Spring 2014

Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction

Michael David Rechenthin
University of Iowa

MACHINE-LEARNING CLASSIFICATION TECHNIQUES FOR THE ANALYSIS AND PREDICTION OF HIGH-FREQUENCY STOCK DIRECTION

by

Michael David Rechenthin

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration (Management Sciences)
in the Graduate College of
The University of Iowa

May 2014

Thesis Supervisor: Professor W. Nick Street

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Michael David Rechenthin

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Business Administration (Management Sciences) at the May 2014 graduation.

Thesis committee: _____
W. Nick Street, Thesis Supervisor

_____
Gautam Pant

_____
Padmini Srinivasan

_____
Tong Yao

_____
Kang Zhao

ACKNOWLEDGEMENTS

I would like to dedicate this thesis to my wife, Abby, and to my parents, Dr. and Mrs. David and Ellen Rechenthin. I am deeply grateful for my dad's valued guidance throughout this entire process and for the loving support and encouragement from Abby and my mom.

I would like to express my gratitude for the supervision of my advisor, Dr. Nick Street, whose direction made this paper possible. I would also like to thank my thesis committee, Dr. Padmini Srinivasan, Dr. Gautam Pant, Dr. Kang Zhao, and Dr. Tong Yao, for their insightful comments and advice.

Finally, I would like to acknowledge the generous financial support of University of Iowa's Department of Management Sciences.

ABSTRACT

This thesis explores predictability in the market and then designs a decision support framework that can be used by traders to provide suggested indications of future stock price direction along with an associated probability of making that move. Markets do not remain stable and approaches that are highly predictive at one moment may cease to be so as more traders spot the patterns and adjust their trading techniques. Ideally, if these "concept drifts" could be anticipated, then the trader could store models to use with each specific market condition (or concept) and later apply those models to incoming data. The assumption however is that the future is uncertain, therefore future concepts are still undecided.

Maintaining a model with only the most up-to-date price data is not necessarily the most ideal choice since the market may stabilize and old knowledge may become useful again. Additionally, decreasing training times to enable modified classifiers to work with streaming high-frequency stock data may result in decreases in performance (e.g. accuracy or AUC) due to insufficient learning times. Our framework takes a different approach to learning with drifting concepts, which is to assume that concept drift occurs and builds this into the model. The framework adapts to these market changes by building thousands of traditional base classifiers (SVMs, Decision Trees, and Neural Networks), using random subsets of past data, and covering similar (sector) stocks and heuristically combining the best of these base classifiers. This "ensemble", or pool of multiple models selected to achieve better predictive per-

formance, is then used to predict future market direction. As the market moves, the base classifiers in the ensemble adapt to stay relevant and keep a high level of model performance. Our approach outperforms existing published algorithms.

This thesis also addresses problems specific to learning with stock data streams, specifically class imbalance, attribute creation (e.g. technical and sentiment analysis), dimensionality reduction, and model performance due to release of news and time of day. Popular methods for dealing with each are discussed.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

Predicting stock price direction is something individuals and financial firms have researched for years. Books and papers have been written on the subject, but rarely are the results repeatable. Recent research has shown greater predictability in high-frequency stock data (by second or by minute, rather than daily or weekly), but this research is often under represented in the academic literature. Historically, this is due to the lack of availability of trade-by-trade data, and the difficulty in working with such large quantities of it. Furthermore, determining future market direction in practice requires special consideration since streaming stock data may arrive faster than a model may produce results; a model that takes 30 minutes to arrive at a prediction is of little value if the objective was to predict one minute in the future. This thesis explores the predictability of stock market direction using machine learning classification techniques and high-frequency stock data.

These techniques would be of considerable interest to quantitative traders who produce mathematical models that account for as much of 55% of the total volume of US traded stocks [2]. Our research objective is to build a decision support framework that can be used by traders to provide suggested indications of future stock price direction along with an associated probability of making that move. For example, if a trader wanted to purchase an equity position, knowing whether to buy now or wait and re-evaluate in n seconds could allow the trader to purchase the stock at a lower price than was previously expected. Over time this could make a significant

difference in the profitability of the strategy.

It is argued that the lack of published working models exists because there is little incentive to publish such methods in academic literature. The incentive to instead sell them to a trading firm is much greater, monetarily. Also Timmermann and Granger [216] write of a possible "file drawer" bias in published studies due to the difficulty in publishing empirical results that are often barely or border-line statistically insignificant; but markets, since they are partially driven by human emotions, involve a large degree of error. This may result in a glut of research arguing that the market is efficient, and thus unpredictable; Timmermann and Granger calls this the "reverse file-drawer" bias. It is also possible that many traditional forms of stock market prediction are simply inadequate or sponsoring companies may not wish to divulge successful applications [245]. We demonstrate that inefficiency and moments of predictability exist using 22 million stock transactions. This is discussed further in Chapter 2.

When predicting stock price direction, practitioners typically use one of three approaches. The first is the fundamental approach, which examines the economic factors that drive the price of stock (e.g. a company's financial statements such as the balance sheet or income statement). The second approach is to use traditional technical analysis to anticipate what *others* are thinking based on the price and volume of the stock. Indicators are computed from past prices and volumes and these are used to foresee future changes in prices. The goal of technical analysis is to identify regularities by extracting patterns from noisy data and by inspecting the stock charts

visually. Studies show 80% to 90% of polled professionals and individual investors rely on at least some form of technical analysis [157, 162, 163, 213]. With recent breakthroughs in technology and algorithms, technical analysis has morphed into a more quantitative and statistical approach [154]; this is what we call quantitative technical analysis and it is the third approach to predicting market direction. Whereas traditional technical analysis is *visual*, quantitative technical analysis is *numerical*, which allows us to easily program the rules into a computer. This is the method explored in this thesis.

Markets do not remain stable; indicators that be highly predictive at one moment may cease to be so as more traders spot the patterns and implement them in their trading approaches. Widespread adoption of a particular trading strategy is enough to drive the price either up or down enough to eliminate the pattern [201, 215]. This *concept drift* complicates the learning of models and is unique to streaming data. As the concept changes, model performance may decrease, requiring an update in the training data and/or change in the quantitative technical analysis indicators used as attributes. Modern machine learning classification techniques provide solutions and this, along with quantitative technical analysis, allows us to outperform existing published methods of stock market direction.

We begin Chapter 3 with an introduction to the discussion of machine learning for streaming data and in particular high-frequency stock data. This includes descriptions of traditional supervised learning methods and different ways of evaluating classifiers that are used for analysis later in the thesis. Chapter 4 then discusses

the two main approaches to learning from streaming data: adaptive (online) and wrapper methods. Adaptive methods learn data incrementally (as instances arrive) and efficiently with a single pass through the data; they are ideal for use with high-frequency streaming data because they require limited time and memory. *Forgetting factors* can be integrated in the model to give less weight to older data, thus *gradually* making older data obsolete. Wrapper methods use traditional classification algorithms, such as support vector machines or neural networks (both are explained in Chapter 3), that learn on collected batches of data. The models are then chosen and combined to form predictions, such as through the use of ensembles.

High-frequency streaming stock data requires special consideration for three main reasons: first, the market is constantly changing, so models quickly become obsolete; second, speedy processing is required to make fast predictions; and third, specific volumes of data are needed to make precise decisions. In Chapter 5 we address problems specific to stock data such as imbalanced datasets, too much and irregularly spaced data, and attribute creation and selection.

In Chapter 6, we discuss our new wrapper-based ensemble method that provides a solution for all three considerations outlined in Chapter 5. This method is created by building thousands of models in parallel using price and volume data covering different periods of time and using different stocks, and then efficiently switching between these models as time progresses. We then demonstrate an improvement over existing methods using our new approach.

Lastly in Chapter 7 we summarize our thesis and discuss an additional idea

for future work; specifically using misclassification costs to optimize decisions rather than AUC.

CHAPTER 2
BACKGROUND OF MARKET PREDICTABILITY

2.1 Overview

This thesis examines the use of machine learning algorithms for the prediction of stock price direction in the near term (e.g. seconds to minutes in the future). Mainstream finance theory has traditionally held the view that financial prices are efficient and follow a random walk, so we address this question throughout this chapter. How then is it possible to predict future prices if markets are efficient and thus unpredictable? In Section 2.2 market efficiency is defined and from this we explain the Efficient Market Hypothesis. In Section 2.3 we explore market inefficiency by running a series of experiments using conditional probabilities to demonstrate that the market has moments of high predictability that are not explained by traditional market dynamics. Starting in Section 2.4 we explain the differences between the fundamental and technical approaches to predicting stock prices, and provide a brief introduction to how these quantitative methods can be used along with modern machine learning algorithms to predict future price direction.

