Ridge Reg	Orthogonal Matching Pursuit	Bayesian Ridge Reg	ARD Reg	Theil-Sen Reg
alpha9t-5	0.000000	0.073231	0.091666	-0.531629	-0.004946	-0.002140
alpha9t-3	0.000000	-0.002812	0.091666	0.405462	-0.004946	-0.002140
alpha9t-1	0.000000	-0.040993	0.091666	-1.269013	-0.995382	-0.002140
alpha98t-8	0.000000	0.048245	0.091666	0.801900	-0.004946	-0.002140
alpha98t-2	0.000000	-0.771450	0.091666	0.609926	-0.004946	-0.002135
alpha98t-15	-0.000125	-0.494184	0.091666	-0.810118	-0.004946	-0.608780
alpha98t-1	0.000000	0.876238	0.091666	0.619361	-0.004946	-0.002041
alpha97t-19	0.000000	0.369626	0.091666	0.183154	-0.004946	-0.002139
alpha97t-13	0.000000	-1.881110	0.091666	-0.976052	-0.004946	-0.002075
alpha96t-20	0.000000	-0.287959	0.091666	-0.005013	-0.004946	-0.002140
alpha95t-5	0.000000	0.576942	0.218069	-0.021729	-0.004946	-0.002140
alpha95t-20	0.000000	0.140619	0.091666	-0.021585	-0.004946	-0.002140
alpha95t-14	0.000000	0.163857	0.091666	-0.013480	-0.004946	-0.002140
alpha95t-1	0.000000	0.021749	0.091666	0.017860	-0.004946	-0.002140
alpha94t-9	0.000000	0.060366	0.091666	-0.035136	-0.004946	-0.002140
alpha94t-10	0.000000	0.009945	0.091666	-0.032092	-0.004946	-0.002140
alpha93t-11	0.000000	0.252260	0.091666	-0.679506	-0.004946	-0.002140
alpha89t-6	0.000000	0.030977	0.091666	-0.017309	-0.004946	-0.002140
alpha89t-20	0.000000	0.332725	0.091666	-0.018108	-0.004946	-0.002140
alpha89t-2	0.000000	0.010064	0.091666	-0.016606	-0.004946	-0.002140
alpha89t-19	0.000000	0.078825	0.091666	-0.012966	-0.004946	-0.002140
alpha89t-12	0.000000	-0.188866	0.091666	-0.043146	-0.004946	-0.002140
alpha89t-10	0.000000	0.542514	0.091666	-0.013045	-0.004946	-0.002140
alpha89t-1	0.000000	0.149702	0.091666	-0.264749	-0.004946	-0.002141
alpha87t-5	0.000000	0.153810	0.091666	0.291295	-0.004946	-0.002140
alpha85t-7	0.000000	0.057126	0.091666	-0.069226	-0.004946	-0.002140
alpha85t-4	0.000000	0.123872	0.091666	-0.015945	-0.004946	-0.002140
alpha85t-2	0.000000	-0.025615	0.091666	-0.013159	-0.004946	-0.002140
alpha85t-15	0.000321	-0.092223	0.091666	1.357402	-0.004946	1.562214
alpha84t-7	-0.000001	-0.897487	0.091666	1.391410	-0.004946	-0.002308
alpha84t-20	0.000000	-0.036811	0.091666	1.260091	-0.004946	-0.002136
alpha83t-2	-0.000322	0.125766	0.091666	1.280173	-0.004946	-1.566108
alpha82t-10	-0.000001	0.776421	0.091666	-0.988156	-0.004946	-0.002368
alpha81t-18	0.000000	-0.067757	0.091666	-0.030714	-0.004946	-0.002140
alpha81t-10	0.000000	0.441815	0.091666	-0.015474	-0.004946	-0.002140
alpha80t-2	0.000000	0.025798	0.091666	-0.323257	-0.004946	-0.002140
alpha79t-8	0.000000	-0.190662	0.091666	0.049365	-0.004946	-0.002140
alpha78t-20	0.000000	-1.163769	0.091666	3.268469	1.533697	-0.002139
alpha78t-14	0.000000	-4.325387	-10.340719	-8.999399	-4.513858	-0.002143
