The University of Hong Kong

Department of Computer Science

COMP 4801

Final Year Project 2017-2018:

Financial Data Forecaster:

Pair Trading with SVM strategy

Final Report

Date: 20/4/2018

Supervisor: Dr. C.L. Yip

Member: Ng Kwun Ting BEng(CompSC)

UID: 3035100897

Abstract

With the rapid development of computer technology, nowadays data scientists use lots of algorithms and scientific methods to do data analysis with computer. Within that, one of the famous topic is to predict the trend of financial market. Hence, Financial data forecaster is the core part of algorithmic trading in the finance industry. This project aims to build a program to predict the future value of financial products.

Acknowledgement

I would like to thank my supervisor Dr. C.L. Yip for helping and guiding me a lot throughout the process of my project.

Table of Contents

Page

List of Figures

1. **Background and Objectives**

Background:

Investing on the financial markets is what a knowledge-demanded task that many professionals in the related fields and people are doing. With the computer technology nowadays, the term “Algorithmic Trading” is prevalent to the industry. In fact, there are lots of companies and individuals out there, who possess very complexing algorithms that can actually benefit from the financial market steadily. Although, there are many papers published to tell their trading algorithms, the really powerful ones are not published because of commercial secret and that is what we call the “Black Box”.

Objectives:

The goal of our project is to use scientific methods for building one of the powerful algorithms to forecast the financial data in the future. Hopefully, it can actually trade and benefit from the financial market, that is, “open the Black Box”.

2. **Theoretical Background**

2.1 Brief background description:

The technologies used in this project involve the following theoretical background:

Pair trading:

Pair Trading is a method and trading strategy/concept that to find out a pair of stocks that are positive correlated. When there is a deviation between the two prices of the two stocks, there will be a force that the one (stock's price) gone above the average will go down and the other one(stock's price) gone below the average will go up until they reach an average/balance.

Machine Learning:

Machine learning is a particular aspect of artificial intelligence, it mainly acts as a classifier to sort different items by types. In this project, classifier such as support vector machine(SVM) are used to sort data to different types.

Time-series Analysis:

Time-series analysis is a hot aspect in statistics that considers the time-series data, based on regression methods. In this project, the financial time-series data and statistics technique with two time series data is used

Technical indicators:

Technical indicators are developed by the finance industry to help understanding and predicting the trend of stock markets. It can be divided into two categories: trend indicators such as MACD and momentum indicators such as RSI and KD.

2.2 More theoretical background description on important techniques used:

2.2.1 Pair Trading

Pairs trading is a market-neutral trading strategy that matches a long position with a short position in a pair of highly correlated instruments such as two stocks, exchange-traded funds (ETFs), currencies, commodities or options. Pairs traders wait for weakness in the correlation and then go long the under-performer while simultaneously short selling the over-performer, closing the positions as the relationship returns to statistical norms.

The strategy's profit is derived from the difference in price change between the two instruments, rather than from the direction each moves. Therefore, a profit can be realized if the long position goes up more than the short, or the short position goes down more than the long (in a perfect situation, the long position rises and the short position falls, but that's not a requirement for making a profit). It's possible for pairs traders to profit during a variety of market conditions, including periods when the market goes up, down or sideways – and during periods of either low or high volatility.

2.2.2 Technical indicator

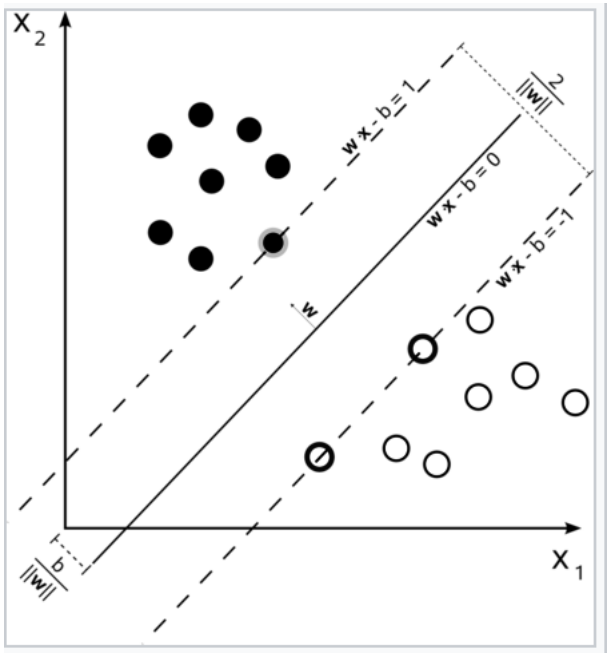
Technical indicators are mathematical calculations based on the price, volume, or open interest of a security or contract. By analyzing historical data, technical analysts use indicators to predict future price movements. Examples of common technical indicators include Relative Strength Index, Money Flow Index, Stochastics, MACD and Bollinger Bands.

Examples of Technical Indicators

The following chart shows some of the most common technical indicators, including moving averages, the relative strength index (RSI), and the moving average convergence-divergence (MACD).

2.2.3 SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.



Kernel trick

The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary.

2.2.4 Grid search with n-fold cross validation

In machine learning, two tasks are commonly done at the same time in data pipelines: cross validation and (hyper)parameter tuning. Cross validation is the process of training learners using one set of data and testing it using a different set. Parameter tuning is the process to selecting the values for a model's parameters that maximize the accuracy of the model.

While Grid Search, for example, a typical soft-margin SVM classifier equipped with an RBF kernel has at least two hyperparameters that need to be tuned for good performance on unseen data: a regularization constant C and a kernel hyperparameter γ. Both parameters are continuous, so to perform grid search, one selects a finite set of "reasonable" values for each,

for example,

C in {10,100,1000}

gamma in {0.1,0.2,0.5,1.0}

Grid search then trains an SVM with each pair (C, γ) in the Cartesian product of these two sets and evaluates their performance on a held-out validation set (or by internal cross-validation on the training set, in which case multiple SVMs are trained per pair). Finally, the grid search algorithm outputs the settings that achieved the highest score in the validation procedure.

3.Scope

3.1 Data and language

In this project, historical data of year 2002-2017 (component stocks from HSI(Hang Seng Index), SSE50(Shanghai Stock Exchange Index), DJIA(Dow Jones Industrial Average) are used and collected from google/yahoo finance API. The forecaster is implemented by python.

