Quantamental Trading Strategy: Using Machine Learning Methods on Fundamental Stock Indicators

Group Name

Quantamental Capital:

Hongyang (Bruce) Yang
Haojian Zhang
Lan Jiang
Wei Zhou
Yanbing Liu

Prepared for Math 5220 Quant Methods for Investment Management

Courtesy of Dr. Alex Greyserman

Abstract:
This report provides a practical approach to using machine learning methodology selecting S&P 500 stocks based on quarterly fundamental financial ratios. Linear regression, random forest, ridge regression, stepwise regression, and generalized boosted regression model serve as the key algorithms. The predictor variables are the 20 financial ratios (e.g., EPS, ROA, ROE, etc). The response variable is the log return of next quarter price. The stocks selected are then used to construct a portfolio applying minimum variance theory. Finally, we test the portfolio by implementing a quarterly buy-and-hold strategy. Our results show that the portfolio of the selected stocks by the machine learning methods outperform the long-only strategy on the S&P 500 index, thus illustrating that applying machine learning algorithms to fundamental data contribute to better stock selection.

Section 1: Introduction

The goal of our project is to develop and test an automatic trading system that uses advance machine learning algorithms on traditional financial indicators such as P/E ratios and profit margins etc. In this project, we design a trading strategy that makes the quarterly investment on S&P 500 stocks, uses long-only strategy and takes no leverage. The introduction part will explain the motivation of our investment, and provide the general structure of our strategy.

The first step of our portfolio construction process is to conduct stock selection based on quarterly fundamental financial ratios related to all companies listed on S&P 500 index. The reason of using fundamental financial ratios is that many fundamental financial factors like PE ratio and profit margin issued by public companies quarterly indicate overall profitability, operating efficiency, capital structure, ability of generating future cash flows and other valuable information of the corresponding companies, and thus should have predictive power on reflecting related stocks' future performance. We want to extract this valuable information from these fundamental factors, and base our stock selection process on this information. In our project, 20 financial factors: Revenue Growth, EPS, ROA, ROE, P/E ratio, P/S ratio, Net Profit Margin, Gross Profit Margin, Operating Margin, Price to Book ratio, Price to Cash Flow ratio, Cash Ratio, Enterprise Multiple, EV/CFO, Long Term Debt/Total Assets, Working Capital Ratio, Debt to Equity ratio, Quick Ratio, Days Sales of Inventory, and Days Payable Outstanding are calculated for each stock listed on S&P 500 over 27-year time interval (1990-2017) and considered in our stock selection process.

Every quarter (Q1-3/31, Q2-6/30, Q3-9/30, Q4-12/31) we have companies' financial reports, and we rebalance portfolio quarterly by selecting stocks according to the above 20 financial factors. The traditional way of doing fundamental analysis is using fundamental factors like P/E, price/sales (P/S) ratio, and PEG ratio directly to rank and pick stocks. But this could be inconsiderate, because this method has not addressed autocorrelation between factors, and it fails to justify the weights assigned to each factor when ranking stocks. In our project, we use machine learning algorithms (Linear Regression, Random Forest, Ridge, Stepwise Regression, and Generalized Boosting Regression) to assign weights to each factor dynamically, and select top 20% stocks each quarter based on the ranking of predicted returns generated by the best performing machine learning algorithm over the past 5 years of training periods before each rebalancing day on a rolling basis.

After we select stocks for our portfolio on each rebalancing day, we test asset allocation methodologies: Mean-Variance, Min-Variance, and Equally-Weighted-Allocation on selected stocks using in sample data (1990-2007). We pick minimum-variance as our portfolio allocation method based on the sharpe ratio of the in-sample period.

Finally, we compare the P&L of our strategy with SPX, and the equally weighted benchmark, summarize the competency of our strategy, and list areas for potential future improvements.

Section 2: Data

2.1 Data Source

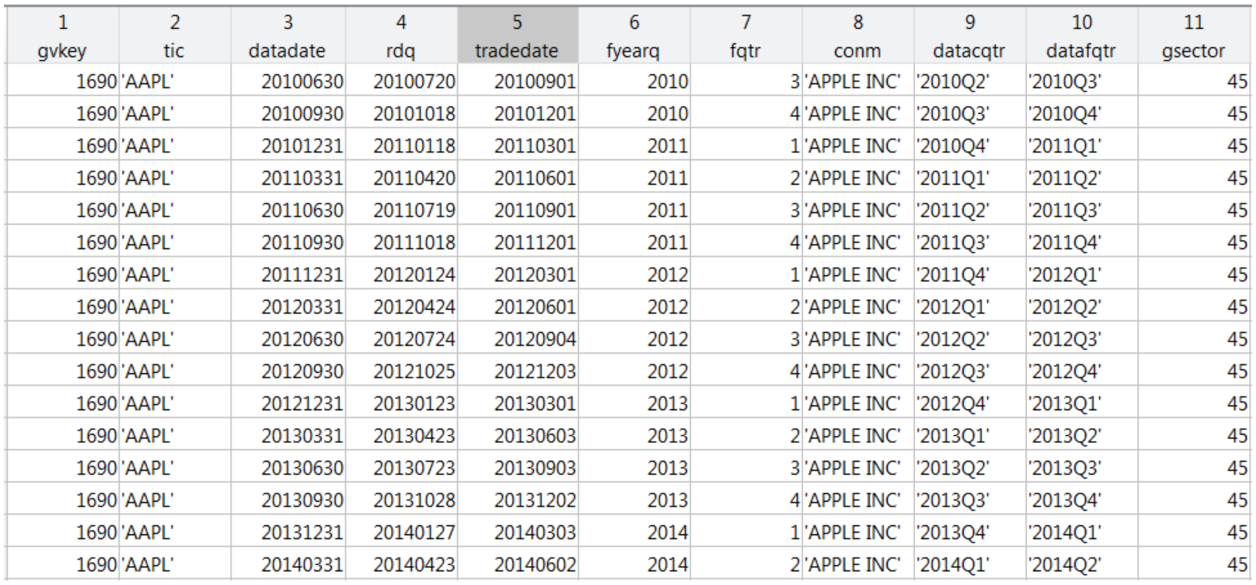
The data for this project is mainly taken from Compustat database accessed through Wharton Research Data Services (WRDS). The dataset used here consists of a period of 27 years that goes from 01/01/1990 to 06/01/2017. We use all historical S&P 500 component stocks (about 1142 stocks) including delisted or bankrupt companies. The adjusted close price goes on a daily basis (trading days) and generates 6,438,964 observations. The fundamental data goes on a quarterly basis and generates 91,216 observations.

2.2 Data Process and Clean

In order to build the dataset for machine learning to train, we select top 20 most popular financial ratios and calculated these factors from the fundamental raw data from the WRDS; *Appendix A* shows the detailed formula to calculate each factor. Also, in order to build a sector-neutral portfolio, we split the dataset into 11 sectors (10-Energy, 15-Materials, 20-Industrials, 25-Consumer Discretionary, 30-Consumer Staples, 35-Health Care, 40-Financials, 45-Information Technology, 50-Telecommunication services, 55-Utilities, 60-Real Estate). Last, we handle missing data in each sector separately: if one factor has more than 5% missing data, we delete this factor; if a certain stock generates the most missing data, we delete this stock, and thus we remove 46 stocks. Therefore, after performing these steps, the overall missing data is reduced to less than 7% of each sector. Finally, we delete this 7% missing data.

2.3 Trade Dates Adjustments

We also adjust the trade date by 2 months lag after the standard quarter end date. For quarter end date between 01/01 and 03/31, we adjust our trade date to 06/01. For quarter end date between 04/01 and 06/30, we adjust our trade date to 09/01. Similarly, for quarter end date between 07/01 and 09/30, we adjust our trade date to 12/01. Lastly, for quarter end date between 10/01 and 12/31, we adjust our trade date to next year 03/01. In this way, whether a company has a standard quarter end date or a non-standard quarter end date, we will have its earnings reported by our adjusted trade dates. For example, Apple released its earnings report on 2010/07/20 for the quarter end date 2010/06/30, we will use this data to trade on 2010/09/01.



1	2	3	4	5	6	7	8	9	10	11
gvkey	tic	datadate	rdq	tradedate	fyearq	fqtr	conm	datacqtr	datafqtr	gsector
1690	'AAPL'	20100630	20100720	20100901	2010	3	'APPLE INC'	'2010Q2'	'2010Q3'	45
1690	'AAPL'	20100930	20101018	20101201	2010	4	'APPLE INC'	'2010Q3'	'2010Q4'	45
1690	'AAPL'	20101231	20110118	20110301	2011	1	'APPLE INC'	'2010Q4'	'2011Q1'	45
1690	'AAPL'	20110331	20110420	20110601	2011	2	'APPLE INC'	'2011Q1'	'2011Q2'	45
1690	'AAPL'	20110630	20110719	20110901	2011	3	'APPLE INC'	'2011Q2'	'2011Q3'	45
1690	'AAPL'	20110930	20111018	20111201	2011	4	'APPLE INC'	'2011Q3'	'2011Q4'	45
1690	'AAPL'	20111231	20120124	20120301	2012	1	'APPLE INC'	'2011Q4'	'2012Q1'	45
1690	'AAPL'	20120331	20120424	20120601	2012	2	'APPLE INC'	'2012Q1'	'2012Q2'	45
1690	'AAPL'	20120630	20120724	20120904	2012	3	'APPLE INC'	'2012Q2'	'2012Q3'	45
1690	'AAPL'	20120930	20121025	20121203	2012	4	'APPLE INC'	'2012Q3'	'2012Q4'	45
1690	'AAPL'	20121231	20130123	20130301	2013	1	'APPLE INC'	'2012Q4'	'2013Q1'	45
1690	'AAPL'	20130331	20130423	20130603	2013	2	'APPLE INC'	'2013Q1'	'2013Q2'	45
1690	'AAPL'	20130630	20130723	20130903	2013	3	'APPLE INC'	'2013Q2'	'2013Q3'	45
1690	'AAPL'	20130930	20131028	20131202	2013	4	'APPLE INC'	'2013Q3'	'2013Q4'	45
1690	'AAPL'	20131231	20140127	20140303	2014	1	'APPLE INC'	'2013Q4'	'2014Q1'	45
1690	'AAPL'	20140331	20140423	20140602	2014	2	'APPLE INC'	'2014Q1'	'2014Q2'	45