alpha78t-13	-0.940358	4.526455	0.091666	3.813354	1.889171	-0.002153
alpha78t	0.940357	4.526455	0.091666	3.813354	1.889171	-0.002126

alpha76t-18	0.000000	0.124551	-0.359579	-0.015552	-0.004946	-0.002140
alpha71t-2	0.000000	4.201904	0.091666	0.926078	-0.004946	-0.002141
alpha71t-16	-0.192867	-4.394479	0.091666	0.250861	-0.004946	-0.002156
alpha71t-14	0.192866	-4.394479	-13.505113	0.250861	-0.004946	-0.002124
alpha71t-1	-1.196735	-0.353804	0.091666	-0.022069	-0.004946	-0.002156
alpha71t	1.196734	-0.353804	0.091666	-0.022069	-0.004946	-0.002124
alpha70t-14	0.000000	-0.036225	0.091666	-0.019721	-0.004946	-0.002140
alpha70t-13	0.000000	-0.112549	0.091666	-0.016662	-0.004946	-0.002140
alpha70t-1	0.000000	0.143356	0.132573	-0.012775	-0.004946	-0.002140
alpha70t	0.000000	0.094500	0.091666	-0.017686	-0.004946	-0.002140
alpha68t-20	0.000000	0.103452	0.091666	-0.014408	-0.004946	-0.002140
alpha67t-3	0.000000	-0.009929	0.091666	0.008592	-0.004946	-0.002140
alpha67t-1	0.000000	-0.099970	0.091666	-0.019861	-0.004946	-0.002140
alpha66t-17	0.000000	-0.136210	0.091666	-0.007064	-0.004946	-0.002140
alpha65t-9	0.000000	0.003249	0.091666	-0.032688	-0.004946	-0.002140
alpha65t-2	0.000000	0.012572	0.091666	0.079935	-0.004946	-0.002140
alpha63t-7	0.000000	-0.033266	0.091666	-0.096458	-0.004946	-0.002140
alpha63t-1	0.000000	-0.021695	0.091666	0.485679	6.493642	-0.002130
alpha60t-19	0.002489	0.016247	0.091666	0.447672	4.114057	12.100712
alpha60t-13	-0.002490	-0.002115	0.091666	0.533358	3.202683	-12.104982
alpha5t-12	0.000000	0.011836	0.091666	0.156120	-0.004946	-0.002140
alpha59t-15	0.000000	-0.020992	0.091666	-0.017634	-0.004946	-0.002140
alpha59t-13	0.000000	0.127478	0.091666	-0.007845	-0.004946	-0.002140
alpha59t	0.000000	-0.136503	0.091666	0.013944	-0.004946	-0.002140
alpha52t-4	0.000000	0.113648	0.091666	-0.014501	-0.004946	-0.002140
alpha52t-16	0.000000	1.182471	0.091666	5.364806	2.261059	-0.002137
alpha4t-5	0.412288	-3.761648	0.091666	-3.613840	-1.453866	-0.002100
alpha49t-12	-0.412289	-3.761648	0.091666	-3.613839	-1.453866	-0.002183
alpha48t-6	0.000000	-0.002902	0.091666	-0.013828	-0.004946	-0.002140
alpha48t-4	0.000000	0.107098	0.091666	0.002156	-0.004946	-0.002140
alpha48t-20	0.000000	0.297330	0.091666	-0.041875	-0.004946	-0.002140
alpha48t-2	0.000000	-0.192398	0.091666	-0.015586	-0.004946	-0.002140
alpha47t-17	0.952303	-0.236221	0.091666	-0.008498	-0.004946	-0.002131
alpha47t-14	-0.952304	-0.236221	0.091666	-0.008498	-0.004946	-0.002149
alpha46t-6	0.000000	-0.023647	0.091666	-0.020213	-0.004946	-0.002140
alpha46t-15	0.000000	0.115978	0.091666	-0.015808	-0.004946	-0.002140
alpha46t-1	0.000000	-0.009331	0.091666	-0.017749	-0.004946	-0.002140
alpha44t-16	0.000000	0.095842	0.091666	-0.027178	-0.004946	-0.002140