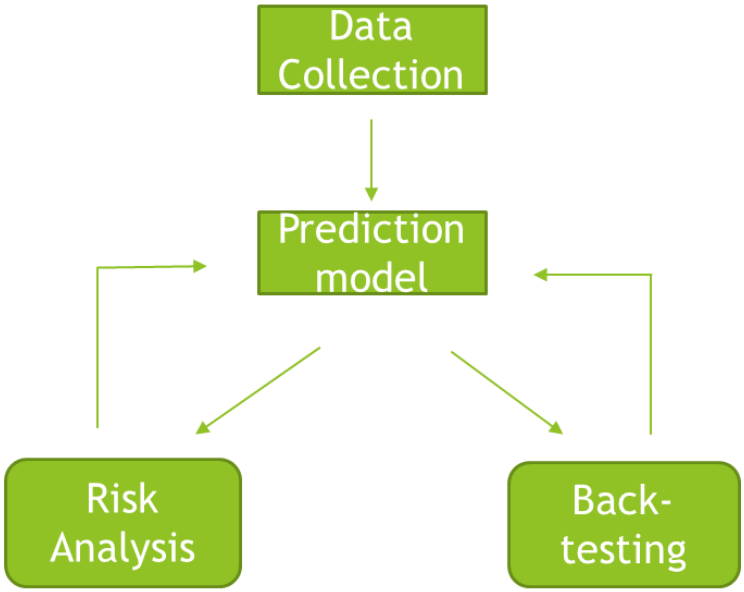
4. Methodology

4.1 General model concept

The general model of a financial forecaster is as follows:

Figure 1. General model

1. Data Collection
2. Prediction model
3. Back-testing
4. Risk Analysis



As by the efficient market hypothesis (Malkiel, April 2003) of some economists, the financial market has reflected all the information of factors on the price , and therefore , it cannot be predicted. However, some research (Khaidem, L., Saha, S., & Roy Dey, S. (2016)) state that the efficient market is a concept of the whole financial markets in different countries. Within the efficient market, there are a mixture of smaller efficient and inefficient markets and this is why many predictions come.

As we believe in the second claims above that there are small inefficient financial markets contained in the big general efficient financial market, we plan to develop below method to find out the predictable stocks, which integrates pair trading, technical indicators, machine learning, statistics method.

5. Experiments and Results:

5.1 SVM with typical pair trading strategy:

5.1.1 Steps Description:

The steps are as follows:

1. Because of the efficient market hypothesis(EMH), it is difficult to predict the market trend, therefore, I use the following algorithm, trying to predict the market.

2. First, I use the concept of pair trading to match some stocks in pairs by a time-series analysis model : Cointegration model , which can prove the two stocks are positive correlated statistically.

3. To this step, I know that, the pairs I find will have one stock's price(in a pair) go down and the other one's price(in a pair) go up in the future until the two prices attain an average(concept of pair trading).

4. Then I use 3 prediction rules as the trading signal, only if the data fulfil all the 3 criterions, the trading signal can then be generated.

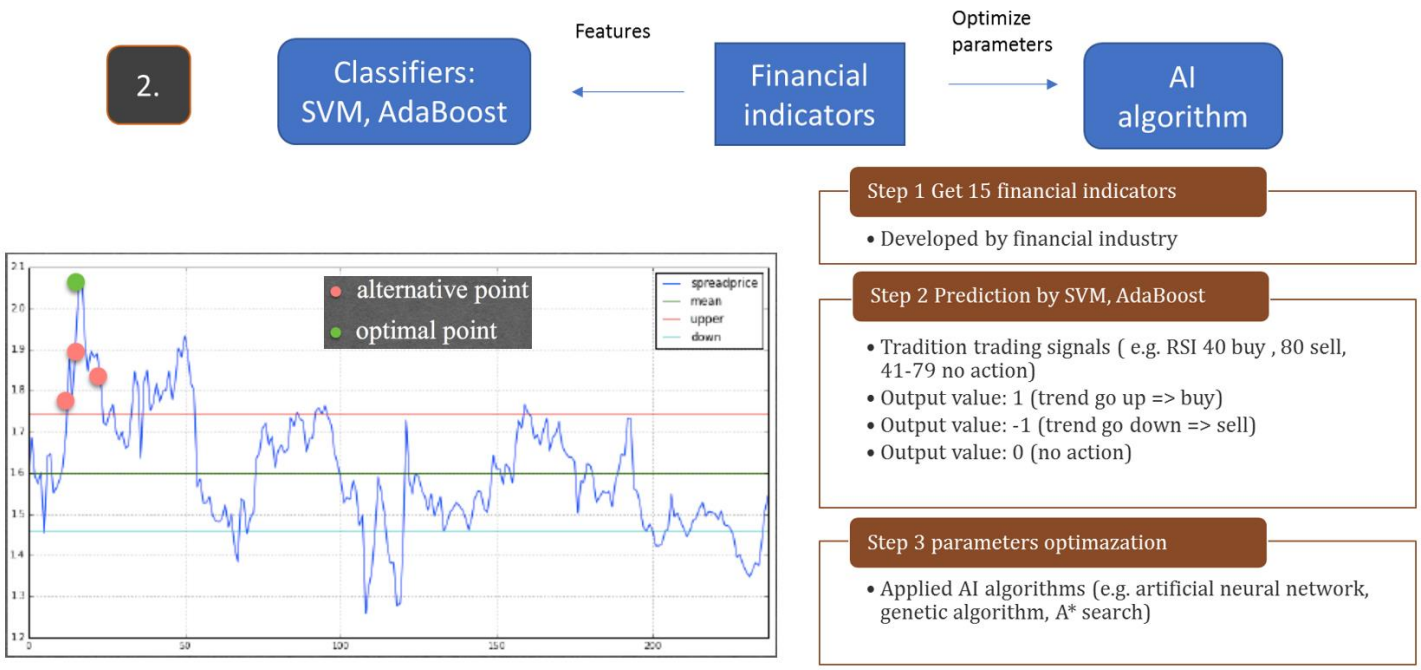
5. The first prediction rule is the traditional strategy of pair trading, which uses the spread(the difference of prices of the pair) and the parameters from the regression in Cointegration model (alpha and beta) and the statistics from the spread of formation period (mean and standard deviation).

6. The second prediction rule is the most important part of my algorithm. It is mainly use the classifiers in machine learning (SVM, Adaboost) and use technical indicators(sush as RSI,KD,MACD) in the finance industry as the features to classify when is the entry point when the first prediction rule is fulfilled.

