Financial Data Mining III

Price movement prediction in Hong Kong equity market

Project Adviser: Dr. CHAN, Lai-Wan Lauren

Report by:

Ying Ting Chung (1008630404)
e-mail: tcying8@cse.cuhk.edu.hk

Department of Computer Science and Engineering,
The Chinese University of Hong Kong

2010 - 2011

Abstract

This project predicts buying signals of Hang Seng Index with data mining. A stop-loss or return margin is embedded in signal definitions. K-nearest neighbour is evaluated by fixed sliding window test. It becomes an effective classifier, using functions of time as input attribute. Prediction accuracy is satisfactory and is compared with that of neural network. Issues regarding actual usage of the trading signals are discussed.

Keyword: financial data mining, financial forecast

Acknowledgment

To my FYP advisor professor Chan I owe a word of thanks. Technical advice she gave is valuable, but not the most valuable thing she has given. Professor Chan constantly reminded us of a mindset of tackling a problem, which is equally important as the problem itself —to focus on one objective at a time, to understand its nature, to place a thesis in the academic context... that pinpoint the problem of us and of our generation. We did not live up to full expectation in the project. But the experience is a seed that will grow into other aspects of life.

Table of Contents

List of figures

Chapter 1 Introduction

1.1 Project Scope

Financial market is the most active market in the world. An often quoted example is the depth of the currency market: average daily turnover in currency market has grown to $3.98 trillion as of April 2010 (BIS Monetary and Economic Department, 2010). In Hong Kong, securities market has the most attention, wherein for the first 10 months of 2010, average daily turnover exceeded 66 billion. It presents enduring opportunities to business and research community.

While there is a variety of market role from issuer to arbitrager, etc., the knowledge of future price of a financial instrument certainly hands all of them the competitive edge. Of course, price forecast is especially valuable to financial traders. In this project, we focus on *price movement prediction in Hong Kong equity market using data mining techniques*. And it is priority that the forecast is in a form usable by traders.

1.2 Background

Price Predictability

Predictability of financial market has been studied and remains controversial. Random walk hypothesis assumes prices are completely stochastic; Efficient Market Hypothesis says that asset price reacts to new information so rapidly that opportunities for systematically profitable trading are eliminated. But financial market is dynamic, complex, and analysing its predictability theoretically is as hard as to actually predict it. In fact, empirical studies of market anomalies exist (Schwert, 2002) and statistical models were made to exploit the market (Pauly & Schell, 1989). In the same spirit, this project takes on an empirical, experimental, and pragmatic approach to financial prediction. Emphasis is put on accuracy of prediction results, and if they are in a form usable by the trading community.

Fundamental Analysis

There are different approaches to financial prediction. Fundamental analysis posits that security price will reflect intrinsic value of the underlying asset, and establishes ways to evaluate it. In the context of equity, investors may look at economic condition, industry landscape, corporate management, financial ratios and so on, to make buy or sell decisions. Such information is not frequently released. This paradigm gives little explanation as for daily fluctuation or sudden swings of security price. So traders cannot optimise short-term positions by this alone.

Technical Analysis

Technical analysts believe that historical performance of stocks indicate their future performance (Investopedia). Usually historical price, volume, or open interest are charted or calculated to some number indicators. Oscillators are indicators used to filter overbought or oversold conditions as defined; Trend lines indicate current price trend and assume little deviation from trend in near future; Chart and candlestick patterns are recurring price patterns believed to indicate trend continuation or reversal; Support-Resistance lines and Fibonacci lines indicate a lower or upper bound for the price. Austere techniques like Elliott Wave Theory and Gann Theory are popular among professional traders but do not attract academic attention. Most of the indicators predict direction of change of price rather than the price itself.

Ensemble Approach

Traders may weight forecast drawn from different approaches. Market sentiment, expert opinion, or even conspiracy theory could too be taken into account. The judgment of how to compromise different forecasts is subjective. It demands expertise during analysis and discipline during trade execution.

1.3 Financial Data Mining

Data mining can be defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. Applied to stock market forecasting, it includes uncovering market trends, planning investment strategies, identifying the best time to purchase the stocks and what stocks to purchase. Methods were explored for financial modelling. To name a few examples, building a data mining model to look for frequent itemsets (candlestick patterns) on stock series (Vásquez, F., Osorio, D., & Losada, 2009); or trying to classify price trends using web news articles (Huang & Li, 2008). More traditional techniques include neural networks, k-nearest neighbours (for outlier detection), decision tree analysis, regression, ARIMA, Bayesian learning, principal component analysis, and genetic algorithm (for attribute selections).

Advantage

It is early to conclude data mining techniques can outperform traditional approaches and human speculation in prediction price movement. Nevertheless market data is growing. An investment bank typically deals with 8000TB of data per day. Data mining allows extraction and analysis of information from huge and multidimensional data set, which is impossible in other approaches. Information technology has also made market activity faster and more volatile than ever. Efficient data mining can provide stronger basis to fast and decisive algorithmic trading. As data mining algorithms become computational feasible on personal computers, plus the growing population of self-directed traders, this new technique also has the prospect to be promoted to public.

Methodology

Most of data mining modelling is based on rigorous mathematical or statistical formulation. When we build such models for real world phenomena, the most ideal condition is on one hand, the phenomena operates as clock-worked as the laws of classical physics; on

the other hand, mathematical constructs and operations are expressive enough to map one-to-one directly to those essential real world entities and mechanisms. But whether current algorithms are sufficient to learn real world entities and mechanisms are dubitable. In a large extent, success of forecasting relies on human ability to compromise: finding an appropriate representation of the problem, and mapping it to the suitable model.

Therefore we can hardly avoid making hypotheses about the way market works in the first step of data mining process. In fact, scientific method is inspirational to data mining process:

1. Hypothesis formation	(Feature extraction and selection)
2.Theoryformulation	(Model selection)
3. Experiment or observation	(Model evaluation and comparison)
4. Technology	(Using discovered knowledge)

Certainly there were experiences in which a wrong hypothesis predicted better than a 'less wrong' hypothesis as measured. Path to discovery is no straight line. Only good hypotheses will be generalisable in the long run but they could be hidden under false discoveries. *Therefore we will not give up a hypothesis or a model only because its performance measure is not the best.* Justification to hypotheses is weighed.

1.4 Problem Formulation

Problem Definition

Investors come into a wide spectrum of trading styles. It is a challenge to support all financial decisions because:

1. Investors trade in local market. Time zone, settlement rules, transaction cost, market structures, laws and risk differ.
2. They trade different types of financial instruments. Pricing mechanisms and factors that affect future price totally differ.
3. Traders specialise in different trading styles and set up different trading rules, for example carry trade, or intraday scalping. Forecast needed by these traders differ in form and content.
4. Market participants have different level of access to market information. They have different capital and position size. Their ability to execute buy or sell according to a trading signal differs.
5. Financial institutions have different risk management policies. Even presented with same information, usage and results would be different.

To delimit a tractable problem, we focus on *price movement prediction in Hong Kong equity market* in this project:

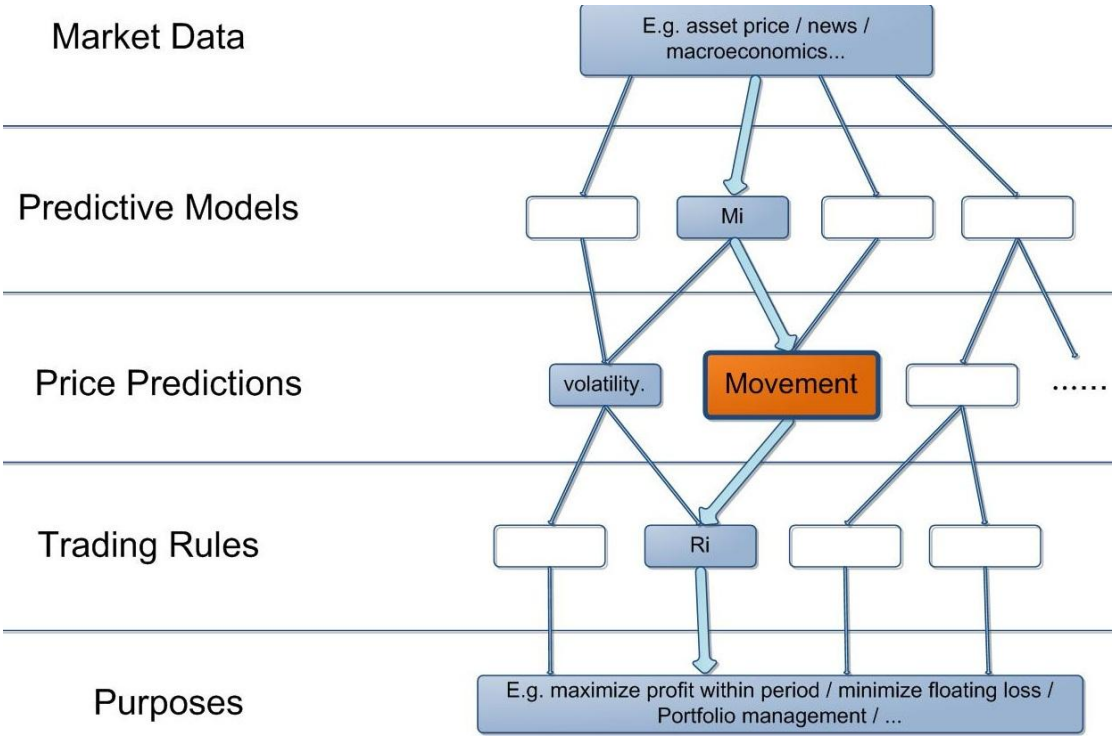


FIGURE 1: PREDICTION PROBLEM FRAMEWORK

With proper risk and money management, even modestly accurately forecasts can produce fair trading profits in different market conditions. Still it is an edge to improve accuracy in the modelling phase. We do not confine to any set of prediction output yet, since each model performs differently for different output sets; similarly we do not confine to any predictive model, as to allow room to discover the best model for a given output set.

