MSBD 5013 – Statistical Prediction

Project Report

On

Return and Strategy on Stock Investment

Chung Kin Fung	20391508
Geng Xu	20405672
Lee Kam Chiu	20403636
Shen Yuying	20407101

18th December, 2016

Table of Contents

1. **Introduction** 1
2. **Previous Work** 2
3. **The Data and Pre-processing** 4
   3.1 Opening Price 4
   3.2 Highest Price 4
   3.3 Lowest Price 5
   3.4 Closing Price 5
   3.5 Trading Amount 5
   3.6 Constraints 5
4. **Design and Methods** 7
   4.1 Trading Rule Based Methods 7
   4.2 Statistical Learning Methods 9
      4.2.1 Model Inputs 10
      4.2.2 Handling Time-Series Data 11
      4.2.3 The Prediction Configuration 11
      4.2.4 Trading Signals 12
   4.3 Slope with Simple Moving Average 13
5. **Evaluation and Results** 15
   5.1 Analysis 16
6. **Conclusion** 17

Appendix A: R Codes 18

1. Introduction

Stock markets go up and down, lots of individuals, enterprises, and organizations are trying to explain and then try to predict the future of stock markets. Some people may say that price prediction is just smoke and mirrors, but still, there are a number of stock analysts, stock brokers and investors have attracted to this particular topic. However, finding clues or indicators for future stock price with high percentage of accuracy is extremely difficult because stock price reflects the future value of a company which is basically determined by currently available information rather than past prices. In order to predict the future movement of the stocks, fundamental analysis and technical analysis are mostly used methods.

Fundamental analysis is the studying of the listed company's basic business and earning powers which including earning per shares (EPS), price earnings ratio (P/E), cash flow, dividend yield, etc. which are easily obtainable from the company annual report or from other financial institutions analysis report. Technical analysis is more complicated which is the mixture of external conditions together with the demand and supply for that particular stock. Those external conditions include a lot of economic factors such as economic strength of the community, inflation rate, interest rate, unemployment rate, liquidity, market sentiment, etc. Moreover, a lot of analysis in a myriad of technical indicators such as moving average (SMA, EMA, WMA), moving average convergence/divergence (MACD), relative strength index (RSI), momentum, stochastic, etc. are also used in the analysis.

Long term investors pay more attention on the fundamental factors and then recognize the technical factors but short term investors tend to focus mainly on the technical factors.

The objective of this project is to find out the best trading strategy in order to achieve the greatest net investment return with a starting amount of $100 in 100 stocks training data.

2. Previous Work

Some previous works are done by one of the group member GENG Xu (20405672) in his undergraduate study about stock price forecasting. That project alerted a pessimistic result on applying traditional machine learning techniques to stock price forecasting. Details of the project is available on his github https://github.com/PhilipGeng/stockPrediction.

The project was divided into two parts. Firstly, HMM-GMM (Hidden Markov Model with Gaussian Mixture Model) was adapted for modelling sequence of data. It converted the regression problem to classification problem by discretizing profit into classes. However, the result is not better than SMA. The classifier is not predictive as expected. It is more like conducting series smoothing. The figure below shows a prediction series along with the real series.

S=50, M=10, T=3; err_train: 0.1253; err_test:0.1138

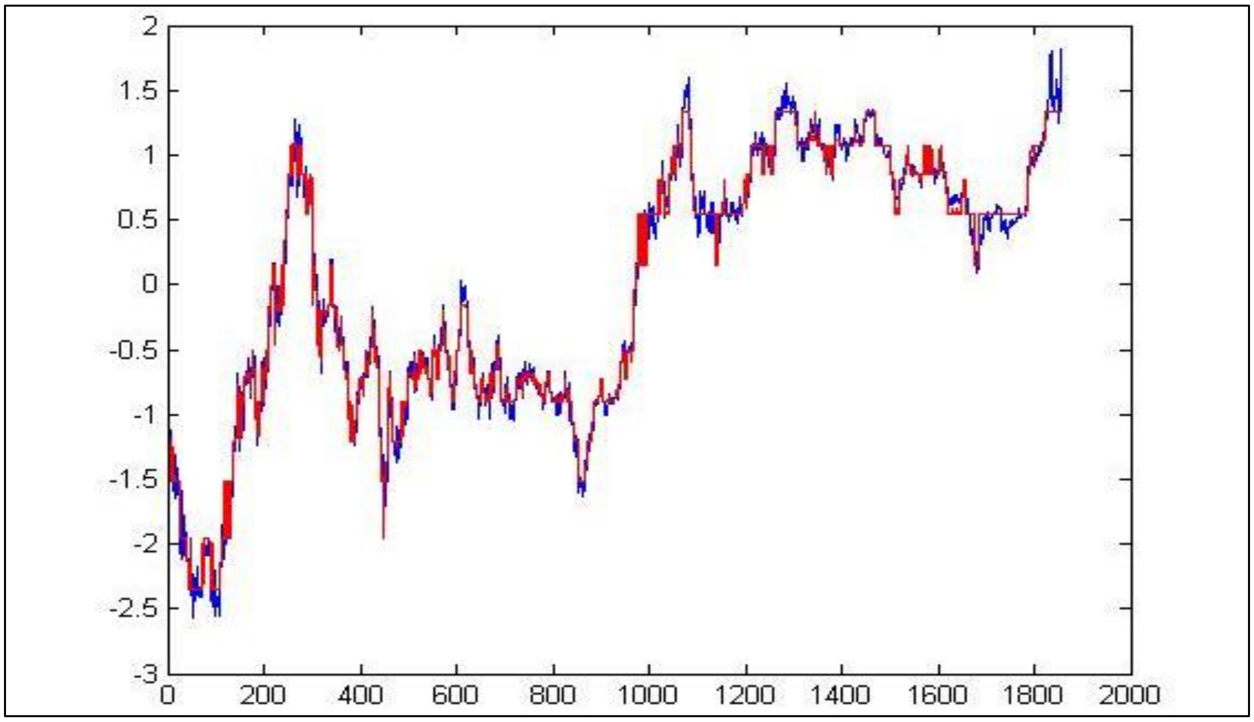


Figure 2.1

prediction series along with real series

Secondly, SVM is used to classify the trend of stock (moving up or down). There are 16 input features coming from indicators that works well:

WILLR, ROCR3/12, MOM1/3, EMA6/12, MACD, OBV, RSI6/12, ATR14, MFI14, CCI12/20, TRIX

PCA is performed to reduce the number of features to 8. Top 8 eigenvectors recover approximately 90% of variance:

Stock	Stock1	Stock11	Stock13	Stock23	Stock293	Stock857
Variance	0.9063	0.9027	0.9118	0.9124	0.9035	0.8996

The leave-one-out validation result is shown below:

Stock	Stock1	Stock11	Stock13	Stock23	Stock293	Stock857
accuracy	0.5285	0.6860	0.7115	0.6851	0.6813	0.6918

The best result is around 70%. However, the classifier hardly predicts any turning point, where a lot of money can be made. Also, the prediction of day x+1 is more likely to be the same with label of day x.

All in all, we think it is considerably hard to truly predict the stock price purely from series data. In this project, we devoted more on designing a better strategy, rather than forecasting the future trend.

3. The Data and Pre-processing

There are five tables are provided in the dataset with information on opening price, highest price, lowest price, closing price and trading amount of 731 trading days of 100 stocks and Figure 3.1 is an example of one stock movement in candlestick chart which can easily identify the opening, high, low, closing price and volume in one time frame (seconds, minutes, days, etc.) or tick transactions.

