From Software Engineer to Day Trader in 3 Easy Steps: A Comparison of Software Engineering (SE) Data Mining with Financial Data Mining[1]

Gary D. Boetticher
University of Houston – Clear Lake
2700 Bay Area Boulevard
Houston, Texas 77058
1 281.283.3805
Boetticher@uhcl.edu

ABSTRACT
One of the research objectives in Software Engineering Data Mining is to produce useful, usable, verifiable, and repeatable models for better managing software processes and projects. This objective implies that an organization will be more profitable as consequence of better management.

Financial events, such as bailouts, high unemployment rates, foreclosures, etc. have received extensive news coverage since Fall, 2008. Many professionals are concerned about their job status and their retirement accounts. From the context of the Software Engineering community, one question arises: *To what extent can the skills and knowledge attained in Software Engineering Data Mining apply towards Financial Data Mining?*

A reader may be motivated to explore both types of data mining due to the potential rewards. Furthermore, by studying both domains makes it possible to determine the financial impact of delivering software on time or with fewer defects.

This paper examines 3 aspects related to both types of data mining. The underlying data used for constructing models, the models themselves, and validation techniques.

Categories and Subject Descriptors
D.2.8 [**Metrics**]: Performance measures

General Terms
Measurement, Economics, Experimentation.

Keywords
Software engineering, data mining, day trading, financial data mining, stock market

1. INTRODUCTION
From 1982 through 2007 the stock market (S&P) grew by an annual rate of 9 percent. These 26 years of prosperity lulled many investors into a "buy and hold" mindset. In 2008, the average U.S. portfolio lost approximately 39 percent. To fully recover these losses would take 9 years at an annual rate of 6 percent net profit. Such a drastic change in market conditions challenges the traditional "buy and hold" strategy. An alternative approach, which seeks to identify buying and selling opportunities, might be more appropriate given these volatile times.

Shifting the focus from "what to buy" to "when to buy" suggests the application of financial data mining for identifying buying and selling opportunities.

The potential for huge financial gains and the intellectual challenges are sufficient incentives for researching this domain. Those who have studied data mining in the Software Engineering domain may wonder about the extent of reusability of their Software Engineering data mining skills in the financial domain. Even if one is not motivated to pursue financial data mining, studying financial data mining could be beneficial in exploring the financial impact of delivering software on time or with fewer defects upon a company's profit margin relative to the ebb and flow of the stock market.

To migrate from Software Engineering Data Mining to Financial Data Mining, three aspects of each data mining domain are presented. The first aspect considers the nature of data, in terms of construct, quality, and quantity from each domain. Understanding the nature of data helps determine plausible approaches to building models relative to the domain. The second aspect considers the relative models, metrics, and indicators for each domain and how they may be used in the decision making process. The third aspect examines the validation process of models. This is an important step for it gives credibility to a model. This paper elaborates on each of these aspects.

This paper is organized as follows. Section 2 presents a justification for financial data mining. Section 3 describes the nature of data from the two domains. Section 4 explores metrics, indicators, and models from both domains. In section 5 validation techniques are described. Section 6 discusses these aspects plus other aspects to consider when performing financial data mining using actual money. Section 7 offers a conclusion.

2. WHY FINANCIAL DATA MINING?
If you believe you or anyone else has a system that can predict the future of the stock market, the joke is on you [17].

There are numerous papers on financial data mining that apply some type of machine learner to a financial data set. Below are some examples.

Frick [10] combines Point-And-Figure (PNF) charting and Genetic Algorithms. Given a series of price movements, a corresponding PNF chart is created. The Genetic Algorithm generates a set of rules to apply to the PNF chart and the results are assessed.

Mizuno et al. [14] build a neural network model against the Tokyo Stock Exchange Prices Index (TOPIX) spanning a 5-year period (1982-1987). The inputs consist of 4 technical indicators and three outputs (Buy/Hold/Sell). They use an "Equalized Learning Method." This method duplicates vectors with Buy/Sell output values as a way of redistributing instance counts. They tested their models against the TOPIX from 1986 to 1987. Their best model produces an annual ROI of 20%. However, the buy-and-hold strategy during this period produced an annual ROI of 21%.