alpha43t-1	0.000000	0.145947	0.091666	-0.005638	-0.004946	-0.002140
alpha42t-4	0.000000	0.205358	0.091666	-0.011041	-0.004946	-0.002140
alpha42t-19	0.000000	0.191122	0.091666	0.028033	-0.004946	-0.002140
alpha40t-9	0.000000	-0.044823	0.091666	0.012404	-0.004946	-0.002140
alpha40t-7	0.000000	0.092622	0.091666	-0.029021	-0.004946	-0.002140
alpha40t-6	0.000000	0.261099	-0.116753	-0.015520	-0.004946	-0.002140

alpha40t-14	0.000000	0.387993	0.091666	-0.020358	-0.004946	-0.002140
alpha40t-11	0.000000	0.126620	0.091666	0.161004	-0.004946	-0.002140
alpha40t-10	0.000000	0.262936	0.091666	0.363996	-0.004946	-0.002140
alpha3t-6	0.000000	0.077784	0.091666	-0.020896	-0.004946	-0.002140
alpha3t-4	0.000000	0.027165	0.091666	0.012681	-0.004946	-0.002140
alpha3t-16	0.000000	-0.093703	0.091666	0.280041	-0.004946	-0.002140
alpha3t-13	0.000000	0.120394	0.091666	-0.016157	-0.004946	-0.002140
alpha3t-10	0.000000	0.012608	0.091666	-0.013207	-0.004946	-0.002140
alpha38t-5	0.000000	0.855917	0.091666	-4.458913	-2.159460	-0.002143
alpha38t-4	0.000000	3.010475	0.091666	5.364169	2.812548	-0.002138
alpha38t-2	0.000000	-0.638273	0.091666	-1.622646	-1.224397	-0.002141
alpha38t-19	0.000000	3.024213	0.091666	5.023991	2.273100	-0.002138
alpha38t-16	0.000000	-0.035390	0.091666	-0.010015	-0.004946	-0.002140
alpha38t-14	0.000000	0.009963	0.091666	-0.018245	-0.004946	-0.002140
alpha38t-10	0.000000	0.160033	0.091666	-0.025310	-0.004946	-0.002140
alpha38t	0.000000	-0.194545	0.091666	-0.025009	-0.004946	-0.002140
alpha37t-8	0.000000	-0.092711	0.091666	-0.015646	-0.004946	-0.002140
alpha37t-20	0.000000	0.313083	0.091666	-0.036032	-0.004946	-0.002140
alpha37t-12	0.000000	0.190805	0.091666	-0.019717	-0.004946	-0.002140
alpha35t-4	0.000000	-0.134124	0.091666	-0.020108	-0.004946	-0.002140
alpha34t-1	0.000000	0.178915	0.091666	0.040124	-0.004946	-0.002140
alpha34t	-0.000001	0.126745	0.091666	-0.055204	-0.004946	-0.001974
alpha2t-19	0.000000	-0.000670	0.091666	0.249858	-0.004946	-0.002541
alpha2t-17	0.000000	0.074299	0.091666	-0.264085	-0.004946	-0.001824
alpha2t-14	0.000000	0.196191	0.091666	0.008066	-0.004946	-0.002140
alpha29t-8	0.000000	-0.206027	0.091666	-0.038749	-0.004946	-0.002140
alpha29t	0.000000	-0.081869	-0.487272	-0.549683	-0.004946	-0.002140
alpha28t-4	0.000000	0.427288	0.091666	0.044717	-0.004946	-0.002140
alpha25t-19	0.000000	0.109351	1.719546	0.639656	-0.004946	-0.002140
alpha24t-8	0.000000	-0.078924	0.091666	-0.017230	-0.004946	-0.002140
alpha24t-16	0.000000	-0.011582	0.091666	-0.015693	-0.004946	-0.002140
alpha24t-1	0.000000	-0.486221	0.091666	-0.165077	-0.004946	-0.002140
alpha23t-7	0.000000	-0.185048	0.091666	0.030258	-0.004946	-0.002140
alpha23t-6	0.000000	-0.529561	0.091666	-0.613884	-0.004946	-0.002140
alpha23t-20	0.000000	0.181565	0.091666	-0.000142	-0.004946	-0.002140