Price movement cannot escape the categories of *magnitude, direction, and timeframe*:

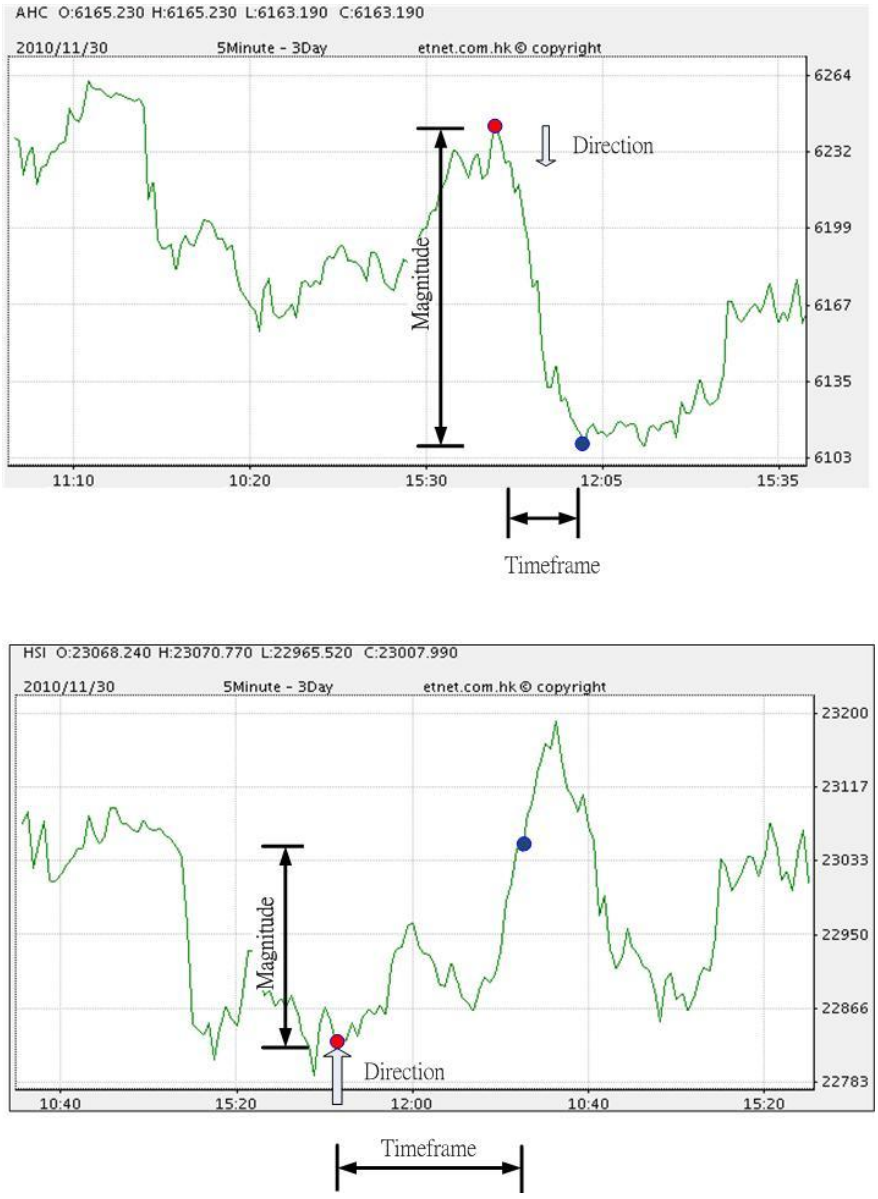


FIGURE 2: PRICE MOVEMENT

Therefore, the prediction problem can be defined as: *given some attribute vectors at time $t \leq i$ and all their associated observed labels, predict the class label for each new attribute vector at time i+1, where the labels are nominal or numeric descriptions of future change of price in terms of magnitude or direction, valid within certain timeframe.*

Problem Analysis

Financial prediction has always been formulated as a time series problem. Time series forecasting models exist, but it is reported that they do not handle non-linear and non-stationary processes as good as some data mining techniques.

Given that price movement is influenced by multitude of factors, if we express class label as a function of past attributes, meanwhile introducing lag variables, then we are already in the curse of dimensionality. Another problem lies in selection of output. Consider the simplest binary classification of daily market {up, down}, there is a prospect to achieve high forecast accuracy. During application, result is susceptible to overtrading and net loss as time and magnitude dimensions of price movement are neglected. Suppose we extract significant local optima as class labels to avoid overtrading. Because of the imbalance number of optima versus non-optima, the classifier may simply learn naive prediction, if performance measure is not well-designed.

1.5 Project Objective

In semester one, we set off to understand the working domain, then to develop prototype data mining process and results. We have taken prediction of Hang Seng Index (HSI) price movement as the first step, for the following reasons:

1. Data for HSI covers more dimension and longer period than ordinary stocks
2. It could be argued that it is price movement of HSI is more predictable because of large number of market participants. Being a spot for global hot money, and the

underlying asset of HSI Futures, Options, CBBCs, where more speculative traders gather, HSI could exhibit more trending and technical properties. At the very least, more hypotheses could be explored due to data availability.

3. Flexible hedging options are possible with HSI, so the set of usable price forecasts is expected to be larger.
4. HSI is an important indicator of market sentiment. Ordinary stocks in Hong Kong equity market have a high correlation to HSI. There is potential that we could re-use HSI forecasts to predict some of other stocks.

Chapter 2 Data Mining Process

This chapter presents the data mining process and results. Steps of data mining are inter-dependent, so I did not actually proceed linearly as laid out.

2.1 Data Collection

In this phase, we search for potentially useful datasets and download them from websites. Reputable source ensures data availability during use phase. Data sources are listed in Appendix A.

We implemented a web crawler as part of our FDMIII Java project. We downloaded historical daily prices of some major world indices, plus those of all listing Hong Kong stocks with stock code less than 4000. There were ETFs coded between 2800 and 3078 and they were separated from equities. Data mainly came from Yahoo! Finance and have the format:

Date	Open	High	Low	Close	Volume	Adj Close
30/9/2010	79.6	79.8	79.2	79.7	31379300	79.62
29/9/2010	81.1	81.4	80.3	80.3	22206100	80.22
28/9/2010	82	82	80.15	80.3	18066800	80.22

Flat files are often in CSV format. One minor issue is that CSV format is not standardized so we have to check if spaces are considered part of a field, or fields are enclosed within double-quotes (Comma-separated values, 2010). After all they are readily imported into a local host MySQL database.

2.2 Data Pre-processing

In this phase we want to increase data reliability and consistency. We avoid using spreadsheet programs since they often alter date and number format automatically. Financial figures are often big, and we shall make sure they are in the right format and do not overflow in the database.

Data Cleansing

Some values are indeed missing in the dataset. For instance daily volume of HSI before 2001 September was not recorded. For consistency, we have to drop all those records. The second issue is that the dataset contains holiday entries where volumes are recorded zero. This could distort the trend say if we do moving average smoothing. Such entries are deleted. Also, days of some Chinese holidays are not recorded in Hong Kong time series, when for US or other market, price is recorded as usual trading days. Our policy is that when join intermarket time series on the fly, we duplicate prices of previous trading day as a dummy record of the missing day. Similarly we index time series on the fly to make sure the time points align.

Corporate actions such as stock splits, ex-rights may cause sudden jumps in price. We would use adjusted price whenever possible. Seven years of Hong Kong stocks data contain about 20 days of half-day trading, which is considered insignificant.

Outlier Treatment

Outlier is a form of price movement where the change is abrupt, large in very short timeframe. It is paradoxical that it could be huge risk or catchable price difference. It is almost hopeless to forecast outliers. For example, it was witnessed that stocks, including blue chips, deviated as much as ten percent during the last ten minute of trading day 28th May, 2008 (Thomson Financial News Limited, 2008). Later it was explained by a change in market

mechanism. But its effect was plain unexpected to most people except few insiders. However, we do not plan to remove such outliers and rather to accept it as limitation of data mining approach.

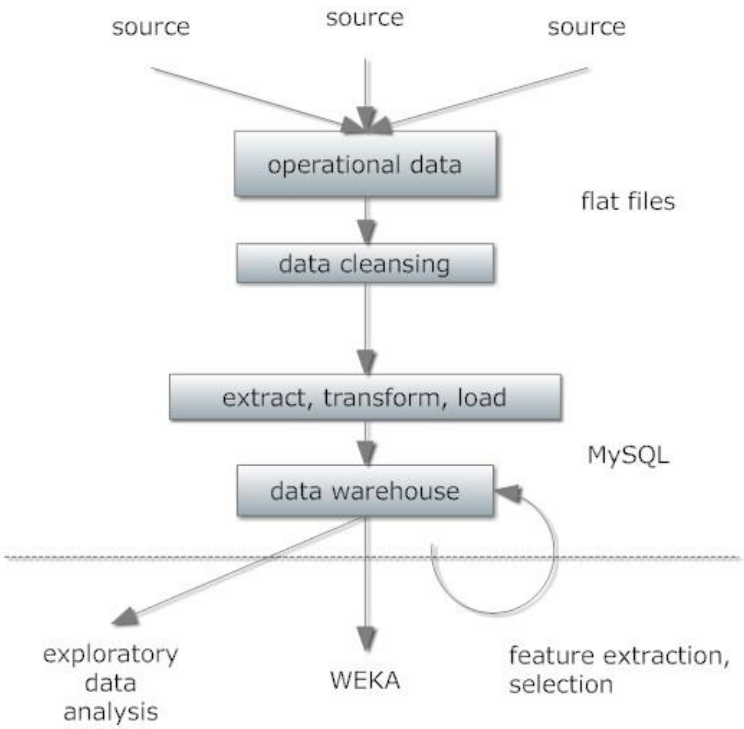


FIGURE 3: DATA PRE-PROCESSING FLOW

2.2.1 Data Integration: FDMIII implementation

We implemented data extraction and transformation in *FDMIII*. Important code snapshot is given in Appendix B, while this section describes its main modules and functions.

fdm.Main class sets some of the parameters for the sub-routines, calls them statically and iterates them in different order. Programmers have a flexible control over the flow of data mining process like that of using script.

fdm.settings.Settings class allows getting parameters from an outside text file. Information such as database password might be separated from the code.

If a data website stores file with an URL template, we learn it and code it into the *fdm.ods.Crawler* class.

fdm.ode.DB class represents a MySQL database. It extends from *weka.experiment.DatabaseUtils*. It supports import of CSV and ARFF files into database, and export of query results. It converts the data between query result set, array, and the internal data structures of WEKA (*weka.core.Instances*). Attribute selection and record subsampling are done through MySQL. Exported CSV files can be imported to Gretl, for example, for statistical and visual data analysis.

fdm.etl.Extraction class is responsible for derivation of new attributes out of existing data.

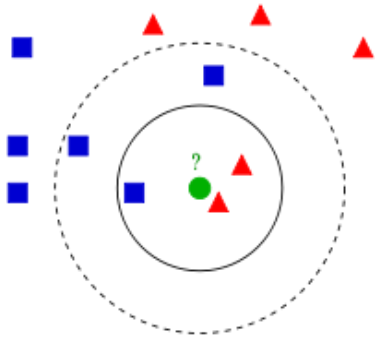
fdm.etl.SlidePlot2D class extends from *weka.gui.visualize.Plot2D*. It queries from the database and plot the distribution of a chosen class over two chosen dimensions (input variable). Assuming the data is a time series, each time the plotter plots a number of N data points, corresponding to a closed interval of N time points in the time series. By holding down “left” or “right” key the plot traverses backward or forward in time. Thus it is used to interactively visualize classification error and the change in class distribution over time.

Fdm.eval.SlideIBk class extends from *weka.classifiers.lazy.IBk*. It implements the sliding window k-NN classifier that is to be described next chapter.

2.3 K-Nearest Neighbour

2.3.1 Theory

K-nearest neighbours algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. It is a type of instance-based learning where function is approximated locally. It is known as an algorithm which is simple, easy to understand, and is suitable for exploring a problem when parametric estimates of probability densities are difficult to determine (Peterson, 2009).



The k nearest neighbouring points will hold majority voting to decide which class the query point (green point above) should belong. Votes can be weighted such that nearer neighbours carry more voting power.