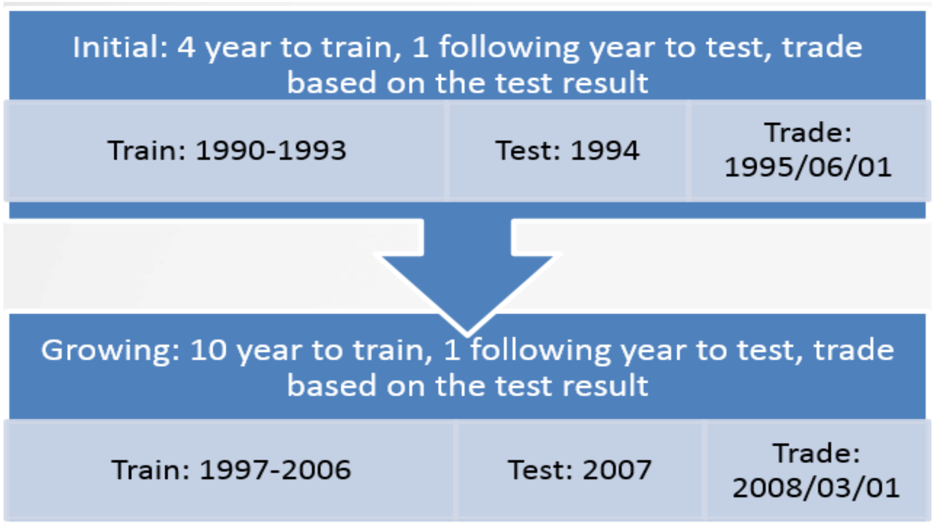
In addition, we delete all records that indicate a release date (rdq) after our trade date, this including about 0.84% of the dataset. So we assure that on our trade date, 99% of the companies having their earnings reports readily to be used.

Section 3: Forecasting Model

Our machine learning forecasting model contains 5 algorithms: Linear Regression, Random Forest, Ridge Regression, Stepwise Regression, and Generalized Boosted Regression Model. The formula and the loss function of these algorithms are presented in *Appendix B*. We utilize packages in R (glmnet, tree, randomForest, gbm, and caret) to implement those algorithms.

3.1 Rolling Windows

We use a maximum 10-year growing rolling windows to train the model, followed by an one-year rolling window to test, and then we trade according to the test results. The training-testing-trading cycle of our strategy can be summarized by the following chart:


Initial: 4 year to train, 1 following year to test, trade based on the test result
Train: 1990-1993
Test: 1994
Trade: 1995/06/01
Growing: 10 year to train, 1 following year to test, trade based on the test result
Train: 1997-2006
Test: 2007
Trade: 2008/03/01

3.2 Model Selection Approach

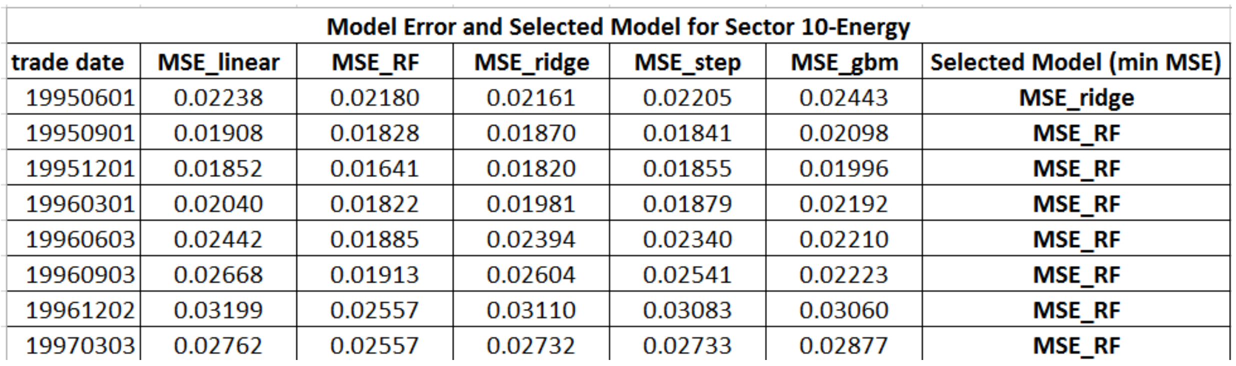
We have been through many processes to land on these five models. First, we want supervised regression models that have already been developed in R or Python packages, so that we can access to these packages to adjust relevant parameters efficiently. Second, we need a feature selection model to remove undesirable features, thus reducing the overfitting issues, improving model accuracy and expediting the training procedure. Third, we prefer to have a white-box model that we can observe every single factor with its coefficients in our model, and it is also easier for portfolio managers who do not know machine learning to comprehend. Essentially, the concept of Quantamental means to combine quant methods and fundamental analysis together.

3.3 Model Implementation

Our implementation can be concluded in the following 4 steps:

1. Train and test to get a model error for each of the 5 models.

2. Choose the model that has the lowest MSE in that certain period. For example, in the first trade period 1995/06/01 of Energy Sector, we choose Ridge regression as our model to select stocks. In the second trade period 1995/09/01, we choose Random Forest as our model to select stocks.

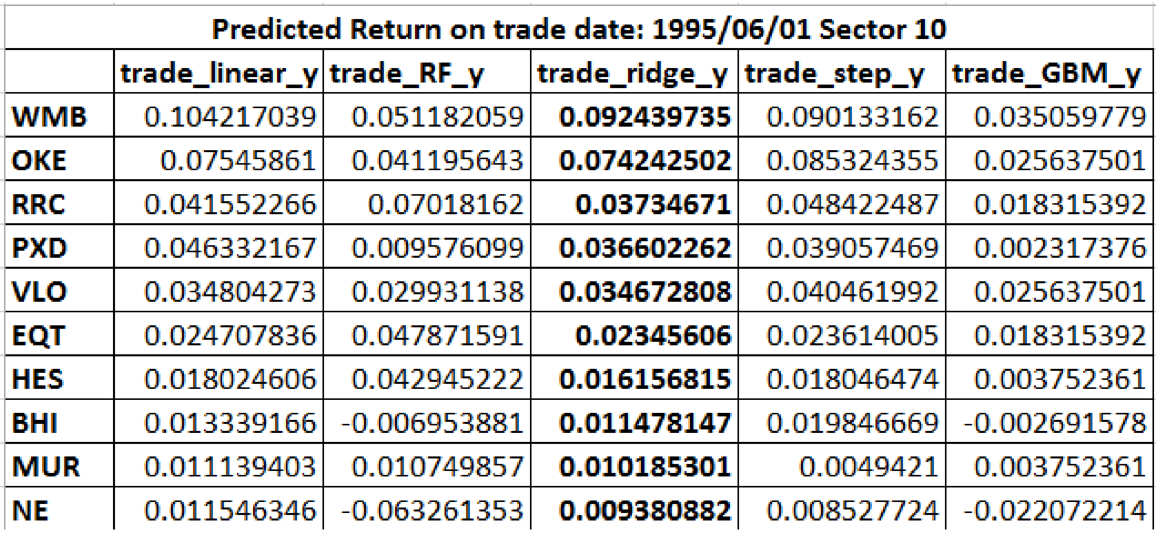


Model Error and Selected Model for Sector 10-Energy

trade date	MSE_linear	MSE_RF	MSE_ridge	MSE_step	MSE_gbm	Selected Model (min MSE)
19950601	0.02238	0.02180	0.02161	0.02205	0.02443	**MSE_ridge**
19950901	0.01908	0.01828	0.01870	0.01841	0.02098	**MSE_RF**
19951201	0.01852	0.01641	0.01820	0.01855	0.01996	**MSE_RF**
19960301	0.02040	0.01822	0.01981	0.01879	0.02192	**MSE_RF**
19960603	0.02442	0.01885	0.02394	0.02340	0.02210	**MSE_RF**
19960903	0.02668	0.01913	0.02604	0.02541	0.02223	**MSE_RF**
19961202	0.03199	0.02557	0.03110	0.03083	0.03060	**MSE_RF**
19970303	0.02762	0.02557	0.02732	0.02733	0.02877	**MSE_RF**

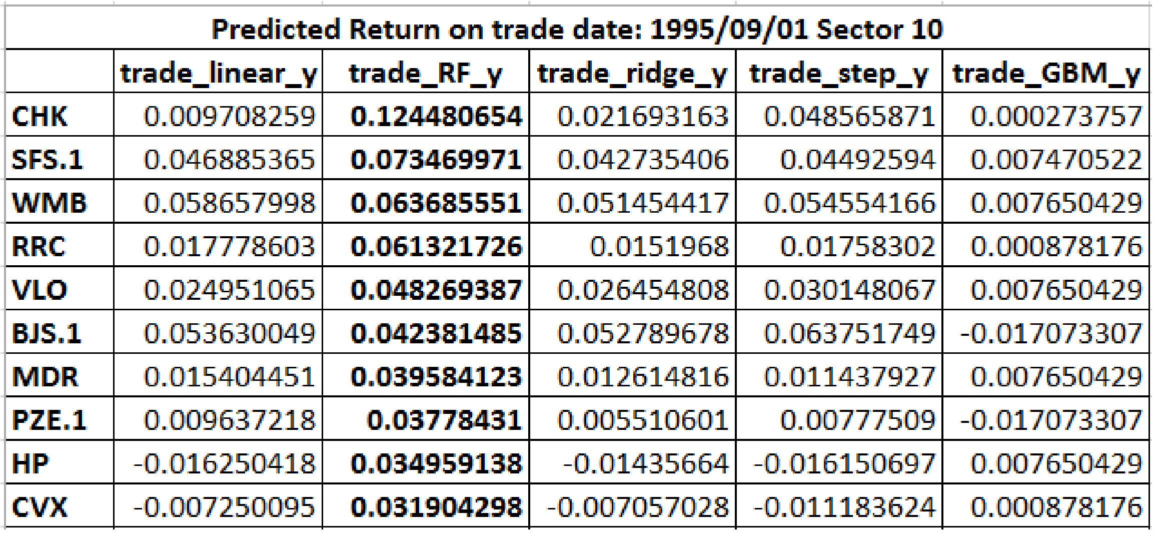
3. Use the predicted return in the selected model to pick up top 20% stocks from each sector. We predict next quarter return (predicted y) using current information (test Xs) based on the trained model.