Chenoweth et al. [7] uses a Directional Index (Directional Movement divided by True Range). The inputs consists of S&P500; S&P500, lagged 1 day; S&P500, lagged 2 days; U.S. Treasury Rate, lagged 2 months; U.S. Treasury Rate, lagged 3 months; and 30 Year Government bonds. They trained on data spanning 1982 through 1988 and tested on data from 1989 through 1993. Their best model achieves an annual rate of return of 16.39 percent.

Financial Data Mining is often met with a certain degree of skepticism. The efficient market hypothesis [9] states that everyone has access to the same information; therefore any success appears to be a matter of chance. What is often lacking in the financial data mining research domain are results using real money. In this situation where a researcher becomes trader and is much more vulnerable to the financial pressures and may undermine the model (assuming it is not totally automated).

Thus, if it is not possible to make money using financial data mining, then there is no reason to even consider migrating from Software Engineering Data Mining to Financial Data Mining.

To address this concern, the author presents his 2008 and 2009 financial results and 2009 results in Tables 1 and 2 respectively. The first three rows of each table show the statistical results for *long* trades (buy to enter, sell to exit), *short* trades (sell to enter, buy to exit), and the sum of the *long* and *short* trades. The second three rows of each table show composite trades. A composite trades consists of one or more raw trades that may have multiple entry points, but a single entry point. An example of a composite trade might buy one share of *IBM* at 100, buy a second share at 90, then sell both shares at 98. The first raw trade lost two points (98 - 100), the second raw trade gained eight points (98 – 90), and the composite trade would have netted 6 points.

Table 1: 2008 Trading Results

	Winners	Losers	Pct.
Raw Trades (Long)	412	193	68%
Raw Trades (Short)	408	184	69%
Raw Trades (Total)	820	377	69%
Composite Trades (Long)	308	38	89%
Composite Trades (Short)	272	24	92%
Composite Trade (Total)	580	62	90%

Table 2: 2009 Trading Results

	Winners	Losers	Pct.
Raw Trades (Long)	46	15	75%
Raw Trades (Short)	25	14	64%
Raw Trades (Total)	71	29	71%
Composite Trades (Long)	37	0	100%
Composite Trades (Short)	20	0	100%
Composite Trade (Total)	57	0	100%

One may argue that statistical success does not necessarily equate to financial success. For 2008, the trading portfolio increased by 120 percent over 5.3 months of trading. For 2009, the trading portfolio is up 9 percent over 1.5 months of trading.

3. DATA SETS

At the foundation of any type of data mining is the data. This section compares software engineering data with financial data in terms of repeatability, reliability, and complexity.

3.1 Software Engineering Data

The largest public collection of software engineering data is the Predictor Models in Software Engineering (PROMISE) repository. As of April, 5, 2009 the PROMISE repository contained 88 software engineering-based data sets [6]. Fifty-six of these sets relate to defect prediction, 11 to effort estimation, 7 general, 5 model-base SE, and 9 are text mining. Reuse of data across these sub-domains is not possible. Next, we focus on the two most prominent sub-domains defect data and effort prediction.

Defect Data

More than half (31 out of 56) of the defect data sets have the following layout: There are $n - 1$ independent variables which refer to various software metrics (size, language, or complexity), and a dependent variable which refers to the number of defects per module. Most of these 31 data sets come from a Turkish Telecommunications company or NASA. Comparing these 31 defect data sets with the other 25 data sets would not be fruitful due to major differences in layout and meaning.

Performing comparisons between the 31 data sets with the same general layout is difficult for several reasons. Two data sets may appear to contain the same metrics, but they may differ in how they define one or particular metric. A common example is how *Source Lines of Code* (*SLOC*) is defined. SLOC counts may include/exclude a variety of components such as comments, compiler directives, etc. There are at least 1000 permutations in defining SLOC [16]. Differences in SLOC count would over-under-state the amount of defects within an application.

A second problem with the 31 data sets relates to defect counts. Defect count is highly resource dependent. It depends upon the number of persons reviewing some, their ability to identify defects, and the total amount of time dedicated to reviewing the code. Thus, defect counts are dependent on resource allocation It does not appear that any of the 31 defect data sets contain resource allocation metrics.