When k-NN is applied to time series problem, its "memory-based reasoning" property is prominent. Suppose we use k-NN to predict certain two-class label for some price movement. And *sliding window validation* is used, looking back 4 days each time:

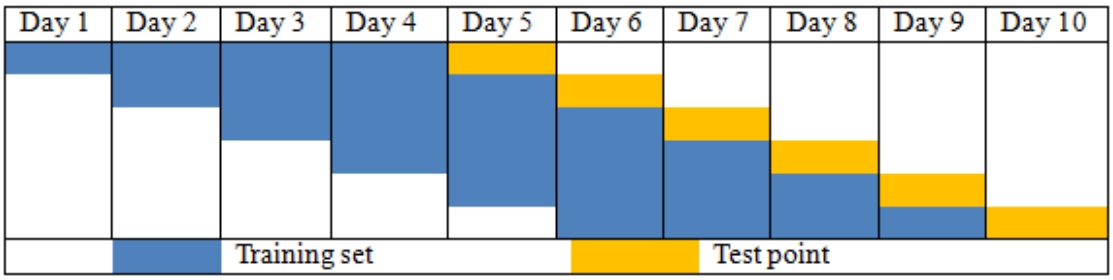
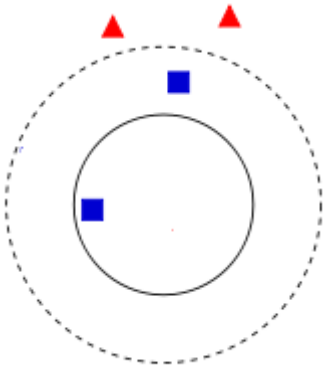
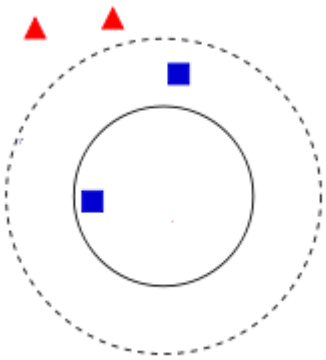


FIGURE 4: SLIDING WINDOW TEST

At day 5, test point is confronted with points from day 1 to day 4:



At day 6, a new test point would be confronted with points from day 2 to day 5:



If we define the all points in the feature space as "memory" of the k-NN model, then as window proceeds through time, some data points are being "forgotten", and memory is revised by the new test point. In contrast with other models, k-NN does not produce a target function in the form $x(t+1) = F(Y_1(t), Y_1(t-1), \dots, Y_2(t), Y_2(t-1), \dots)$. Yet it takes large number of historical data points into account without the need to introduce lag variable, i.e. without explosive growth in number of dimension. Also, the learning process is time-dependent and model changes gradually through time. With suitably selected features, k-NN could become an adaptive model, which fits the fast changing financial market.

2.3.2 Output Selection

To define an output usable by traders, we explore N-bottom in Semester 1:

For each day t, if its daily closing price is lower than the daily low price after N days, then day t is an N-bottom.

To plot 8-day bottom of HSI within the period October 2008 to October 2010, in which red labels are positive 8-day bottoms and blue ones are negatives:

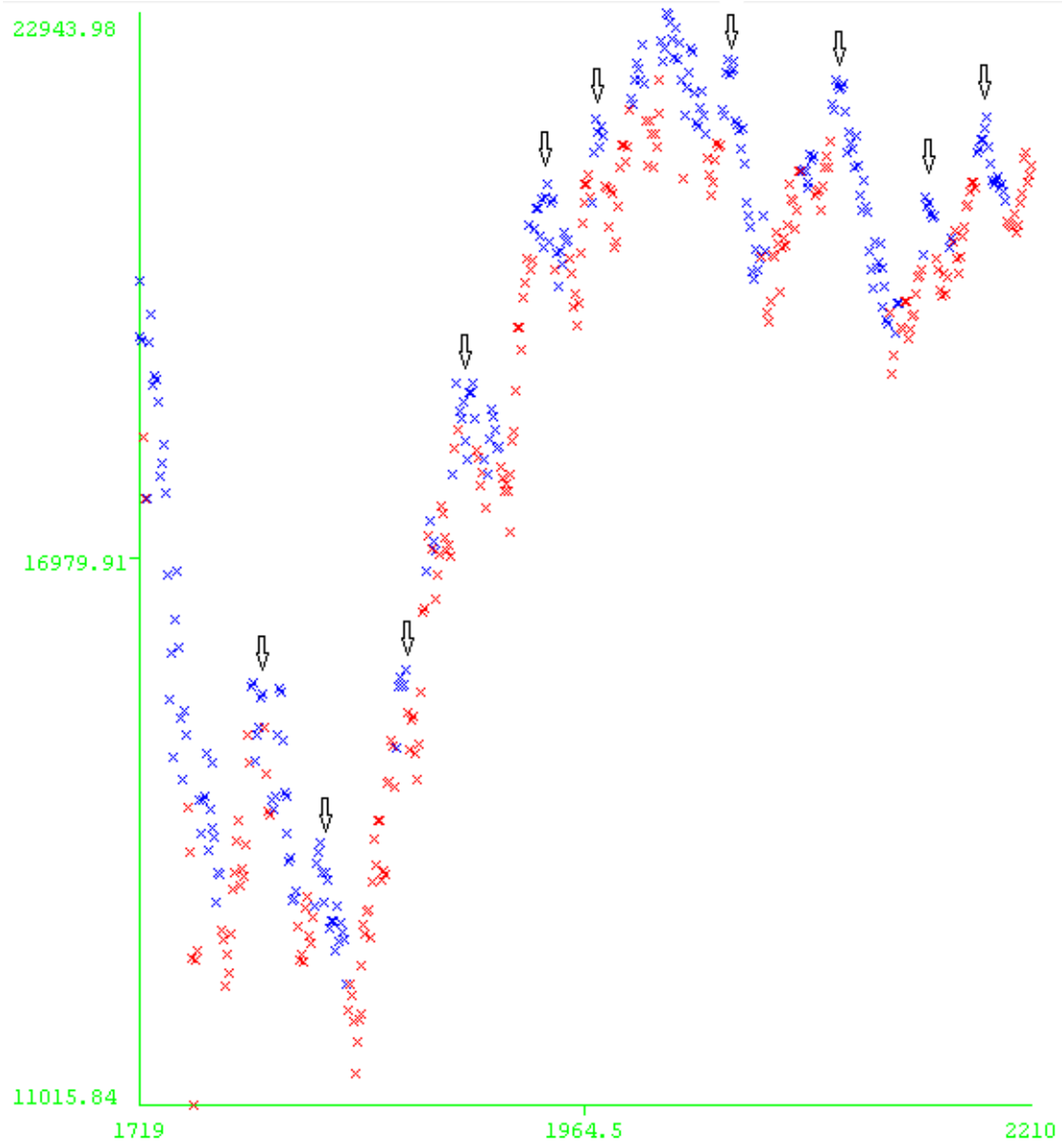


FIGURE 5: HSI 8-DAY BOTTOMS

This definition identifies price bottoms well. More importantly it does not mis-identify tops as bottoms.

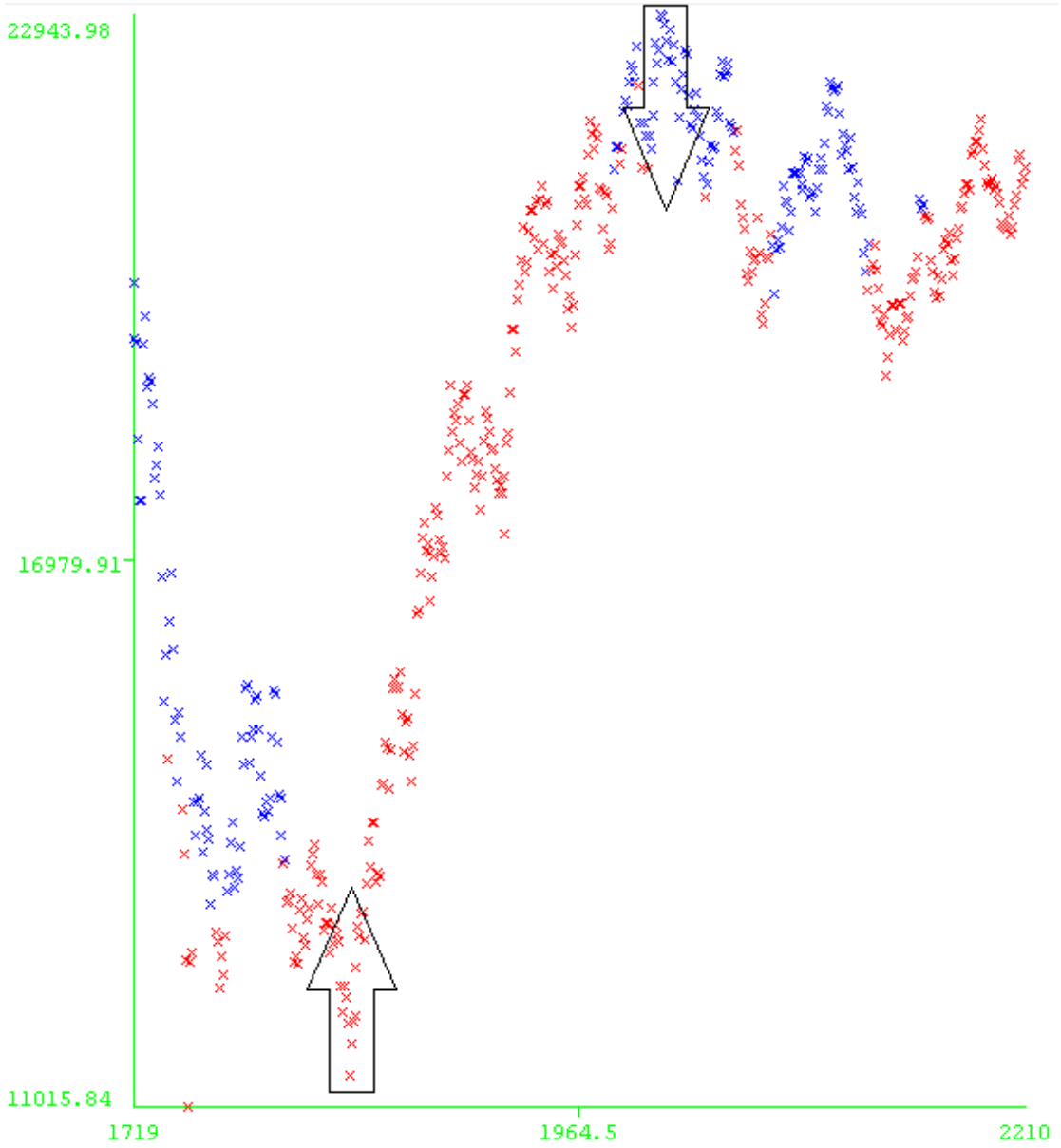


FIGURE 6: HSI 50-DAY BOTTOMS

50-day bottoms identify tops and bottoms in a longer timeframe, and reveal a "fractal" structure of the market where tops and bottoms are defined in different time scales. The N (day) parameter allows modelling of price movement in different timeframe. Generally, a wave defined in longer timeframe is also larger in magnitude.