In this example, we use ridge predicted return to pick stocks for the trade period 1995/06/01, the selected top 20% stocks are: WMB, OKE, RRC, PXD, VLO, EQT, HES, BHI, MUR, and NE. We then trade these stocks during the period between 1995/06/01 and 1995/09/01.



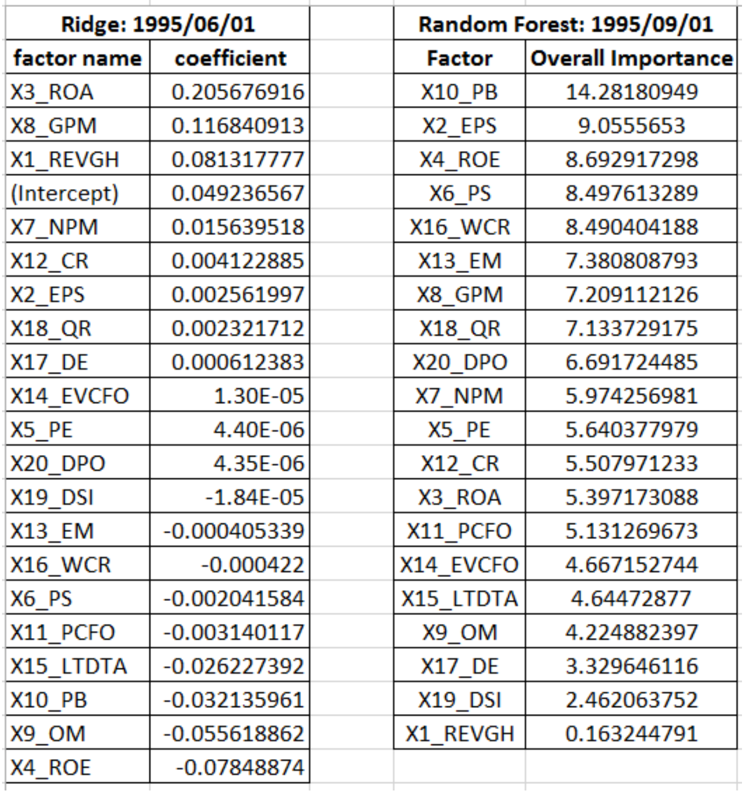
Predicted Return on trade date: 1995/06/01 Sector 10					
	trade_linear_y	**trade_RF_y**	**trade_ridge_y**	**trade_step_y**	**trade_GBM_y**
WMB	0.104217039	0.051182059	**0.092439735**	0.090133162	0.035059779
OKE	0.07545861	0.041195643	**0.074242502**	0.085324355	0.025637501
RRC	0.041552266	0.07018162	**0.03734671**	0.048422487	0.018315392
PXD	0.046332167	0.009576099	**0.036602262**	0.039057469	0.002317376
VLO	0.034804273	0.029931138	**0.034672808**	0.040461992	0.025637501
EQT	0.024707836	0.047871591	**0.02345606**	0.023614005	0.018315392
HES	0.018024606	0.042945222	**0.016156815**	0.018046474	0.003752361
BHI	0.013339166	-0.006953881	**0.011478147**	0.019846669	-0.002691578
MUR	0.011139403	0.010749857	**0.010185301**	0.0049421	0.003752361
NE	0.011546346	-0.063261353	**0.009380882**	0.008527724	-0.022072214

In the second trade period of 1995/09/01, we use random forest to pick the stocks, the top 20% stocks are: CHK, SFS.1, WMB, RRC, VLO, BJS.1, MDR, PZE.1, HP, and CVX. We then trade these stocks from 1995/09/01 to 1995/12/01. As for the stocks owned at previous quarters, such as WMB, RRC, and VLO, we just need to use the portfolio weights to adjust their shares.



Predicted Return on trade date: 1995/09/01 Sector 10					
	trade_linear_y	**trade_RF_y**	**trade_ridge_y**	**trade_step_y**	**trade_GBM_y**
CHK	0.009708259	**0.124480654**	0.021693163	0.048565871	0.000273757
SFS.1	0.046885365	**0.073469971**	0.042735406	0.04492594	0.007470522
WMB	0.058657998	**0.063685551**	0.051454417	0.054554166	0.007650429
RRC	0.017778603	**0.061321726**	0.0151968	0.01758302	0.000878176
VLO	0.024951065	**0.048269387**	0.026454808	0.030148067	0.007650429
BJS.1	0.053630049	**0.042381485**	0.052789678	0.063751749	-0.017073307
MDR	0.015404451	**0.039584123**	0.012614816	0.011437927	0.007650429
PZE.1	0.009637218	**0.03778431**	0.005510601	0.00777509	-0.017073307
HP	-0.016250418	**0.034959138**	-0.01435664	-0.016150697	0.007650429
CVX	-0.007250095	**0.031904298**	-0.007057028	-0.011183624	0.000878176

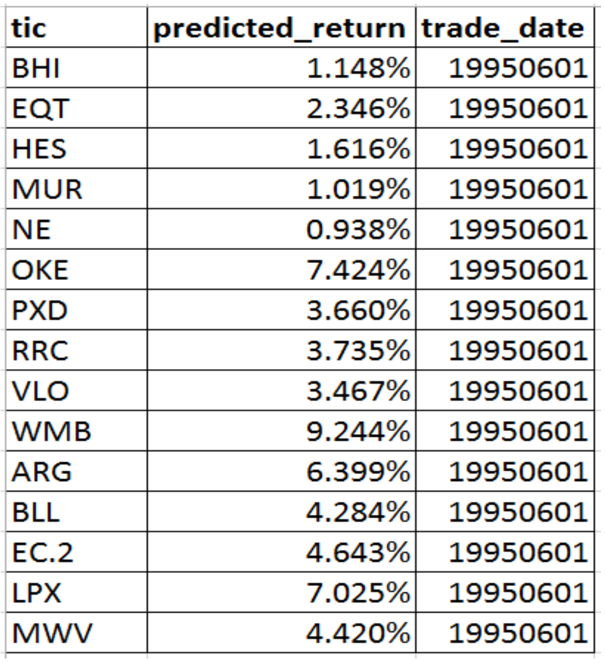
4. We check on the corresponding models' features and its coefficients or importance level to ensure that there are no abnormal results. e.g. assign 0 to all features.



Ridge: 1995/06/01	
factor name	coefficient
X3_ROA	0.205676916
X8_GPM	0.116840913
X1_REVGH	0.081317777
(Intercept)	0.049236567
X7_NPM	0.015639518
X12_CR	0.004122885
X2_EPS	0.002561997
X18_QR	0.002321712
X17_DE	0.000612383
X14_EVCFO	1.30E-05
X5_PE	4.40E-06
X20_DPO	4.35E-06
X19_DSI	-1.84E-05
X13_EM	-0.000405339
X16_WCR	-0.000422
X6_PS	-0.002041584
X11_PCFO	-0.003140117
X15_LTDTA	-0.026227392
X10_PB	-0.032135961
X9_OM	-0.055618862
X4_ROE	-0.07848874

Random Forest: 1995/09/01	
Factor	Overall Importance
X10_PB	14.28180949
X2_EPS	9.0555653
X4_ROE	8.692917298
X6_PS	8.497613289
X16_WCR	8.490404188
X13_EM	7.380808793
X8_GPM	7.209112126
X18_QR	7.133729175
X20_DPO	6.691724485
X7_NPM	5.974256981
X5_PE	5.640377979
X12_CR	5.507971233
X3_ROA	5.397173088
X11_PCFO	5.131269673
X14_EVCFO	4.667152744
X15_LTDTA	4.64472877
X9_OM	4.224882397
X17_DE	3.329646116
X19_DSI	2.462063752
X1_REVGH	0.163244791

We finish these steps for each sector. Then we get a final table of all selected stocks with its tic name, predicted returns for next quarter, and the corresponding trading periods.



tic	predicted_return	trade_date
BHI	1.148%	19950601
EQT	2.346%	19950601
HES	1.616%	19950601
MUR	1.019%	19950601
NE	0.938%	19950601
OKE	7.424%	19950601
PXD	3.660%	19950601
RRC	3.735%	19950601
VLO	3.467%	19950601
WMB	9.244%	19950601
ARG	6.399%	19950601
BLL	4.284%	19950601
EC.2	4.643%	19950601
LPX	7.025%	19950601
MWV	4.420%	19950601