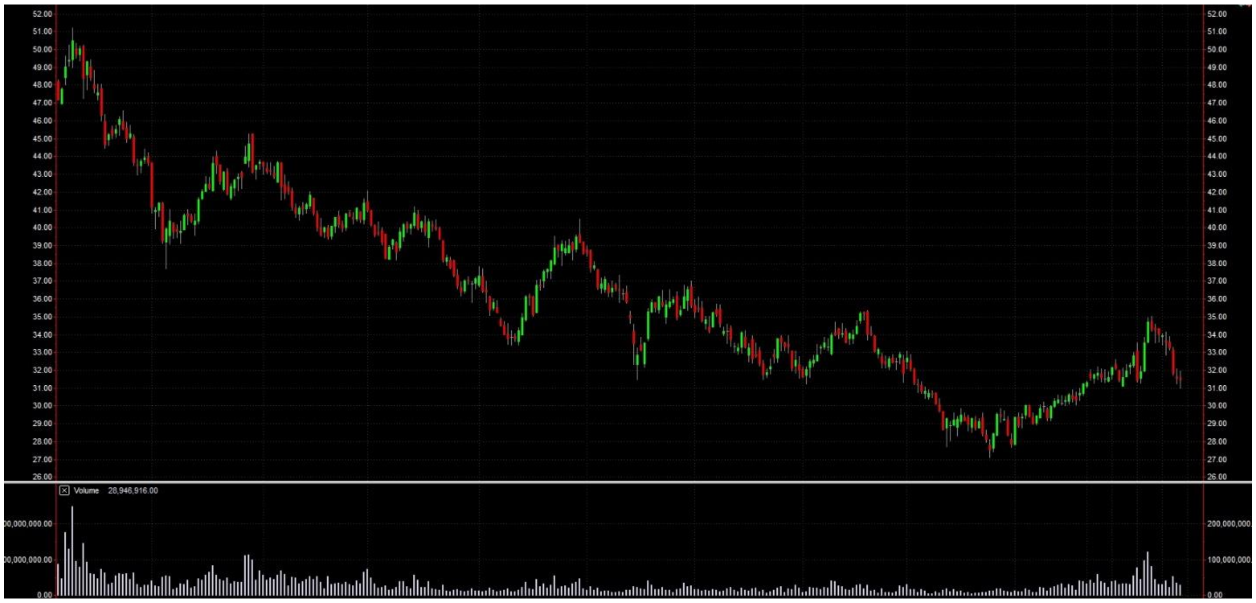


Figure 3.1

The candlestick chart of one stock

3.1 Opening Price

The opening price of a stock is the price at which a security first trades upon the opening of an exchange on a given trading day. It is different from the previous day's closing price because the price is decided by the balance between demand and supply fluctuation and there are a lot of factors can affect the attractiveness of the stock between the closing and the following day's opening.

3.2 Highest Price

The highest price of a stock is the price at which a security ever trades at the highest price on a given trading day regardless of the volume. It may be the same as the opening price, lowest price or closing price.

3.3 Lowest Price

The lowest price of a stock is the price at which a security ever trades at the lowest price on a given trading day regardless of the volume. It may be the same as the opening price, highest price or closing price.

3.4 Closing Price

The closing price of a stock is the final price at which a security is traded on a given trading day and it represents the most up to date value of the listed company.

These four prices can be the same if all the transactions are traded at the same price.

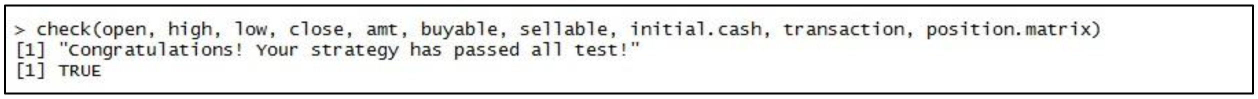
3.5 Trading Amount

Trading amount is the sum of trading number times trading price of a stock traded on a given trading day. Usually a large trading amount can push the stock up or make it decline.

3.6 Constraints

- We have an initial cash of $100 and the balance of each day must be positive.
- Each transaction in buying or selling must deduct the 0.2% transaction fee.
- All transactions must fulfil the T+1 policy. We cannot buy and sell the same stocks in the same trading day.

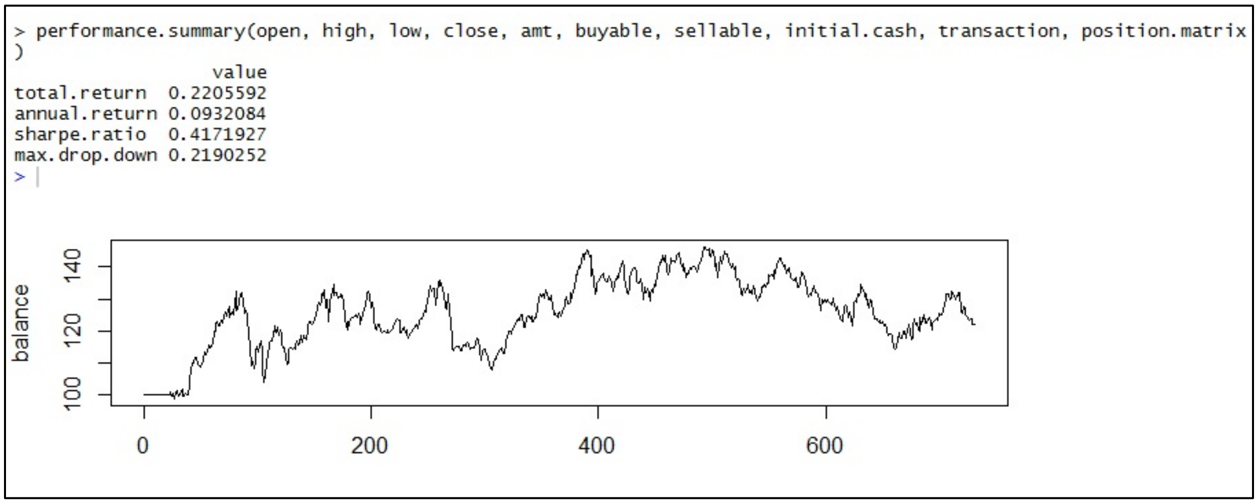
- We have to buy the stock at the opening price and sell the stock at the closing price and no short selling (i.e. must buy the stock first and then sell it) is allowed.
- No forward looking is allowed and we can only use the information on or before that time point. For example, assuming we want to buy some stocks at the opening price today, what we know is only the previous days' information and today's opening price. We can't use today's highest, lowest, closing and amount to make such buying decision in the strategy.
- We have to use those provided information to calculate the net investment return (after deducting the transaction fees) after a number of trading (buying and selling). Two functions "check" and "performance" are provided to check the strategy whether it satisfied the rules (Figure 3.2) and calculate the net investment return (Figure 3.3) respectively.



```
> check(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix)
[1] "Congratulations! Your strategy has passed all test!"
[1] TRUE
```

Figure 3.2

An example of the checking result



```
> performance.summary(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix
)
                  value
total.return  0.2205592
annual.return 0.0932084
sharpe.ratio  0.4171927
max.drop.down 0.2190252
> |
```

Figure 3.3

An example of the investment return

4. Design and Methods

4.1 Trading Rule Based Methods

A simple moving average (SAM) is the unweighted mean of the previous data. It is an arithmetic calculation by adding the price of a stock for a time periods and then dividing by the number of time periods. The formula of SMA is:

$$SMA = \frac{1}{n}(P_m + P_{m-1} + \dots + P_{m-(n-1)})$$

Moving averages are important analytical tools used to identify current price trends and the potential for a change in an established trend. It can smooth out the volatility and makes it easier to view the price trend of a stock. If the SMA points up, this means that the stock's price is increasing. If it is pointing down, it means that the stock's price is decreasing. The longer the timeframe for the SMA, the smoother the SMA. A shorter term moving average is more volatile, but its reading is closer to the source data. The main advantage of the SMA is that it offers a smoothed line, less prone to whipsawing up and down in response to slight, temporary price swings back and forth. Therefore, it provides a more stable level indicating support and resistance. Its weakness is that it is slower to respond to rapid price changes that often occur at market reversal points.