Consider to actually use the above target labels in trading assets. Suppose the classifier has 100% accuracy. Whenever a positive prediction arrives, we buy and hold the asset. By definition after N days, asset price would be equal to initial bought price in the worst case. This definition has two typical problems as a trading signal. 1. Within those N days asset price could get as low as to liquidate trading position, let alone in reality 2 to 3 successive prediction errors are enough to do great harm to a trader. 2. Correct predictions giving small gains while wrong predictions incurring huge loss.

Stop-N-Bottom

So in Semester 2 we extend and modify the definition of N-day-bottom:

For each time point t, if its closing price is lower than the minimum low price within N time points in the future by m%, then time point t is a stop-N-bottom.

Under this definition, asset price after N time points is guaranteed not lower than current closing price. Also within these N time points, price would not fluctuate below current price. Operationally, consider a trader is working on daily time scale and has set a stop-loss margin of m%. If a positive prediction arrives, he buys and holds the asset. By definition within N days, initial investment will not decrease by more than m%, and N days later asset price would break even at least.

If the asset price goes below the stop-loss margin at some moment, then the trading signal is falsified immediately. The trader could take stop-loss action. If the classifier has a constant false positive rate, coupled with a definite loss percentage each time, it means risk management is possible with trading signal like this.

Although it is named a “bottom”, it is a trading signal backed by trading rules only. It does not necessarily match the local minima that human eye would identify intuitively on price chart. Stop-N-bottom is interpreted as “buying timings” rather than a price bottom.

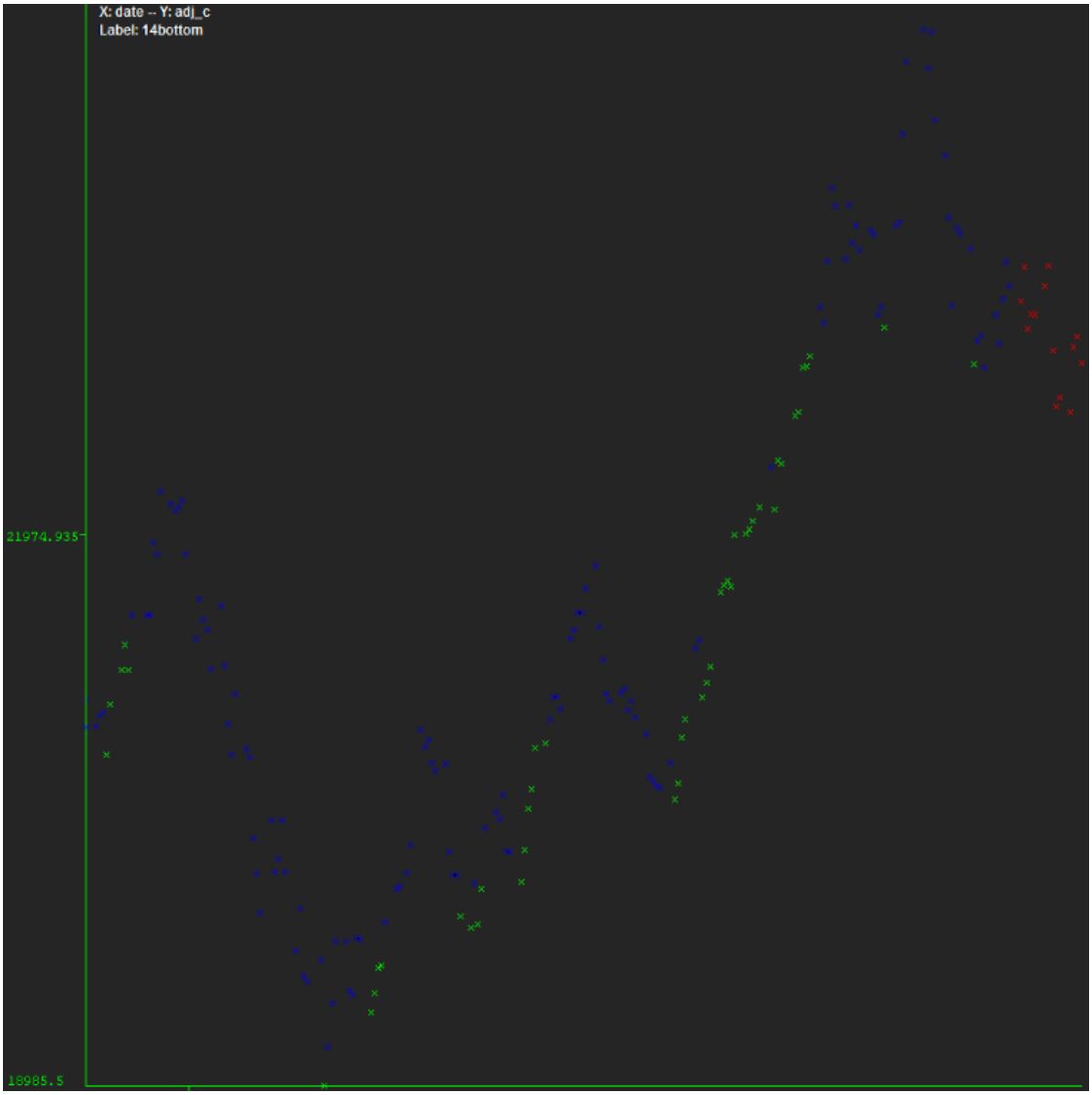


FIGURE 7: STOP-14-BOTTOM, 1% STOP-LOSS MARGIN

The above plot show labels of stop-14-bottom of Hang Seng Index for the second half of year 2010. Green points are positive labels. Blue points are negative. Since the definition of target label of day *t* requires information beyond time t, some points in the historical time series are undefined. For integrity, they are coloured red. But for most classifiers these points could be discarded without much affect.

Hold-N-Bottom

In Semester 2 we also consider another target definition:

For each time point t, if its closing price is lower than the low price by m% after N time points, then time point t is a hold-N-bottom.

Under this definition, asset price after N time points is guaranteed to be higher than current closing by r%, where *r* is a user-defined profit margin. Operationally, consider a trader is working on daily time scale and has set a profit margin of r%. If a positive prediction arrives, he buys and holds the asset. By definition after N days, initial investment will at least has a return of *r*%.

In principle, this definition would avoid catching "top of the wave" to be "bottoms". Figure 8**: HOLD-10-BOTTOM, 3% PROFIT MARGIN** is a graphical example of this target definition.

Stop-14-Bottom

In the improved definitions there are fewer positive signals. This matches the intuition that buying opportunities does not occur often for any asset. This avoids the overtrading problem at the level of label definition, because the definition already embeds an entry/exit strategy.

Since the new definitions extend the definition of N-bottoms, they are subsets of the original labels. Trading signal is more useful as we impose more conditions, but resulting class would be too imbalanced. Mining rare cases is a highly challenging task.

We visualize and compare targets of different parameters on the Hang Seng Index series. We make a trade-off between usefulness and frequency of positive labels. Stop-14-bottom with 1% stop-loss margin is chosen to be the target definition (Figure 7: **STOP-14-BOTTOM, 1% STOP-LOSS MARGIN.**

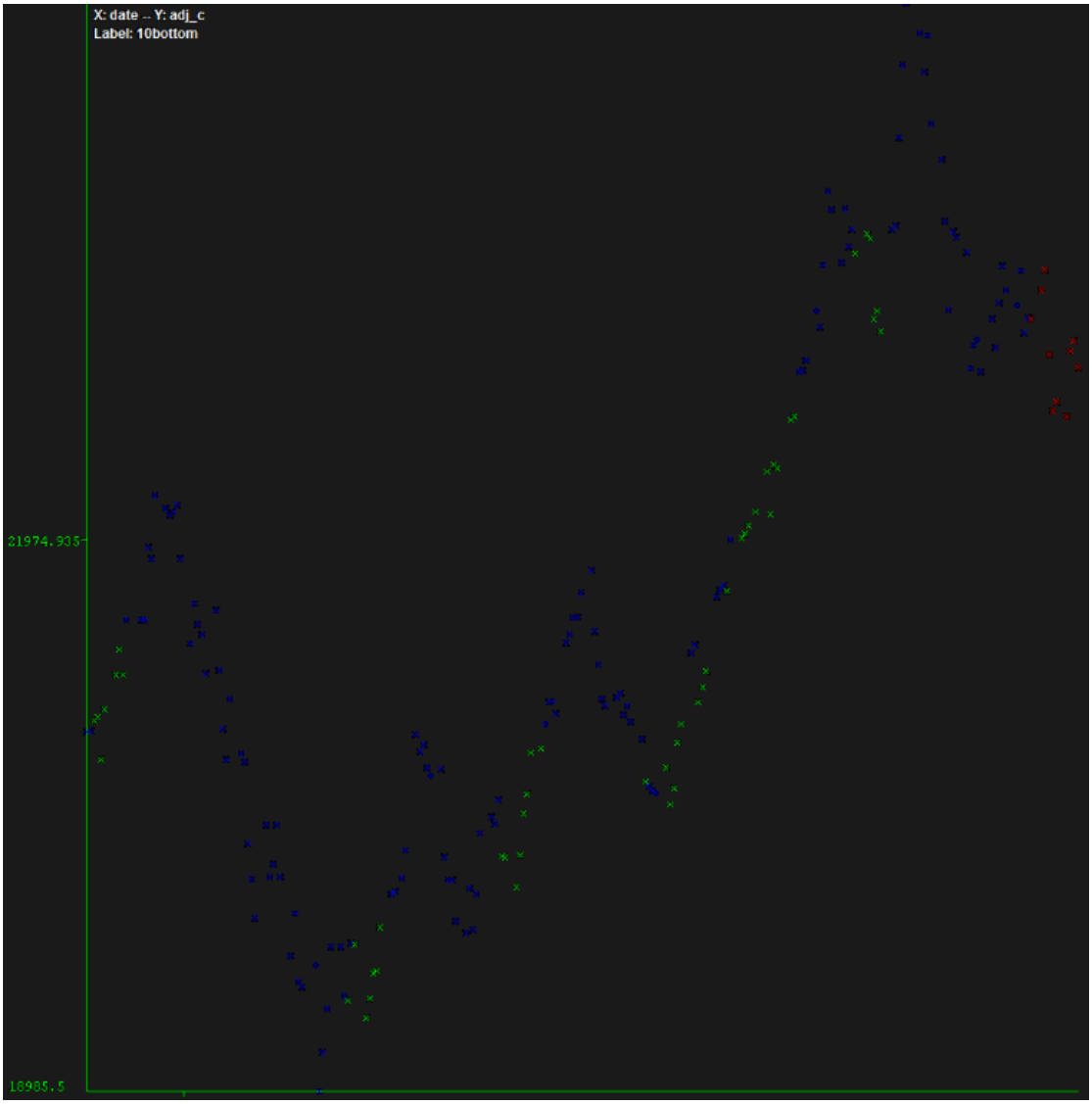


FIGURE 8: HOLD-10-BOTTOM, 3% PROFIT MARGIN

2.3.3 Input Selection

The choice of input variable is limited by practical issues. If a trader uses a classifier to speculate future price movement, but the classifier needs information not available until the end of trading hours, such as closing price of current day. In this case, trader would have to estimate the information before market actually closes, derive the indicators and evaluate the model in real time. If the model is very sensitive to input values, the prediction could deviate a lot. Buy and sell decisions have to defer till end of trading hours, when market is illiquid and sometimes volatile. So execution price could deviate from target and order size is limited.