Section 4: Portfolio Optimization

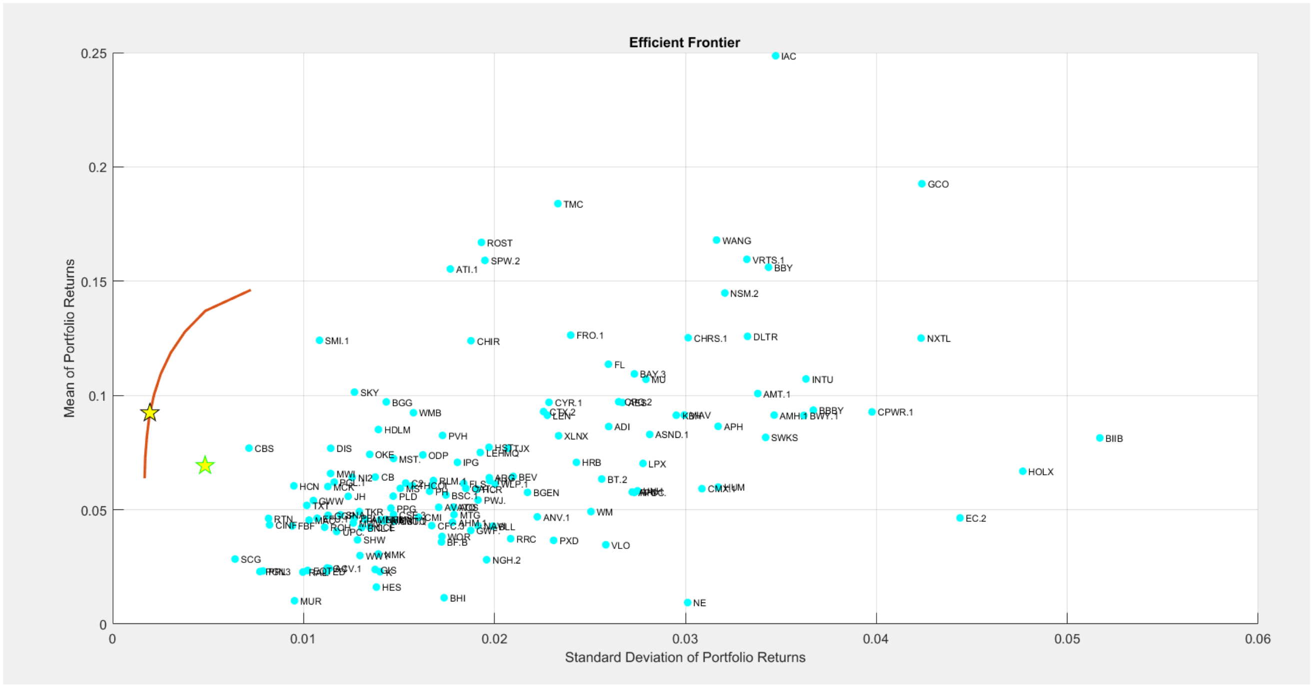
We use two optimization methods: Mean-variance and Min-variance to decide the weights of each stock, and then use equal-weighted portfolio as our benchmark. We perform the portfolio optimization by Matlab Financial Toolbox-Portfolio Object.

4.1 Mean-Variance Portfolio Constraints

We first use mean-variance optimization to allocate the stocks we have picked.

- Expected return: Predicted next quarter return.
- Covariance Matrix: 1 year historical daily return.
- Constraint of weights: Upper bound: 5%, Lower bound: 0% (Long only).
- Fully invest our capital for each rebalancing day: Sum of weights=100%.
- LowerBudget = UpperBudget = 1 (no leverage)

Here is a graph of the efficient frontier from the first trading period:



The location of green star in the graph is where our benchmark at and the yellow star on the red curve is the mean-variance result; the rest of the points are stocks plotted based on its predicted return and standard deviation. The result shows that our approach is legitimate. Also,the mean-variance result has a predicted return higher than that of the benchmark and risk lower than the benchmark risk.

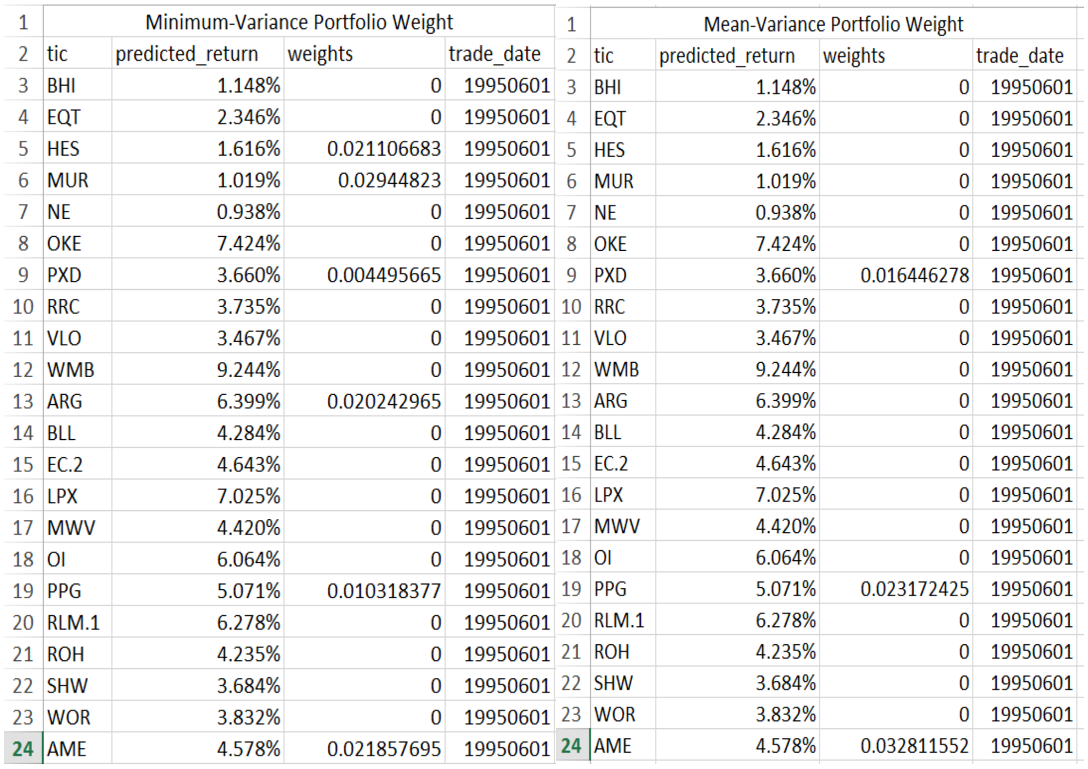
4.2 Minimum-Variance Portfolio Constraints

Then, we try the min-variance optimization approach to allocate the stocks we have picked.

- Expected return: equal to 0.

- Covariance Matrix: 1 year historical daily return.
- Constraint of weights: Upper bound: 5%, Lower bound: 0% (Long only).
- Fully invest our capital for each rebalancing day: Sum of weights=100%.
- LowerBudget = UpperBudget = 1 (no leverage).

Here is a comparison of the result for mean-variance and min-variance optimized weights, the weights should be different:



1	Minimum-Variance Portfolio Weight			
2	tic	predicted_return	weights	trade_date
3	BHI	1.148%	0	19950601
4	EQT	2.346%	0	19950601
5	HES	1.616%	0.021106683	19950601
6	MUR	1.019%	0.02944823	19950601
7	NE	0.938%	0	19950601
8	OKE	7.424%	0	19950601
9	PXD	3.660%	0.004495665	19950601
10	RRC	3.735%	0	19950601
11	VLO	3.467%	0	19950601
12	WMB	9.244%	0	19950601
13	ARG	6.399%	0.020242965	19950601
14	BLL	4.284%	0	19950601
15	EC.2	4.643%	0	19950601
16	LPX	7.025%	0	19950601
17	MWV	4.420%	0	19950601
18	OI	6.064%	0	19950601
19	PPG	5.071%	0.010318377	19950601
20	RLM.1	6.278%	0	19950601
21	ROH	4.235%	0	19950601
22	SHW	3.684%	0	19950601
23	WOR	3.832%	0	19950601
24	AME	4.578%	0.021857695	19950601

1	Mean-Variance Portfolio Weight			
2	tic	predicted_return	weights	trade_date
3	BHI	1.148%	0	19950601
4	EQT	2.346%	0	19950601
5	HES	1.616%	0	19950601
6	MUR	1.019%	0	19950601
7	NE	0.938%	0	19950601
8	OKE	7.424%	0	19950601
9	PXD	3.660%	0.016446278	19950601
10	RRC	3.735%	0	19950601
11	VLO	3.467%	0	19950601
12	WMB	9.244%	0	19950601
13	ARG	6.399%	0	19950601
14	BLL	4.284%	0	19950601
15	EC.2	4.643%	0	19950601
16	LPX	7.025%	0	19950601
17	MWV	4.420%	0	19950601
18	OI	6.064%	0	19950601
19	PPG	5.071%	0.023172425	19950601
20	RLM.1	6.278%	0	19950601
21	ROH	4.235%	0	19950601
22	SHW	3.684%	0	19950601
23	WOR	3.832%	0	19950601
24	AME	4.578%	0.032811552	19950601

Section 5: Transaction Costs

Generally, fees for each trade are measured based on broker fees, exchange fees, and SEC fees. In the real-world scenarios, a fund or trading firm might have different execution costs for many reasons. Despite these possible variations in cost, after going through several scenarios we consider our transaction cost to be 1/1000 of the value of the trade. We believe our fee assumption to be sufficient and reasonable for this part of the project.

Here is our formula to calculate the transaction cost: $\sum|(S(t)-S(t-1)) \times P(t)| \times 0.1\%$, S is buy or sell share based on weights and P is stock price.

Section 6: Risk Management

After the procedure of building portfolio and structuring with appropriate functions, we equip decision rules that would be applied to risk management of each trade. Fundamentally, due to the nature of long-only strategy, the risk was controlled internally through our portfolio optimization methods:

- Minimize variance and Maximum Sharpe ratio.
- Limits on position sizes (maximum of position size is 5% of portfolio value).
- No leverage.