2.2 Market efficiency

2.2.1 Definition

Mainstream finance theory has traditionally held the view that financial prices are efficient and follow a random walk and are thus unpredictable. The definition of a random walk is a process by which the changes from one time period to the next

are independent of one another, and are identically distributed. One of the earliest studies of market efficiency was done by Fama in 1965 to describe how equity prices at any point in time best represent the actual intrinsic value, with the prices updating instantaneously to information [71]. Efficiency is associated with a trendless and unpredictable financial market.

According to the theory of random walks and market efficiency, the future direction of a stock is no more predictable than the path of a series of cumulative random numbers [71]. Statistically it can be said that each successive price is independent from the past; each series of price changes has no memory. If testing for market independence, the probability of market directional-movement at time t is compared against time $t-1$. The same should hold as more prior information is added since, according to market efficiency, the past cannot be used to predict the future.

The theory of random walks and efficiency of market prices was expanded by Fama in [71] to the Efficient Market Hypothesis (EMH) in the 1960's. The theory states that the current market's price is the correct one, and any past information is already reflected in the price. According to the EMH, although no market participant is *all knowing*, collectively they know as much as can be known; for as a group, they *are* the market. These individuals are constantly updating his or her beliefs about the direction of the market, and although he or she will disagree on the direction of the stock, this will lead, as noted by Fama, to "a discrepancy between the actual price and the intrinsic price, with the competing market participants causing the stock to

wander randomly around its intrinsic value [71]."

If markets are indeed efficient, then it implies that markets never over- or under-react during the trading day. Any effort that an average investor dedicates to analyzing and trading securities is wasted, since one cannot hope to consistently *beat* the market benchmark. Any attempt to predict future prices is futile and although high rates of return may be achieved, they are on the average, proportional to risk. In addition, high risk may achieve high rates of return, but it also can deliver high rates of loss. For example, when flipping a fair coin, a roughly 50% chance of getting heads would be expected; however, expecting heads ten times consecutively would come with high risk. The concept of an efficient market implies that consistently predicting the market carries a high degree of risk[1].

2.2.2 Purely random?

The Efficient Market hypothesis implies that consistently beating the returns of the market benchmark comes with a high degree of risk. Although the probability of outperforming the market is extremely small, given enough people trying to do so, eventually someone will succeed by pure randomness. This argument can be used to explain successful traders or the great wealth of Warren Buffett ($53.3 billion), who is also considered one of the world's greatest investors.

William Sharpe, as quoted in [156], describes Warren Buffett as an anomaly

[1]How much risk to take for a given return is a subject of study within financial engineering first laid out by Markowitz in 1952 [159]; this provides a quantitative framework using probability theory along with statistics [69]. This is beyond the scope of this dissertation.

– a "3-sigma event." Burton Malkiel in [158] writes "In any activity in which large number of people are engaged, although the average is likely to predominate, the unexpected is bound to happen. The very small number of really good performers we find in the investment management business actually is not at all inconsistent with the laws of chance."

As described in [34, 183][2], if everyone in the United States flipped a coin every day, after 25 days, it would be expected that nine individuals would have flipped continuous heads[3]. Instead of flipping a coin, the argument could be transformed into *outperforming the market* (or benchmark). In the case of Warren Buffet, he outperformed the market (Standard and Poor's 500 benchmark) 39 out of 48 years [226]; as a binomial probability with a 50% likelihood of outperforming the market, this probability would be $\sum_{n=39}^{48} \binom{48}{n} 0.50^n \left(1 - 0.50\right)^{48-n} = 0.0000076 = 7.6 \times 10^{-6}$ of an individual performing *at least* as well. Reverting back to the coin example, if everyone in the United States flipped a coin for 48 days, 2283 individuals would have landed on heads at least 39 times; while rare, some individuals will simply succeed due to chance.

Buffett [34] counters the coin flipping argument by insisting that if a large number of the 2283 individuals who got heads at least 39 out of 48 flips came from a particular region (which he calls *Graham-and-Doddsville*), then statistical indepen-

[2]We updated the experiment from the original 1984 paper by researching Warren Buffett's returns via his investment firm Berkshire Hathaway Inc and updating the population of the United States to 2013 levels.

[3]The probability would be $0.50^{25} = 2.98 \times 10^{-8}$ and this would be multiplied by 300 million

dence could not be assumed and therefore there must be something noteworthy about their investment style (or in the example, coin flipping style). Buffett attributes his success though to thoughtful analysis of companies and their stock price (i.e. buying companies whose value is more than the price the market gives the stock at the time). Research in [160, 183] analyzing his investment style give credibility to this.

A distinction should be made here between investing and trading approaches. Buffett makes his money through investing, which is the process of building wealth over an extended period of time by buying and holding stock in profitable companies. Trading is the process of buying and selling stocks more frequently, often by examining irregularities in price. Since trades are done more frequently, small profits of a few pennies per share can amount to significant sums over time. The point here is that investing and trading are based on the assumption of market inefficiencies; predictability is possible and not purely random. While this thesis does not make claims of model profitability or that all stocks are predictable at all times or during all intervals, having an idea about the future market direction in the short-term (e.g. a few minutes in the future) can lead to significant sums of money.

2.3 Market inefficiency with conditional probabilities

2.3.1 Demonstrating market inefficiency

Andrew Lo tells a joke in [152]:

An economist strolls down the street with a companion. They come upon a $100 bill lying on the ground, and as the companion reaches down to pick it up, the

economist says, 'Don't bother – if it were a genuine $100 bill, someone would have already picked it up.'

While this is an exaggeration of reasoning, many of the followers of the Efficient Market Hypothesis hold their view that the market is efficient no matter the evidence to the contrary. This is still a hotly contested topic within finance and economics, split between those who believe the market has moments of predictability versus those who do not. This is an important topic for this thesis since efficiency would make stock prediction futile and our research would end here.

To demonstrate statistical efficiency (or inefficiency) within the equity market, conditional probabilities of upward versus downward market movements given prior price movement are examined. The binary representation of price movements, can be written as $Pr(\triangle p_t = \{\text{up,down}\}|\triangle p_{t-1} = \{\text{up,down}\}, \triangle p_{t-2} = \{\text{up,down}\}, \ldots, \triangle p_{t-m} = \{\text{up,down}\})$ where p is the price and $\triangle p_t = p_t - p_{t-1}$. Market independence would have us believe that $Pr(\triangle p_t = \text{up}|\triangle p_{t-1} = \text{down})$ should equal $Pr(\triangle p_t = \text{up})$. Upward movements are abbreviated as $(+)$ and downward movements as $(-)$; for example, the conditional probability of an upward movement given two previous downward movements is written as$Pr(+|-,-)$.

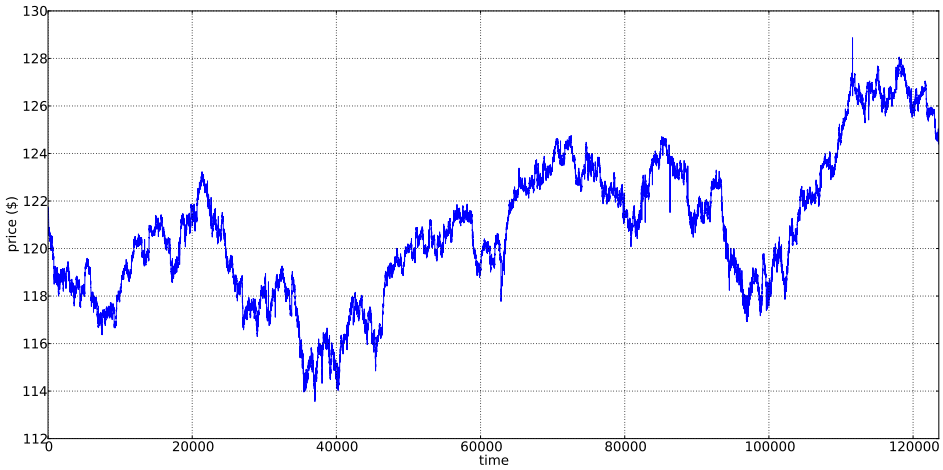
To illustrate this visually, we offer a brief comparison between random and actual data. Figure 2.1b shows the appearance of a downward trend, which appears simplistically predictable. However, this chart was created by randomly choosing, with equal probability, an upward or downward movement. Figure 2.1a is actual 1-minute intra-day data for the stock SPY over the period January 3 through December

30, 2005. The two charts appear remarkably the same in terms of existence of trends and potential predictability. But when the 2^m conditional probabilities for memory depth m are computed for both datasets displayed in Figure 2.1, the results are very different. For the random data there is, as expected, roughly 0.50 conditional probability that the market will go up given prior information. However, for the actual intra-day 1-minute data, a probability of an upward movement occurs with roughly 0.50 probability, but only 0.422 that the market will go up given a downward movement in price, $Pr(+|-)$. With entirely independent data, this should not be true.