2.3.4 Sliding Window Test

Sliding window test is a common backtesting method of trading strategy in finance industry. In complex environment like the market we do not expect variables to simply follow an i.i.d. probability distribution. For example there are periods when market is more volatile and other times when market is quiet. Market could also be dominated by hot issues at the time, i.e. dependency changes. Below is a plot of HSI against two technical indicators. Positive stop-14-bottom labels are shown in red, negative ones in blue. Local distribution of red points is "shifting" from centre to top right.

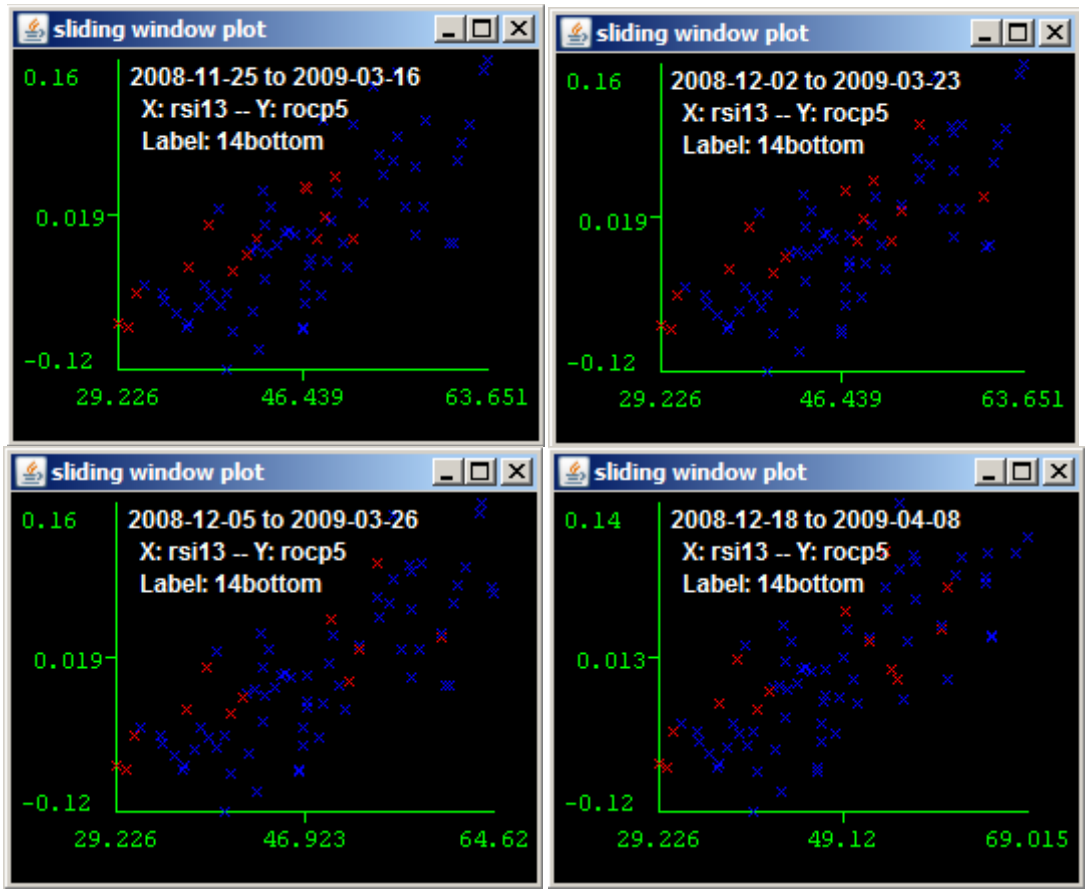


FIGURE 9: DATASET SHIFT

2.3.5 Results and Discussion

As a tactic to search for explanatory variables, we first try to classify daily HSI 5-day bottoms from 1st January 2006 to 23rd December 2010. Simple majority voting of k-NN and fixed sliding window test is used. We measure precision and recall with respective to the positive class. After trials we settle with following attributes:

Slope of SMA(22) and EMA(8, 13, 18)
Price Deviation from SMA(9, 22) and EMA(8, 16)
Percentage aggregate return for last 5 and 6 days
Daily gap open

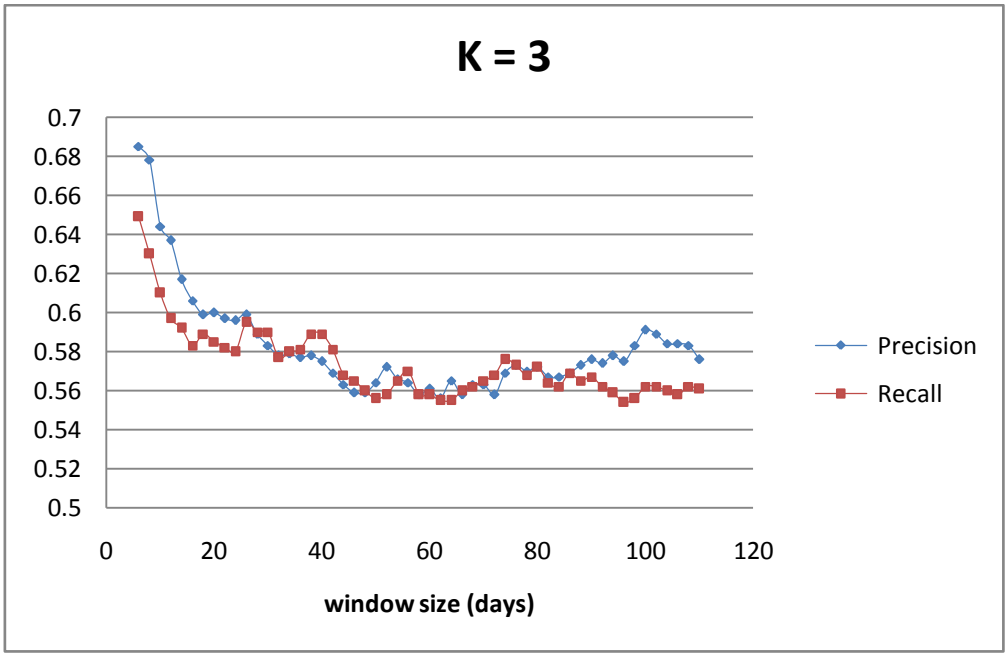


FIGURE 10: EFFECT OF CHANGING WINDOW SIZE

As the model look back for more days, performance deteriorates. For a window size of 240, precision and recall decrease down to 0.529, reflecting that classification is only interfered by more noise. On the other hand, the 5-bottom label tends to occur continuously in

the price series. It is more so in a rising trend. Therefore for extreme case as a window size of 6 days, the neighbours will easily all be the same positive bottoms. Positive labels would win the majority voting. Forecast for the next day would be positive as well irrespective of any input variable. It also shows that the model fails to learn from history as it takes only 6 days into account.

FIGURE 11: EFFECT OF CHANGING K

The case of 6 day window size is reminiscent of a k-NN model that only has a memory of 6 days, and is ignorant of any history beyond memory.

We could extend an extra dimension in the feature space. The new axis is an exponential function of the time index of HSI price series. Result is that for current day *t*, recent data points are closer along that axis. Points back in time are exponential getting further away. It emulates a k-NN model that has very good short-term memory, but long-term memory decays. Including time index (or its derivative) is an uncommon practice. It hints asset price is dependent on time in some way, which is an uncommon hypothesis. We still experiment with it up to window size of 540 days:

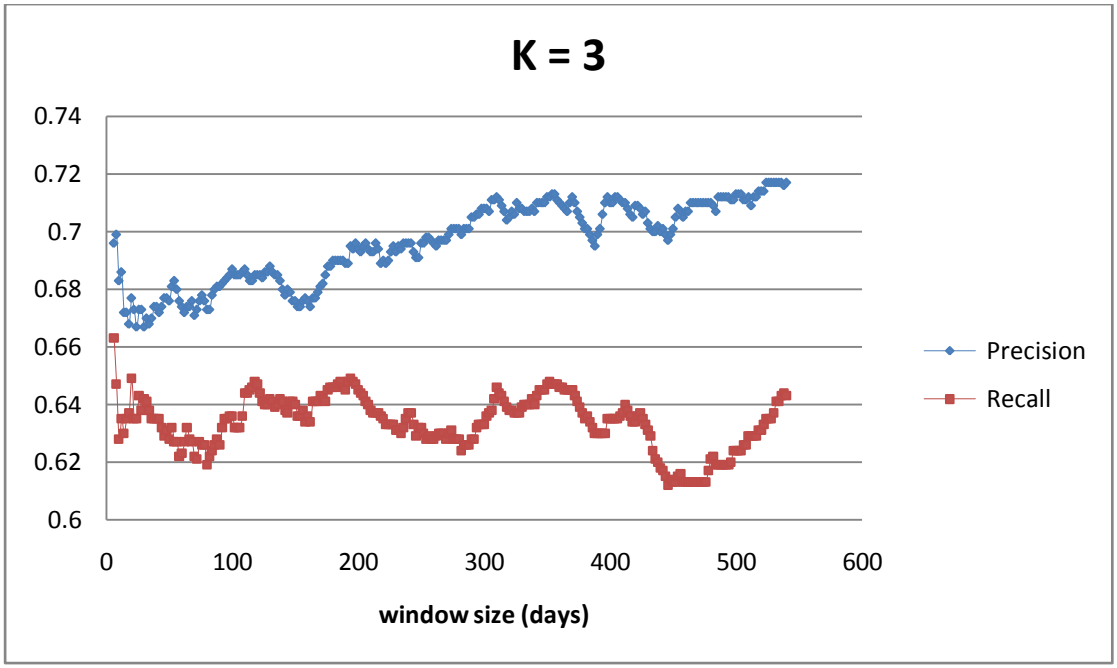


FIGURE 12: EMULATING K-NN MEMORY DECAY

We could apply several sine functions of different periodicity to the time index as well. In effect, data points of same periodicity are pulled closer together in the feature space. This is to emulate the cyclic nature of financial market.