Section 7: Alternative Risk Management: Turbulence

During the year of 2008, most Funds incurred a maximum drawdown greater than -50%. In order to prevent 2008 scenario, we have done some research and found a way to measure the market fragility, which is called Turbulence. Financial turbulence is defined as “a condition in which asset prices, given their historical patterns of behavior, behave in an uncharacteristic fashion, including extreme price moves, decoupling of correlated assets, and convergence of uncorrelated assets, it often coincides with excessive risk aversion, illiquidity, and devaluation of risky assets” (Kritzman and Li, 2010). In this way, we only need the historical close prices of an asset class to calculate its turbulence. Once we obtain the historical average returns and the related covariance matrix, the following formula can be used to calculate the turbulence of any asset classes:

$$d_t = (\mathbf{y}_t - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{y}_t - \boldsymbol{\mu})' . \tag{2}$$

where

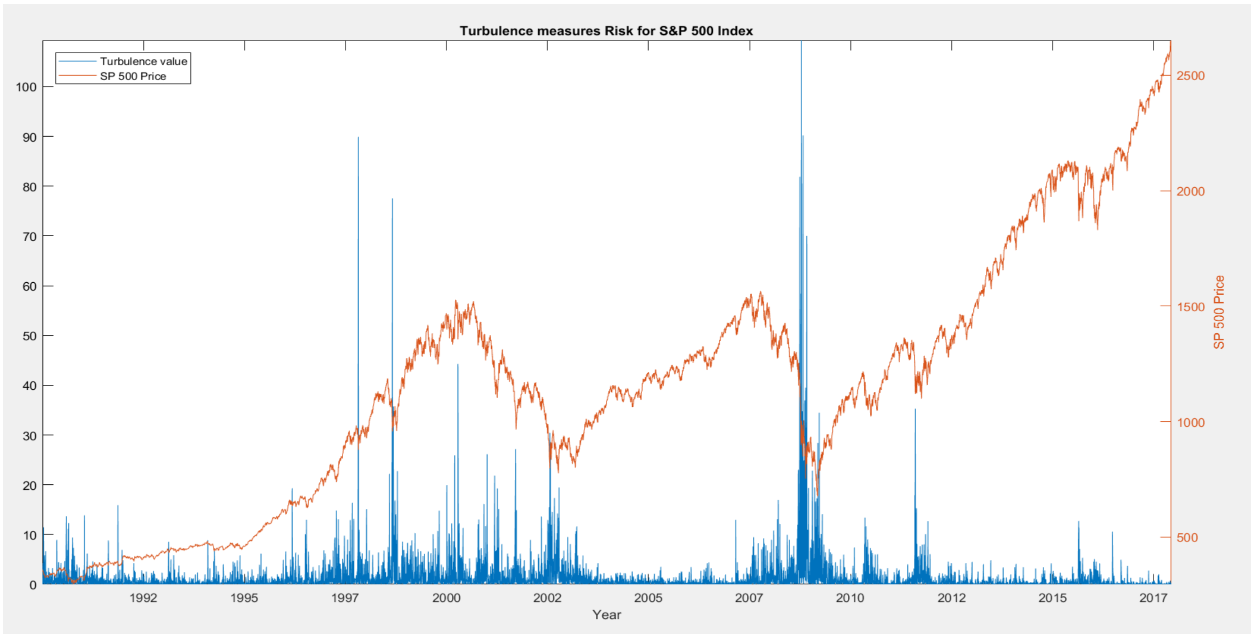
d_t = turbulence for a particular time period t (scalar)

$\mathbf{y}_t$ = vector of asset returns for period t ($1 \times n$ vector)

$\boldsymbol{\mu}$ = sample average vector of historical returns ($1 \times n$ vector)

$\boldsymbol{\Sigma}$ = sample covariance matrix of historical returns ($n \times n$ matrix)

We can calculate SPX’s turbulence based on this formula: the higher the turbulence value, the more fragile the market is. The turbulence value successfully captures each market crash period, including the 2001 dot-com bubble burst and 2008 financial crisis, etc. The graph below illustrate this point.



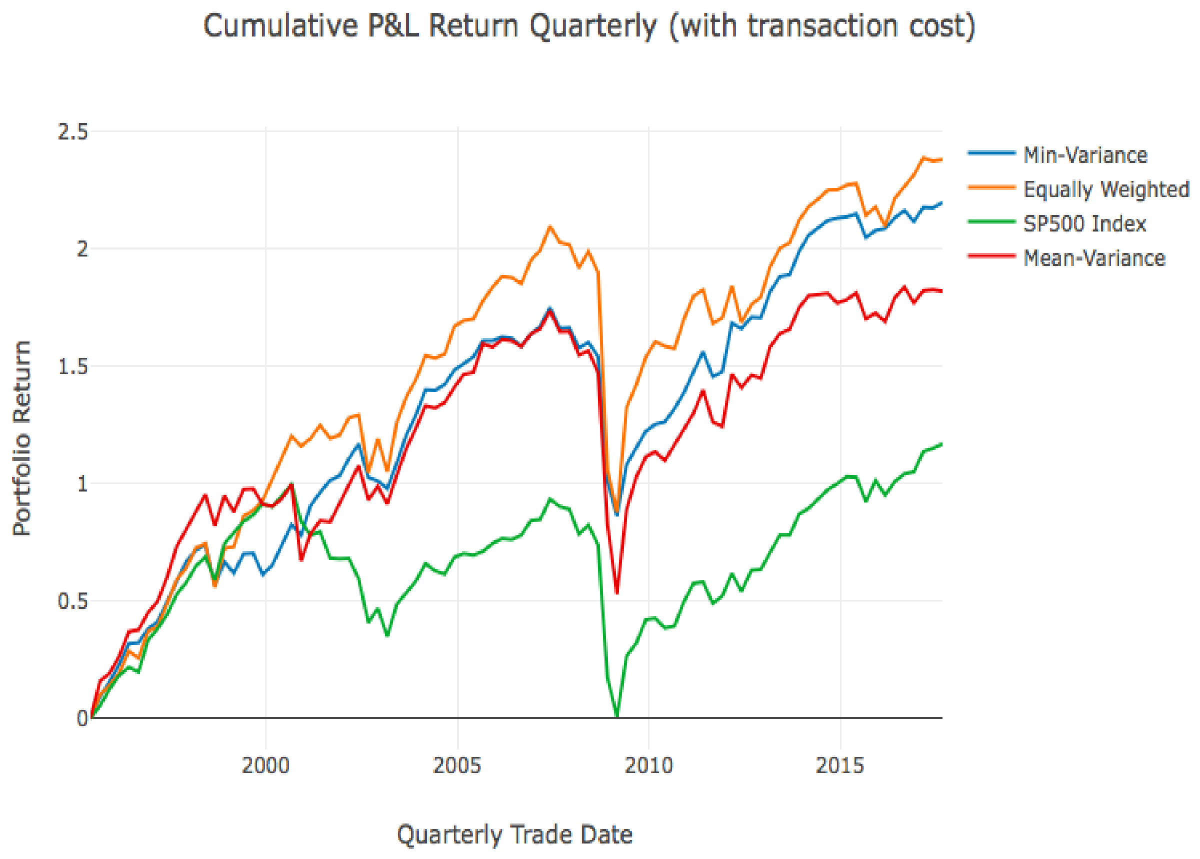
We can use the turbulence as an early warning signal to get out of market. We actually develop a simple turbulence strategy:

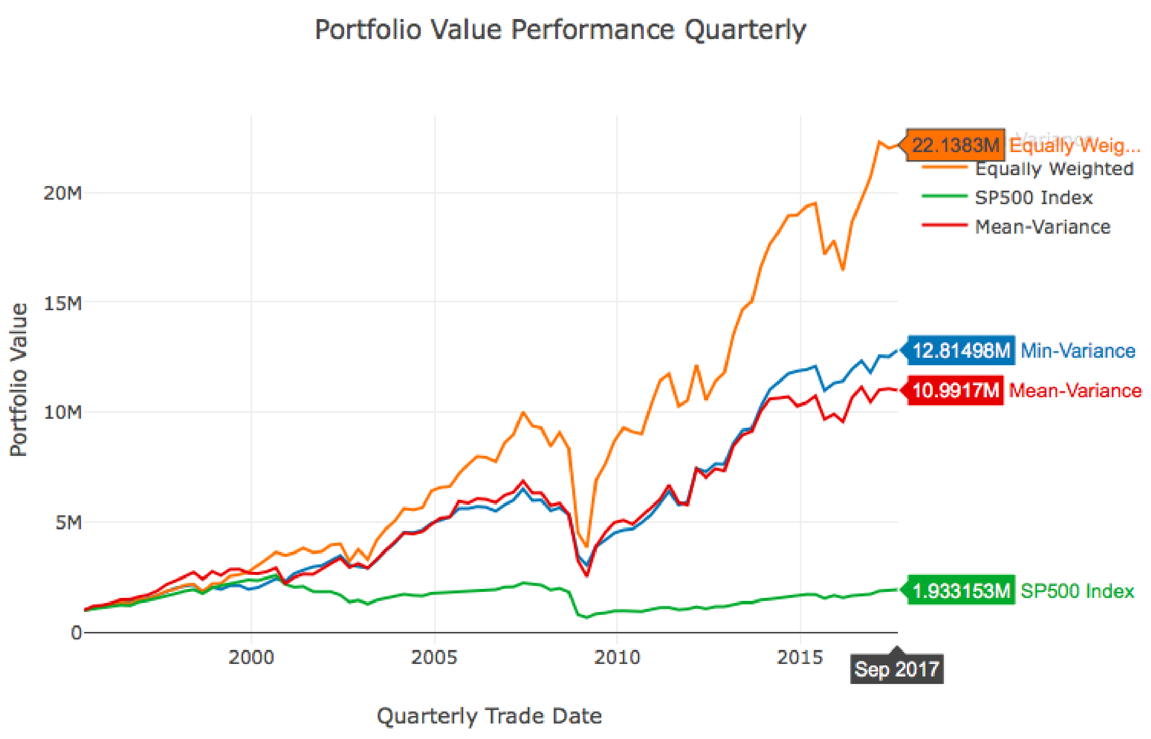
- Calculate turbulence for all stocks in the portfolio and set a threshold (if current value > threshold, then mark 1), then we can generate a daily signal.
- If 70%-80% stocks in our portfolio have the signal in the same day, then we should quit the market and sell everything or pull out 50% of our money at least.

However, we have not put this signal into our automated trading strategy because the decision to quit a market is so important that we cannot rely on machines to make the decision. Most of time, a decision such as this needs the board members to take a majority vote to decide.

Sector 8: Performance Analysis

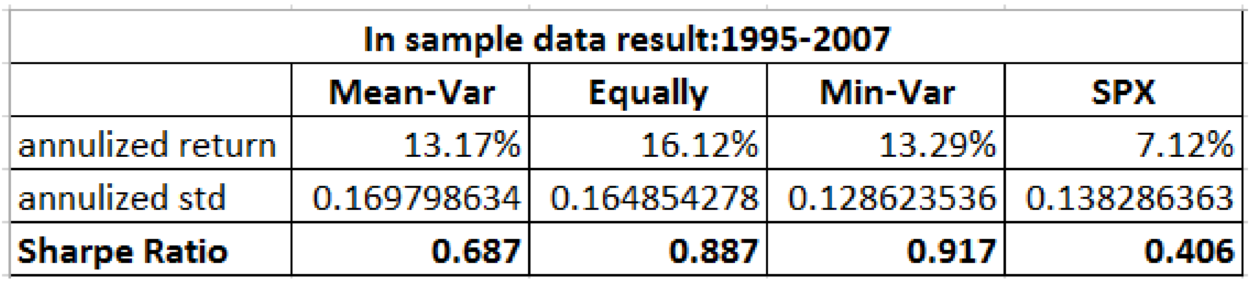
The results show that our strategy outperforms the market. The following back-test figures including equally weighted portfolio, mean-variance portfolio and minimum-variance portfolio indicate greater performance than that of the benchmark S&P 500 index. More importantly, all the statistics can show that the portfolio outperforms the S&P 500 index not only in the in-sample training period (before 2007) but also in the real trade period (after 2007).