Effort Estimation

Within the 11 effort estimation data sets, 3 contain data related to some type COCOMO model, 3 are human-based models, and the

other 5 do not fit any pattern. Since COCOMO is the most recognized effort estimation data set with the PROMISE repository, we examine these data sets.

One must use caution when comparing the COCOMO-based data sets from the PROMISE repository for three reasons. Some of the COCOMO data collected spans more than 12 years (early 80s through early 90s). Thus, older COCOMO data may not contain metrics that reflect technological improvements in the software development process. Second, the COCOMO model contains many metrics that are subjective in nature. They are dependent upon an estimator's assessment of the various facets of a software project. Third, many estimates consist of aggregate values. For example, the Process Maturity (*PMAT*) metric is based on 18 estimates.

3.2 Financial Data

Financial data is essentially a series of ticks, where a tick represents a point in time where some financial instrument (e.g. a stock) is traded at a particular price. The price activity is segmented by a time-interval, such as one hour. During that time period the first trade price (the *Open*), the highest trade priced (the *High*), the lowest trade price (the *Low*), and the last trade price (the *Close*) are recorded. The total number of items traded (e.g. number of shares) during this period, called the *Volume*, is also recorded. Normally, the *Open*, *High*, *Low*, *Close*, and *Volume* are abbreviated as *OHLCV*.

The *OHLCV* may be captured for multiple time frames ranging from 30 seconds up to quarterly (3 months). Choosing a particular time frame is a matter of personal preference.

Daily, weekly, and/or monthly data may be acquired for free. A common source is Yahoo finance (http://finance.yahoo.com/). Intraday data may be purchased for a nominal cost from a commercial data vendor [4, 15].

It is possible to get data for thousands of different stocks spanning decades. Data spanning a relatively long time frame may need to be adjusted to account for inflation. Also, a stock may experience a stock split. This would cause a drastic change in the price of a stock without changing the underlying value.

4. EQUATIONS/TECHNICAL INDICATORS

In the financial data mining domain the term *Technical Indicator* is used to describe some type of equation.

In the Software Engineering domain, effort estimation equations, such as COCOMO [5] or Function Point Analysis [3] are more recognizable than defect prediction equations such as Akiyama [2], Halstead [12], Lipow [13], Gaffney [11], UNISYS [8]. This lack of recognition seems to be mostly a consequence of the infancy of the Software Engineering domain.

There are at least 485 technical indicators available in the financial domain. A good introduction to some of the more common technical indicators is [1]. Some of the more common technical indicators include Relative Strength Index (RSI), and Simple Moving Average (SMA).

Technical indicators by themselves are really metrics. What makes them indicators is by imposing some condition for making a buy or sell decision. Figure 1 illustrates this idea with the RSI indicator. When the RSI crosses below the 70 threshold suggests it is time to sell. When the RSI crosses above the 30 threshold suggests it is time to buy.