The most popular usage is to compare a pair of SMA with each covering different time frames. If a shorter-term SMA crosses above a longer-term SMA, an uptrend is expected. On the other hand, a short-term SMA crosses under a longer-term SMA signals a downward movement in the trend (Figure 4.1).

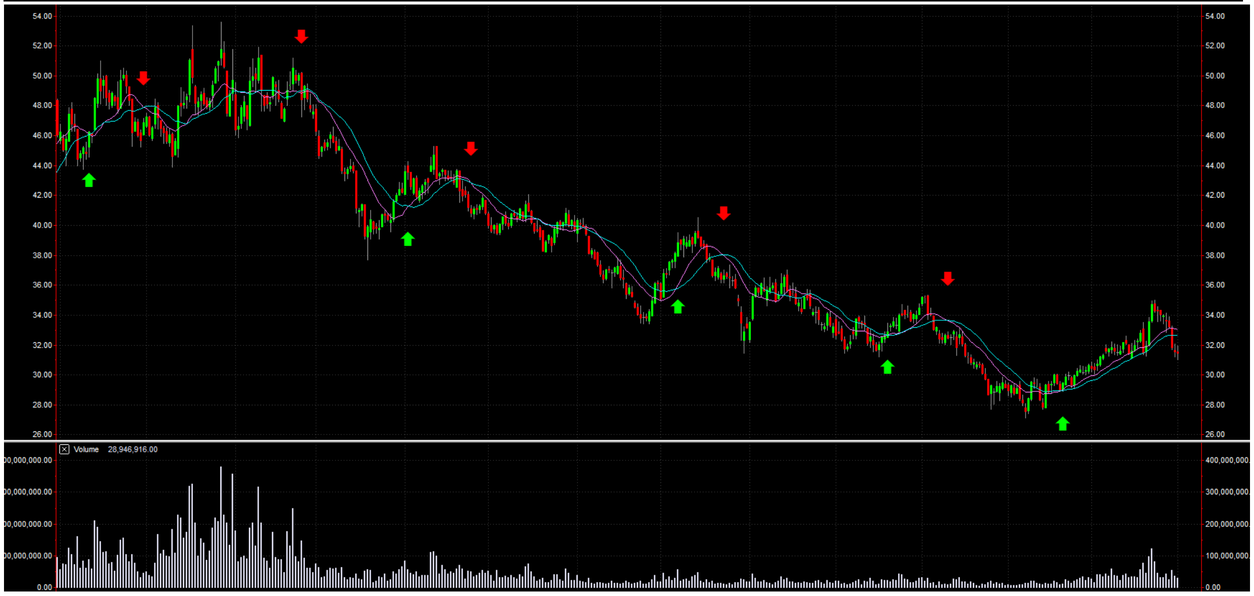
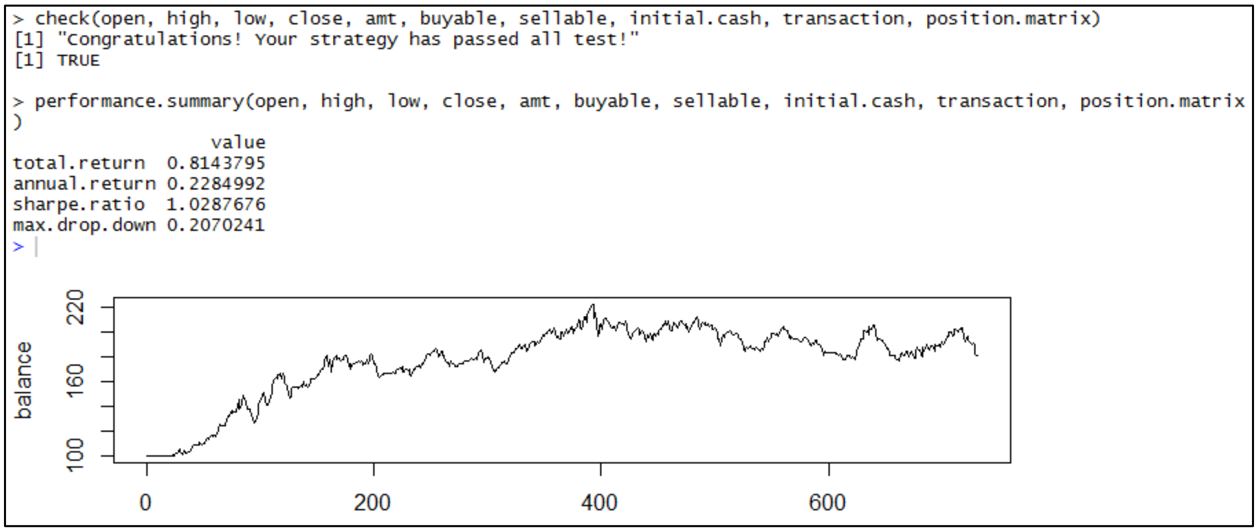


Figure 4.1

A simple moving average showing the uptrend and downtrend in crossing each other

One of a popular trading patterns that uses SMA including the “golden cross” and a “death cross”. A bullish signal is considered once a golden cross occurs when the 10-day SMA crosses above the 20-day SMA. A bearish signal is considered once a death cross occurs when the 10-day SMA crosses under the 20-day SMA. These trends can be considered together with high trading volumes.

For each day, if SMA 13 crosses above SMA 20, buying signal triggered and buy the stock with the largest price if more than one stock fulfils the condition. On the other hand, if SMA 13 crosses below SMA 20, selling signal came up and sell all the holding stocks.



```
> check(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix)
[1] "Congratulations! Your strategy has passed all test!"
[1] TRUE

> performance.summary(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix
)
                 value
total.return  0.8143795
annual.return 0.2284992
sharpe.ratio  1.0287676
max.drop.down 0.2070241
>
```

Figure 4.2

Investment return of SMA 13 crosses SMA 20 with investment cash of $16

In this experiment, we fine-tuned the best result in this strategy is SMA 13 crosses SMA 20 with reserved cash and investment cash of $16 and $16 respectively and the total return is 81.43795% which is shown in Figure 4.2.

In order to find out the best total return in this strategy, different combinations are calculated with investment cash from $5 to $25, fast MA from 5 to 21 and slow MA from 13 to 55 which running over 15,000 iterations and the return varies from -0.1737894 to 0.8143795.

4.2 Statistical Learning Methods

We would like to incorporate methods from statistical learning into our strategy, since it is a prediction problem. There are two main types of statistical learning methods: prediction and classification. We believe classification is not exactly a prediction in nature. For example, in our textbook ISLR, the authors claim that they achieved a > 60% accuracy of classifying whether tomorrow will rise or fall. This is not very useful, because if we know tomorrow will fall[rise], we would sell[buy] today. The problem is that there are empirical evidences that the chance that the stock (or index) will rise or fall tomorrow is around 50%. This means even if we can predict 100% whether it will rise or fall tomorrow, this will incur huge transaction cost in practice, since we are basically buying and selling every day. In view of this, we take the prediction approach and consider only the prediction methods.