Therefore, we conclude that our Quantamental method for stock selections does generate a better results than the market portfolio does. If we only analyze the portfolio value performance from the figure, it is noticeable that the equally weighted portfolio, the benchmark, has higher value than minimum-variance and mean-variance portfolio. However, the portfolio value is not the only consideration when selecting the optimal portfolio. We have the following reasons to conclude that the minimum-variance portfolio is a better method in the real trade period.

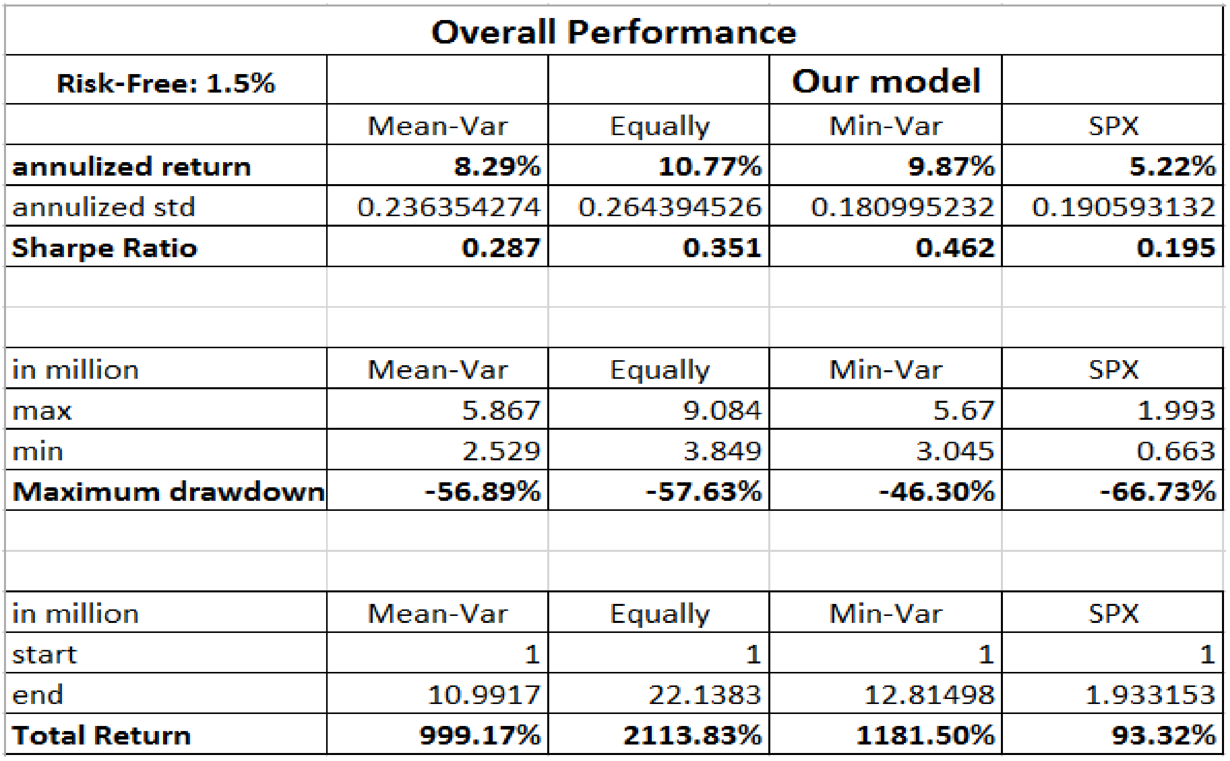
First of all, the equally-weighted portfolio is not robust enough. We notice that the performance of the equally-weighted allocation fully depends on the predicted returns that we calculate from our model. Therefore, the predicted returns will vary every time we run our model. Secondly, minimum-variance portfolio allocation takes the risk factor into consideration, making it more reliable in real trade. From the following table, we find that the minimum-variance allocation has a higher Sharpe ratio than that of the equally-weighted allocation during the in-sample period. Thus, we choose the minimum-variance as our portfolio allocation method.



In sample data result:1995-2007

	Mean-Var	**Equally**	**Min-Var**	**SPX**
annulized return	13.17%	16.12%	13.29%	7.12%
annulized std	0.169798634	0.164854278	0.128623536	0.138286363
Sharpe Ratio	**0.687**	**0.887**	**0.917**	**0.406**

In the out of sample trade period, minimum-variance still shows an outstanding performance. As we can see from the following overall performance table, the minimum-variance has the second highest annualized return 9.87%, the lowest maximum drawdown -46.3% and the highest Sharpe ratio 0.462, while SPX only has a sharpe ratio of 0.195.



Overall Performance				
Risk-Free: 1.5%			**Our model**	
	Mean-Var	Equally	Min-Var	SPX
annulized return	**8.29%**	**10.77%**	**9.87%**	**5.22%**
annulized std	0.236354274	0.264394526	0.180995232	0.190593132
Sharpe Ratio	**0.287**	**0.351**	**0.462**	**0.195**
in million	Mean-Var	Equally	Min-Var	SPX
max	5.867	9.084	5.67	1.993
min	2.529	3.849	3.045	0.663
Maximum drawdown	**-56.89%**	**-57.63%**	**-46.30%**	**-66.73%**
in million	Mean-Var	Equally	Min-Var	SPX
start	1	1	1	1
end	10.9917	22.1383	12.81498	1.933153
Total Return	**999.17%**	**2113.83%**	**1181.50%**	**93.32%**

Section 9: Conclusion

Applying machine learning algorithms to the fundamental financial data can filter out stocks with a relative bad earnings, thus providing a better way to select stocks. Minimum-variance method reduces the portfolio risk and thus yields a higher sharpe ratio. From the total returns of our portfolio compared to the benchmark, our trading strategy outperforms the S&P 500 index. More importantly, combined with our trading strategy, the portfolio allocation method is proven to improve the overall performance of our portfolio. Last but not the least, when Sharpe ratio considered our portfolio also outperforms the S&P 500.

Section 10: Potential Strategy Improvements

In our forecasting model, the average training time for each sector is about 1.5 hours and the total training time is 16.5 hours. Due to time constraints, we only run our model twice. To increase the stability of our model, it needs to run more than 10 times (it needs AWS and Google Cloud), and we need to make sure each time the portfolio can outperform the market.

Another area that can be improved is the model accuracy. Our current methodology basically selects the minimum MSE, and assigns 1 to the selected model and 0 to the other models. To improve our models, the ensemble methods or functional ANOVA can be applied here so that it can assign weights to all models: higher weights to lower MSE models, lower weights to higher MSE models. Thus, we can improve the accuracy of the model.

Furthermore, we have a quarterly portfolio rebalance period, and we can use turbulence or some stop-loss mechanism to cut some stocks that already incur a loss more than 10%. Thus we can use a monthly rebalance period to better measure the risk.

Reference:

1. Kritzman, M., & Li, Y. (2010). *Skulls, financial turbulence, and risk management*. Financial Analysts Journal, 66(5), 30-41.

2. Tibshirani, R., James, G., Witten, D., & Hastie, T. (2013*). An introduction to statistical learning-with applications in R*.

3. Ruppert, D. (2011). *Statistics and data analysis for financial engineering* (Vol. 13). New York: Springer.

Appendix A: Financial Ratios

A1. Revenue Growth
Revenue (T) / Revenue (T-1) -1
WRDS: REVTQ (T) /REVTQ (T-1) -1

A2. EPS
(Net Income -Dividends on Preferred Stock) / Average Outstanding Shares
WRDS: EPSPXQ (Earnings per Share (Basic), Excluding Extraordinary Items)

A3. ROA: Return on Asset
Net income / total asset
WRDS: NIQ (Net Income (Loss)) / ATQ (Assets, Total)

A4. ROE: Return on Equity
Net income / Average Stockholders' Equity
WRDS: NIQ (Net Income (Loss)) / CEQQ (Common/Ordinary Equity, Total)

A5. PE Ratio: Price to Earnings per share
Current market close price/ EPS
WRDS: Current market close price/ EPSPXQ (Earnings per Share (Basic) -Excluding Extraordinary Items)

A6. PS Ratio: Price to Sales
Current market close price / sales per share
WRDS: Current market close price / [REVTQ (Revenue, Total) / CSHOQ (Common Shares Outstanding)]

A7. Net Profit Margin
Net Income / Revenue
WRDS: NIQ (Net Income (Loss)) / REVTQ (Revenue, Total)

A8. Gross Profit Margin
Gross Profit / Revenue
WRDS: [REVTQ (Revenue, Total) –COGSQ (Cost of Goods Sold)] / REVTQ (Revenue, Total)

A9. Operating Margin
Operating income/net sales
WRDS: OIADPQ (Operating Income after Depreciation, Quarterly) / SALEQ (Sales/Turnover (Net))

A10. Price to Book ratio
Current market close price / Book Value per Share
WRDS: Current market close price / {[ATQ (Assets, Total) –LTQ (Liabilities, Total) / CSHOQ}

A11. Price to cash flow ratio

Current market close price / cash flow from operations per share
WRDS: Current market close price / { [(NIQ (Net Income (Loss)) + DPQ (Depreciation and Amortization, Total) + (WCAPQ (T)-WCAPQ (T-1))] / CSHOQ }

A12. Cash Ratio
Cash / Current Liabilities
WRDS: CHEQ (Cash and Short-Term Investments)/ LCTQ (Current Liabilities, Total)

A13. Enterprise Multiple
Enterprise Value / EBITDA
WRDS: EV = PRCCQ (Price Close, Quarter) * CSHOQ (Common Shares Outstanding) + DLTTQ (Long-Term Debt, Total) + DLCQ (Debt in Current Liabilities) -CHEQ (Cash and Short-Term Investments)
WRDS: EBITDA = REVTQ (Revenue, Total) –XOPRQ (Operating Expense, Total) + DPQ (Depreciation and Amortization, Total)