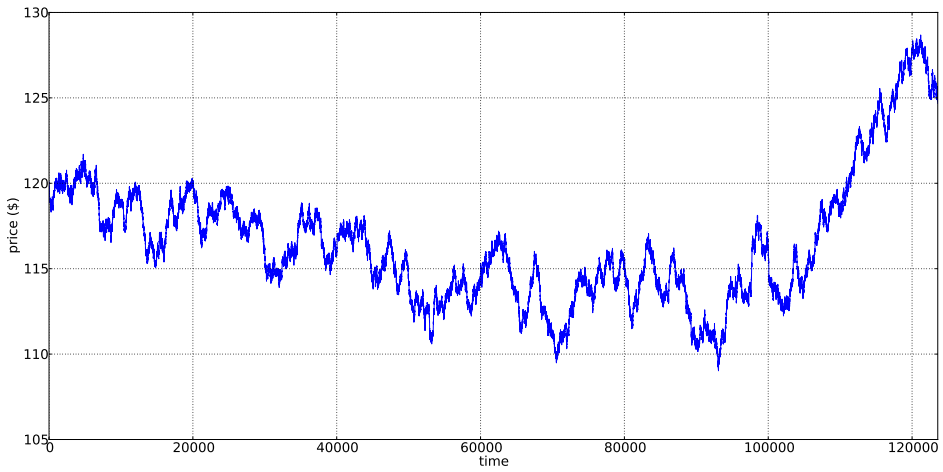
2.3.2 Existing research demonstrating predictability

The first paper that we are aware of that used probabilities to examine the existence of market trends and thus market inefficiencies in high-frequency data was Niederhoffer et al. [170]. The authors in that paper found that the stock examined had a higher probability to reverse directions from the previous price change than to continue in the same direction. Much of the existence of predictability was explained by traditional market dynamics, or the bouncing of the prices between the *bid* and *ask*, also called the *bid-ask spread*. Niederhoffer et al. calls this "the natural consequence of the mechanics of trading on the stock exchange."

The bid-ask spread is a small region of price that brackets the underlying value of the asset. The bid is the highest price an individual is willing to pay, and the ask is the lowest price an individual is willing to sell his or her stock at the moment. The



(a) SPY stock price per minute (January 3 - December 30, 2005



(b) Artificially created random stock price

Figure 2.1: Real versus random stock prices

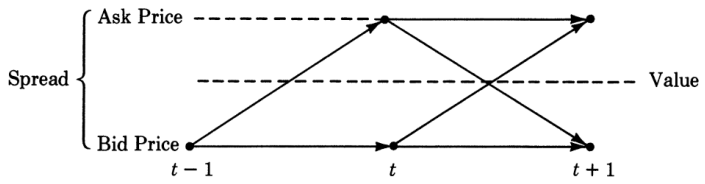


Figure 2.2: Bid-ask spread schematic [192]

"value" can be thought of as somewhere between the bid and ask (see Figure 2.2).

For example, let us consider that Participant 1 wants to buy 100 shares of stock at a price of $10.01; this is currently the best bid. Another participant, Participant 2, is attempting to sell his or her shares at a price of $10.02; this is currently the best ask. A trade does not take place until Participant 1 either pays Participant 2's ask of $10.02, or Participant 2 lowers his ask to Participant 1's bid of $10.01. Of course in an actual market, there are often hundreds or even thousands of participants at any given time who can participate in transactions. In an efficient market, the bid and ask fluctuate randomly [192].

Furthermore, in widely traded stocks with multiple active participants, there may be thousands of shares available at a bid or ask at any given time. In the short run, these orders act as a barrier to continued price movement in either direction. The larger the number of orders, or participants, at a given price level, the longer the price will stay constrained within a small price bound. Only after the bid or ask is eliminated, will the stock move to another price point [170].

Alexander [6] attempted to show predictability, and therefore the existence of trends with another approach by using quantitative rules based on prior price

history to create profits by buying and selling. If markets are random, zero profits would be expected over a baseline amount; however, if a model can be introduced that shows apparent profitability, then this opens the possibility of markets that do occasionally trend. According to Timmermann and Granger [216], the existence of a single successful trading model would be sufficient to demonstrate a violation of the market efficiency hypothesis. A number of empirical studies using daily data, such as Neely et al. [169], Chang and Osler [40], Levich and Thomas [144], and Sweeney [211] found profitability of trading rules in excess of the risks taken. The consensus of these papers is that the market is predictable, by way of trading rule profitability, at least part of the time.

Timmermann [215] however, found forecasting models using daily and longer interval data to predict stock returns mostly performed poorly. He did find some evidence of short-lived instances of predictability, thus requiring the examination of intra-day trading data. The theory is that if there are more instances of a particular high-probability pattern during a timespan, they will more likely be spotted by traders and implemented in their trading strategy. This widespread adoption of a particular trading approach drives the asset price either up or down enough to eliminate the pattern. Furthermore, while it is common for professional traders to use intra-day data, this short of a time horizon is often underrepresented in the academic literature.

Ohira et al. [171], Tanaka-Yamawaki [212], Sazuka [199], and Hashimoto et al. [100] examined market data at the lowest intra-day level available, trade-by-trade (sometimes known as tick data) and found extremely high levels of predictability. For

example, in [171] and [212] the authors report predictability as high as 79.7% and 75.0% respectively. While the movements are clearly predictable and raise doubt as to the efficiency of the currency market, we theorize here that much of the predictability in those two papers can be explained by the noisy continuation[4] of the bid-ask market dynamics[5].

To escape the noisy influence of bid-ask market dynamics, some researchers have sampled the market at even intervals such as 5, 10, and 60 minute intervals. A paper by Reboredo et al. [189] found profitability over a benchmark for 5, 10, 30, and 60 minute intervals of intra-day data using Markov switching, artificial neural networks and support vector machine regression models. Additionally Wang and Yang [225] found intra-day market inefficiency in the energy markets using 30 minute intra-day prices.

2.3.3 Our research demonstrating predictability

2.3.3.1 Introduction

Unlike [189, 225], our research demonstrates, in most cases, the market has gone back to efficiency and thus is unpredictable after a one-minute timespan. Furthermore, while we agree that predictability exists in the tick-by-tick market, we again

[4]Continuation is a term used by [170] and refers to the pattern where the signs of at least two non-zero consecutive changes are in the same direction. See also Section 2.3.3.4.

[5]While the currency-spot market is different from the equity market, such as the absence of a reported last trade/transaction, market dynamics still apply [17]. The large number of participants and lack of centralized reporting facility cause the bid and ask to fluctuate in the currency-spot market, similar to the last trade/transaction in the equity market bouncing between the bid and ask.

believe that the high levels of predictability found in [171, 212] have more to do with the bid-ask spread than the past price change. We empirically examine the conditional probabilities of trade-by-trade (tick) data along with nine temporal timespans of 1, 3, 5, 10, and 30 seconds and 1, 5, and 30 minutes for 52 separate one-week periods in 2005 of a popularly traded stock, the Standard and Poor's 500 index (symbol: SPY). By investigating the conditional probabilities we find that the market escapes the confines of the bid-ask spread after a 5 to 10 second timespan and find trends with seemingly high levels of predictability; trends have high occurrences of continuing rather than going against the trend, unless the trend is broken. Additionally, while the *bid-ask bounce* has been discussed in academic literature previously, we believe this is the first study of this size (data set includes 15 billion in share volume) and level of detail (number of intervals examined) that examines when a stock escapes the confines of the bid and ask spread.

2.3.3.2 Dataset and preprocessing steps

The stock that was used to examine conditional probabilities of upward versus downward price movements is one of the most widely traded stocks in the world, the Standard and Poor's 500 Index (symbol: SPY). It is an electronically traded fund (ETF) that holds all 500 Standard and Poor's stocks and is considered representative of the overall US market. The sheer number of transactions makes this an interesting stock to observe, and makes analysis easier given the need to examine longer-length series. The problem with sparseness, or the lack of transactions when

sampling at narrow time intervals, is minimized since volumes per day for SPY in 2005 averaged 63,186,191 $\pm$ 19,474,197; the average number of transactions per day was 91,981 $\pm$ 23,332. This large volume of transactions leads to a more efficient and unpredictable stock as greater number of participants are driving the stock to an equilibrium; findings of predictability would be especially noteworthy.