alpha23t-2	0.000000	-0.580723	-0.948936	0.008031	-0.004946	-0.002140
alpha23t-12	0.000000	0.018994	0.091666	-0.020693	-0.004946	-0.002140
alpha23t-10	0.000000	0.923967	0.091666	0.035116	-0.004946	-0.002140
alpha23t-1	0.000000	2.572081	0.091666	-0.298462	-0.004946	-0.002140
alpha23t	0.000000	0.247105	0.091666	-0.105936	-0.004946	-0.002140
alpha22t-6	0.106569	-5.214397	0.091666	-0.470342	-0.004946	-0.002122
alpha22t-18	-0.106570	-5.214397	-1.688975	-0.470342	-0.004946	-0.002159
alpha22t	0.000000	-0.417845	0.091666	-0.528330	-0.004946	-0.002140
alpha21t-10	0.000000	-0.436566	0.091666	-0.005292	-0.004946	-0.002140

alpha20t-20	0.000000	0.043641	0.091666	0.012365	-0.004946	-0.002140
alpha20t-15	0.000000	0.560727	0.091666	-0.050077	-0.004946	-0.002140
alpha20t-13	0.000000	0.160461	0.091666	-0.005462	-0.004946	-0.002140
alpha19t-3	0.000000	-0.100863	0.091666	-0.021661	-0.004946	-0.002140
alpha191t-5	0.000000	0.932824	0.091666	-0.018178	-0.004946	-0.002140
alpha191t-4	0.038619	-1.775560	0.091666	-0.050733	-0.004946	-0.002103
alpha191t-3	-0.038620	-1.775560	-0.161963	-0.050733	-0.004946	-0.002177
alpha191t-2	0.000000	-0.255884	0.091666	-0.040036	-0.004946	-0.002140
alpha191t-14	0.000000	0.095115	0.091666	-0.010834	-0.004946	-0.002140
alpha191t-12	0.000000	0.178821	0.091666	0.048426	-0.004946	-0.002140
alpha191t-11	0.000000	0.344734	0.091666	-0.057616	-0.004946	-0.002140
alpha191t-1	0.000000	-0.079305	0.091666	-0.035801	-0.004946	-0.002140
alpha191t	0.000000	-0.008097	0.091666	0.007301	-0.004946	-0.002140
alpha18t-3	0.000000	0.085046	0.091666	-0.017174	-0.004946	-0.002140
alpha189t-2	0.000000	0.168085	0.091666	-0.018170	-0.004946	-0.002140
alpha189t-19	0.000000	-0.010117	0.091666	-0.032541	-0.004946	-0.002140
alpha189t-1	0.000000	-0.057944	0.091666	-0.017996	-0.004946	-0.002140
alpha189t	0.000000	0.261421	0.091666	-0.014238	-0.004946	-0.002140
alpha187t-2	0.000000	0.292659	0.169057	-0.021582	-0.004946	-0.002140
alpha187t-1	0.000000	-0.044137	0.091666	-0.029670	-0.004946	-0.002140
alpha186t-14	0.000000	0.279472	0.091666	0.099968	-0.004946	-0.002140
alpha180t-9	0.000000	0.456993	0.091666	-0.184138	-0.004946	-0.002140
alpha180t-4	0.000000	-0.456899	-0.114983	0.020160	-0.004946	-0.002140
alpha180t-2	0.000000	-0.511383	0.091666	0.003516	-0.004946	-0.002140
alpha180t-17	0.000000	-0.014784	0.091666	-0.024550	-0.004946	-0.002140
alpha178t-5	0.059162	1.539487	0.091666	-0.015826	-0.004946	-0.002143
alpha178t-4	-0.059162	1.539487	0.091666	-0.015826	-0.004946	-0.002137
alpha178t-14	0.000000	0.024975	0.091666	-0.015689	-0.004946	-0.002140
alpha178t-13	0.000000	-0.035271	0.091666	-0.016344	-0.004946	-0.002140
alpha178t-1	0.000000	0.245926	0.091666	-0.005802	-0.004946	-0.002140
alpha178t	0.000000	-0.246032	0.091666	-0.016294	-0.004946	-0.002140