A14. Enterprise Value to Cash Flow from Operations (EV/CFO)
Enterprise Value / cash flow from operations
WRDS:[PRCCQ (Price Close, Quarter) * CSHOQ + DLTTQ (Long-Term Debt, Total) + DLCQ (Debt in Current Liabilities) -CHEQ (Cash and Short-Term Investments)] /[(NIQ (Net Income (Loss)) + DPQ (Depreciation and Amortization, Total) –(WCAPQ (T)-WCAPQ (T-1))]

A15. Long Term Debt to Total Assets
WRDS: DLTTQ (Long-Term Debt, Total) / ATQ (Assets, Total)

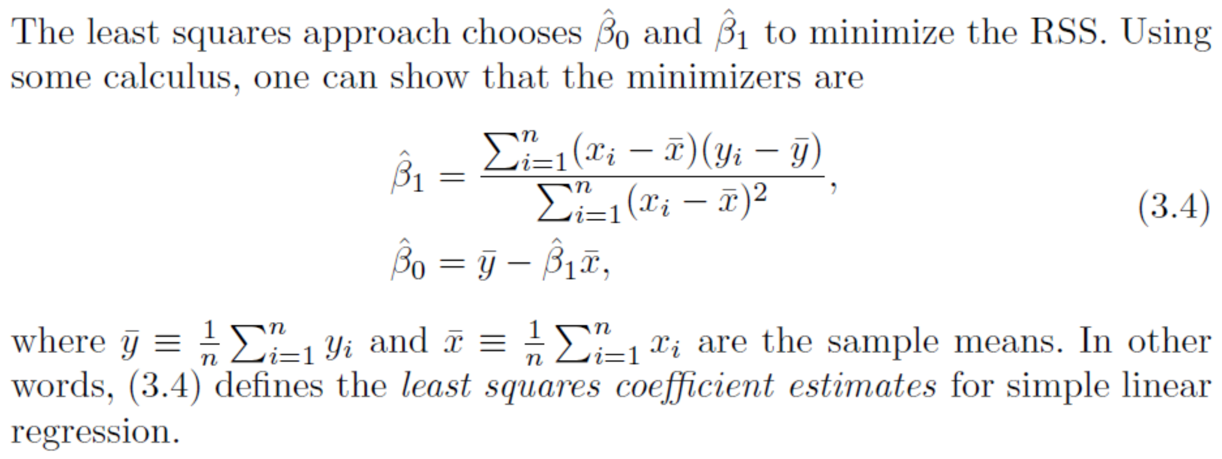
A16. Working Capital Ratio
Current assets / Current liabilities
WRDS: ACTQ (Current Assets, Total) / LCTQ (Current Liabilities, Total)

A17. Debt-Equity Ratio
Total Liabilities / Shareholders' Equity
WRDS: LTQ (Liabilities, Total) / CEQQ (Common/Ordinary Equity, Total)

A18. Quick Ratio
Current asset –Inventory / Current liability
WRDS: [ACTQ (Current Assets, Total) -INVTQ (Inventory, Finished Goods)] / LCTQ (Current Liabilities, Total)
A19. Days Sales of Inventory
Inventory/COGS*365
WRDS: INVTQ (Inventory, Finished Goods) / COGSQ (Cost of Goods Sold) * 365
A20. Days Payable Outstanding
Accounts Payable / COGS * 365
WRDS: APQ (Account Payable/Creditors, Trade) / COGSQ (Cost of Goods Sold) * 365

Appendix B: Machine learning algorithms formula and loss function

B1. Linear Regression Loss Function

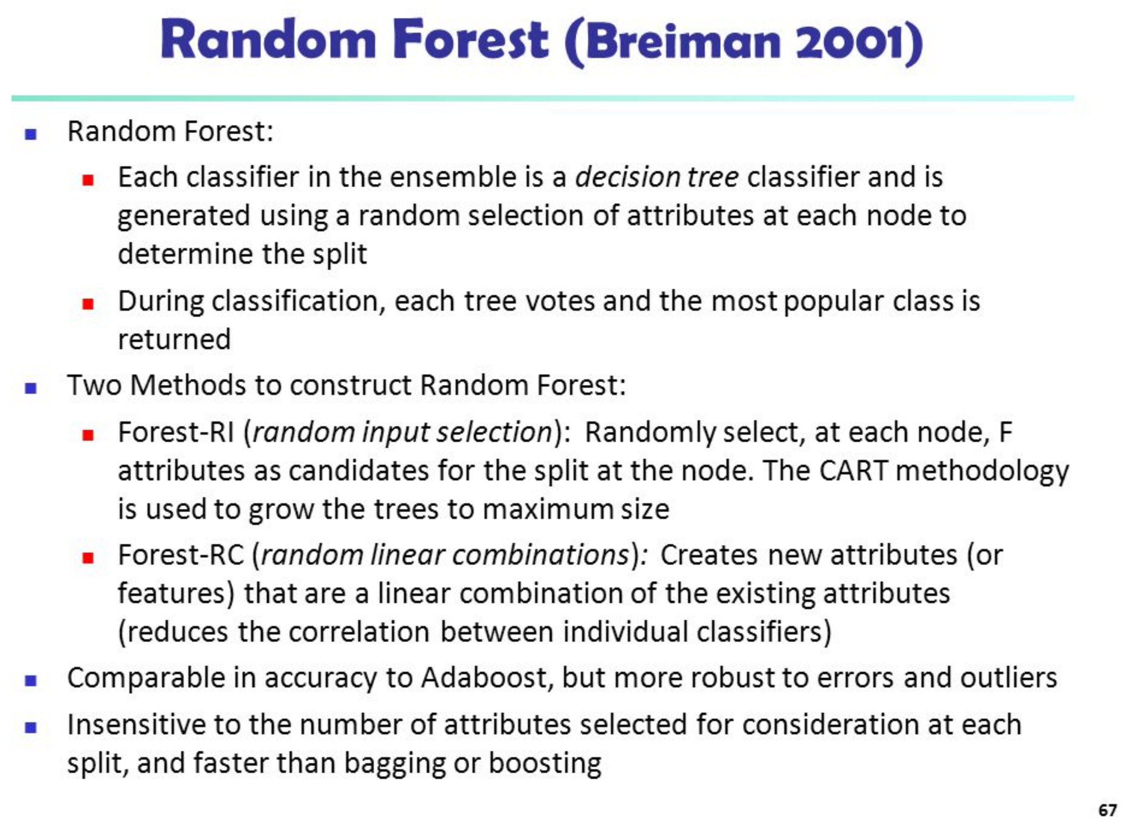


The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. Using some calculus, one can show that the minimizers are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}, \tag{3.4}$$

where $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$ are the sample means. In other words, (3.4) defines the *least squares coefficient estimates* for simple linear regression.

B2. Random Forest Algorithms



Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

67

B3. Ridge Regression Loss Function

Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

B4. Stepwise Regression with Forward and Backward Selection

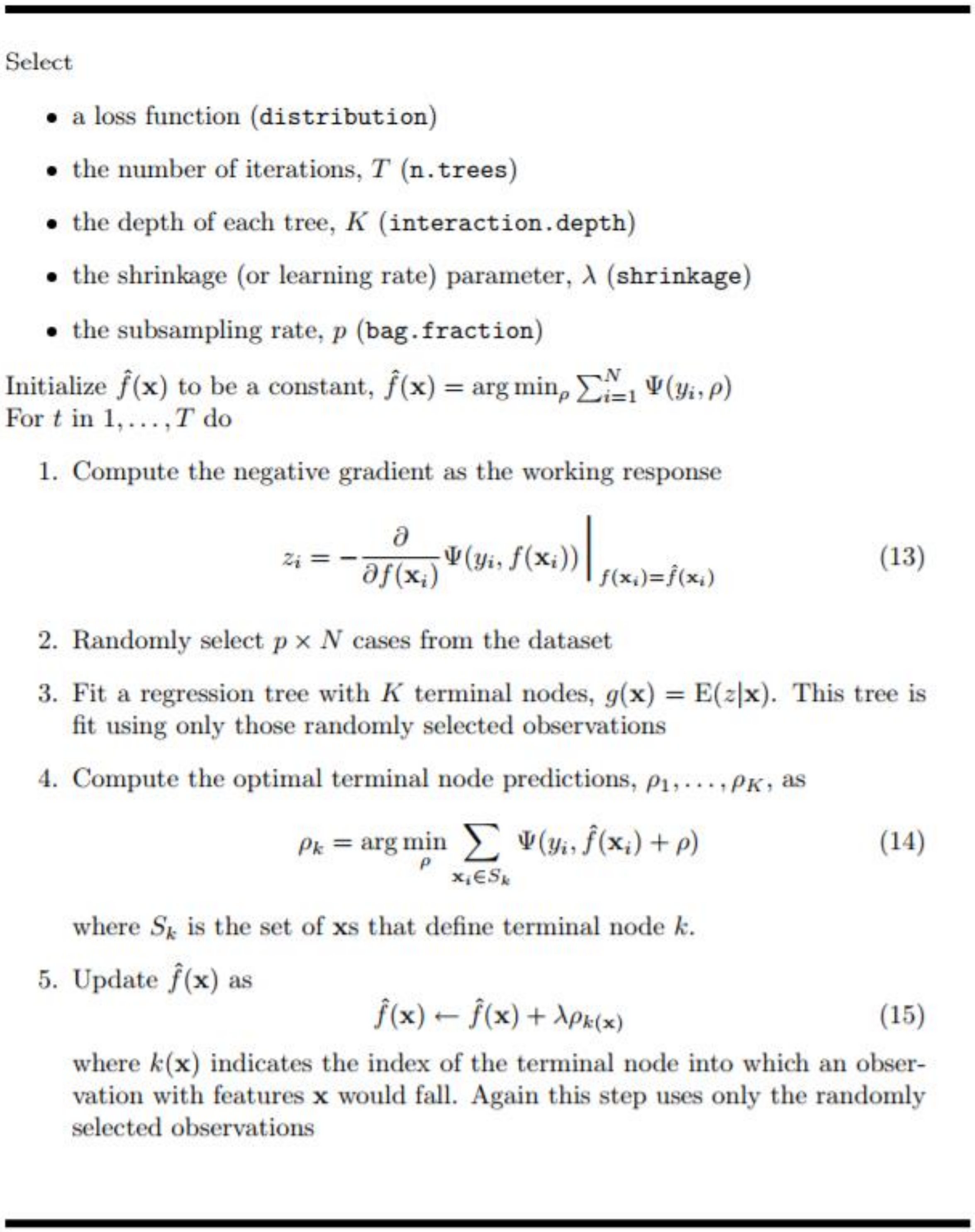
Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