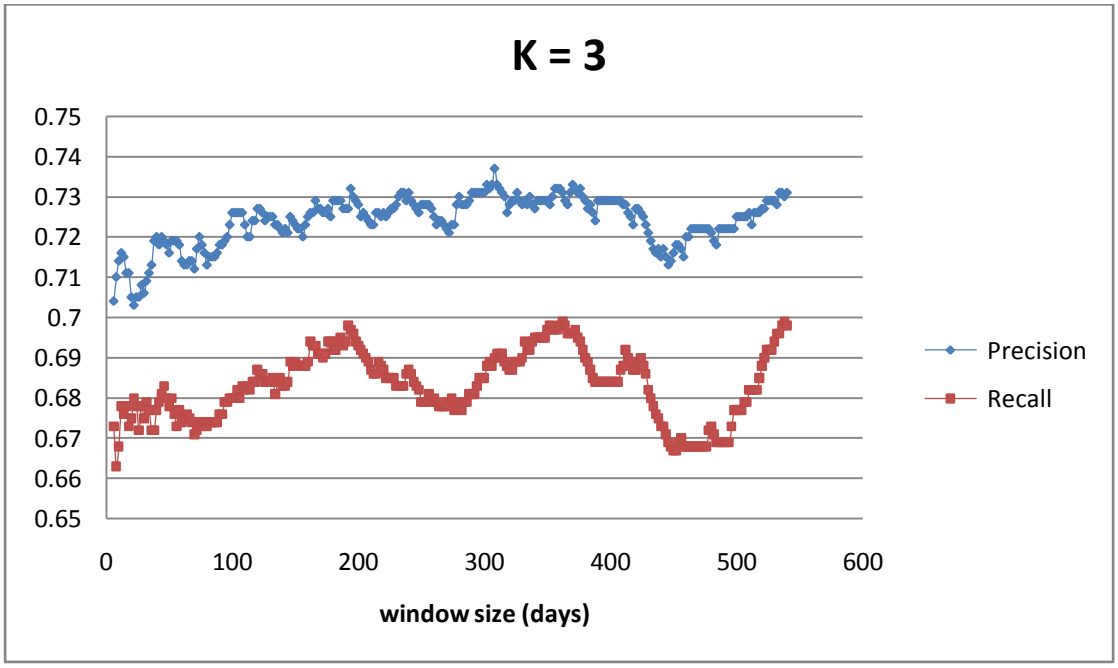


FIGURE 13: EMULATING MARKET CYCLES

For window size of 540, using 3 neighbours, above experiment presents a precision of 0.731 and a recall of 0.698. Confusion matrix is as follow, having error rate of 0.24:

```
=== Confusion Matrix ===

   a   b   c     actual class
 324   0  77 |   a = -1
   1   2   2 |   b = 0
  93   0 215 |   c = 1
```

Letter c stands for positive 5-day bottom, letter a are negative ones. Class b includes undefined targets but the number is insignificant.

All experiments above are using 5-day bottom, which is a less challenging prediction target. We replicate the previous experiment with target 14-days bottom of stop-loss margin of 1%. More satisfactory results occur for k = 3 and window size = 120 days. Precision for this setup is 0.601 while recall is 0.476. Confusion matrix is as follow, having error rate of 0.196:

```
=== Confusion Matrix ===

   a   b   c     actual class
 791   0  79 |   a = -1
   2  12   0 |   b = 0
 131   0 119 |   c = 1
```

Although error rate is more or less identical, false positive rate is much higher. It means traders will suffer from more false entry and loss will accumulate.

2.3.6 Future Work

In constructing an adaptive k-NN classifier, change in variable dependency over time should be considered too. Input variables contributing to predictive accuracy could become irrelevant as time traverse. Various methods exist addressing this problem. For example adopting a weighted Euclidean distance is equivalent to using weighted attributes. A learning feedback mechanism is desirable. During sliding window test, prediction errors can be immediately summarised. Importance of individual attributes is adjusted accordingly as time traverse. In the paper *Discriminant Adaptive Nearest Neighbor Classification* method for global dimension reduction, that combines local dimension information, was proposed (Hastie & Tishirani, 1996a). However such scheme is not easy to implement.

2.4 Multilayer Perceptron

2.4.1 Theory

Due to presence of noise in market data and a weak understanding of market mechanism, we turn to experimenting multilayer perceptron (MLP). MLP is often used to discover non-linear relationship. It is a canonical neural network model consists of a multi-layer, feedforward architecture. (A. & M.).

2.4.2 Experiments

As a trial, we replicate the attribute set in Chapter 2.3.5, using stop-14-days bottom with 1% stop-loss as well. Attributes of day *t* are used to predict target signal at day *t + 1*. The same asset and period are used for training and validation. Network structure is fixed as a three-layer feedforward network. Backpropagation training algorithm is used.

We use the WEKA UI to do the trials. The time series is split into two halves. First half is used to train the model and remaining data is used for validation. Before split the ratio of positive to negative labels is 1:3.35. After split, the same ratio is 1:3.09 is the validation set. Market condition is not biased after dataset split. Holding other parameters constant at default values, we vary the number of epochs and learning rate. Results are presented in Appendix A. It reaches bottleneck at learning rate of 0.3 and training iteration of 10000 times:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 401   0  67 |   a = -1
   0   0   0 |   b = 0
 106   0  46 |   c = 1
```

The dataset is also tested with split test of different ratio. It shows that 50%-50% split produces more generalised model. Extending training period back into year 2006 for example, does not improve the result either.

Chapter 3 Contribution of Work

Winter Semester

This chapter focuses on my involvement in the project. Being a complete beginner in data mining in winter semester, I took the course *Techniques of Data Mining* and spent a month or two to familiarise with the subject. At the same time, I was reading some survey paper but could not decide the predictive target and classifier to use. So I started to implement the web crawler, pre-processed the data, learnt to code with external technical analysis library, and learnt the interface of WEKA first.

After one to two meetings with advisor, we decided to take k-Nearest Neighbour as the first step. However my laptop crashed a week before FYP report deadline. Schedule of the whole project hampered.

Experiment results were preliminary. I was responsible for the textual part of the report in Semester 1 (except cover page and references).

Spring Semester

During vacation I started to revise the implementation in order to automate experimental trials. Some time was devoted to integrate MySQL with WEKA API. WEKA API was more complicated of me to learn than expected. Professor Chan noticed the problem and advised us to learn MATLAB instead. But I was attached to the former implementations and regretfully did not take the advice. The *FDMIII* implementation took its current shape at the end of February. After that I also developed an error visualiser. Then weakness of the k-Nearest Neighbour approach quickly emerged after I conducted a number of trials.

The transfer to neural network needed a deep understanding and technical knowledge, of which I had not got enough. Certainly the project has room to further develop.

Appendix A Neural Network trials

Confusion matrices are in format: $\begin{bmatrix} true\ negative & false\ positive \\ false\ negative & true\ positive \end{bmatrix}$

Learning rate increases in rows. Number of iterations increases in columns.

	500	5000	10000	20000	30000
0.25	$\begin{bmatrix} 456 & 12 \\ 151 & 1 \end{bmatrix}$	$\begin{bmatrix} 402 & 66 \\ 109 & 43 \end{bmatrix}$	$\begin{bmatrix} 400 & 68 \\ 107 & 45 \end{bmatrix}$	$\begin{bmatrix} 400 & 68 \\ 110 & 42 \end{bmatrix}$	$\begin{bmatrix} 405 & 63 \\ 113 & 39 \end{bmatrix}$
0.3	$\begin{bmatrix} 445 & 23 \\ 148 & 4 \end{bmatrix}$	$\begin{bmatrix} 402 & 66 \\ 106 & 46 \end{bmatrix}$	$\begin{bmatrix} 401 & 67 \\ 106 & 46 \end{bmatrix}$	$\begin{bmatrix} 399 & 76 \\ 106 & 46 \end{bmatrix}$	$\begin{bmatrix} 392 & 76 \\ 106 & 46 \end{bmatrix}$
0.35	$\begin{bmatrix} 435 & 33 \\ 145 & 7 \end{bmatrix}$	$\begin{bmatrix} 393 & 75 \\ 109 & 43 \end{bmatrix}$	$\begin{bmatrix} 402 & 66 \\ 114 & 38 \end{bmatrix}$	$\begin{bmatrix} 402 & 66 \\ 113 & 39 \end{bmatrix}$	$\begin{bmatrix} 404 & 64 \\ 115 & 37 \end{bmatrix}$
0.5	$\begin{bmatrix} 393 & 73 \\ 129 & 23 \end{bmatrix}$	$\begin{bmatrix} 351 & 117 \\ 91 & 61 \end{bmatrix}$	$\begin{bmatrix} 349 & 119 \\ 91 & 61 \end{bmatrix}$	$\begin{bmatrix} 356 & 112 \\ 92 & 60 \end{bmatrix}$	$\begin{bmatrix} 306 & 108 \\ 94 & 58 \end{bmatrix}$

Learning rate fixed at 0.3, number of iterations 10000. Split ratio vary.

Split	Confusion matrix	Precision	False positive rate
40-60	$\begin{bmatrix} 487 & 82 \\ 138 & 37 \end{bmatrix}$	0.311	0.144
50-50	$\begin{bmatrix} 401 & 67 \\ 106 & 46 \end{bmatrix}$	0.407	0.143
60-40	$\begin{bmatrix} 308 & 63 \\ 88 & 40 \end{bmatrix}$	0.388	0.17
66-33	$\begin{bmatrix} 247 & 68 \\ 64 & 43 \end{bmatrix}$	0.387	0.216
70-30	$\begin{bmatrix} 267 & 16 \\ 80 & 9 \end{bmatrix}$	0.36	0.057
75-25	$\begin{bmatrix} 220 & 17 \\ 59 & 14 \end{bmatrix}$	0.452	0.072

Appendix B Source of Data

List of Stock Codes, HKEx
http://www.hkex.com.hk/eng/market/sec_tradinfo/stockcode/eisdeqty.htm

Historical Stock Price, Yahoo! Finance
http://hk.finance.yahoo.com/q/hp?s=0001.HK

List of Securities Market Makers and Listed ETFs, HKEx
http://www.hkex.com.hk/eng/prod/secprod/etf/SMM_List.htm

Trading Calendar and Trading Hours, HKEx
http://www.hkex.com.hk/eng/market/sec_tradinfo/tradcal/tradcal.htm

Daily trading value, volume and number of deals, Securities Statistics Archive, HKEx
http://www.hkex.com.hk/eng/stat/smstat/statarch/statarchive.htm

Historical US Dollar / Hong Kong Dollar (USD/HKD)
http://www.mataf.net/en/histo/USD-HKD

Free Historical Trade Data
http://www.tradingblox.com/tradingblox/free-historical-data.htm

Designated Securities Eligible for Short Selling, HKEx
http://www.hkex.com.hk/eng/market/sec_tradinfo/stkcdorder.htm

Company/Securities Profile, HKEx
http://www.hkex.com.hk/eng/invest/company/profilemenu_page_e.asp

Appendix C Code Snippet of FDMIII

FDMIII is a Java project that sorts to integrate the data mining process. Mainly the convenience comes with combining MySQL, WEKA API, file I/O, and all sorts of data structure conversion. Whole project consists of 9 Java classes and 4 of them are shown below. Part of the codes is omitted but work flow within this project remains.

Main

```
package fdm;
/* import declarations */

/**
 * @description  Batch routines in data mining process
 * @author       Ying Ting Chung
 */
public class Main {
  /* FileName = TableName + ".csv" */
  private static DB db = null;
  private static WekaFilter f = null;
  /**
   * Routine. Download data from various sites. Save data as flat files in the directory of
ODS.
   * Convert files to CSV format preferably.
   */
  public static void collect()   {
    String odsDir = Settings.getProperty("ODS.dir");
    Crawler c = new Crawler();
    c.YahooBatchDownload(odsDir + "temp\\", odsDir + "Listed ETFs.csv");
  }

  /**
   * Routine. Connect to MySQL. Define correct table schema according to data file.
   * Batch import from CSV to MySQL table.
   * @throws Exception
   */
  public static void CSVtoMySQL() throws Exception       {
    db.importYahooCSV(Settings.getProperty("ODS.dir") + "quote\\index\\", "_HSI.csv");
    db.importDealCSV(Settings.getProperty("ODS.dir"), "hkmkt.csv");
  }