The simplest prediction approach is linear regression. But it is impossible to predict future prices based on historical prices via linear regression in the short term, because there is no “turning” ability of linear regression. In the long term, it works better on indices than individual stocks. Since our horizon in this experiment is around 3 years, we are basically looking for a short-term approach. Based on the good results from our trading-rule based models, we believe it may be possible to apply linear regression on the different technical indicators.

4.2.1 Model Inputs

There are a few qualities that we desire on the indicators. Besides the obvious quality that it should have great predictive power, another important quality is that we want it to be applicable across stocks of different prices. Thirdly, since we will run it on learning algorithms, it would be less computationally intensive if it is naturally bounded, so that we do not need to normalize it every time we run the algorithm. These considerations all point to indicators residing in the range [0, 1], so the natural choices are momentum indicators. We pick the William %R and the RSI.

We pick William%R for its intuitiveness. The formula for William%R is:

$$William\%R = \frac{Highest(n\ day) - \ Today's\ close}{Highest(n\ day) - Lowest(n\ day)}$$

It gives a clear gauge on the relative position of today’s closing price on the past n days window. However it is a bit reverse, with a high today’s close resulting a small William%R value. We prefer subtracting it by 1 and call it

$$RecentScore = \frac{Today's\ close - Lowest(n\ day)}{Highest(n\ day) - Lowest(n\ day)}$$

RecentScore is purely based on price. It is a flat reflection on the current standing and it does not tell us what actually happens on the look-back period. So we choose RSI as our second indicator.

$$RSI = 100\left(\frac{RS}{1+RS}\right), where\ RS = \frac{Average\ Gain\ in\ n\ day}{Average\ Loss\ in\ n\ day}$$

and Average Gain = [(previous Average Gain)*13 + current Gain]/14 and similar for Average Loss. This tells us more information on the look-back period because it has some information on how each day looks like in the look-back period. From our trading-rule based testing, we find that a look-back period of 20 days is reasonably powerful. We could have chosen different window sizes for William %R and RSI. But we feel the model is more coherent with equal window size.

During our trading-rule testing, we realize that the days since the highest point has some predictive power. So we will use the DAYS_SINCE_ALL_TIME_HIGH and DAYS_SINCE_ALL_TIME_LOW variables in our regression model too. As mentioned before, we will use the normalized version: DAYS_SINCE_ALL_TIME_HIGH / NUMBER_DAYS_SINCE_BEGINNING.

4.2.2 Handling Time-Series Data

The most difficult part in designing our strategy based on regression is that we are dealing with data in time-series format. In conventional prediction problem, we predict a scalar output given a vector of inputs. But in our case, suppose we want to predict tomorrow's daily return of a particular stock. Our raw inputs are the time-series data with a specified window size from our indicators. Suppose we choose a look-back period of 20 days. This will give us 80 covariates. This leads to two problems. First, it is intuitively true that the most recent values have greatest predictive power and so including too many covariates will create spurious relationships. Second, these covariates are highly correlated and their multi-collinearity also annihilates our regression. In view of this, we will collapse the 20days data into a vector of 4 inputs, namely:

- Smooth RecentScore and RSI by SMA(5). This is because RecentScore and RSI oscillate wildly.
- Simply use the most recent DAYS_SINCE_ALL_TIME_HIGH and DAYS_SINCE_ALL_TIME_LOW

4.2.3 The Prediction Configuration

Since the stock market is full of noise, it is impossible to accurately predict the return on tomorrow's prices. We take a less ambitious route by predicting the return on the next F days based on the past H days. We think F = 10 is reasonable because it is not too short to avoid the noise, and not too long that it can still be accurate (we do not think we can predict what happens in the next 20 trading days ≈ 1 month). Given F = 10, we think H should be at least twice as long. But we cannot choose H to be too big because of two reasons: 1) Days further back have less predictive power and 2) Our horizon is not very long. We settle down on H = 30. Also, we choose an update frequency of FREQ = 10 days to avoid too much computation. Moreover, we mentioned that linear regression cannot predict trend reversals, so we will use two linear regression models. Finally, we avoid the forward looking problem by running the algorithms after the "next 10 days". For example, we perform the first regression training on day 40 by using day 1 – day 30 day to predict the highest and lowest returns between day 31 – day 40. Since update frequency is FREQ = 10, the second regression training is run on day 50 by using day 11 – day 40 day to predict the highest and lowest returns between day 41 – day 50, etc. The whole configuration is as follows:

- Linear Regression model A: use the past H = 30 days data to predict the highest return on the future F = 10 days, where return = HIGHEST_NEXT10/PERIOD_OPEN_PRICE – 1
- Linear Regression model B: use the past H = 30 days data to predict the lowest return on the future F = 10 days, where return = LOWEST_NEXT10/PERIOD_OPEN_PRICE – 1
- Update the models A and B every FREQ = 10 days
- Run regression training on day n using data on day n – 39 to n – 10 to predict highest and lowest return on day n – 9 to n.

4.2.4 Trading Signals

First we describe the trading signals of our system II before adding regression prediction:

- If we have cash, use betSize = $5.5 to buy all stocks with buy signals (can buy > 1 stocks per day). If cash ≤ 5.5, use 0.5*(cash available) to buy.
- If several stocks have buy signals, randomly arrange a buying order.
- Buy Signal 1: RecentScore > 0.85 AND [(Days_since_highest < 100) OR (Days since highest > 100 AND All_time_Score > 70)]
- Sell Signal 1: if today's close < min(closing prices in past 5 days)*stopRatio

We incorporate our regression predictions into our trading-rule based system as follows:

- Buy Signal 2 from regression model A: predicted highest return on the future 10 days > 8%

Unfortunately, the sell signal "predicted lowest return on the future 10 days < sell_threshold = -10%" achieves very bad performances. A detail check on the portfolio allocation reveals that the regression model B predicts strongly negative returns on most stocks that are big-gainers. The reason is that big-gainers tend to have higher variances. We have to tolerate some degree of loss and wait out to the top. Therefore, we use the prediction of regression model B to arrange the buying order:

- If several stocks have buy signals, invest in the stocks with "least predicted lowest return in absolute value", ie. buy the stocks which we predict that it will lose the least.