Trade-by-trade data was retrieved from Wharton Research Data Services for the period January 3, 2005 to December 31, 2005. As noted in [30, 70, 101, 140, 187], high-frequency trading data, such as the type used in this paper, requires special consideration. All late-trades, trades reported out-of-sequence, or trades with special settlement-conditions are excluded since their prices are not comparable to adjacent trades. The data was then reduced to temporal timespans of 1, 3, 5, 10, 20 and 30 second and 1, 5 and 30 minute data using a volume-weighted average price approach (VWAP). This is calculated using the following formula:

$$P_{\text{VWAP}} = \frac{\sum_j P_j V_j}{\sum_j V_j}$$

where j are the individual trades that take place over the period of time and P_j is the price and V_j is the volume of trade j. Using a volume-weighted average price allows for a more realistic analysis of price movements, rather than sampling the last reported execution during a specific timespan. In addition, half-trading days such as the day after Thanksgiving and before Independence day were eliminated.

Trades were next encoded as either upward (+) or downward (−) as compared against the previous transaction[6]. The data was split into one-week periods covering

[6]Similar to existing literature, non-movements were eliminated from the study.

all 52 weeks in 2005. One-week periods allow for enough instances (of prior information) of memory depth 5 (our longest depth) and as explained in [216], the existence of predictability in markets will eventually lead to their decline once those anomalies become "public knowledge." Traders who use forecasting models will bid up prices of stocks that are expected to rise, and sell off stocks that are expected to drop, thus eliminating their predictability. To prevent this elimination of possible predictability – the reversion back to randomness – we used 52 one-week timespans. From here, 2^m conditional probabilities for depth $m = 5$ are computed. The computation of separate weekly conditional probabilities allows us to analyze how the predictability changes over a weekly period.

Since the stock's previous day's closing price and the current day's opening price are often different, conditional probabilities for individual days are computed separately. This discontinuity is a distinct disadvantage of equity data over currency data, such as was used in [171, 212]. The worldwide nature of currency trading allows for the market to be continually open somewhere on Earth, except for weekends.

The conditional probabilities of directional movements for the timespans can be seen in Appendix A. The 30-minute timespan probabilities were not included in this paper because of the lack of data for all of the events, and as this paper demonstrates in the next section, predictability can not be assumed for the 30-minute timespan.

2.3.3.3 Experiment 1: Test of market independence

As explained previously, to test for market efficiency, the probability of a given market directional-movement at time t is compared against the directional-movement at time $t-1$. Under the assumption of efficiency, more prior information should not increase predictability. A violation of independence allows the possibility of predictable trends based on prior information.

Conditional probabilities of upward versus downward price movement, given prior price movement for each timespan, were computed separately for each of the 52 weeks in 2005. In addition, we calculated binomial 95% confidence intervals to determine the number of weeks that were statistically-significantly outside of the bounds of error. If prices followed a random walk, the probability of an upward movement given prior information would be expected to equal the probability of an upward movement for that particular timespan, while taking into consideration the 95% confidence intervals. The week's probability was determined to be statistically significant if the 95% confidence interval's lower bound is above, or upper bound is below, the probability of the same directional movement. For example, the probability of an upward movement in price, given prior information ($Pr(+|\{+,-\}\dots\{+,-\})$), is significantly greater or less than the probability of an upward movement in price ($Pr(+)$).

In Figure 2.3, the probability of an upward movement is plotted for each of the timespans, which is roughly 50% probable that the market will go up. The variance of probabilities increases as the time between spans increases due to the

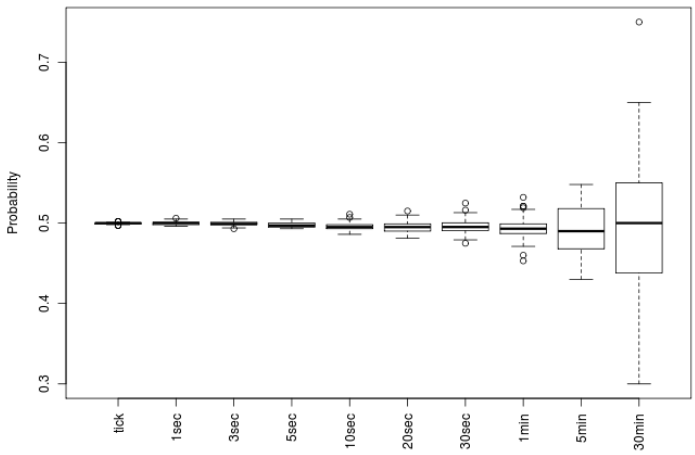
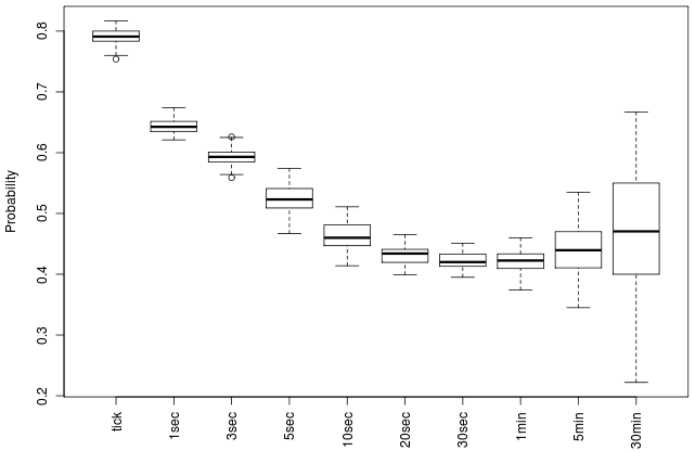


Figure 2.3: Boxplot of Pr(+) for different timespans aggregated over 52 separate weeks

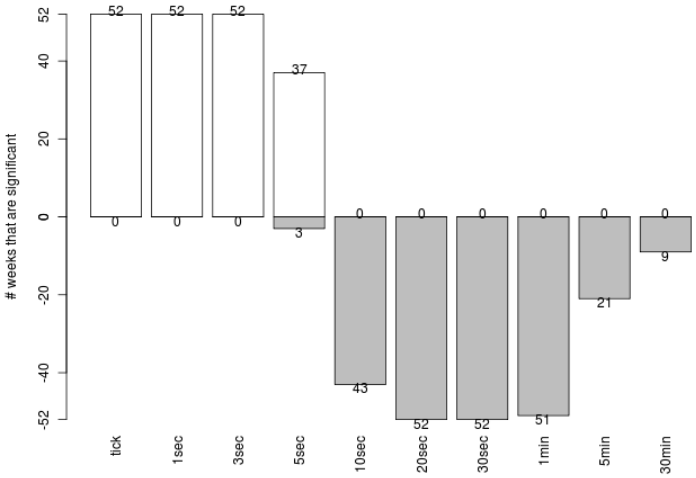
decrease in data points. Figure 2.4a displays the probability of an upward movement given a downward movement, $Pr(+|-)$. The trade-by-trade (tick) data shows the highest predictability given prior information with a 79.0% conditional probability of an upward movement given a downward movement. At the 30 minute timespan, there is only a 46.2% probability that the market will move up given a previous downward movement. This can be subtracted from 1 to get the probability of a downward given a previous downward movement, $1 - Pr(+|-) = Pr(-|-)$. The probabilities over the 52 weeks for the 30 minute timespan range from a high of 66.7% to a low of 22.2%[7].

The number of weeks out of 52 that are statistically significant can be seen in Figure 2.4b. From this chart it can be seen that all 52 weeks of the tick data were statistically significantly above $Pr(+)$, while with 5 second data a total of 40

[7]For a summary of the conditional probabilities over each of the different timespans, excluding the 30 minute timespan, please see Appendix A.



(a) Pr(+|−) for different timespans



(b) The number of weeks that are statistically significant for the corresponding timespan where the 95% confidence interval's lower bound and upper bound are above or below the appropriate Pr(+) respectively

Figure 2.4: Boxplot of Pr(+|−) for different timespans, along with the number of significant weeks

weeks were statistically-significant, with 37 weeks significantly above $Pr(+)$ and 3 weeks below$Pr(+)$. For the 30 minute timespan, only 9 weeks out of the 52 weeks are statistically significantly below that timespan's $Pr(+)$.

In addition, we use an independent samples t-test with a 95% confidence interval to test the hypothesis below for each of the timespans over the 52 week period:

$H_0 : Pr(+) = Pr(+|-)$; accept independence.

$H_1 : Pr(+) \neq Pr(+|-)$; reject independence.