  /**
   * Routine. Connect to DBMS. Batch import from ARFF to DBMS relation.
   * @throws Exception
   */
  public static void ARFFtoDB() throws Exception {
    db.importARFF("hsi_nstopBottoms", Settings.getProperty("DW.dir"), "hsi_nstopBottoms.arff");
    db.importARFF("hsi_nstopTops", Settings.getProperty("DW.dir"), "hsi_nstopTops.arff");
  }

  /**
   * Routine. Extract, transform, and load.
   * SQL query to join/drop columns and rows. Feature extraction and selection, record sub-
sampling.
   * @throws Exception
   */
  public static void etl() throws Exception      {
    Extraction extract = new Extraction();
    Instances data;
    data = db.exportResultSet("SELECT date, adj_c, l FROM _hsi ORDER BY date");

    data = extract.stopBottoms(data, 1, 2, 2);
    data.deleteAttributeAt(1);
    data.deleteAttributeAt(1);

    System.out.println(data.toSummaryString());
```

```
  db.importInstances("hsi_stop2bottom", data);
}

/**
 * Routine. Exploratory data analysis.
 * @throws Exception
 */
public static void eda() throws Exception      {
  Instances data = null;

  data = db.exportResultSet("SELECT * FROM" +
      " (SELECT date, adj_c FROM _hsi) A" +
      " NATURAL JOIN (SELECT date, 14bottom FROM hsi_stop1bottom) Z" +
      " WHERE date > '2006-01-01'" +
      " ORDER BY date");

  // data = f.fNumericToNominal("last", data);
  System.out.println(data.toSummaryString());

  SlidePlot2D plotter = new SlidePlot2D(data);
  plotter.setY(1);
  plotter.setX(0);
  plotter.setC(2);
  plotter.setTime(650);
  plotter.setWindowSize(350);
  plotter.updatePlot();

  JFrame frame = new JFrame("sliding window plot");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.getContentPane().setLayout(new BorderLayout());
  frame.setSize(new Dimension(1024, 800));
  frame.setLocationRelativeTo(null);
  frame.addKeyListener(plotter);
  frame.getContentPane().add(plotter);
  frame.setVisible(true);
}

/**
 * Routine. k-NN Experiment.
 * @param windowSize - look-back period
 * @param horizonSize - look-ahead period
 * @throws Exception
 */
public static void kNNexperiment(String param) throws Exception {
  StringBuilder sb = new StringBuilder();
  File log = new File(Settings.getProperty("log.dir") + "tmp.txt");
  File csvlog = new File(Settings.getProperty("log.dir") + "tmp.csv");
  BufferedWriter bw = new BufferedWriter(new FileWriter(log));
  BufferedWriter csvbw = new BufferedWriter(new FileWriter(csvlog));

  DateFormat dateFormatter = new SimpleDateFormat("MMddHHmmss");

  TwoClassStats stat;

  Instances masterData = db.exportResultSet("SELECT * FROM" +
      " (SELECT date, adj_c FROM _hsi) A" +
      " NATURAL JOIN (SELECT date, ema8slope, ema13slope, ema18slope FROM hsi_emaslope) C" +
      " NATURAL JOIN (SELECT date, sma22slope FROM hsi_smaslope) H" +
      " NATURAL JOIN (SELECT date, rocp5, rocp6 FROM hsi_rocp) D" +
      " NATURAL JOIN (SELECT date, devema8, devema16 FROM hsi_ldevema) E" +
      " NATURAL JOIN (SELECT date, devsma9, devsma22 FROM hsi_ldevsma) F" +
      " NATURAL JOIN (SELECT date, devsma10 FROM hsi_hdevsma) G" +
      " NATURAL JOIN (SELECT date, gap FROM hsi_gap) I" +
      " NATURAL JOIN (SELECT date, 14bottom FROM hsi_stop1bottom) Z" +
      " WHERE date > '2006-01-01'" +
      " ORDER BY date");

  Instances inputData = new Instances(masterData);
  if (inputData.classIndex() == -1)    {
      inputData.setClassIndex(inputData.numAttributes() - 1);
  }
  inputData.deleteAttributeAt(0);      // delete date attribute
  inputData.deleteAttributeAt(0);      // delete price attribute
  inputData = f.fAddID("1", "t", inputData);
```

```
    inputData = f.fAddExpression("sin(a1/6)", inputData);
    inputData = f.fAddExpression("sin(a1/4)", inputData);
    inputData = f.fAddExpression("exp(a1/20)", inputData);
    inputData.deleteAttributeAt(0); // delete time index
    inputData = f.fNumericToNominal(Integer.toString(inputData.classIndex()+1), inputData);
    System.out.println(inputData.toSummaryString());
    bw.write(inputData.toSummaryString());
    bw.newLine();

    /* Precision: measure of risk
     * Recall:    measure of opportunity */
    String header = "K, WindowSize, Precision, Recall, F1";
    System.out.println(header);
    bw.write(header);
    bw.newLine();
    csvbw.write(header);
    csvbw.newLine();

    SlideIBk kNN = new SlideIBk(inputData);
    // for (int k = 5; k <= 21; k=k+2)   {
    for (int k = 3; k <= 3; k=k+2)       {
    // for(int windowSize = 6; windowSize <= 110 ; windowSize=windowSize+2)       {
        for(int windowSize = 45; windowSize <= 45; windowSize=windowSize+1)       {
        if (windowSize < k)      continue;
        sb.delete(0, sb.length());
      //kNN.setOptions(weka.core.Utils.splitOptions("-I"));
        kNN.setKNN(k);
        kNN.setWindowSize(windowSize);
        kNN.setHorizonSize(1);
        kNN.buildClassifer();

        stat = kNN.getConfusionMatrix().getTwoClassStats(2);

        sb.append(kNN.getKNN()).append(", ").append(kNN.getWindowSize());
        sb.append(", ").append(String.format("%.3f", stat.getPrecision()));
        sb.append(", ").append(String.format("%.3f", stat.getRecall()));
        sb.append(", ").append(String.format("%.3f", stat.getFMeasure()));
        System.out.println(sb.toString());
        bw.write(sb.toString());
        bw.newLine();
        csvbw.write(sb.toString());
        csvbw.newLine();
        }
    }
    bw.close();
    csvbw.close();

    String newPath = Settings.getProperty("log.dir") + "e" + dateFormatter.format(new Date());
    log.renameTo(new File(newPath + ".txt"));
    csvlog.renameTo(new File(newPath + ".csv"));
    csvlog = new File(newPath + ".csv");
    // 'precision' is a reserved word in MySQL
    // db.importCSV("k INT,windowSize INT,PPV REAL,recall REAL,F1 REAL",
csvlog.getParentFile().getCanonicalPath() + "\\", csvlog.getName());

    Instances rst = Instances.mergeInstances(f.fNumericToNominal("last", masterData),
kNN.getPredictions());
    System.out.println(rst.toSummaryString());
    SlidePlot2D plotter = new SlidePlot2D(rst);
    plotter.setY(1);
    plotter.setX(0);
    plotter.setC(14);
    plotter.setTime(650);
    plotter.setWindowSize(200);
    plotter.updatePlot();

    JFrame frame = new JFrame("sliding window experiment");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(new BorderLayout());
    frame.setSize(new Dimension(800, 600));
    frame.setLocationRelativeTo(null);
    frame.addKeyListener(plotter);
    frame.getContentPane().add(plotter);
    frame.setVisible(true);
```

```
  }

  /* Workflow:
   * 1. collect()
   * 2. Identify inconsistency, duplication in data; handle missing values
   * 3. CSVtoMySQL() / ARFFtoDB()
   * 4. etl()
   * 5. eda()
   * 6. experiment()
   */
  /**
   * Organise and execute data mining routines.
   * @param args
   */
  public static void main(String[] args) {
    try {
      f = new WekaFilter();
      db = new DB();
      // collect();
      // CSVtoMySQL();
      // ARFFtoDB();
      // etl();
      // eda();
      for (int i = 3; i < 4; i++) {
        kNNexperiment(Integer.toString(i));
      }
      db.disconnectFromDatabase();
    } catch (Exception e) {
          e.printStackTrace();
    }
  }
}
```

Extraction

```
package fdm.etl;
import com.tictactec.ta.lib.*;
import weka.core.*;

/**
 * @description Extract new attribute from existing ones
 * @author      Ying Ting Chung
 */
public class Extraction {

  /**
   * Convert double array to WEKA instances
   *
   * @param attName - attribute name
   * @param col - double array
   * @return single column {@code weka.core.Instance}
   */
  public Instances doubleArrayToAttribute(String attName, double[] col) {
    FastVector atts = new FastVector();
    atts.addElement(new Attribute(attName));
    double[] vals;
    Instances newCol = new Instances(attName, atts, 0);
    for(int i = 0; i < col.length; i++) {
      vals = new double[1];
      vals[0] = col[i];
      newCol.add(new Instance(1, vals));
    }
    return newCol;
  }

  /**
   * After N time points, defined price arrives above current price by certain percentage.
   * Positive label represented by 1.0.
   * Profit taking is safe for long and hold at the end of look ahead period, if low price as
the future series.
   *
   * @param curPrice - asset price measure at current time point
   * @param futurePrice - asset price measure at {@code horizonSize} later
   * @param horizonSize - number of time points look ahead
   * @param ror - rate of return desired
   * @return bottoms -
   */
  private double[] bottom(double[] curPrice, double[] futurePrice, int horizonSize, double ror)
{
    int timePointCnt = curPrice.length;
    double[] bottoms = new double[timePointCnt];

    for(int t = 0; t < timePointCnt; t++){
      try {
        if(curPrice[t] * (1+ror) < (futurePrice[t+horizonSize]))
          bottoms[t] = 1;
        else
          bottoms[t] = -1;
      } catch (ArrayIndexOutOfBoundsException e)  {
        bottoms[t] = 0;
      }
    }
    return bottoms;
  }

  /**
   * Within N time points, defined future price does not shake lower than current price.
   * Positive class represented by 1.0. Profit taking possible for long and hold within look
ahead period
   * Negative class represented by -1.0.
   *
   * @param curPrice - asset price measure at current time point
   * @param futurePrice - asset price measure at {@code horizonSize} later
   * @param horizonSize - number of time points look ahead
   * @param stopMargin - stop loss in percentage of current price
```

```
   * @return bottoms -
   */
  private double[] nBottom(double[] curPrice, double[] futurePrice, int horizonSize, double
stopMargin) {
    int timePointCnt = curPrice.length;
    double[] bottoms = new double[timePointCnt];

    for(int t = 0; t < timePointCnt; t++){
      bottoms[t] = 1;
      try {
        for(int j = 1; j < horizonSize; j++){
          if(curPrice[t] * (1 - stopMargin/100) > (futurePrice[t+j])) {
            bottoms[t] = -1;
          }
        }
        if(curPrice[t] > (futurePrice[t+horizonSize]))  {
          bottoms[t] = -1;
        }
      } catch (ArrayIndexOutOfBoundsException e)  {
        bottoms[t] = 0;}
    }
    return bottoms;
  }