alpha176t-9	0.000000	-0.076832	0.091666	-1.051721	-7.798798	-0.002136
alpha176t-6	0.000000	0.076076	0.091666	0.215460	-0.004946	-0.002180
alpha176t-16	0.000000	0.072486	0.091666	-0.431175	-0.004946	-0.001780
alpha176t-15	-0.000001	-0.030093	0.091666	0.240635	-0.004946	-0.002446
alpha176t-11	0.000000	-0.037462	0.091666	1.158590	-0.004946	-0.002139
alpha174t-6	-0.000001	-0.373839	0.091666	0.614114	5.576211	-0.002495
alpha174t-18	0.000000	0.008728	0.091666	-0.680076	0.382440	-0.002134
alpha173t-9	0.000000	0.044758	0.091666	-0.375290	-0.340046	-0.002151
alpha173t-7	0.000000	0.075916	0.091666	-1.822388	1.411358	-0.002137
alpha173t-6	0.000000	0.031674	0.091666	0.267515	-0.004946	-0.002141
alpha173t-2	0.000000	0.023500	0.018916	-0.021393	-0.004946	-0.001814
alpha173t-19	0.000000	0.167197	0.091666	-0.026119	-0.004946	-0.002140
alpha173t-17	0.000000	-0.008953	0.091666	-0.029127	-0.004946	-0.002140

alpha173t-13	0.000000	0.053020	0.091666	-0.014566	-0.004946	-0.002140
alpha173t-11	0.000000	0.010018	0.091666	-0.012731	-0.004946	-0.002140
alpha173t-1	0.000000	-0.104051	0.091666	-0.019919	-0.004946	-0.002140
alpha173t	0.000000	0.010007	0.091666	-0.005824	-0.004946	-0.002140
alpha172t-15	0.000000	0.013623	0.091666	0.460587	-0.004946	-0.002140
alpha171t-6	0.000000	-0.040448	0.091666	-0.159588	-0.004946	-0.002140
alpha171t-11	0.000000	0.107421	0.091666	-0.027133	-0.004946	-0.002140
alpha171t	0.026851	0.095076	0.091666	-0.013091	-0.004946	-0.002111
alpha170t-10	-0.026852	0.095076	0.091666	-0.013091	-0.004946	-0.002169
alpha170t-1	0.000000	0.008180	0.091666	-0.008365	-0.004946	-0.002140
alpha170t	0.000000	0.127145	0.091666	-0.019461	-0.004946	-0.002140
alpha169t-6	0.000000	0.462414	0.091666	-0.000851	-0.004946	-0.002140
alpha169t-5	-0.037522	-0.883055	0.091666	-0.020725	-0.004946	-0.002153
alpha169t-10	0.037521	-0.883055	-0.106417	-0.020725	-0.004946	-0.002128
alpha169t-1	0.000000	0.197854	0.091666	0.083856	-0.004946	-0.002140
alpha169t	0.000000	0.120451	0.091666	0.050207	-0.004946	-0.002141
alpha168t-4	0.000000	0.087089	0.091666	-0.003182	-0.004946	-0.002140
alpha167t-8	0.000000	0.132216	0.091666	0.003566	-0.004946	-0.002140
alpha164t-20	0.000000	0.462407	0.091666	0.004159	-0.004946	-0.002140
alpha164t-2	0.000000	-1.096155	-0.171999	-0.035563	-0.004946	-0.002140
alpha164t-18	0.000000	0.099710	0.091666	0.009137	-0.004946	-0.002140
alpha164t-14	0.000000	-1.181031	0.091666	-0.111703	-0.004946	-0.002140
alpha164t	0.000000	0.006789	0.091666	-0.030390	-0.004946	-0.002140
alpha162t-8	0.000000	-0.101218	0.091666	-0.013811	-0.004946	-0.002140
alpha161t-19	0.000000	-0.023157	0.091666	0.379964	-0.004946	-0.002140
alpha161t-11	0.000000	-0.023939	0.091666	-0.613909	-0.004946	-0.002141
alpha161t-1	0.000000	-0.096208	0.091666	0.124639	-0.004946	-0.002140