7. In order to further against the efficient market hypothesis(EMH), I use the "white noise" test (a "white noise" means a time-series is a random process that can not be predicted) to judge whether the trading period(time-series) is a white noise or not, if it is not a white noise, it means an autocorrelation existed for the stock tested in that trading period.

8. Optimizing the parameters in step5 (e.g. the traditional entry point 1.5sd) and step6(e.g. parameters and put/short signal of RSI) by AI algorithm.

For some explanations of step6 to 8:



By traditional pair trading strategy, the entry point is the red line (when the spread excesses mu +/- 1.5 sd), but the spread may continue to become larger in value and the optimal entry point is the peak (green point of the above graph). Of course, we don't know when is the optimal entry point, but by adding the second prediction rule (classifers, technical indicators as the features), we can obtain an entry point(red point of the above graph) at least as good as the traditional entry point because the algorithm must fulfil the above two prediction rules to generate the entry signal.

5.1.2 prediction model architecture

The prediction model architecture of the above description:

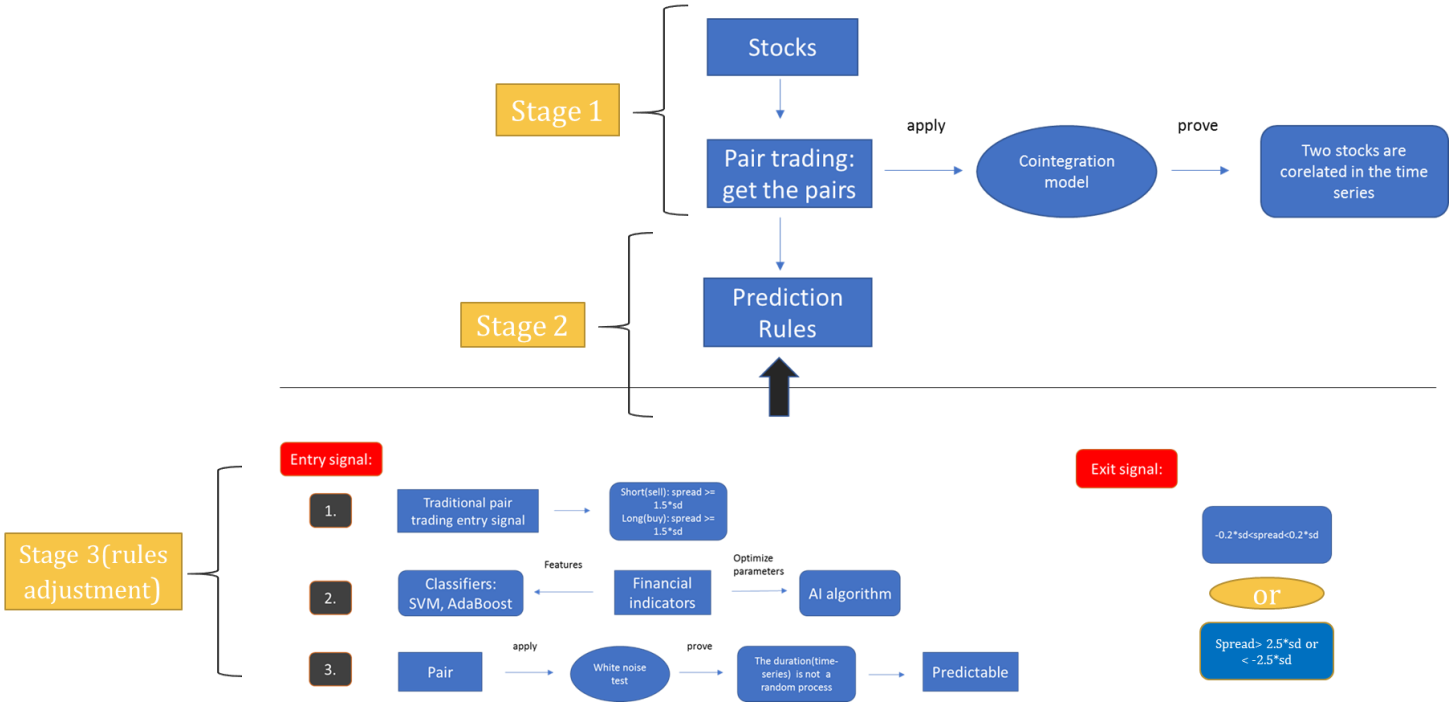


Figure 2. prediction model architecture of SVM with typical pair trading strategy

5.1.3 Some Results from typical pair trading strategy

the pair trading uses the formation period as the formation of a pair and the trading period as the back testing.

The pair formation uses the cointegration model which has the core equation:

$$Spread_t = \log\left(P_t^Y\right) - \left[\hat{\alpha} + \hat{\beta}\log\left(P_t^X\right)\right]$$

Here, Y and X presents the two stocks in a pair and P is the price.

The value of alpha and beta can then be calculated by regression, which are then be used in calculating the spread of trading period.

Also, the mean and standard deviation of spread of formation period are calculated, which are used as the traditional entry signal (spread excesses mu +/- 1.5 sd) and exit signal (spread returns back to mu +/- 0.2 sd or spread excesses mu +/- 2.5 sd(the relationship of pair broken)) in the trading period.

Also, the program contains the Augmented Dickey–Fuller test (ADF) tests, which is used to test a time series data is weakly stationary(weakly stationary means there is autocorrelation of the time series itself and therefore it may be predictable). The ADF tests are used as a prerequisite for the two stocks to test before the cointegration model and there is also an ADF test in the cointegration model to test the spread.

After that, I have built a simulation account for backtesting, and the result is: there is some profits in general, but the amount of profit is very various and depends on the parameter values that I input.