4.3 Slope with Simple Moving Average

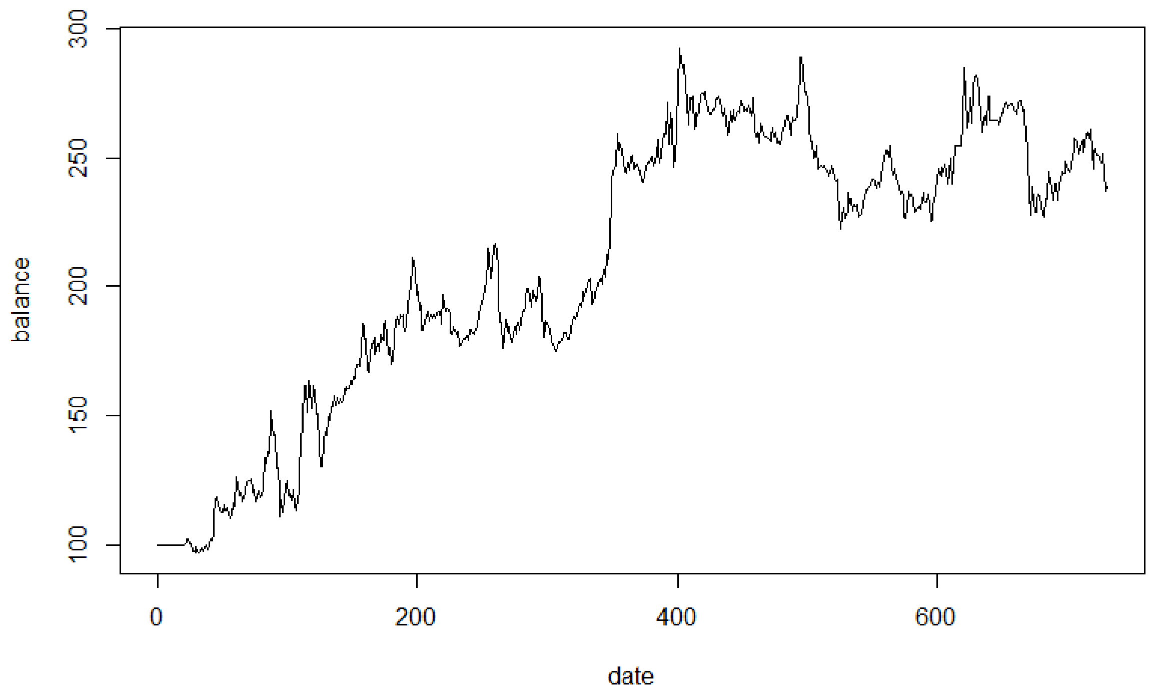
The intuition of this strategy is to indicate buy/sell action by the trend of stock price. The trend of stock price can be indicated by the slope (1st derivative) of the price curve. If the stock is going up, traders are expected to buy this stock. If the stock is going down, traders are expected to sell the stock. The strategy is valid and profitable when the window size of calculating slope is less than the period of stock fluctuation.

First of all, the SMA of close value is calculated for the judgement of slope. SMA here is used to smooth data to eliminate daily fluctuations. The stock is judged to be profitable, thus buyable, if the 1st derivative of the stock price is positive in most of the days in a judgement window. In contrast, the traders are expected to sell the stock if the 1st derivative of the stock price is negative in most of the days in a window.

For each day, if buying signal exists, choose the largest amt to buy. If selling signal exists, sell all stocks that available to sell.

Parameters and performance:
There are some parameters to be tuned. First of all, the reserved cash is set to 15, and investment cash is 13. Secondly, for SMA, I calculated 11-day moving average, which achieves the best performance. Thirdly, the slope is calculated with i-th day price against with (i-5)-th day price. Fourthly, the window size of monotony judgement is 10. Within 10 days, the stock is judged to be monotony if the slope keeps positive or negative in 80% of the days.
The final performance is shown below:

total.return	1.3863362
annual.return	0.3626338
sharpe.ratio	1.0032267
max.drop.down	0.2675137



The overall performance looks very good. However, inspecting the buying record, actually the strategy mostly focus on several stocks in stock 80-100. This strategy may not work well on new data.

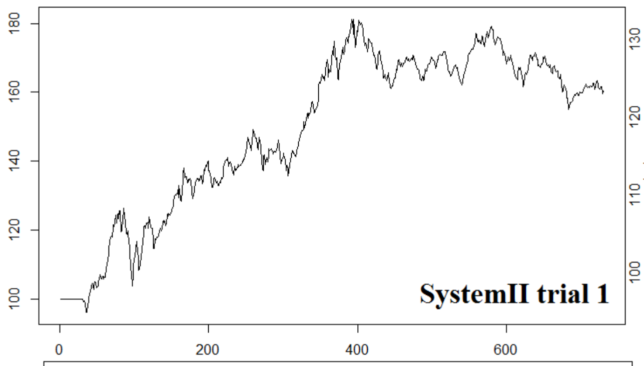
5. Evaluation and Results

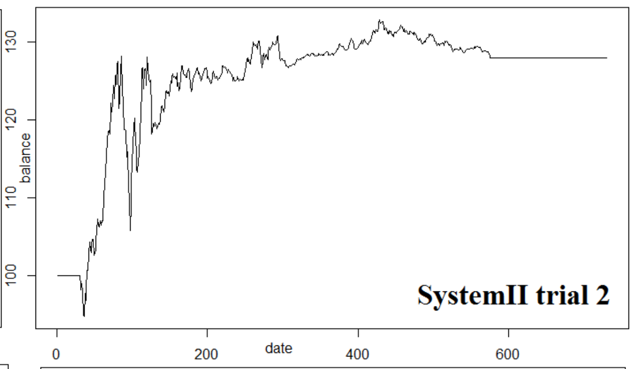
In this section we compare the performances of our systems.

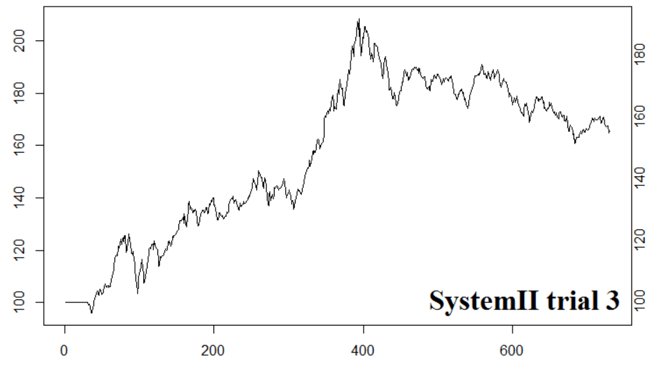
System II	Trial 1	Trial 2	Trial 3	Trial 4

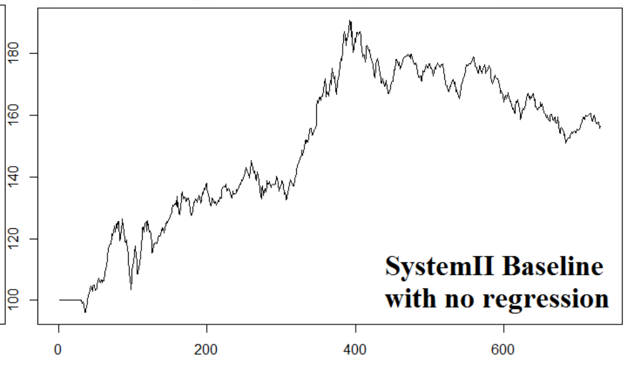
Parameters	stopLossRatio = 0.93
	betSize = 5.5
	buyThreshold = 8%
		stopLossRatio = 0.95
		betSize = 7
		buyThreshold = 8%
			stopLossRatio = 0.92
			betSize = 5.5
			buyThreshold = 10%
				No Regression
				stopLossRatio = 0.92
				betSize = 5.5

Total Return	0.6036	0.2803	0.6562	0.5642
Annualized Return	0.1739	0.09022	0.1881	0.1664
Sharpe Ratio	1.1087	0.8446	1.0699	1.0180
Max. Drawdown	0.1793	0.1749	0.2291	0.2087









5.1 Analysis

System II performs quite well in the baseline (trial 4) achieving 56% total return and Sharpe ratio of 1.018. Adding the prediction from regression algorithm improves the performances further (trial 1, 3) with over 60% total return and Sharpe ratio 1.07 – 1.11. However, when we change the parameters to trial 2, the result is much worse, which requires closer examination.

In the portfolio of trial 2, the total return is down to 28% because we are more heavily invested (betSize $7) in some stocks that achieve mediocre returns. They do not drop too much so that our sell signals are not fired. Close to the end, no buy signals are fired and the portfolio is holding purely cash. Looking at the stock charts reveal that many stocks are falling towards the end of the period, so our buy signals correctly indicate this. Since our parameters in trial 2 would make no difference on buy or sell signals compared to trial 1, the same situation (no buying at the end) happens in trial 1. So why was trial 1 performed much better? We believe the reason lies in the period from days 300 – 400, where trial 1 holds on to big-gainers and enjoy the big ride, while trial 2 dumps the big-gainers because of a tighter stopLossRatio. Trial 3 also keeps the big-gainers with a stopLossRatio 0.92 and thus achieves better returns.