In testing the hypothesis for independence, the null hypothesis is rejected for all but the 30 minute timespan (see Table 2.1). Therefore, in this statistical hypothesis testing, we can reject the independence assumption for the trade-by-trade, 1, 3, 5, 10, 20, 30 second, 1 and 5 minute timespans; the independence still holds for the 30 minute data. We conclude that the market is inefficient, and prior information impacts price movement, until approximately the 30 minute period at which time the market begins to become more efficient.

As previously explained, much of the predictability of, at least, the trade-by-trade data can be explained by the bouncing of the price between the bid and ask. In our examined stock, the average trade size is roughly 700 shares and the median trade is 100 shares; with bid and ask sizes of 5000+ shares per side common, the transaction prices will fluctuate up and down between the bid and the ask until all shares are depleted. The question remains as to when the high levels of predictability cease to be explained by the market dynamics. The next experiment explores this

Table 2.1: Results from t-test for the different timespans and assuming unequal variances

Timespan	t value	p value
tick	-152.67	< 0.0001
1 second	-84.13	< 0.0001
3 second	-42.63	< 0.0001
5 second	-7.66	< 0.0001
10 second	10.32	< 0.0001
20 second	28.18	< 0.0001
30 second	31.89	< 0.0001
1 minute	21.30	< 0.0001
5 minute	7.38	< 0.0001
30 minute	1.68	0.0968

question further.

2.3.3.4 Experiment 2: Escaping the bid/ask spread

Much of the predictability of trade-by-trade (tick) intra-day data can be explained by market dynamics; the price fluctuating between the bid and ask. As explained previously, a *bid* is the best price at which an individual is willing to buy, while an *ask* is the best price at which one is willing to sell. When a stock has a large number of participants placing orders at the same price, but the average transaction size executing against the bid or ask is smaller, it takes time before the stock's bid or ask is eliminated and the stock is allowed to move to the next price point.

Using the same terminology as [170], when the signs of two non-zero consecutive changes are unlike each other, this pattern will be called a *reversal*, and when they are in the same direction, the pattern will be called a *continuation*. Re-examining Figures 2.4a and 2.4b, it can be observed that trade-by-trade data, up to a temporal

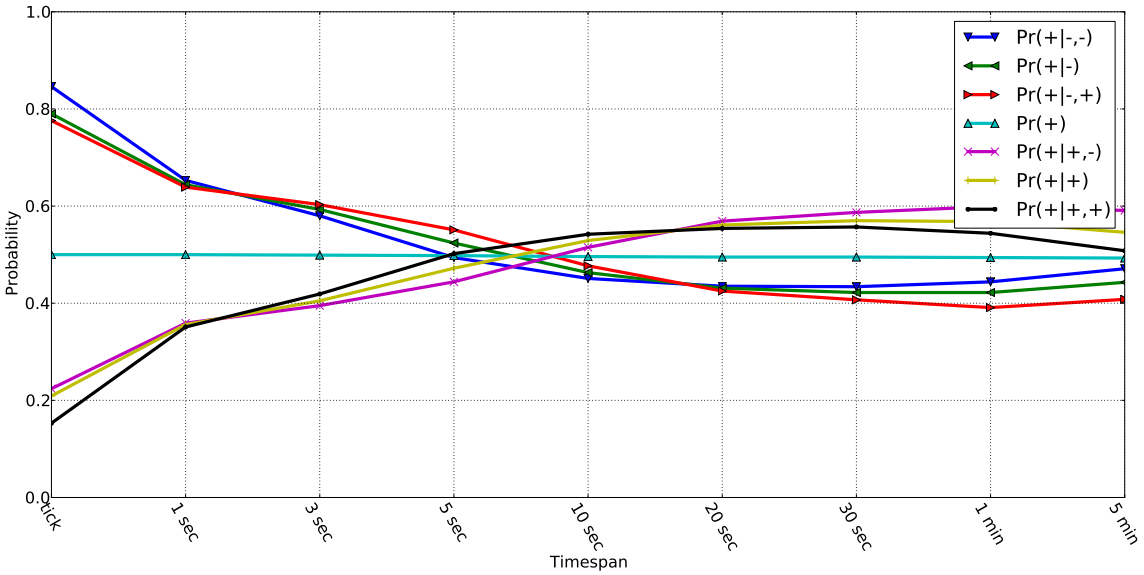


Figure 2.5: Mean conditional probabilities of depth 2 for different timespans. Until 5 to 10 seconds, predictability is higher for *reversals* of trends, after which *continuation* of trend is higher.

timespan of 5 seconds, has a higher probability of reversal, rather than a trend continuation. After 10 seconds, the market has a higher probability of continuation than reversal. This can also be observed in Figure 2.5, where the probabilities for different conditional probabilities up to depth 2 are plotted to show how the probabilities change with the increase in time between data points. Thirty-minute intervals were not included because of the lack of independence. As seen from the chart, market reversals ($Pr(+|-)$,$Pr(+|-,-)$, $Pr(+|-,+)$) occur with a greater likelihood than do continuations ($Pr(+|+)$, $Pr(+|+,+)$, $Pr(+|+,-)$) until 5 to 10 seconds. After this period, continuations occur with greater probability than reversals. While the variance increases as the interval between timespans becomes larger, the number of statistically significant weeks remain high and stable over the 52 weeks until a one to

Table 2.2: Comparing the conditional probabilities of directional movements for reversals versus continuations – brackets are the standard deviations

	Event	Probability		
		tick	5 second	20 second
Reversals	$Pr(+\|-)$	0.79 [0.01]	0.52 [0.02]	0.43 [0.02]
	$Pr(+\|-,-)$	0.85 [0.02]	0.49 [0.03]	0.44 [0.01]
	$Pr(+\|-,-,-)$	0.85 [0.03]	0.48 [0.03]	0.44 [0.02]
	$Pr(+\|-,-,-,-)$	0.84 [0.04]	0.47 [0.03]	0.44 [0.02]
	$Pr(+\|-,-,-,-,-)$	0.84 [0.07]	0.46 [0.03]	0.44 [0.03]
Continuations	$Pr(+\|+)$	0.21 [0.01]	0.47 [0.02]	0.56 [0.01]
	$Pr(+\|+,+)$	0.15 [0.02]	0.50 [0.03]	0.55 [0.01]
	$Pr(+\|+,+,+)$	0.15 [0.03]	0.52 [0.02]	0.56 [0.02]
	$Pr(+\|+,+,+,+)$	0.17 [0.03]	0.53 [0.03]	0.56 [0.02]
	$Pr(+\|+,+,+,+,+)$	0.18 [0.08]	0.53 [0.03]	0.55 [0.03]

five minute timespan.

Table 2.2 displays the probability of continuations and reversals for trade-by-trade, along with a 5 second and 20 second timespan. Reversals occur with higher probabilities for trade-by-trade data. For example, the $Pr(+|-)$ occurs with probability 0.79 and $Pr(+|+)$ occurs with probability of 0.21 which infers a reversal of $1 - Pr(+|-) = Pr(-|+) = 0.79$. By the 20 second timespan, the reversal $Pr(+|-)$ occurs with probability 0.43, which infers a continuation of two downward movements of $1 - Pr(+|-) = Pr(-|-) = 0.57$.

These probabilities are not fleeting. Figure 2.6 shows the trend continuation events using 30 second interval data (with added 95% confidence intervals) by month. All twelve months are statistically significant from a probability of an upward movement during the same period. This further demonstrates the stability of events using high-frequency data.