B5. Generalized Boosted Model (GBM) Algorithms



Select

- a loss function (`distribution`)
- the number of iterations, T (`n.trees`)
- the depth of each tree, K (`interaction.depth`)
- the shrinkage (or learning rate) parameter, λ (`shrinkage`)
- the subsampling rate, p (`bag.fraction`)

Initialize $\hat{f}(\mathbf{x})$ to be a constant, $\hat{f}(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, \rho)$
For t in $1, \ldots, T$ do

1. Compute the negative gradient as the working response

$$z_i = -\frac{\partial}{\partial f(\mathbf{x}_i)} \Psi(y_i, f(\mathbf{x}_i)) \Bigg|_{f(\mathbf{x}_i)=\hat{f}(\mathbf{x}_i)} \qquad (13)$$

2. Randomly select $p \times N$ cases from the dataset

3. Fit a regression tree with K terminal nodes, $g(\mathbf{x}) = \mathrm{E}(z|\mathbf{x})$. This tree is fit using only those randomly selected observations

4. Compute the optimal terminal node predictions, $\rho_1, \ldots, \rho_K$, as

$$\rho_k = \arg\min_{\rho} \sum_{\mathbf{x}_i \in S_k} \Psi(y_i, \hat{f}(\mathbf{x}_i) + \rho) \qquad (14)$$

   where S_k is the set of $\mathbf{x}$s that define terminal node k.

5. Update $\hat{f}(\mathbf{x})$ as

$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \rho_{k(\mathbf{x})} \qquad (15)$$

   where $k(\mathbf{x})$ indicates the index of the terminal node into which an observation with features $\mathbf{x}$ would fall. Again this step uses only the randomly selected observations

Appendix C: Forecasting model code

```
fundamental_ML_model <- function(sector_data,trade_date){
  #######################################################
  #1. model test error to select models
  #2. trade period predicted return to select stocks
  #3. linear regression features
  #4. random forest features
  #5. ridge features
  #6. stepwise regression features
  #7. gbm features

  #look at the data determine the first factor column number
  start_column=12

  #set the rows to 89, because we have 89 stock selections
  #may need to adjust and put into function

  #model error to select model
  model_error=data.frame(MSE_linear=replicate(89,0))
  model_error[,2]=data.frame(MSE_RF=replicate(89,0))
  model_error[,3]=data.frame(MSE_ridge=replicate(89,0))
  model_error[,4]=data.frame(MSE_step=replicate(89,0))
  model_error[,5]=data.frame(MSE_gbm=replicate(89,0))

  #predicte return to select stocks
  predicted_return=list()

  #main model
  LR_features=list()
  RF_features=list()
  ridge_features=list()

  Step_features=list()
  GBM_features=list()

  #understand rolling windows
  #for(i in 1:(length(trade_date)-19)){print(c(i,i+15,i+16,i+19,trade_date[i+20]))}

  for(i in 1:(length(trade_date)-21)){

    ###############################################
    ###########rolling window######################
    ####train the model based on 4 years, 16 quarters data
    #growing window 10 years
    if (i<=25) {
      data_train=sector_data[(sector_data$tradedate <= trade_date[i+15]),]
      train_x=data_train[,c(start_column:(dim(sector_data)[2]-1))]
      train_y=data_train[,dim(sector_data)[2]]
    } else{
      data_train=sector_data[(sector_data$tradedate <= trade_date[i+15]) & sector_data$tradedate >=
trade_date[i-25],]
      train_x=data_train[,c(start_column:(dim(sector_data)[2]-1))]
      train_y=data_train[,dim(sector_data)[2]]
    }

    ####test the model based on 1 year, 4 quarters data
    data_test=sector_data[(sector_data$tradedate <= trade_date[i+19]) & (sector_data$tradedate >=
trade_date[i+16]),]
    test_x=data_test[,c(start_column:(dim(sector_data)[2]-1))]
    test_y=data_test[,dim(sector_data)[2]]

    train=cbind(train_y,train_x)
    test=cbind(test_y,test_x)



    ####trade data for every quarter
    data_trade=sector_data[(sector_data$tradedate == trade_date[i+20]),]
    trade_x=data_trade[,c(start_column:(dim(sector_data)[2]-1))]
    trade_y=data_trade[,dim(sector_data)[2]]
    trade=cbind(trade_x,trade_y)

    row.names(trade_x)=data_trade$tic

    ###########################################
    ##############linear regression###########
    ###########################################
    linear_model=lm(y_return~., data=train)
    linear_pre_y=predict(linear_model,test_x)
    MSE_linear=mean((test_y-linear_pre_y)^2,na.rm=TRUE)
    #MSE_linear

    #LR features
    LR_features[[i]]=summary(linear_model)

    ###########################################
    ################Random Forest#############
    ############################################
    # Tune using algorithm tools
    # Tunning the mtry
    bestmtry <- tuneRF(train[,-1],train[,1], stepFactor=1.5, improve=1e-5, ntree=500,trace=0,plot =
FALSE)
    #plot(bestmtry,type = "l")
    bestmtry=data.frame(bestmtry)
    mytry_optimal=bestmtry$mtry[which.min(bestmtry$OOBError)]
    #mytry_optimal
    RF_Model=randomForest(y_return~.,data = train,ntree=500,mtry=mytry_optimal,importance=TRUE,
na.rm = T,trace=0)

    yhat_bag=predict(RF_Model,test_x)
    MSE_RF=mean((yhat_bag-test_y)^2)
    #MSE_RF
    #importance table
    #varImp(RF_Model)
    #varImpPlot(RF_Model,main='Random Forest Importance Table')

    ########RF features
    RF_features[[i]]=varImp(RF_Model)

    ######################################
    ################ridge################
    ######################################
    x_train_ridge=model.matrix(y_return~., train)[,-1]
    y_train_ridge=train$y_return

    x_test_ridge=model.matrix(y_return~.,test)[,-1]
    y_test_ridge=test$y_return

    #tunning for lambda
    #first run ridge on training set and pick the best lambda
    cv.out_ridge=cv.glmnet(x_train_ridge,y_train_ridge,alpha=1)
    bestlam_ridge=cv.out_ridge$lambda.min

    ridge_model=glmnet(x_train_ridge,y_train_ridge,alpha = 0,lambda = bestlam_ridge)
    ridge_pred_y=predict(ridge_model, newx = x_test_ridge)

    MSE_ridge=mean((ridge_pred_y-y_test_ridge)^2,na.rm=TRUE)

    #ridge features
    ridge_coeffs <- coef(ridge_model)
    ridge_coef=data.frame(name = ridge_coeffs@Dimnames[[1]][ridge_coeffs@i + 1], coefficient =
ridge_coeffs@x)

    ridge_features[[i]]=ridge_coef

    ##############################################
    ##############stepwise regression##########
    ##############################################
    #based on linear regresion
    step_model=stepAIC(linear_model, direction="both",trace = 0)
    step_pre_y=predict(step_model,test_x)

    MSE_step=mean((test_y-step_pre_y)^2,na.rm=TRUE)
    #MSE_step

    #step features
    Step_features[[i]]=summary(step_model)

    #####################################
    ################GBM################
    #####################################
    #Generalized Boosted Regression Models
    gbm_model=gbm(y_return~.,data = train,
                  dist="gaussian",
                  n.tree = 400,
                  shrinkage=0.1,
                  cv.folds = 5)

    gbm_pred_y = predict(gbm_model, test, n.tree = 400, type = 'response')
    MSE_gbm=mean((gbm_pred_y-test_y)^2,na.rm=TRUE)
    #MSE_gbm
    ########GBM features
    GBM_features[[i]]=  summary(gbm_model,plot=FALSE)

    ########################################
    #############get results#############
    ########################################



    #######################################
    #all model trade data
    #trade using linear regression
    trade_linear_y=predict(linear_model,trade_x)
```

```
  #trade using random forest
  trade_RF_y=predict(RF_Model,trade_x)
  #trade using ridge
  x_trade_ridge=model.matrix(y_return~.,trade)[,-1]
  row.names(x_trade_ridge)=data_trade$tic
  trade_ridge_y=predict(ridge_model,x_trade_ridge)
  colnames(trade_ridge_y)=c('trade_ridge_y')

  #trade stepwise regression
  trade_step_y=predict(step_model,trade_x)
  #trade using GBM
  trade_GBM_y=predict(gbm_model,trade_x)

  ###########store model error
  if (length(unique(trade_linear_y))<length(trade_linear_y)*0.2){
    MSE_linear=NA
  }

  if(length(unique(trade_RF_y))<length(trade_RF_y)*0.2){
    MSE_RF=NA
  }

  if(length(unique(trade_ridge_y))<length(trade_ridge_y)*0.2){
    MSE_ridge=NA
  }


  if(length(unique(trade_step_y))<length(trade_step_y)*0.2){
    MSE_step=NA
  }

  if(length(unique(trade_GBM_y))<length(trade_GBM_y)*0.2){
    MSE_gbm=NA
  }

  model_error[i,]=c(MSE_linear,MSE_RF,MSE_ridge,MSE_step,MSE_gbm)

  #store all the predicted returns
  temp_return=cbind(trade_linear_y,trade_RF_y,trade_ridge_y,trade_step_y,trade_GBM_y)
  predicted_return[[i]]=temp_return
  }

output=list(model_error=model_error,
            predicted_return=predicted_return,
             LR_features=LR_features,
             RF_features=RF_features,
             ridge_features=ridge_features,

             Step_features=Step_features,
             GBM_features=GBM_features
             )
  return(output)
}
```