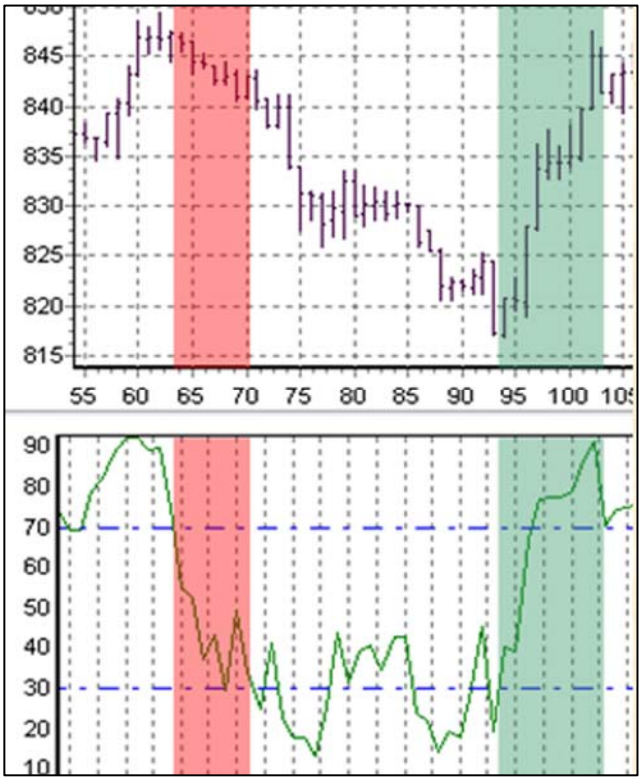

Figure 1: Graph of RSI Indicator

A second example might be whenever two or more simple moving averages (based on different periods) cross. Figure 2 illustrates this idea. In this case, the market had a major downturn as a result of a short moving average crossing below a longer moving average. Short period moving averages crossing above longer period moving averages might indicate that the price will rise.

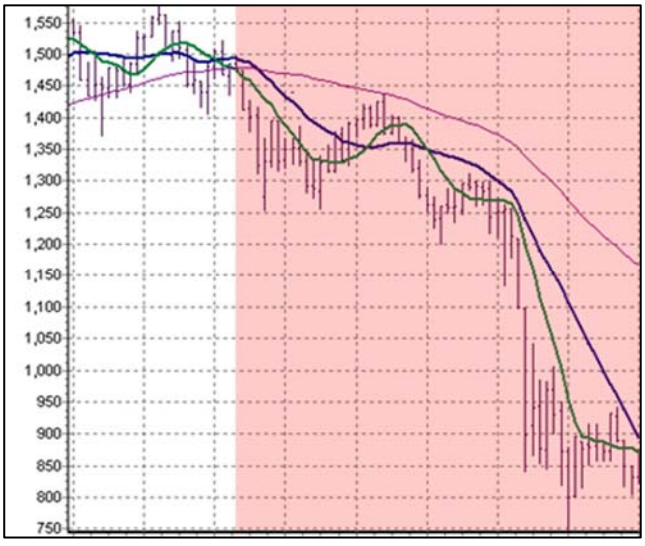

Figure 2: Moving Averages Crossing

Thus, technical indicators would be used to generate a model which is a combination of buy and sell rules. An example of a simple financial model as depicted below.

```
BUY = RSI crosses above 30
SELL = RSI crosses below 70
```

5. MODEL VALIDATION

Both domains validate models by dividing a data set into independent data sets (train and test), using the training data to build a model (assuming that the user chooses not to use a pre-existing model) and assessing the model using test data.

Software Engineering models may be assessed using Pred (percentage of answers with a certain scope of accuracy), Mean Magnitude of Relative Error (MMRE), Mean Absolute Relative Error (MARE), or ROC curves. The Software Engineering community has not reached a consensus regarding which assessment approach to adopt, and which settings (e.g. PRED(25) or PRED(30)) to use.

In the Financial Domain, models can be assessed using statistical methods (e.g. percent of winning trades), but statistical success doe not always guarantee financial success. Thus, a financial model may be assessed by how much profit it earns.

Using only profit may not be sufficient to build a robust model. Consider the results from two financial models in Figure 3. Both models generate the same amount of profit. However, Model A is much more volatile than model B. This makes model B more attractive than A.

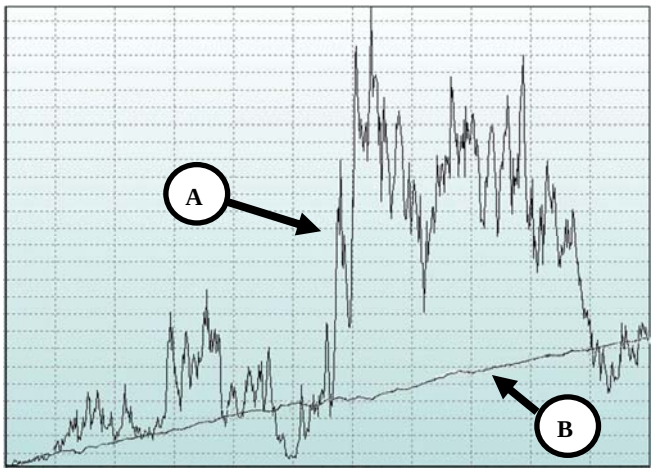

Figure 3: Results from Two Financial Models

The financial validation must take into account certain financial realities. An example may be that a model can not trade more than 10 percent of the average *Volume* for a given stock. To ignore this type of rule may result in a model that has a million dollars in equity buying 20 million shares of a 5-cent stock.