Moreover, we believe the improvements from regression comes from regression model B, where it is deciding the order of which stocks to buy, than from regression model A, where it is predicting the highest return in the future 10 days. The reason is that the buy signals from model A actually fires 14000 – 25000 times averaging in the 691 days where it is in operation. Even if we set the buyThreshold to be 10%, it still fires 14000 times. Similar to the regression model B, we believe model A is exaggerating the predicted highest return. A lot of the buy signals either get filtered out by our trading rule signals or did not get execute because we are lack of cash. So the true value of our regression comes from choosing the correct order, ie. the regression predicted values are not useful, but their ranking is. We emphasize that the prediction improvement does not happen on the basis of overfitting, because the regressions are run on data from 100 different stocks, even on those that we do not buy.
In summary, System II's strength lies in its strong trading-rule based signals. The regression adds on to the good performance by cleverly choosing which stocks to buy first.

6. Conclusion

In this project, we compare trading systems based on pure trading-rules and a hybrid combining trading-rules and local regression. All of our systems achieve stellar returns because of overfitting. Even our hybrid system shows strong signs of overfitting, because it is also heavily rule based. We have just verified Burton Malkiel's main thesis accurately. The prediction from regression improves the hybrid system not at the cost of overfitting.

For future investigation, we believe a more flexible model such as polynomial regression or neural network may achieve promising result. One of our main difficulties is that our trading-rule based models are highly overfitted. Linear regression does not seem to solve this either. The best way to handle high variance is ensemble methods. However, it is not easy to come up with independent learners to form an ensemble because stocks data is highly correlated!

Generally, we hold the belief that small-scale machine learning techniques are inferior to trading-rule based models in stock trading. This is because the stock price incorporates commonly known information and expectations, but it is vulnerable to unexpected news and human irrationality. So only partial information is encoded in the prices. Another main problem is that conventional algorithms are based on IID assumptions on the samples but stocks data is highly correlated and as far as we know, there is not an adequate robust way to handle time-series data for learning algorithms. Ad-hoc methods like our simple smoothing and take the most recent lack theoretical grounding. In conclusion, stock prices are too noisy for most learning algorithms and the challenge of stock prediction is only open to the most robust ones (eg. Deep neural network).

Appendix A: R Codes

Part I:

```
setwd("C:/MSBD5013")
load("MSBD5013_Final_Project.RData")



strategy <- function(open, high, low, close, amt, buyable, sellable, initial.cash, transaction){

 ## necessary packages
 require(TTR)  # functions about technical indicators such as moving average



 # create a new matrix with all values moved downward L rows
 pushback <- function(z, L=1){
   # z is a matrix, L is a integer
   z <- as.matrix(z)
   A <- matrix(NA, nrow=L, ncol=ncol(z))
   y <- z
   y[1:L,] <- A
   y[-(1:L),] <- as.matrix(z[-((nrow(z)-L+1):nrow(z)), ])
   return(y)
 }

 # create a new matrix with all values moved upward L rows
 pushforward <- function(z, L=1){
   # z is a matrix, L is a integer
   z <- as.matrix(z)
   A <- matrix(NA, nrow=L, ncol=ncol(z))
   y <- z
   y[1:(nrow(z)-L), ] <- z[-(1:L),]
   y[(nrow(z)-L+1):nrow(z),]<- A
   return(y)
 }

 # strategy:
 # buying signal:  For each stock, when SMA 13 corsses above SMA 20, buy that stock tomorrow
 #                at the openning price with $16. If more than one stock satisfies the buying condition,
 #                just pick one stock with the highest amount yesterday.
 # selling signal: For each stock, when SMA 13 crosses below SMA 20, sell this stock tomorrow at the
 #                closing price.
 # other settings: When the available cash is less than $16, stop any buying actions.

 reserved.cash <- 16
 investment.cash <- 16

 SMA13 <- apply(close, 2, SMA, n=13)  # SMA 13
 SMA20 <- apply(close, 2, SMA, n=20)  # SMA 20

 lower <- (SMA13 < SMA20)   # indicates when SMA 13 crosses below SMA 20
 upper <- (SMA13 >= SMA20)  # indicates when SMA 13 corsses above SMA 20

 upcrossing <- pushback(lower)*upper
 # an upcross happens when yesterday's SMA 13 is lower than SMA 20 but today's SMA 13 is higher or
equal than SMA 20
 downcrossing <- pushback(upper)*lower
 # a downcross happens when yesterday's SMA 13 is higher or equal than SMA 20 but today's SMA 13 is
lower than SMA 20

 buy.signal <- pushback(upcrossing)*buyable
 # a buy signal happens when observed upcross yesterday, and the stock is "buyable" today
 sell.signal <- pushback(downcrossing)*sellable
 # a sell signal happens when observed downcross yesterday, and the stock is "sellable" today

 available.cash <- initial.cash  # initialize available cash
 position.matrix <- close*0  # initialize position matrix
 for(i in 2:nrow(close)){
    position.matrix[i, ] <-  position.matrix[i-1, ]  # load position in the previous trading day
    # buy action
    if(sum(buy.signal[i,]==1, na.rm = T)!=0 & available.cash>reserved.cash){
      # when there exist buy signals and current available cash is larger than reserved cash,
      # prepare for buying actions
      amt.order <- order(pushback(amt[i,]))  # sort the order of stocks by yesterday's amt
      target.stock <- which(amt.order==max(amt.order[which(buy.signal[i,]==1)]))
      # choose the target stock: firstly buy.signal==1, then pick the stock with max amt (yesterday)
      investment.shares <- round(investment.cash/open[i, target.stock], digits = 2)
      # calculate investment shares for the target stock
      position.matrix[i, target.stock] <- position.matrix[i, target.stock] + investment.shares
      # update position.matrix
      available.cash <- available.cash - investment.shares*open[i, target.stock]*(1+transaction)
      # update available.cash
    }
    # sell action
    if(sum(sell.signal[i,]==1, na.rm = T)!=0){
      # when there exist sell signals, prepare for selling actions
      target.stock <- (sell.signal[i,]==1)
      # choose the target stocks
      available.cash <- available.cash + sum(position.matrix[i,][target.stock]*close[i,][target.stock])*(1-
transaction)
      # update available cash
      position.matrix[i,][target.stock] <- 0
      # update position matrix
    }
    #print(available.cash)
  }

  ## return position.matrix
  return(position.matrix)
}

start <- 1
```

```
end <- nrow(CLOSE)

open <- OPEN[start:end,]
close <- CLOSE[start:end,]
high <- HIGH[start:end,]
low <- LOW[start:end,]
amt <- AMT[start:end,]
buyable <- BUYABLE[start:end,]
sellable <- SELLABLE[start:end,]

position.matrix <- strategy(open, high, low, close, amt, buyable, sellable, initial.cash, transaction)
check(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix)
performance.summary(open, high, low, close, amt, buyable, sellable, initial.cash, transaction,
position.matrix)
```