For example, for a pair from SSE50 (no.: 601988, 600000) using the same formation period(i.e. same alpha and beta to calculate the spread of trading period) :

Trading period: 2015-01-05 to 2015-12-31

Entry signal: spread excesses mu +/- 1.5 sd

Exit signal: spread returns back to mu +/- 0.2 sd or spread excesses mu +/- 2.5 sd

```
=================== Backtesting in a trading period =====================

Trddt
2015-01-05    100000.0
Name: Asset, dtype: float64
Trddt
2015-12-31    143849.353626
Name: Asset, dtype: float64

Backtesting result:
cash used for put/short of each position: 20000
Asset at the beginning of the trading period: 100000.0
Asset at the end of the trading period: 143849.35362571833
Asset percentage change: 143.85 %
Profit gain: 43.85 %
========================================================================
```

Trading period: 2015-01-05 to 2015-06-30

Entry signal: spread excesses mu +/- 1.6 sd

Exit signal: spread returns back to mu +/- 0.3 sd or spread excesses mu +/- 2.8 sd

```
=================== Backtesting in a trading period =====================

Trddt
2015-01-05    100000.0
Name: Asset, dtype: float64
Trddt
2015-06-30    181264.543659
Name: Asset, dtype: float64

Backtesting result:
cash used for put/short of each position: 20000
Asset at the beginning of the trading period: 100000.0
Asset at the end of the trading period: 181264.5436590488
Asset percentage change: 181.26 %
Profit gain: 81.26 %
========================================================================
```

From the above result, we can see that by adjusting the parameters to different values, the trading period is halved but the profit is nearly double. (43.85% to 81.26%)

It means that only by the first prediction rule, the return is very unstable and mainly depends on the parameter values and trading period and some time may suffer small amount of loss. (Of course the return is for the assumption that we can put/short the stocks with the prices on the series and no extra cost is needed)

5.1.4 Some Results from typical pair trading strategy

Problem with the above strategy:

Shortage: very few testing data within a year

- very few amount trading signals generated
  - A pair with only about 5 data on average
  - Not enough to prove the accuracy statistically
  - Adding rule 2 and 3 will only make it worse

5.2 New thought:

In order to generate more signals to make the data high frequent enough to get a more reliable accuracy statistically, We decided to make the prediction window from 'buy in when the 'spread excesses mu +/- 1.5 sd and sell when back to normal level' to 'prediction on next trading day trend'.

There are totally four attempts below, for each one, which is based on some observation and inference to try best to improve the accuracy.

5.3 Attempts 1(Experiment 1):

Thought1: A pair means they have similar trend, make their price together as features which should be able to reflect the trend.

Problem: one stock can have pairs with several other stocks

-If the pair is not similar enough, combines the features will make more noise

Solution: Only use closest pairs

Method to find the closest pairs:

Traditional method:

Cointegration model method

Stated above,

Which has a disadvantage that it only proves whether two stocks can form a pair but not giving a measure criteria to judge which two stocks are the most similar pair. For example, consider that there are stock A,B,C,D,E,F and G. There are totally 15 combinations e.g.{A,B}, {D,G}.

If we want to find out the pairs among them (for every two stocks, whether they are similar to each other on the closing price's time series), we can apply the Cointegration model, which combines regression and ADF test (Augmented Dickey-Fuller test) to prove the pair statistically.

However, the output can be, for example, {A,B} ,{A,C} , {D,E}, {C,D}.

We can see that stock A, B, C actually form more than one pair to some stocks else. If we want to combine the features of a pair to predict next day trend, we have to find the best pair for the stock if it

actually can form more than a pair to other stocks in order to make better accuracy.

Therefore, in this case, the Cointegration model may not be a good choice to find out pairs.

Method applied

Another method to be used instead: SSD method (Smallest square distance)

The SSD method uses the formula below to calculate the smallest square distance between two stocks (X and Y):

$$r_t^i = \frac{P_t^i - P_{t-1}^i}{P_{t-1}^i}, \quad t = 1, 2, 3 \cdots T$$

$$\hat{p}_t^i = \sum_{\tau=1}^{t} (1 + r_\tau^i)$$

$$SSD_{X,Y} = \sum_{t=1}^{T} (\hat{p}_t^X - \hat{p}_t^Y)^2$$

After calculating every combination of two stocks in the same index(HSI, SSE50, DJIA), we can get a list of SSD values for each stock.

We sort the lists in ascending order, as the smaller the SSD value is, the more similar the pair is. Therefore, the first value(element) in the list will be the stock that is most similar with the stock that owns the list.

Only two stocks with the first value(element) being both each other are considered as a “useful” pair. (It means that the SSD method is

used to find out these “useful” pairs in HSI, SSE50 and DJIA to perform the steps and attempts afterward.)

An example to verify the “useful” pair:

Part of the program output:

```
Finding the pair for 2012
Pair formation period to be used: 2002-01-01 to 2012-01-01

Pair found in HSI:
{'2628', '2318'}
{'0388', '1109'}
{'0011', '0012'}
{'0066', '0003'}
{'0019', '0004'}
{'0101', '0883'}
{'1044', '0700'}
{'0386', '0144'}
Pair found in SSE50:
{'600016', '600000'}
{'600030', '600519'}
Pair found in DJIA:
{'V', 'MMM'}
{'XOM', 'CVX'}
{'HD', 'INTC'}
{'NKE', 'MCD'}
{'MRK', 'PFE'}
{'VZ', 'MSFT'}
```

Fig3. An sample output of pairs in a year

For example, the pair 2628 and 2318 in HSI:

Fig4: 2628.HK: 2002-2012 closing price

Fig5: 2318.HK: 2002-2012 closing price

We can see that, by the SSD method to find the closest pairs, the output pairs did have similar trend within a ten-year period.

We use the previous ten-year period as the period to verify a pair, for example, if we input “2012” (the year that we want to predict), the output pairs are based on “01-01-2002 to 01-01-2012” data as fig.

Concrete action of thought 1:

- Step 1: For each stock, calculate SSD with other stocks in the same index (HSI, SSE50, DJIA)

- Step 2: Find the pairs with both have each other as the smallest SSD

- Step 3: For each pair, calculate 7 financial indicators:
  - RSI, momentum, SMA, WMA, EWMA, MACD, CCI

- Step 4: Combine the 7 financial indicators, low, high, close, open price of a pair as features, UP/Down of the stock itself as Label (next trading day trend)
  - Total 22 features

- Step 5: Normalize the data

- Step 6: Train and test each stock of the pair with SVM (previous one year as training set)
  - two training data sets, two testing data sets (SVM Grid Search with 5-fold cross-validation to find the best parameter)

Result of attempt 1:

2012 -2017:

Accuracy(Average):

HSI pairs: 50.94% on 21552 data

SSE50 pairs: 51.55% on 4844 data

DJIA pairs: 51.92% on 20658 data

SSE50,DJIA pairs: **51.85%** on 25502 data

all pairs: **51.43%** on 47054 data

Return(simulation Account):

Simple: 6.80%

Compound: 6.83%

fig6. Sample output of attempt 1 result

```
################################################################################
SVM with pair strategy on 2017[window size=1](overall):

Total Pair: 17

HSI accuracy: 49.07% on 2700 data
HSI Return(simple): 8.59%
HSI Return(compound): 8.12%

SSE50 accuracy: 54.28% on 888 data
SSE50 Return(simple): -5.06%
SSE50 Return(compound): -6.46%

DJIA accuracy: 53.13% on 4122 data
DJIA Return(simple): 13.21%
DJIA Return(compound): 13.84%

Average accuracy: 51.84% on 7710 data
Average Return(simple): 9.43%
Average Return(compound): 9.43%

condiser trade day and trade frequency
Adjusted Average Return(simple): 10.40%
Adjusted Average Return(compound): 10.40%
################################################################################
```

Reflection on attempt 1:

For the pairs in HSI, the average accuracy is 50.94% on 21552 data(total), which is only slightly better than random guess although the testing(predicted) data is huge(in financial stocks price field).