  /**
   * Extract long-and-hold bottoms of varied holding period.
   *
   * @param inst - {@code weka.core.Instances} containing series to be extracted
   * @param index1 - attribute index of series to be extracted
   * @param index2 - attribute index of look-ahead series to be extracted
   * @param ror - rate of return
   * @return - input instances merged with newly extracted instances
   */
  public Instances holdBottoms(Instances inst, int index1, int index2, double ror)  {
    double[] close = inst.attributeToDoubleArray(index1);
    double[] low = inst.attributeToDoubleArray(index2);
    double[] nBottoms;
    for(int horizonSize = 1; horizonSize < 41; horizonSize++) {
      nBottoms = bottom(close, low, horizonSize, ror);
      inst = Instances.mergeInstances(inst, doubleArrayToAttribute(horizonSize + "bottom",
nBottoms));
    }
    return inst;
  }

  /**
   * Extract long and stop-loss bottoms of varied horizon size.
   *
   * @param inst - {@code weka.core.Instances} containing series to be extracted
   * @param index1 - attribute index of series to be extracted
   * @param index2 - attribute index of look-ahead series to be extracted
   * @param stopMargin - e.g. 0.5 means 0.5%, not 50%
   * @return - input instances merged with newly extracted instances
   * @throws Exception
   */
  public Instances stopBottoms(Instances inst, int index1, int index2, double stopMargin)
throws Exception  {
    double[] close = inst.attributeToDoubleArray(index1);
    double[] low = inst.attributeToDoubleArray(index2);
    double[] nBottoms;
    for(int horizonSize = 1; horizonSize < 41; horizonSize++) {
      nBottoms = nBottom(close, low, horizonSize, stopMargin);
      inst = Instances.mergeInstances(inst, doubleArrayToAttribute(horizonSize + "bottom",
nBottoms));
    }
    return inst;
  }
```

```java
  /**
   * Extract percentage gap open.
   *
   * @param inst
   * @param O - open price
   * @param C - close price
   * @return
   * @throws Exception
   */
  public Instances gap(Instances inst, int O, int C) throws Exception {
    int timePointCnt = inst.numInstances();
    fdm.etl.WekaFilter f = new WekaFilter();
    inst = f.fTimeSeriesTranslate(inst, Integer.toString(C+1), -1);
    double[] open = inst.attributeToDoubleArray(O);
    double[] yesterday_close = inst.attributeToDoubleArray(C);
    double[] gap = new double[timePointCnt];
    for(int t = 0; t < timePointCnt; t++) {
      gap[t] = (open[t]-yesterday_close[t]) / yesterday_close[t];
    }
    return Instances.mergeInstances(inst, doubleArrayToAttribute("gap", gap));
  }


/* ...remaining methods has a similar form */
```

SlidePlot2D

```
package fdm.etl;
/* import declarations */

/**
 * @description Fixed sliding window plotter. Best to set time series as master data.
 * Left key goes back in time, right key goes forward.
 * Up and down keys to alter class label. W and S keys to alter y dimension. A and D keys to
alter x dimension.
 *
 * @author    Ying Ting Chung
 */
public class SlidePlot2D extends Plot2D implements KeyListener  {
  /* variable declarations */

  /** Constructor initialising master data copy and label
   * @param master - master data */
  public SlidePlot2D(Instances master)  {
    this.masterData = master;
    label = new JLabel();
    label.setForeground(Color.white);
    this.add(label, -1);
    for(int index = 0; index < master.numAttributes(); index++) {
      if(master.attribute(index).isDate())  {
        this.date = index;
        break;}
    }
  }

  public SlidePlot2D(Instances master, int time, int windowSize)  {
    this.masterData = master;
    this.time = time;
    this.windowSize = windowSize;
    label = new JLabel();
    label.setForeground(Color.white);
    this.add(label, -1);
    for(int index = 0; index < master.numAttributes(); index++) {
      if(master.attribute(index).isDate())  {
        this.date = index;
        break;}
    }
  }

  @Override
  public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == 37) { // left key
      if(time > 0)  {
        time--; updatePlot();}
    } else if (e.getKeyCode() == 39)  { // right key
      if(time + windowSize < masterData.numInstances()) {
        time++; updatePlot();}
    } else if (e.getKeyCode() == 38)  { // up key
      if(cls > 0) {
        cls--; updatePlot();}
    } else if (e.getKeyCode() == 40)  { // down key
      if(cls < masterData.numAttributes() - 1)  {
        cls++; updatePlot();}
    } else if (e.getKeyCode() == 65)  { // 'a' key
      if(X > 0) {
        X--; updatePlot();}
    } else if (e.getKeyCode() == 68)  { // 'd' key
      if(X < masterData.numAttributes() - 1)  {
        X++; updatePlot();}
    } else if (e.getKeyCode() == 87)  { // 'w' key
      if(Y < masterData.numAttributes() - 1)  {
        Y++; updatePlot();}
    } else if (e.getKeyCode() == 83)  { // 's' key
      if(Y > 0) {
        Y--; updatePlot();}
    }
  }
```

```
  @Override
  public void keyReleased(KeyEvent e) {}

  @Override
  public void keyTyped(KeyEvent e) {}

  /** Sets x-coordinate of the plot
   * @param index - attribute index */
  public void setX(int index) {this.X = index;}

  /** Sets y-coordinate of the plot
   * @param index - attribute index */
  public void setY(int index) {this.Y = index;}

  /** Sets index of the attribute to use for colouring
   * @param index - attribute index */
  public void setC(int index) {this.cls = index;}

  /** Sets beginning time point of sliding window
   * @param time -  */
  public void setTime(int time) {this.time = time;}

  /** Sets size of sliding window
   * @param windowSize - number of data points */
  public void setWindowSize(int windowSize) {this.windowSize = windowSize;}

  public void updatePlot()  {
    WekaFilter f = new WekaFilter();
    StringBuilder sb = new StringBuilder();

    slideData = new Instances(masterData, time, windowSize);
    slideData.setClassIndex(cls);

      try {
        if(!slideData.attribute(cls).isDate())  {
          slideData = f.fNumericToNominal(Integer.toString(cls+1), slideData);
        }
        setInstances(slideData);
    } catch (Exception e) {
      e.printStackTrace();
    }

    if (slideData.classIndex() == -1)
        setCindex(slideData.numAttributes() - 1);
      else
        setCindex(slideData.classIndex());

      setXindex(X);
      setYindex(Y);
      sb.append("<html>");
      sb.append(slideData.firstInstance().stringValue(date).substring(0, 10)).append(" to
").append(slideData.lastInstance().stringValue(date).substring(0, 10));
      sb.append("<br>  X: ").append(slideData.attribute(X).name()).append(" -- Y:
").append(slideData.attribute(Y).name());
      sb.append("<br>  Label:
").append(slideData.attribute(slideData.classIndex()).name());
      label.setText(sb.toString());
      repaint();
  }
}
```

SlideIBk

```
package fdm.eval;
/* import declarations */

/**
 * @description fixed sliding window k-Nearest Neighbour classification and evaluation
 * @author      Ying Ting Chung
 */
public class SlideIBk extends IBk {

  private Instances masterData;
  private String[] attributes;
  private ConfusionMatrix confusionMatrix;
  private int windowSize;
  private int horizonSize;
  private Instances trainingSet;
  private Instances testSet;
  private FastVector atts;
  private Instances predData;

  public SlideIBk() {
    super();
  }

  public SlideIBk(int k)  {
    super(k);
  }

  public SlideIBk(Instances master) {
    this.masterData = master;
    this.attributes = new String[masterData.numClasses()];
    for (int i = 0; i < attributes.length; i++) {
      attributes[i] = masterData.classAttribute().value(i);
    }
    atts = new FastVector();
    FastVector labels = new FastVector();
    labels.addElement("-1");  // pred_val == 0
    labels.addElement("0");
    labels.addElement("1");   // pred_val == 2
    atts.addElement(new Attribute("predictions", labels));
  }

  public void setWindowSize(int windowSize) {this.windowSize = windowSize;}
  public void setHorizonSize(int horizonSize) {this.horizonSize = horizonSize;}

  public int getWindowSize()  {return this.windowSize;}
  public int getHorizonSize() {return this.horizonSize;}

  public ConfusionMatrix getConfusionMatrix() {return confusionMatrix;}
  public Instances getPredictions() {return predData;}

  public void buildClassifer() throws Exception {
    IBk kNN = new IBk(getKNN());
    Evaluation eval;
    NominalPrediction nom;

    predData = new Instances("Predictions", atts, 0);

    confusionMatrix = new ConfusionMatrix(attributes);

    for (int i = 0; i < windowSize; i++)  {
      double[] pred_val = new double[predData.numAttributes()];
      pred_val[0] = 1.0;
      predData.add(new Instance(1.0, pred_val));
    }
    for (int t = 0; t + windowSize + horizonSize <= masterData.numInstances(); t++) {
      double[] pred_val = new double[predData.numAttributes()];
      trainingSet = new Instances(masterData, t, windowSize);
      testSet = new Instances(masterData, t + windowSize, horizonSize);
      kNN.buildClassifier(trainingSet);
      eval = new Evaluation(trainingSet);
```

```
      eval.evaluateModel(kNN, testSet);
      nom = (NominalPrediction) eval.predictions().elementAt(0);
      pred_val[0] = nom.predicted();
      predData.add(new Instance(1.0, pred_val));
      confusionMatrix.addPredictions(eval.predictions());
    }
  }
}
```

Bibliography

(n.d.). Retrieved April 2011, from Investopedia: http://www.investopedia.com/terms/t/technicalanalysis.asp

A., B., & M., O. Biologically Inspired Algorithms for Financial Modelling. In B. A., & O. M., *Biologically Inspired Algorithms for Financial Modelling* (p. 16). Springer.

BIS Monetary and Economic Department. (2010). *Triennial Central Bank Survey.*

Comma-separated values. (2010, November 23). Retrieved from Wikipedia: http://en.wikipedia.org/w/index.php?title=Comma-separated_values&oldid=398858843

Hastie, T., & Tishirani, R. (1996a). Discriminant adaptive nearest-neighbor classification. *IEEE Pattern Recognition and Machine Intelligence* (18), 607-616.

Huang, T.-H., & Li, Z.-Y. (2008). Stock Prediction Using Web News Articles.

Pauly, R., & Schell, A. (1989). Calendar Effects in Structural Time Series Models . *Empirical Economics, Springer, vol. 14(3)* , 241-56.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia , 4*, 1883.

Schwert, G. W. (2002). Anomalies and Market Efficiency. *Simon School of Business Working Paper No. FR 02-13* .

Thomson Financial News Limited. (2008, June 6). *HKEx says late fluctuation in some stocks last Friday due to MSCI changes.* Retrieved from http://www.forbes.com/feeds/afx/2008/06/01/afx5067901.html

Vásquez, M., F., A., Osorio, G., D., F., & Losada, H. (2009). Mining Candlesticks Patterns on Stock Series: A Fuzzy Logic Approach. *ADMA '09 Proceedings of the 5th International Conference on Advanced Data Mining and Applications.* Springer-Verlag Berlin, Heidelberg ©2009.