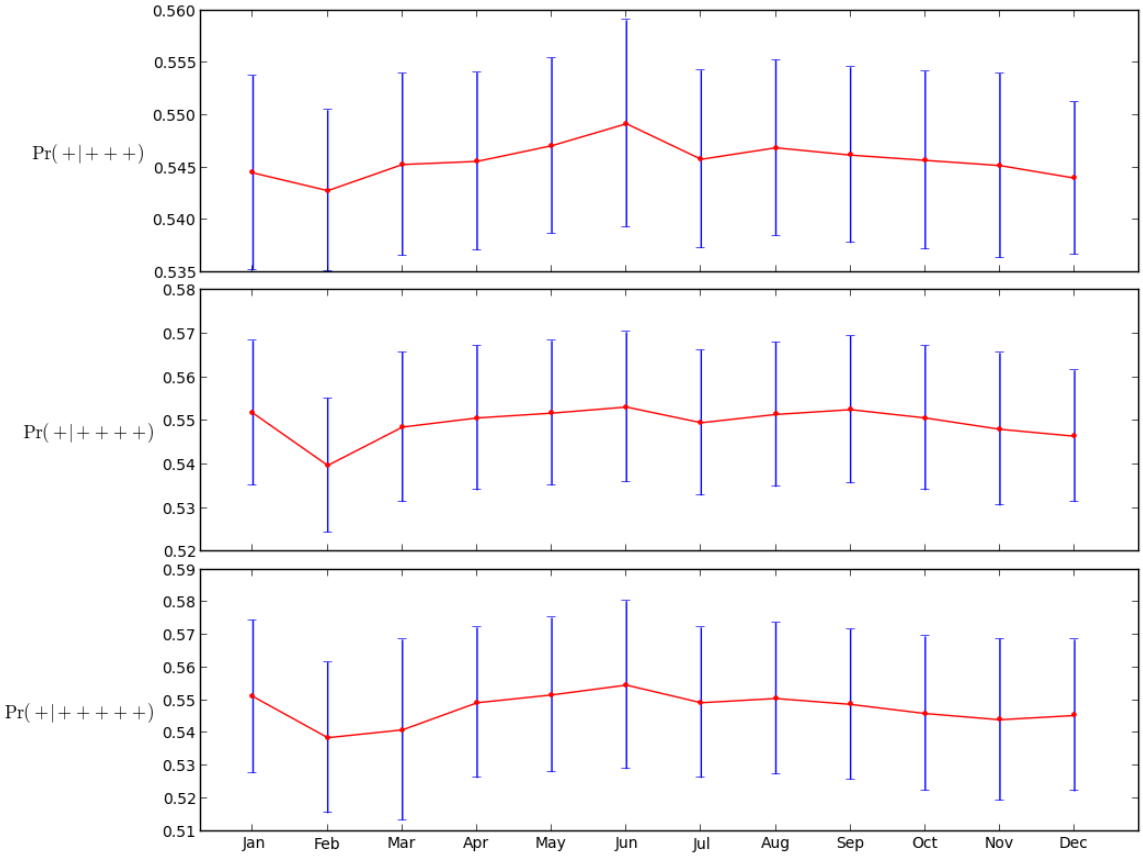


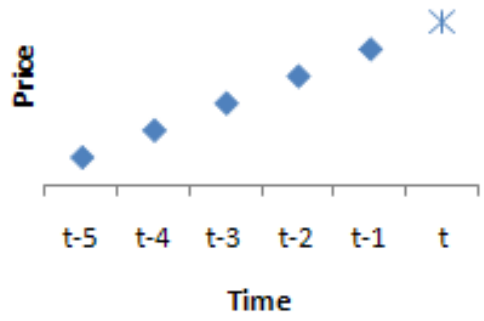
Figure 2.6: Examining monthly stability of events using 30 second interval data

We theorize a 5 to 10 second timespan is the average length of time that the market price breaks the confines of the bid and ask and can move freely outside of these bounds. This observed reversal of directional movements before 5 seconds reflects the price being trapped between the bid and ask. While these movements are clearly predictable, they do not represent actual market changes, merely bounces of price between the bid and ask.

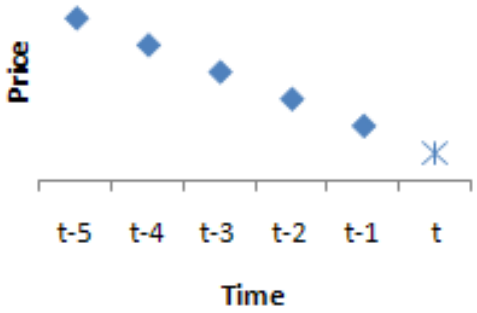
2.3.3.5 Trends with high apparent predictability

An interesting pattern is observed in the data after a 10 second timespan, which requires further analysis. Among some of the highest probabilities observed with the strongest significance over the 52 weeks, is what we call the trend *reversion-to-mean* (Sub-Figures 2.7c and 2.7d). This can be described as the market trending in one direction, followed by an abrupt change in the opposite direction. The probability that the next market directional movement will move in the same direction as the last movement is higher.

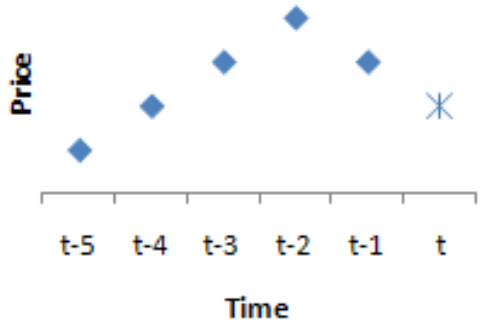
By comparing the trend reversion-to-mean to the trend continuation (Sub-Figures 2.7a and 2.7b), it can be observed that the probability of trend reversion-to-mean occurs with a greater number of *statistically significant number of weeks* (see Table 2.3). For example, in a 30 second temporal timespan, $Pr(+|+,-,-,-)$ occurs with a probability of 59.3% and is statistically significant compared to the probability of an uptick ($Pr(+)$), 42 out of 52 weeks. We compare this to the probability of $Pr(+|+,+,+,+)$, which occurs with a probability of 55.3%, but is only statistically significant 26 out of 52 weeks. Furthermore, the reversion-to-mean probabilities are larger and occur with a greater number of statistical weeks than continuations of the same depth. This pattern occurs in the 20 second, 30 second, 1 minute, and 5 minute timespans.



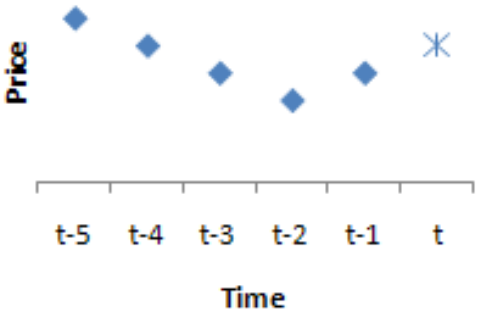
(a) Continuation: $\Pr(+|+,\ldots,+)$



(b) Continuation: $\Pr(-|-,\ldots,-)$



(c) Reversion-to-mean: $\Pr(-|-,+,\ldots,+)$



(d) Reversion-to-mean: $\Pr(+|+,-,\ldots,-)$

Figure 2.7: Examples of high-probability events: the trend continuation and the trend reversion-to-mean

Table 2.3: Comparing the probabilities and the level of significance for reversion-to-mean and trend continuations for a 30 second timespan

	Event	Prob.	# weeks that are stat. sig. from $Pr(+)$
Reversion-to-mean	$Pr(+\|+,-,-)$	0.587[0.024]	49
	$Pr(+\|+,-,-,-)$	0.593[0.034]	42
	$Pr(+\|+,-,-,-,-)$	0.599[0.039]	34
	$Pr(-\|-,+,+)$	0.598[0.026]	50
	$Pr(-\|-,+,+,+)$	0.605[0.029]	49
	$Pr(-\|-,+,+,+,+)$	0.613[0.040]	41
Continuation	$Pr(+\|+,+,+)$	0.556[0.021]	42
	$Pr(+\|+,+,+,+)$	0.553[0.028]	26
	$Pr(+\|+,+,+,+,+)$	0.546[0.046	15
	$Pr(-\|-,-,-)$	0.563[0.021]	45
	$Pr(-\|-,-,-,-)$	0.56[0.029]	30
	$Pr(-\|-,-,-,-,-)$	0.559[0.034]	19

2.3.4 Conclusion

Market inefficiency was examined empirically by analyzing trade-by-trade data at nine timespans. While statistically significant levels of predictability were found, we theorize that the predictability for timespans prior to 5 to 10 seconds is due to traditional market dynamics of prices fluctuating between the stock's bid and ask. By examining the stock's probability of upward movements ($Pr(+)$) versus upward given downward movements ($Pr(+|-)$), it was found that prior to a 5 to 10 second timespan the probabilities of reversal movements occurred with higher probability than continuation of price movements. After 5 to 10 second timespans, continuation of price movements became more probable than reversals. We theorize this to be the point at which the stock examined was escaping the confines of the bid and ask.

The probabilities of market reversions-to-mean were statistically higher than

probabilities of continuations of the same depth; this occurred in 20 second, 30 second, 1 minute, and 5 minute temporal timespans. We also observed higher numbers of statistically significant weeks of market reversions-to-mean as compared to the number of statistically significant weeks of market continuations for the same depth. This suggests that the market is being pulled back to the equilibrium price. The information presented here would be useful for traders when deciding to exit or hold a position; if the probability is higher for the market to reverse directions than continue, a trader may decide to close the position, whereas with a higher probability for the market to continue in the same direction, the trader may decide to hold the position longer.