While for pairs in SSE50 and DJIA, the average accuracy are 51.55% on 4844 data(total) and 51.92% on 20658 data(total), which are both higher than 51.50%.

In conclusion on the attempt 1, the average accuracy of pairs in (SSE50 + DJIA) is 51.85% on 25502 data(total), which shows that the method suggested above do have some prediction ability to predict next day stock price trend, and by this method, the simulation account shows an average 6.83% year return.

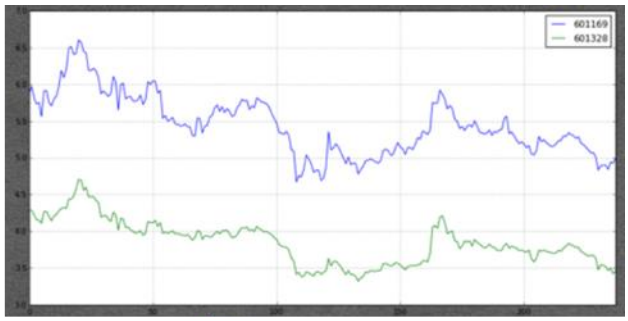
However, the 51.85% accuracy is still low, compared to 55%.

We then consider by adding one more crucial feature in the next attempt for the sake of model improvement.

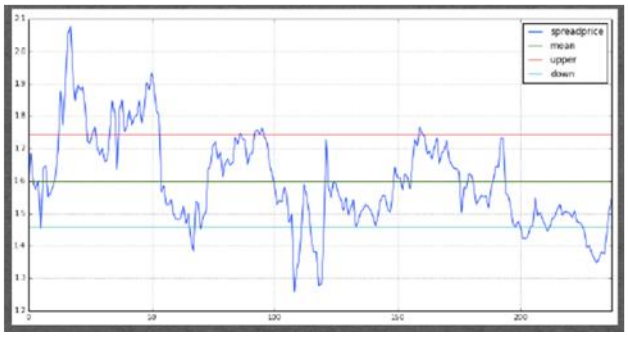
5.4 Attempt 2(Experiment 2):

- Thought2: As the model is using pair, trying to be predicted correctly by SVM
- => the 'spread' should be a main feature in this problem

Spread: spread is a basically the log difference of the prices between the two stocks on the time series, as the following equation:



transform



$$Spread_t = \log\left(P_t^Y\right) - \left[\hat{\alpha} + \hat{\beta}\log\left(P_t^X\right)\right]$$

By adding the spread as one more feature, there are in total 23 features.

Concrete action of thought 2:

- Step1: calculate spread by the equation
- Step 2: add the spread as one more feature to the model
- Step3: Train and test by SVM (23 features)

- SVM Grid Search with 5-fold cross-validation to find the best parameter

Result of Attempt 2:

2012 -2017:

Accuracy(Average):

HSI pairs: 50.53% on 21552 data

SSE50 pairs: 51.61% on 4844 data

DJIA pairs: 51.62% on 20658 data

SSE50,DJIA pairs: **51.62%** on 25502 data

all pairs: **51.12%** on 47054 data

Return(simulation Account):

Simple: 6.44%

fig7. Sample output of attempt 2 result

```
################################################################################
SVM with pair strategy on 2017[window size=1](overall):

Total Pair: 17

HSI accuracy: 49.15% on 2700 data
HSI Return(simple): 8.98%
HSI Return(compound): 8.41%

SSE50 accuracy: 53.27% on 888 data
SSE50 Return(simple): 1.31%
SSE50 Return(compound): -0.21%

DJIA accuracy: 52.50% on 4122 data
DJIA Return(simple): 11.29%
DJIA Return(compound): 12.11%

Average accuracy: 51.41% on 7710 data
Average Return(simple): 9.30%
Average Return(compound): 9.36%

condiser trade day and trade frequency
Adjusted Average Return(simple): 10.26%
Adjusted Average Return(compound): 10.32%
################################################################################
```

Reflection on attempt 2:

- 51.85% (attempt1) => 51.62% (attempt2)

After adding the 'spread' feature, the accuracy of the prediction, on the contrary, has decreased slightly from 51.85% to 51.62%, which is in fact, out of our expectation.

- Observation and inference:

1. Pair is normally used to find out any arbitrage chance

2. The spread is used to find out the timing when

the pair behave from abnormal trend back to the same trend

- it is the time when the pair trend go with different direction (one up and one down)

3. Therefore, the spread itself contains noise

Explanation of point 3: It is because when adding the spread as the feature, this feature is expected to reflect a pair that has the same trend, thus helping to forecast the label. However, sometimes, the two stocks in a pair have trend with opposite directions, and normally, spread is used to find out this period to make arbitrage chance. It means that when this period occurs, the spread becomes a noise as a feature (because the predicted labels of the two stocks of a pair are predicted separately).

Therefore, the following attempt 3 is trying to fix this problem.

5.5 Attempt 3(Experiment 3):

- Based on the observation and inference on attempt 2
- => in order to make the spread feature as “useful”
- => it needs some trick to remove the noisy nature (at least part of it)

Thought 3:

Taking the advantage of how to generate pair trading signal normally

=> trade when the spread excess spread_mean +/- 1.2*spread_sd

-Where spread_mean, spread_sd are calculated from past one-year data

The trick:

Do not trade when the spread excess spread_mean +/- 1.2*spread_sd

=> Which is equal to not trade when the trend go different direction

=> Which is equal to eliminate the noisy period of the pair (part of it)

Explanation: The trick(method) proposed above is trying to remove the noisy period of the spread (when the two stocks in a pair go with different trend on the closing price time series), thus, making the ‘spread’ to be a useful feature to predict the label (because by removing the noisy period, the spread is actually showing the same trend of the pair).