Part II:

```
set.seed(100)   # randomize buying order
strategy <- function(open, high, low, close, amt, buyable, sellable, initial.cash, transaction){

   require(TTR)

   #####  Set parameters  #####
   startDay <- 30      # hold-out period for observation
   stopRatio <- 0.92   # stopLoss bound
   betSize <- 5.5      # amount for each buy
   mlBuyThreshold <- 8 # in %
   REG_FREQ <- 10      # frequency in days of running regression

   #####  Cast data into matrix for easy computations  ########################
   position.matrix <- close*0                       # final output as dataframe
   open <- data.matrix(open, rownames=NA)
   high <- data.matrix(high, rownames=NA)
   low <- data.matrix(low, rownames=NA)
```

```
  close <- data.matrix(close, rownames=NA)
  amt <- data.matrix(amt, rownames=NA)
  buyable <- data.matrix(buyable, rownames=NA)
  sellable <- data.matrix(sellable, rownames=NA)

  #####  portfolio-, strategy-related info  #####
  nDay <- nrow(close)
  nStock <- ncol(close)
  pm <- matrix(0, nDay, nStock)        # position matrix in MATRIX class; convert to df before output
  cash <- rep(0, nDay)                 # daily cash balance
  buyReq <- rep(0, nStock)             # buy request holder
  sellReq <- rep(0, nStock)            # sell request holder
  #stopPrice <- rep(0, nStock)          # stop loss order on price
  #entryPrice <- rep(0, nStock)         # another stop loss order on price
  entryMoney <- rep(0, nStock)         # another
  tradable <- matrix(TRUE, nDay, nStock)    # we stop buying that security for 5 days if we just sell
them

  #####  indicators-related info  #####
  rsi <- apply(close, 2, RSI, n=20)           # indicators for each stock on each day
  lowestLow <- matrix(0, nDay, nStock)        # names are self-explantory
  highestHigh <- matrix(0, nDay, nStock)
  lowestLowDate <- matrix(1, nDay, nStock)
  highestHighDate <- matrix(1, nDay, nStock)
  daySinceHighest <- matrix(0, nDay, nStock)
  daySinceLowest <- matrix(0, nDay, nStock)
  recentScore <- matrix(0, nDay, nStock)      # main indicator:  = 1-william%R

  #############  compute all indicators information  #####################
  #############  NO FUTURE INFO IS USED EXCEPT UP TO THAT DAY  ###########
  for (j in 1:nStock) {
    recentScore[,j] <- 1 - WPR(cbind(high[,j],low[,j],close[,j]), n=20)
  }
```

```
highestHigh[1,] <- high[1,]
lowestLow[1,] <- low[1,]
for (i in 2:nDay) {
    highestHigh[i,] <- highestHigh[i-1,]
    highestHighDate[i,] <- highestHighDate[i-1,]
    lowestLow[i,] <- lowestLow[i-1,]
    lowestLowDate[i,] <- lowestLowDate[i-1,]

    for (j in 1:nStock) {
        if (highestHigh[i,j] <= high[i,j]) {
            highestHigh[i,j] <- high[i,j]
            highestHighDate[i,j] <- i
        }
        if (lowestLow[i,j] >= low[i,j]) {
            lowestLow[i,j] <- low[i,j]
            lowestLowDate[i,j] <- i
        }
    }
}
for (i in 20:nDay) {
    daySinceHighest[i,] <- (i-10 - highestHighDate[i-10,])/(i-10)
    daySinceLowest[i,] <- (i-10 - lowestLowDate[i-10,])/(i-10)
}

#####  Statistical Learning Part -- Linear Regression  #####
nmldata <- floor(nDay/30)                # nmldata = how many training we can do
mldata <- matrix(0, nmldata*100, 6)      # this holds the training data
mldata <- data.frame(mldata)
colnames(mldata) <- c('x1', 'x2', 'x3', 'x4', 'y1', 'y2')
mlcounter <- 0
tempy1 <- rep(0, nStock)
tempy2 <- rep(0, nStock)
mlTest <- matrix(0, nStock, 4)           # this holds the daily data for prediction
mlTest <- data.frame(mlTest)
```

```
colnames(mlTest) <- c('x1', 'x2', 'x3', 'x4')

#############  not trading in first (startDay - 1) days  #####################
#############  so cash level   ########################

cash[1:(startDay-1)] <- initial.cash

############# initialize complete; start trading~  #####################

for (i in startDay:nDay) {

   ################  first update positions, then check signals  #########
   #####  load positions from previous day  #####
   pm[i,] <- pm[i-1,]
   cashAvail <- cash[i-1]

   #####  execute buy request from previous day  #####
   if (i < 41) {                              # regression starts on day40
      amtavg <- colMeans(amt[1:(i-1),])        # if no regression,
      amtavg <- sample(order(amtavg))          # buy in random order
   } else {
      amtavg <- order(-mlPredSell)             # if regression prediction available,
   }                                          # buy the stocks with least predicted loss first

   for (j in 1:nStock) {
      sidx <- amtavg[j]               # index to buy

      if (buyReq[sidx] == 1) {
         if (buyable[i,sidx] == 1) {
            if (cashAvail > 0) {
               moneyInvest <- ifelse(cashAvail < betSize, cashAvail*0.5, betSize)
               sharesBought <-  floor(moneyInvest*100/open[i,sidx])/100
               pm[i,sidx] <- pm[i,sidx] + sharesBought
               entryMoney[sidx] <- entryMoney[sidx] + sharesBought*open[i,sidx]*(1+transaction)
```

```
                cashAvail <- cashAvail - entryMoney[sidx]
                #stopPrice[sidx] <- open[i,sidx]
                #entryPrice[sidx] <- open[i,sidx]
              }
            }
            buyReq[sidx] <- 0
          }
        }

        #####  execute sell request from previous day  #####
        for (j in 1:nStock) {
          if (sellReq[j] == 1) {
            if (sellable[i,j] == 1) {
              cashAvail <- cashAvail + pm[i-1,j]*close[i,j]*(1-transaction)
              pm[i,j] <- 0
              sellReq[j] <- 0
            }
          }
        }

        cash[i] <- cashAvail
        ################  finish updating positions  ########################
        ######################################################################
        ################  begin checking buy/sell signals  ##################

        if (i >= 40) {
          if (i %% REG_FREQ == 0) {        # run regression training every REG_FREQ days
                                  # for detail explanation, please refer to report section 3.2
            recentScoreAvg5 <- colMeans(recentScore[(i-14):(i-10),])
            rsiAvg5 <- colMeans(rsi[(i-14):(i-10),])/100
            mlIdx <- (mlcounter*nStock+1):(mlcounter*nStock+nStock)
            mldata[mlIdx,1:4] <- cbind(recentScoreAvg5, rsiAvg5, daySinceHighest[i,],
daySinceLowest[i,])
            for (j in 1:nStock) {
                tempy1[j]  <- periodReturn(max(high[(i-9):i,j]), open[i-9,j])
                tempy2[j] <- periodReturn(min(low[(i-9):i,j]), open[i-9,j])
            }
            mldata[mlIdx,5] <- tempy1
            mldata[mlIdx,6] <- tempy2

            reg.buy <- lm(y1 ~ x1+x2+x3+x4, data=mldata[mlIdx,])
            reg.sell <- lm(y2 ~ x1+x2+x3+x4, data=mldata[mlIdx,])

            mlcounter <- mlcounter + 1
        }

        #####  run regression predictions everyday after day40  #####
        mlTest[,1:4] <- cbind(recentScore[i,], rsi[i,]/100, daySinceHighest[i,], daySinceLowest[i,])
        mlPredBuy <- predict(reg.buy, mlTest)        # predicted highest return on future 10 days
        mlPredSell <- predict(reg.sell, mlTest)      # predicted lowest return on future 10 days
    }

    #####  buy signals  #####
    for (j in 1:nStock) {
        if (tradable[i,j] && pm[i,j] < 2) {
            buySignal <- rep(FALSE, 3)

            if (recentScore[i,j] > 0.85) {
                buySignal[1] <- TRUE
                buySignal[2] <- TRUE
            }

            if (buySignal[2] && (i - highestHighDate[i,j] > 100)) {
                allScore <- currentScore(highestHigh[i,j], lowestLow[i,j], close[i,j])
                if (allScore < 70) {
                    buySignal[2] <- FALSE
                }
            }
        if (i < 40) {      # buy signals from regression after day40
          buySignal[3] <- TRUE
        } else {
          if (mlPredBuy[j] > mlBuyThreshold) {
            buySignal[3] <- TRUE
          }
        }

        if (sum(buySignal) >= 2) {
          buyReq[j] <- 1
        }
      }
    }

    #####  sell signals  #####
    for (j in 1:nStock) {
      if (pm[i,j] > 0) {
        sellSignal <- rep(FALSE, 4)
        stopLossPrice <- min(close[(i-5):(i-1),j])
        if (close[i,j] < stopLossPrice*stopRatio) {
          sellSignal[1] <- TRUE
        }

        if (sum(sellSignal) >= 1) {
          sellReq[j] <- 1
          tradable[(i+1):(i+5),j] <- FALSE      # prevent buying this asset just after selling
        }

      }
    }

  }
  position.matrix[,1:nStock] <- pm
```

```
    return(position.matrix)
}



####################################
position.matrix <- strategy(OPEN, HIGH, LOW, CLOSE, AMT, BUYABLE, SELLABLE, initial.cash,
transaction)
performance.summary(OPEN, HIGH, LOW, CLOSE, AMT, BUYABLE, SELLABLE, initial.cash,
transaction, position.matrix)
```