Given that the chosen stock (symbol: SPY) is an exchange-traded fund which is comprised of the 500 stocks within the Standard and Poor's 500 index, the results are especially noteworthy. This stock, being one of the most widely traded stocks in the world, would suggest that all inefficiencies were spotted by others and implemented in their trading approach. This widespread adoption of the high-probability events would drive the asset price either up or down enough to eliminate the pattern. However, this is not what was found; stable, high-probability market movement, were still found for this popular stock.

Further research would be necessary to determine the timespans at which other stocks become inefficient/efficient. While the 30 minute market is the timespan at which the examined stock became efficient, surely this would be different for each stock, for stocks have different levels of trading activity and levels of participation.

This continued study would be necessary to understand in order to implement an ideal trading model that takes advantage of inefficient markets and enable traders to make better trading decisions.

While many in mainstream finance have reservations of market inefficiency, or the existence of trends, Malkiel states that there may be a possible explanation of why trends might perpetuate themselves [158]:

...it has been argued that the crowd instinct of mass psychology makes it so. When investors see the price of a speculative favorite going higher and higher, they want to jump on the bandwagon and join the rise. Indeed, the price rise itself helps fuel the enthusiasm in a self-fulfilling prophecy. Each rise in price just whets the appetite and make investors expect a further rise.

Whereas the efficient market hypothesis expects traders and investors to act rationally and make the best decisions, behavioral economists argue that in the short-run people do not make rational decisions to maximize profit. Humans are susceptible to acting irrationally and making poor decisions when faced with greed [41]. Grossman and Stiglitz [93] argue that perfectly efficient markets are impossible; if markets are efficient then there would be little reason to trade and markets would collapse. These behavioral economists instead concentrate on the consequences of irrational actions.

Why is market inefficiency so contested? Timmermann and Granger [216] write of a possible "file drawer" bias in published studies due to the difficulty in publishing empirical results that are often barely or border-line statistically insignifi-

cant; but markets, since they are partially driven by human emotions, involve a large degree of error. This may result in a glut of research arguing that the market is efficient, and thus unpredictable; Timmermann and Granger calls this the "reverse file-drawer" bias. Additionally, it is possible that it is not so much *if* a profitable system can be created, it is if a researcher has enough incentive to publish a method in an academic journal. An investment bank or other for-profit company would offer a greater incentive – money. It is also possible that many traditional forms of stock market prediction are simply inadequate or sponsoring companies may not wish to divulge successful applications [245].

2.4 Methods of predicting

2.4.1 Introduction

In the previous section we raised doubts as to the efficiency of the market and demonstrate that during specific market movements high levels of market predictability exist. For those who believe that markets are predictable, there are two approaches for predicting stock prices and therefore stock direction; these are the "fundamental" and the "technical" approaches with practitioners typically called *fundamentalists* and *technicians* respectively. The fundamentalist looks at the external and economic factors to determine price change. The belief is that since stocks are shares of a corporation, examining the fundamental indicators such as profits, sales, debt levels, and dividends should provide an outlook into the future direction of the price. The technician believes that the past performance of stocks can help forecast future price

movements. He studies historical prices to try to understand the psychology of other market participants (the crowds). The technician attempts to identify regularities in the time-series of price or volume information; the thought is that price patterns move in trends, and that these patterns often repeat themselves [6, 158]. We have already seen one example of quantitative technical analysis using conditional probabilities to examine market movements in Section 2.3. A further explanation follows.

2.4.2 Fundamental and technical *type* approaches

2.4.2.1 Fundamental analysis

As explained previously, the fundamental approach examines the economic factors that drive the price of stock with the aim to reveal the intrinsic (fundamental) value of the company [229]. For example companies release financial information four times per year, with three quarterly 10-Q statements and a final year end 10-K statement. Contained in these reports are the profits, sales, and debt levels which can help to determine valuation of the business. The idea seems reasonable considering stocks are shares of businesses and research [8, 36, 37, 51, 75, 188, 218] seems to give credibility to this approach. The problem, as can be seen in Figure 2.8, is these financial statements are released periodically thus they are of little use when explaining the intra-day prices such as those plotted in Figure 2.9[8]. In the first plot the stock Piedmont Natural Gas (symbol: PNY) is shown over the course of 2012 and pinpoints days where financial statements are released to the public. It does appear

[8]A slight price discrepancy exist between this and Figure 2.8. In Figure 2.8 the price is adjusted for the stock dividend, whereas in Figure 2.9 the price excludes this adjustment.

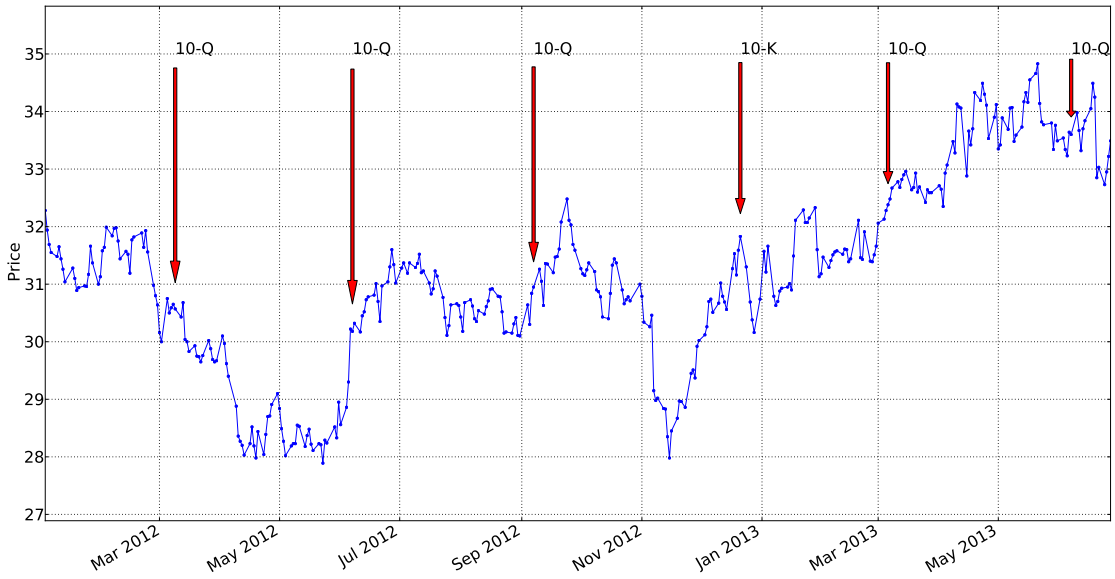


Figure 2.8: Examining the daily stock price of Piedmont Natural Gas (symbol: PNY); arrows mark the dates that 10-Q (quarterly financial statements) and 10-K (annual financial statements) are released

that leading up to, and after, many of these releases, the stock has large price swings of volatility. This does not explain the intra-day movements shown in the second plot. On that day of January 23, 2012 no financial information was released, yet the stock moved erratically over the course of the day.

It could be argued that the stock Piedmont Natural Gas moved during the course of the day because the price of natural gas/oil or interest rates or other stocks moved erratically during this time. How though does this explain the drop of 3% of Piedmont Natural Gas after the opening of the market after the terrorist attack on September 11, 2001; a company that does business primarily in the Carolinas and Tennessee? Or the 8% drop of Microsoft, a company that sells software internationally? Examining fundamentals (i.e. balance sheet, income statements, or any

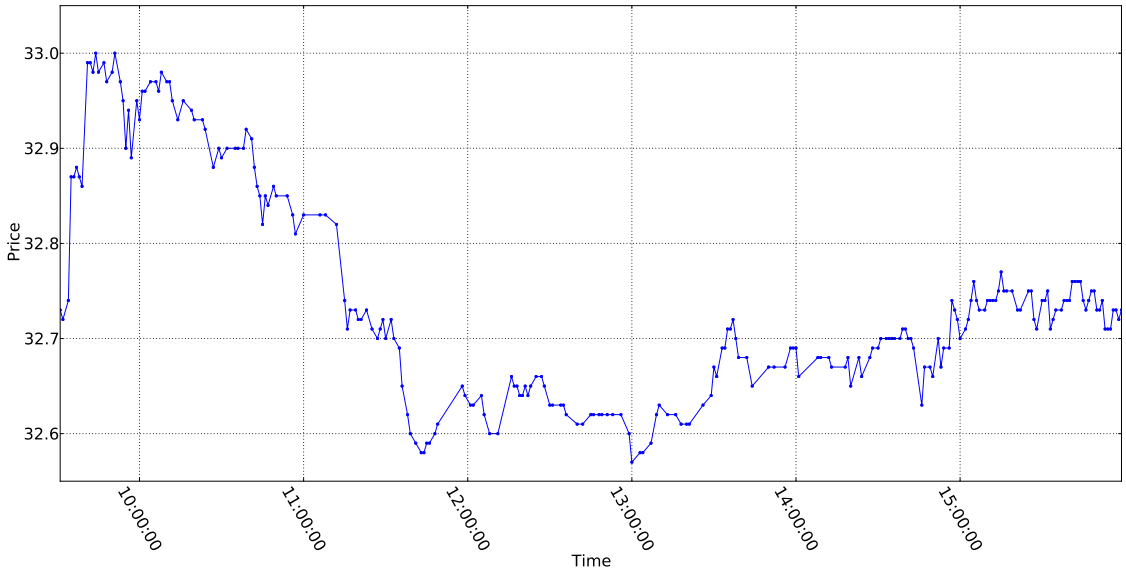


Figure 2.9: The intra-day stock price of Piedmont Natural Gas (symbol: PNY) for January 23, 2012

news that has a direct or indirect bearing on the economy) would offer little to explain the drop. Behavioral economists would explain intra-day movements in terms of psychology (how people think) and would blame this on both fear and greed. According to [153] these are the two most common culprits in the downfall of rational thinking. Fundamental analysis though has a difficult time explaining these short term movements and for this reason the fundamental approach is typically used when predicting prices months and years in the future. This thesis goal is to predict stock direction seconds and minutes into the future, thus another approach is needed, and this approach is technical analysis.