- Concrete action of thought 3:

- Step1: Eliminate the row of data where
  spread >spread_mean + 1.2*spread_sd and spread < spread_mean − 1.2*spread_sd

- Step2: Train and test by SVM
  - SVM Grid Search with 5-fold cross-validation to find the best parameter

Result of attempt 3:

2012 -2017:

Accuracy(Average):

HSI pairs: 50.67% on 8508 data

SSE50 pairs: 51.04% on 1914 data

DJIA pairs: 52.14% on 6978 data

SSE50,DJIA pairs: **51.90%** on 25502 data

all pairs: **51.30%** on 47054 data

Return(simulation Account):

Simple: **8.86**%

Fig8. A sample output of attempt 3 result

```
################################################################################
SVM with pair strategy on 2017[window size=1](overall):

Total Pair: 13

HSI accuracy: 50.08% on 1276 data
HSI Return(simple): 8.84%
HSI Return(compound): 9.01%

SSE50 accuracy: 49.41% on 170 data
SSE50 Return(simple): 2.95%
SSE50 Return(compound): 3.90%

DJIA accuracy: 52.51% on 1472 data
DJIA Return(simple): 4.40%
DJIA Return(compound): 4.55%

Average accuracy: 51.27% on 2918 data
Average Return(simple): 5.89%
Average Return(compound): 6.17%

condiser trade day and trade frequency
Adjusted Average Return(simple): 13.11%
Adjusted Average Return(compound): 13.73%
################################################################################
```

Reflection on attempt 3:

- Accuracy:
- 51.85% (attempt1) => 51.62% (attempt2) => 51.90% (attempt3)
- Return:
- 6.83% (attempt1) = > 6.49% (attempt2) = > 9.11% (attempt3)

We can see that by the action of attempt 3, the accuracy and return both have some improvement, it reflects that, by removing the noisy period of the pair, the additional 'spread' feature is useful (at least some) to predict the label (next day trend).

However, it only shows little improvement on the accuracy of prediction (from 51.85% of attempt 1 to 51.90% of attempt 3, only 0.05% increase), which is again out of our expectation.

As the core concept and application we used in this project is pair trading with machine learning, all the features (23 features with 11 features from each stock of a pair + 'spread' feature) are relevant to the trend of a pair to increase the prediction power. The 'spread' feature, as the most important one (the feature that is most reflecting the trend of a pair), should be good enough to make some improvement on the prediction accuracy.

However, the result on the contrary, showing only very little improvement. There must be some reasons of this undesirable result and therefore, the following attempt 4 is to investigate this reason and try to get a further improved model.

5.6 Attempt 4(Experiment 4):

Thought 4:

=> There should be somethings wrong/omitted when eliminating the noisy period

=> The 'spread' feature should be fine by the adjustment of Attempt3

=> The features, the 'low', 'high', 'close', 'open' should be fine too

- as it is simply the data of the exact days

=> But when we consider the 7 technical indicators, there are actually a problem here

=> Most of the technical indicators we selected are using past 5-14 days data to calculate the value

- for example, SMA is the average of the past 5 days closing price data (we set the SMA parameter to be 5 in this project)

=> However, parts of the technical indicators still contain the days within the noisy period that were just eliminated

=> It needs a trick to reduce the effect (by remove some days of data right after the noisy periods)

Explanation: It is because when we eliminate the noisy periods, the technical indicators' data remained just after the period still contains some information of the tails of the noisy periods we just eliminated. It is inappropriate, as the days involved have already be removed. Therefore, we should also remove some days of data right after the noisy periods.

The trick:

To reduce the effect stated above, although we can remove some days of data right after the noisy periods, there are still two problems:

1. Most of the parameters we set to the technical indicators are 5 (5 days) and for RSI, it is 14 (14 days as suggested by the financial industry), the number of days we should choose to remove more have no certain answer. (For example, if we choose to remove 5 days more, the effect (stated above) may still be crucial, if we choose to remove more data (e.g. 14 days), it would be too much and less data remains, also, it is unrealistic that, considers in real trading situation, we have to wait 14 trading days each time the noisy period finished (as the model is to predict the next day trend, if we have to wait 14 trading days, it is rather better to change the prediction window to larger value, for example, prediction on the trend 10 days after))

2. Sometimes, the noisy period may be a 'fake' noisy period, we may unnecessarily remove the data. (As the elimination criteria in attempt 3 is according to:

   spread >spread_mean + 1.2*spread_sd and spread < spread_mean – 1.2*spread_sd

   some values of the data fulfil this criteria but soon back to the interval (spread between spread_mean +/- 1.2*spread_sd))

Concrete action of thought 4:

Eliminate 6 more trade day data after fulling the requirement:

1. spread >spread_mean + 1.2*spread_sd and spread < spread_mean − 1.2*spread_sd

2. the consecutive days of a noisy period **> 9 days** => Assume to be a valid noisy period

Explanation:

The two requirements/criteria above are trying to solve the two problems of the trick stated above.

For requirement 1:

As stated above too, most of the technical indicators' parameters are set to 5 (5 days), as trials, we tried to eliminate 6 and 8 more trade day data respectively. The result showed that '6' is a better choice than '8'.

For requirement 2:

We make a counter to count the consecutive days of a noisy period. If it is larger than 9 days, it is assumed to be a valid noisy period(it means that we remove 6 more trade days' data only if it is a valid noisy period).
Again, as trials, we tried '9' and '12' days and the result showed that '9' is a better choice than '12'.

Result of attempt 4:

2012 -2017:

Accuracy(Average):

HSI pairs: 50.51% on 8274 data

SSE50 pairs: 53.64% on 1870 data

DJIA pairs: 52.60% on 6532 data

SSE50,DJIA pairs: **52.83%** on 8402 data

all pairs: **51.68%** on 16676 data

Return(simulation Account):

Simple: **10.51**%

Compound: **12.89**%

Fig9. A sample output of attempt 4

```
################################################################################
SVM with pair strategy on 2014[window size=1](overall):

Total Pair: 20

HSI accuracy: 51.31% on 992 data
HSI Return(simple): -0.32%
HSI Return(compound): -0.05%

SSE50 accuracy: 53.78% on 476 data
SSE50 Return(simple): -3.24%
SSE50 Return(compound): -1.63%

DJIA accuracy: 53.48% on 1524 data
DJIA Return(simple): 6.05%
DJIA Return(compound): 6.24%

Average accuracy: 52.81% on 2992 data
Average Return(simple): 2.11%
Average Return(compound): 2.54%

condiser trade day and trade frequency
Adjusted Average Return(simple): 7.05%
Adjusted Average Return(compound): 8.49%
################################################################################
```

Results comparison:

- Accuracy:
  51.85% (attempt 1) => 51.62% (attempt 2) => 51.90% (attempt 3) => 52.83% (attempt 4)

- Return:
  6.83% (attempt 1) = > 6.49% (attempt 2) => 9.11% (attempt 3) => 12.89% (attempt 4)

We can see that by removing more data after the tail of noisy periods according to the two requirements in attempt 4, the model shows an improvement on both the prediction accuracy and return (annual average return).