Part III:

```
setwd("C:\\Users\\Philip\\Desktop\\1ust\\MSBD5013FinalProject\\MSBD5013FinalProject") #you need
to change the directory
load("MSBD5013_Final_Project.RData")

pushback <- function(z, L=1){
  # create a new matrix with all values moved downward L rows
  # z is a matrix, L is a integer
  z <- as.matrix(z)
  A <- matrix(NA, nrow=L, ncol=ncol(z))
  y <- z
  y[1:L,] <- A
  y[-(1:L),] <- as.matrix(z[-((nrow(z)-L+1):nrow(z)), ])
  return(y)
}

pushforward <- function(z, L=1){
  # create a new matrix with all values moved upward L rows
  # z is a matrix, L is a integer
  z <- as.matrix(z)
  A <- matrix(NA, nrow=L, ncol=ncol(z))
  y <- z
  y[1:(nrow(z)-L), ] <- z[-(1:L),]
  y[(nrow(z)-L+1):nrow(z),]<- A
  return(y)
}
# util
plt <- function(x,y){
  plot(normalize(MA5[y,x]),type='l',col='green')
  lines(normalize(MA15[y,x]),type='l',col='red')
  lines(normalize(MA60[y,x]),type='l',col='blue')
  lines(normalize(close[y,x]),type='l')
  lines(standardize(S_MA15[y,x]),type='l',col='pink')
  abline(h = 0, v = 0, col = "gray60")
}
# util
normalize <- function(a){
  max = max(a,na.rm=TRUE)
  min = min(a,na.rm=TRUE)
  ret = 2*(a-min)/(max-min)-1
  return(ret)
}
# util
standardize <- function(a){
  max = max(a,na.rm = TRUE)
  min = min(a,na.rm = TRUE)
  ret = a/(max-min)
  return(ret)
}

#calculate slope against several days before
slope <- function(a,day=5){
  yesterday <- pushback(a,day)
  slope <- (a-yesterday)/yesterday
  return(slope)
}
```

```
#check monotony within ndays
#if 0, ignore
#if 1, increasing
#if -1, decreasing
monotony <- function(data,nday=10){
  a = slope(data)
  r = a*0
  s = sign(a)
  for(i in 1:ncol(s)){
    r[,i] = runSum(s[,i],n=nday)
  }
  r = apply(r,c(1,2),FUN=function(x){
    if(is.na(x)){
      return(0)
    }
    if(x > nday*0.8){
      return(1)
    }
    if(x < -nday*0.8){
      return(-1)
    }
    return(0)
  })
  return(r)
}

strategy <- function(open, high, low, close, amt, buyable, sellable, initial.cash, transaction){

  ## necessary packages
  require(TTR)  # functions about technical indicators such as moving average

  # strategy:
```

```
  # buy condition: for each stock, when it keeps increasing in 80% of days in 10 days, buy this stock
tomorrow at openning price (need satisfy buyable
  # condition). The investment money should be around 13. When more than one stocks satisfy buy
condition, just pick
  # one stock with highest amount yesterday
  # sell condition: for each stock in current position, when it keeps decreasing in 80% of days in 10 days,
  # sell this stock tomorrow at closing price (need satisfy sellable condition)
  # other settings: when observing available cash is less than 10, stop any buying actions

  reserved.cash <- 15
  investment.cash <- 13
  MA10 <- apply(close, 2, SMA, n=11)

  MA10_m_buy <- (monotony(MA10,5)>0)
  MA10_m_sell <- (monotony(MA10,5)<0)

  # a buy signal happens when observed monotony increasing, and the stock is "buyable" today
  buy.signal <- buyable*pushback(MA10_m_buy)
  # a sell signal happens when observed monotony decreasing, and the stock is "sellable" today
  sell.signal <- sellable*pushback(MA10_m_sell)

  available.cash <- initial.cash  # initialize available cash
  position.matrix <- close*0  # initialize position matrix

  for(i in 2:nrow(close)){
    position.matrix[i, ] <-  position.matrix[i-1, ]  # load position in the previous trading day
    # buy action
    if(sum(buy.signal[i,]==1, na.rm = T)!=0 & available.cash>reserved.cash){
      # when there exist buy signals and current available cash is larger than reserved cash,
      # prepare for buying actions
      buyable_amt = pushback(amt[i,])
      amt.order <- order(pushback(amt[i,]))  # sort the order of stocks by yesterday's amt
      target.stock <- which(amt.order==max(amt.order[which(buy.signal[i,]==1)]))
      # choose the target stock: firstly buy.signal==1, then pick the stock with max amt (yesterday)
      investment.shares <- round(investment.cash/open[i, target.stock], digits = 2)
      # calculate investment shares for the target stock
      position.matrix[i, target.stock] <- position.matrix[i, target.stock] + investment.shares
      # update position.matrix
      available.cash <- available.cash - investment.shares*open[i, target.stock]*(1+transaction)
      # update available.cash
    }
    # sell action
    if(sum(sell.signal[i,]==1, na.rm = T)!=0){
      # when there exist sell signals, prepare for selling actions
      target.stock <- (sell.signal[i,]==1)
      # choose the target stocks
      available.cash <- available.cash + sum(position.matrix[i,][target.stock]*close[i,][target.stock])*(1-
transaction)
      # update available cash
      position.matrix[i,][target.stock] <- 0
      # update position matrix
    }
    #print(available.cash)
  }

  ## return position.matrix
  return(position.matrix)
}

start <- 1
end <- nrow(CLOSE)

open <- OPEN[start:end,]
close <- CLOSE[start:end,]
high <- HIGH[start:end,]
low <- LOW[start:end,]
amt <- AMT[start:end,]
buyable <- BUYABLE[start:end,]
```

```
sellable <- SELLABLE[start:end,]

position.matrix <- strategy(open, high, low, close, amt, buyable, sellable, initial.cash, transaction)
check(open, high, low, close, amt, buyable, sellable, initial.cash, transaction, position.matrix)
performance.summary(open, high, low, close, amt, buyable, sellable, initial.cash, transaction,
position.matrix)
```