alpha15t-9	0.000000	0.118867	0.091666	-0.005216	-0.004946	-0.002140
alpha15t-2	0.000000	0.839119	0.091666	-0.029568	-0.004946	-0.002140
alpha15t-15	0.000000	-0.910442	0.091666	0.019194	-0.004946	-0.002140
alpha15t-13	0.000000	0.050103	0.091666	-0.136585	-0.004946	-0.002140
alpha159t-3	0.000000	0.057815	0.091666	0.008626	-0.004946	-0.002140
alpha158t-9	0.000000	0.657902	-0.196867	-0.130349	-0.004946	-0.002140
alpha158t-5	0.000000	0.138136	0.091666	-0.044506	-0.004946	-0.002140
alpha158t-4	0.000000	0.434674	0.091666	-0.024615	-0.004946	-0.002140
alpha158t-3	0.000000	-0.047740	0.091666	-0.023119	-0.004946	-0.002140
alpha158t-15	-0.045689	1.353724	0.091666	-0.012803	-0.004946	-0.002152
alpha158t-14	0.045688	1.353724	0.091666	-0.012803	-0.004946	-0.002128
alpha158t-11	0.000000	-0.312136	0.091666	-0.034569	-0.004946	-0.002140
alpha158t-10	0.000000	-0.021988	0.091666	-0.012720	-0.004946	-0.002140
alpha155t-9	0.000000	-0.956036	0.091666	-0.046435	-0.004946	-0.002140
alpha155t-6	0.772900	0.433038	0.091666	-0.009339	-0.004946	-0.002145
alpha155t-2	-0.772901	0.433038	0.091666	-0.009339	-0.004946	-0.002135
alpha155t-18	0.000000	-0.484281	0.091666	-0.014323	-0.004946	-0.002140

alpha153t-8	0.000000	-0.678383	0.091666	0.068280	-0.004946	-0.002140
alpha153t-19	0.000000	-0.064956	0.091666	-0.011254	-0.004946	-0.002140
alpha153t-15	0.000000	-2.033786	0.091666	-0.052753	-0.004946	-0.002140
alpha153t-1	0.167697	3.349633	0.091666	0.047350	-0.004946	-0.002154
alpha152t-4	-0.167698	3.349633	0.091666	0.047350	-0.004946	-0.002126
alpha150t-8	0.000000	-0.288153	0.091666	-0.069099	-0.004946	-0.002140
alpha150t-6	0.000000	0.110241	0.091666	-0.045806	-0.004946	-0.002140
alpha150t-4	0.000000	0.383750	0.091666	-0.026540	-0.004946	-0.002140
alpha150t-3	0.000000	0.141657	0.091666	-0.024443	-0.004946	-0.002140
alpha150t-19	0.000000	0.168756	0.091666	-0.043932	-0.004946	-0.002140
alpha150t-1	0.000000	-0.095719	0.091666	0.114809	-0.004946	-0.002140
alpha14t-9	0.000000	-0.169181	0.091666	0.644180	-8.229403	-0.002140
alpha14t-7	0.000000	-0.010875	0.091666	-0.965476	-2.512170	-0.002142
alpha14t-5	0.000000	-0.145501	0.091666	-0.025485	-0.004946	-0.002140
alpha14t-4	0.000000	0.366409	0.091666	-0.055230	-0.004946	-0.002140
alpha14t-3	0.000000	-0.229610	0.091666	-0.090709	-0.004946	-0.002140
alpha145t-18	0.000000	-0.039800	0.091666	0.122244	-0.004946	-0.002140
alpha145t-1	0.000000	0.189118	0.091666	0.124072	-0.004946	-0.002140
alpha145t	0.000000	0.215605	0.091666	-0.052289	-0.004946	-0.002140
alpha13t-3	0.000000	0.024719	0.091666	-0.100166	-0.004946	-0.002140
alpha13t-20	0.000000	-0.677239	0.091666	-0.042382	-0.004946	-0.002140
alpha13t-19	0.000000	0.550602	0.091666	-0.051799	-0.004946	-0.002140
alpha13t-18	0.000000	0.304002	0.091666	0.004180	-0.004946	-0.002140