For accuracy, it is enhanced from 51.90% (attempt 3) to 52.83% (attempt 4).

In conclusion, although the model prediction accuracy is increased only nearly **1%** (from 51.85% to 52.83% by the additional technique of attempts 2-4), the return is actually nearly doubled. It reflects that, on next day trend prediction, every 1% accuracy improvement is very crucial because the accumulated effect of higher frequency of trading (more trading signals/data generated).

6.The Final Model:

6.1 Final Algorithm:

Below is our final algorithm of the prediction model:

- Step 1:For each stock, calculate SSD with other stocks in the same index(HSI, SSE50, DJIA)
- Step 2: Find the pairs with both have each other as the smallest SSD
- Step 3: For each pair, calculate 7 financial indicators:

  - RSI, momentum, SMA, WMA, EWMA, MACD, CCI

- Step 4: Combine the 7 financial indicators, low, high, close, open price of a pair as features
  -UP/Down of the stock itself as Label (next trading day trend)
- Step5: Normalize the features
- Step 6: calculate and add the 'spread' feature to the model (Total 23 features)
- Step 7: calculate spread_mean and spread_sd
- Step 8: eliminate noisy period data based on spread_mean and spread_sd
- Step 9: eliminate 6 more data based on valid noisy period
- Step 10 : Train and test by SVM
- - SVM Grid Search with 5-fold cross-validation to find the best parameter