6. DISCUSSION

By studying these three areas, the data, metrics/indicators, and validation, a person may be able to migrate their SE data mining knowledge and skills into the financial domain in order to build viable financial models.

Unfortunately having a viable financial model does not prepare a person for the demands of actual trading. One must master the emotional pressures associated with trading real money and exercise an extensive amount of discipline. The best way to improve in these areas is practice paper trading, then eventually trade with real money.

A second aspect of trading is money management. Very few models are 100 percent successful. Therefore, it is important that a person know how to avoid accumulating large losses.

7. CONCLUSIONS

This paper compares some of the features of SE data mining with Financial data mining in terms of data, metrics/indicators, and model validation.

Creating a financially viable model does not guarantee financial success. Attaining success includes mastering one's emotions and money management skills.

8. REFERENCES

[1] S. Achelis, S., Technical Analysis from A-Z, McGraw Hill Professional, 2000, Pp. 1 – 380.

[2] F. Akiyama, "An Example of Software System Debugging," *Information Processing*, vol. 71, pp. 353-379, 1971.

[3] A.J., Albrecht, "Measuring Application Development Productivity," Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation, 1979, pp. 83–92.

[4] Ashkon Technology, 2004, Information available at: www.ashkon.com

[5] Barry Boehm, et al. *Software cost estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ:Prentice-Hall, 2000.

[6] Boetticher, G., Menzies, T., and T. Ostrand, PROMISE Repository of empirical software engineering data http://promisedata.org/repository, West Virginia University, Department of Computer Science, 2007

[7] Chenoweth, T., Obradovic, Z. and S. Lee, "Embedding Technical Analysis into Neural Network Based Trading Systems," *Applied Artificial Intelligence*, vol 10, no. 6., 1996, Pp. 523-541.

[8] Compton, T., and C. Withrow, "Prediction and Control of Ada Software Defects," *J. Systems and Software*, vol. 12, pp. 199-207, 1990.

[9] Eugene F. Fama, "Random Walks in Stock Market Prices," *Financial Analysts Journal*, Sept./Oct. 1965.

[10] Frick, et al., Genetic-Based Trading Rules - A New Tool to Beat the Market With -- First Empirical Results, in Aktuarielle Ansätze für Finanz-Risiken, Proceedings of 6th International AFIR Colloquium, Nürnberg, 1.-3. October 1996, (Editor Pete Albrecht) Verlag Versicherungswirtschaft e.V. Karlsruhe, Volume I/II, pp. 997 - 1018 (with coauthors A. Frick, R. Herrmann, M. Kreidler and A. Narr).

[11] J.R. Gaffney, "Estimating the Number of Faults in Code," *IEEE Trans. Software Eng.*, vol. 10, no. 4, 1984.

[12] M.H. Halstead, *Elements of Software Science*. Elsevier North-Holland, 1975.

[13] M. Lipow, "Number of Faults per Line of Code," *IEEE Transactions on. Software Engineering*, vol. 8, no. 4, pp. 437-439, 1982.

[14] Mizuno, et al., Application of Neural Network To Technical Analysis of Stock Market Prediction, Studies in Informatics and Control (With Emphasis on Useful Applications of Advanced Technology), 7 2, June 1998.

[15] Alexandre Nikolenko, ANFutures website. Information available at: http://www.anfutures.com/

[16] Park, Robert E. et al. Software Size Measurement: A Framework for Counting Source Statements (CMU/SEI-92-TR-20, ADA 258-304). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992.

[17] Ralph Wanger, A Zebra in Lion Country: Ralph Wanger's Investment Survival Guide Everett Mattlin, Publisher: New York, NY : Simon & Schuster, c1997

[1] Investing is a **very serious and risky endeavor**. This paper is not intended to prepare the reader for investing of **any kind**. The author and the University of Houston at Clear Lake assume **no responsibility** for any direct or indirect expenses or losses incurred by the reader or any individuals affiliated with the reader as a result of applying software and/or techniques associated with this paper.