alpha139t-7	0.000000	-0.314411	0.091666	-0.025253	-0.004946	-0.002140
alpha139t-4	-0.296407	-0.059821	0.091666	0.004819	-0.004946	-0.002107
alpha136t-8	0.296406	-0.059821	0.091666	0.004819	-0.004946	-0.002173
alpha136t	0.000120	0.022798	0.091666	-0.202877	-0.004946	0.583091
alpha135t-4	0.000000	0.226967	0.091666	0.253518	-0.004946	-0.002140
alpha135t-20	0.000136	-0.268513	0.091666	0.804311	-0.682178	0.661075
alpha135t-19	0.000000	-0.087490	0.091666	-0.037708	-0.004946	-0.002140
alpha129t-2	0.000000	0.002217	0.091666	-0.042543	-0.004946	-0.002140
alpha129t-13	0.000000	0.176898	0.091666	-0.018771	-0.004946	-0.002140
alpha126t-7	0.000000	-0.128331	0.091666	-0.016365	-0.004946	-0.002140
alpha126t-6	0.000000	0.564478	0.091666	-0.025940	-0.004946	-0.002140
alpha126t-4	0.000000	-0.048027	0.091666	-0.006773	-0.004946	-0.002140
alpha126t-3	-12.046189	-0.688673	0.091666	0.008713	-0.004946	-0.002136
alpha126t-2	12.046188	-0.688673	0.091666	0.008713	-0.004946	-0.002144
alpha126t-17	0.000000	0.043752	0.091666	-0.018013	-0.004946	-0.002140
alpha126t-16	0.000000	0.037597	0.091666	-0.016184	-0.004946	-0.002140
alpha126t-13	0.000000	-0.067180	0.091666	-0.016321	-0.004946	-0.002140
alpha126t-12	0.000000	-0.023957	0.091666	0.004607	-0.004946	-0.002140
alpha126t-1	0.000000	0.284990	0.091666	-0.043404	-0.004946	-0.002140
alpha126t	0.000000	0.085329	0.091666	0.002509	-0.004946	-0.002140
alpha124t-3	0.000000	-0.241857	0.091666	-0.270765	-0.004946	-0.002140

alpha124t-17	0.271610	0.797579	0.091666	-0.634508	-0.360261	-0.002141
alpha124t-1	-0.271611	0.797579	0.091666	-0.634508	-0.360261	-0.002141
alpha124t	0.000000	-0.034540	0.091666	-0.157801	-0.004946	-0.002140
alpha122t-11	0.000000	-0.164094	0.091666	-1.874725	-0.004946	-0.002140
alpha11t-8	0.000000	0.113774	0.091666	1.406354	-0.004946	-0.002139
alpha11t-7	0.000000	0.220351	0.091666	0.705540	-0.976282	-0.002141
alpha11t-4	0.000000	-0.219183	0.091666	1.639713	0.571332	-0.002141
alpha11t-2	0.000000	-0.895311	0.091666	0.592555	-0.004946	-0.002142
alpha11t-17	0.000000	0.782659	0.091666	-1.433330	-0.004946	-0.002139
alpha11t-13	0.000000	0.004463	0.091666	0.016315	-0.004946	-0.002140
alpha11t	0.000000	0.123322	0.091666	-0.088136	-0.004946	-0.002140
alpha118t-13	0.000000	0.392808	0.091666	0.009449	-0.004946	-0.002140
alpha117t-19	0.000000	0.088736	0.091666	-0.051613	-0.004946	-0.002140
alpha117t-14	0.000000	0.165924	0.091666	-0.005915	-0.004946	-0.002140
alpha117t-12	0.000000	-0.119188	0.091666	-0.020801	-0.004946	-0.002140
alpha116t-16	0.000000	-0.055594	0.091666	-0.025266	-0.004946	-0.002140
alpha114t-7	-0.388758	0.111672	0.091666	-0.003084	-0.004946	-0.002141
alpha114t-3	0.388757	0.111672	0.091666	-0.003084	-0.004946	-0.002140
alpha114t-2	0.000000	-0.035344	0.091666	-0.018892	-0.004946	-0.002140