6.2 The Designed Algorithmic Trading System Overview:

6.2.1 System Working Principle based on the Final Algorithm Designed:

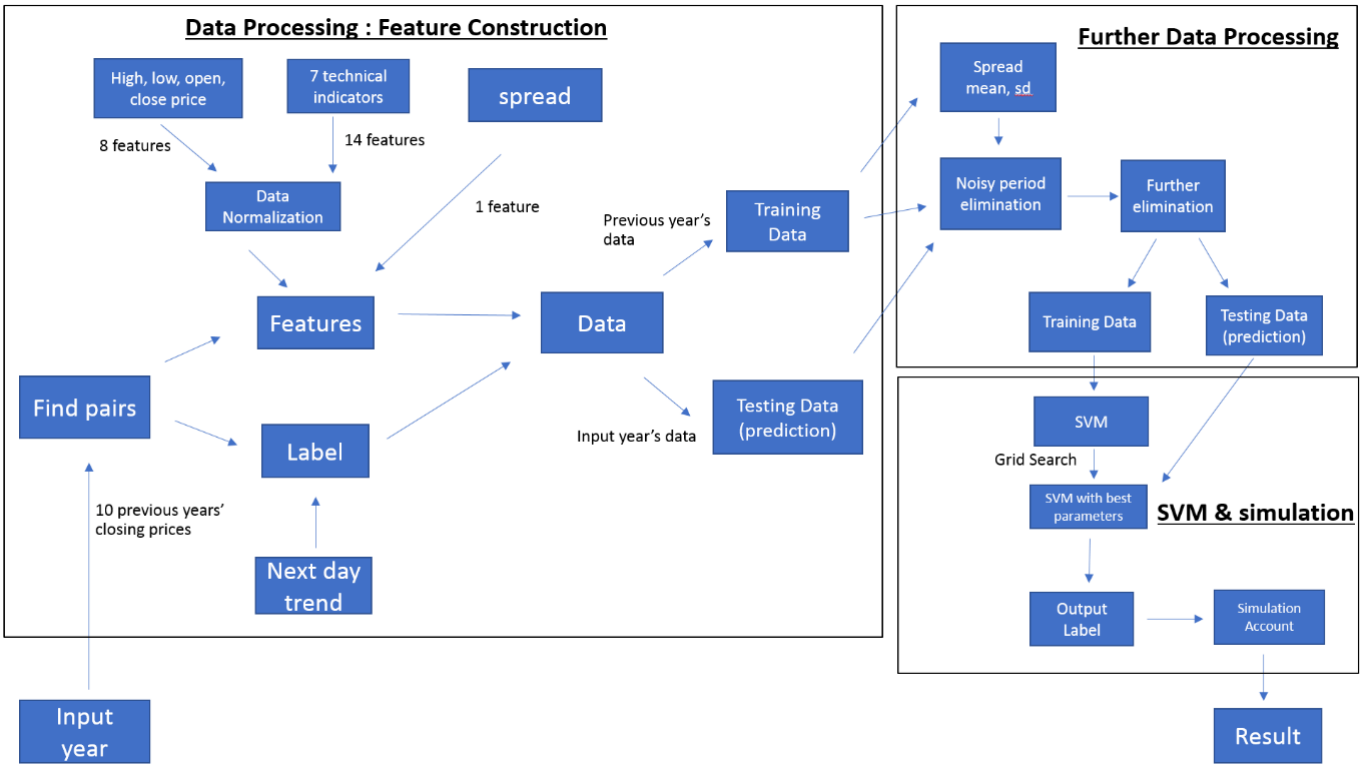


Fig10. System Working Principle

6.2.2 System Realization:

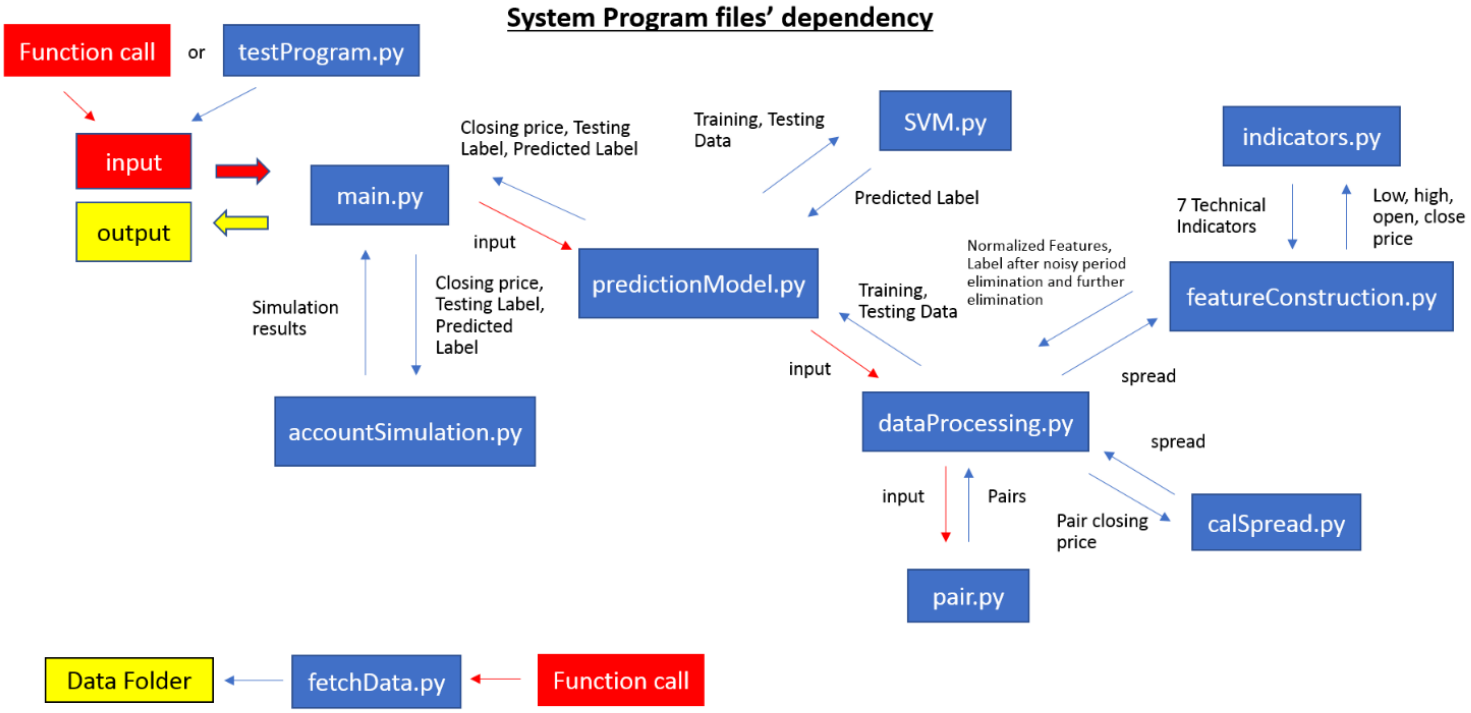


Fig11. System Realization

7.Deliverables

In total, there are 4 deliverables:

1. JupyterNotebook folder
   - contains some demonstration of the program

2. program folder

-The program (folder) contains 11 py files and one folder:

-main.py (This is the main py file of the project)

-accountSimulation.py (This py file is to simulate account to calculate the return)

-predictionModel.py (This py file is to apply SVM and ensemble learning to the data)

-SVM.py (This py file is to train and predict data by SVM grid search with 10-fold cross-validation)

-dataProcessing.py (This py file is to process data (data processing))

-featureConstruction.py (This py file is to construct the features needed for the model)

-indicators.py (This py file is to calculate the 7 technical indicators used)

-calSpread.py (This py file is to calculate the spread of a pair)

-pair.py (This py file is to find out the pair of HSI,SSE50 and DJIA using 10-year closing price data)

-fetchData.py (This py file is to fetch the HSI,SSE50 and DJIA components' stocks data from Yahoo Finance)

-data (folder)

-testProgram.py (For testing the program)

3. result folder

-contains the first 3 attempt experiemnts(algorithm) results , and the final (algorithm) result

4. readme.txt

-some explanation of folders and program requirements

8. Future Research and Summary

Future Research:

The designed model makes accuracy improvement mainly based on the ‘spread’ feature. Also, only 7 technical indicators are used.

Therefore, future research can take more technical indicators

into consideration, and for each of the technical indicators, there are some mixed strategies developed from the financial industry, which are some patterns to judge the stocks’ price trend.

In conclusion, we can apply and train an artificial neural network to recognize the patterns stated above to make further improvement of the model as one of the future research’s direction.

Summary:

Financial data prediction is a very attractive and challenging task that many people want to solve for years but is still a mystery(there is no sure winning strategy, at least, published for people). We know that to predict financial data in the future is an extremely complicated task and has a long way to go deeper and deeper.

Through this project, we learnt how to apply machine learning technique on financial data, and also, we learnt the use of pair trading and concept, technical indicators, and some financial market knowledges. Moreover, we learnt some data processing technique when it comes to financial data and feature construction. These all are both invaluable experience for the future research of algorithmic trading related topic.

9. Abbreviations

1. SVM: Support Vector Machine
2. AI: Artificial Intelligence
3. RSI: Relative Strength Index
4. SMA: Simple Moving Average
5. WMA: Weighted Moving Average
6. EWMA: Exponentially Weighted Moving Average
7. MACD: Moving Average Convergence Divergence
8. CCI: Commodity Channel Index
9. ANN: Artificial Neural Network

10. References

1. Malkiel, B. G. (April 2003). The Efficient Market Hypothesis and Its Critics. April 2003: CEPS Working Paper No. 91.

2. The Application of SVM to Algorithmic Trading
   http://cs229.stanford.edu/proj2008/Blokker-TheApplicationofSVMtoAlgorithmicTrading.pdf

3. Introduction to Technical Indicators and Oscillators
   http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:introduction_to_technical_indicators_and_oscillators

4. Dow Jones Industrial Average
   https://en.wikipedia.org/wiki/Dow_Jones_Industrial_Average

5. Hang Seng Index
   https://en.wikipedia.org/wiki/Hang_Seng_Index

6. SSE 50 Index
   https://en.wikipedia.org/wiki/SSE_50_Index

7. Statistical Arbitrage – Trading a cointegrated pair
   http://gekkoquant.com/2013/01/21/statistical-arbitrage-trading-a-cointegrated-pair/

8. Quantitative Investment Using Python
   Chapter 1-12, 22-32

9. Statistical Arbitrage: Algorithmic Trading Insights and Techniques
   Chapter 1-4

10. Python for Finance: Analyze Big Financial Data

    Chapter 1-7

11. Ensemble methods

    http://scikit-learn.org/stable/modules/ensemble.html

12. SVM (Support Vector Machine)—Theory

https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72