alpha114t-18	0.000000	-0.108928	0.091666	-0.019080	-0.004946	-0.002140
alpha114t-1	0.000000	-0.042024	0.091666	-0.519065	-0.004946	-0.002140
alpha114t	0.000000	-0.242703	0.091666	-0.153963	-0.004946	-0.002140
alpha112t-2	0.000000	-0.079467	0.091666	-2.512769	-0.004946	-0.002141
alpha111t-18	0.000000	0.005773	0.091666	-0.015090	-0.004946	-0.002140
alpha111t-1	0.000000	-0.031813	0.091666	-0.021004	-0.004946	-0.002140
alpha111t	0.000000	0.155071	0.091666	-0.011272	-0.004946	-0.002140
alpha109t-7	0.000000	-0.232865	0.091666	-0.002739	-0.004946	-0.002140
alpha109t-1	0.000000	-0.084596	0.091666	-0.087827	-0.004946	-0.002140
alpha106t-3	0.000000	-0.050797	0.091666	0.060448	-0.004946	-0.002140
alpha104t-7	0.000000	-0.072901	0.091666	0.250035	-0.004946	-0.002140
alpha104t-1	0.000000	0.224700	0.091666	-0.015236	-0.004946	-0.002140
alpha102t-14	0.000000	0.109467	0.091666	-0.021017	-0.004946	-0.002140
alpha102t-12	0.000000	0.154522	0.091666	-0.030777	-0.004946	-0.002140
alpha102t-1	0.000000	-0.109915	0.091666	0.013379	-0.004946	-0.002140
alpha102t	0.000000	0.171435	0.091666	-0.011894	-0.004946	-0.002140
alpha100t-19	0.000000	-0.145383	0.091666	-0.015101	-0.004946	-0.002140
alpha100t-13	0.000000	-0.329899	0.091666	-0.047549	-0.004946	-0.002140
alpha100t-10	0.000000	0.070490	0.091666	-0.014934	-0.004946	-0.002140
alpha100t-1	0.000000	0.053440	0.091666	-0.037452	-0.004946	-0.002140

Append 3 Net Asset Value for Portfolios

Date	long-short		long		short	
	monthly	nav	monthly	nav	monthly	nav
2017/12/5	-0.0916	0.9084	-0.0005	0.9995	0.0911	1.0911

2018/1/4	-0.0961	0.8211	0.0383	1.0378	0.1344	1.2377
2018/2/2	0.1806	0.9694	0.0088	1.0469	-0.1718	1.0251
2018/3/12	0.1890	1.1527	0.0093	1.0566	-0.1798	0.8408
2018/4/12	-0.1768	0.9489	-0.0403	1.0140	0.1365	0.9556
2018/5/15	-0.0409	0.9101	0.0118	1.0260	0.0527	1.0059
2018/6/13	-0.1231	0.7981	-0.0346	0.9906	0.0885	1.0950
2018/7/13	0.1774	0.9397	-0.0639	0.9273	-0.2413	0.8307
2018/8/13	0.0455	0.9824	-0.0252	0.9039	-0.0707	0.7720
2018/9/11	0.1003	1.0810	-0.0520	0.8569	-0.1523	0.6544
2018/10/18	0.1277	1.2190	-0.0921	0.7780	-0.2197	0.5106
2018/11/16	-0.0045	1.2135	0.1218	0.8728	0.1263	0.5751
2018/12/17	0.0821	1.3131	-0.0397	0.8381	-0.1219	0.5050
2019/1/17	-0.0787	1.2098	-0.0179	0.8231	0.0608	0.5357
2019/2/22	0.0380	1.2558	0.1438	0.9414	0.1058	0.5924
2019/3/25	0.0437	1.3107	0.0991	1.0347	0.0554	0.6252
2019/4/24	0.0986	1.4400	0.0463	1.0827	-0.0523	0.5925
2019/5/28	0.0998	1.5837	-0.1021	0.9721	-0.2019	0.4728
2019/6/27	-0.0010	1.5822	0.0365	1.0075	0.0374	0.4905
2019/7/26	0.0887	1.7225	0.0055	1.0131	-0.0831	0.4498
2019/8/26	0.1343	1.9538	-0.0235	0.9893	-0.1578	0.3788
2019/9/25	0.0242	2.0011	0.0351	1.0240	0.0108	0.3829
2019/10/31	0.1619	2.3251	-0.0126	1.0110	-0.1746	0.3161