2.4.2.2 Technical analysis

Technicians (those who follow technical analysis) attempt to anticipate what *others* are thinking based on the price and volume of the stock (see Figure 2.10). Indicators are computed from past history of the prices and volumes and these are used as signals to foresee future changes in prices. The core concept behind technical analysis is the idea of a *trend*, with future prices more likely to continue in the direction of the current trend than to reverse. Our experiments in Section 2.3 give credibility to this idea in the short term. Technicians argue the trend is caused by an imbalance of stock between supply and demand, with demand for the stock causing the increase in the stock price [25]. Instead of interpreting the *fundamentals* of a news story, the technician examines how the *market* reacts to the news. If the stock fails to react to the news, it may be that the news is incorrect or does not accurately reflect the underlying supply and demand [5].

One suggestion of why technical analysis may work is that followers often create a self-fulfilling argument; traders, not wanting to miss the price increase, buy in anticipation of the indicator, thereby helping to fuel the enthusiasm for the stock and pushing it to higher levels [158]. This can work in the other direction as well, traders, not wanting to miss the price decrease, sell in anticipation of the indicator, thereby helping to fuel pessimism for the stock and pushing it to lower levels.

An example of a popular indicator that appears to be based more on a self-fulfilling concept than on any specific reason is the head-and-shoulders indicator seen in Figure 2.11. The characteristic of this pattern as described in [227] is:

Figure 2.10: The stock Exxon on January 3, 2012 with the high, low and closing prices shown on the top plot and the transaction volumes shown on the bottom plot

1. The "head" should be significantly taller than the "shoulder"
2. The top and bottom of the shoulders should be of roughly equal height
3. The spacing between the shoulders and head should be roughly symmetric

When the price intersects the "neckline", which is a straight line connecting the bottom of the left and right shoulders, the trader should *sell short*. Research by Weller et al. [227] using intra-day tick-by-tick data on all stocks in the S&P 100 index from 1999 to 2005, find statistically significant large movements consistent with the predictions.

The use of technical analysis remain high, with studies showing adherence among professionals and individual investors of up to 80% to 90% [157, 162, 163, 213]. As described earlier, a skew toward technical analysis exists in the literature when

Figure 2.11: Example of a head-and-shoulders technical analysis indicator

considering shorter time horizons partially because fundamental analysis, such as financial income statements, do not explain all of the price moves during the trading day. Popular books on technical analysis include [1, 64, 167, 168].

2.4.2.3 Quantitative technical analysis

The goal of technical analysis is to identify regularities by extracting patterns from noisy data. According to practitioners, specific patterns are empirically found to be forebearers of either upward or downward moves in the market. Prior to widespread use of computers these patterns were uncovered visually. With recent breakthroughs in technology and algorithms, technical analysis has morphed into a more quantitative and statistical approach [154]. We differentiate between technical analysis and call

this quantitative technical analysis. Whereas traditional technical analysis is *visual*, and often full of "vague rules", quantitative technical analysis is *numerical*. This allows us to easily program the rules into a computer.

An example of a quantitative technical indicator is the Bollinger Band, which was named after John Bollinger, who popularized it. It is comprised of a middle, upper, and lower band. The middle band is a moving average of size n of the high, low, and closing transaction prices. The upper band is d standard deviations (generally two) above the middle band, and the lower is d standard deviations below. It is formalized below with t representing time:

$$\begin{aligned} \text{MiddleBand}(n) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\text{high}_{t-i}+\text{low}_{t-i}+\text{close}_{t-i}}{3} \\ \text{UpperBand}(d,n) &= \text{MidBand}(n)+(d*\sigma) \\ \text{LowerBand}(d,n) &= \text{MidBand}(n)-(d*\sigma) \end{aligned} \tag{2.1}$$

The Bollinger Band is an indicator of oversold (i.e. inexpensive, therefore buy the stock) and overbought (i.e. expensive, therefore sell the stock) conditions. When the price is near the upper or lower bands, it indicates that a reversal is imminent. This is visualized in Figure 2.12 with oversold conditions marked with a green triangle and overbought conditions marked with a red upside down triangle.

Our last example is the Moving Average Convergence Divergence Oscillator

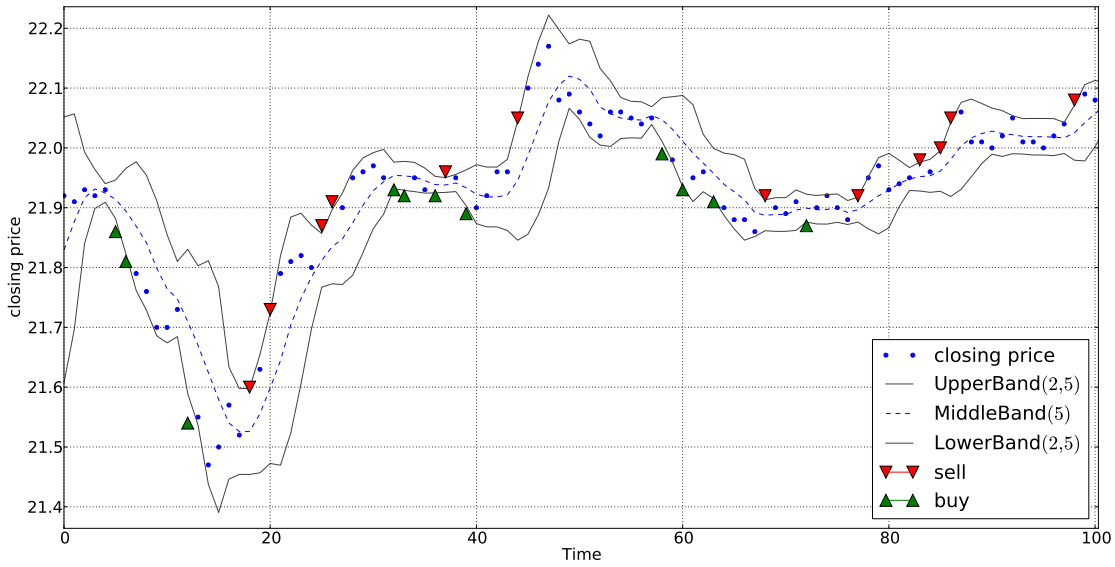


Figure 2.12: Example of using Bollinger Bands as an quantitative technical analysis indicator

(MACD) which was developed by Gerald Appel with the formalization following:

$$\begin{aligned}
\text{ShortEMA}(n_1) &= \text{exponential moving average of size } n_1 \\
\text{LongEMA}(n_2) &= \text{exponential moving average of size } n_2 \\
\text{MACD}(n_1, n_2) &= \text{ShortEMA}(n_1)_t - \text{LongEMA}(n_2)_t \\
\text{SignalLine}(n_1, n_2, n_3) &= \text{exponential moving average of MACD}(n_1, n_2) \text{ of size } n_3 \\
\text{DiffMACDSignal}(n_1, n_2, n_3) &= \text{MACD}(n_1, n_2)_t - \text{SignalLine}(n_1, n_2, n_3)_t
\end{aligned}$$

The MACD itself is the difference between a short and long-term exponential moving average and it generates overbought and oversold positions when an exponential moving average of the MACD goes above or below zero (i.e. when the DiffMACDSignal(n_1, n_2, n_3) becomes positive or negative). This is represented graphically in Figure 2.13; the bottom plot visualizes the DiffMACDSignal(n_1, n_2, n_3) which

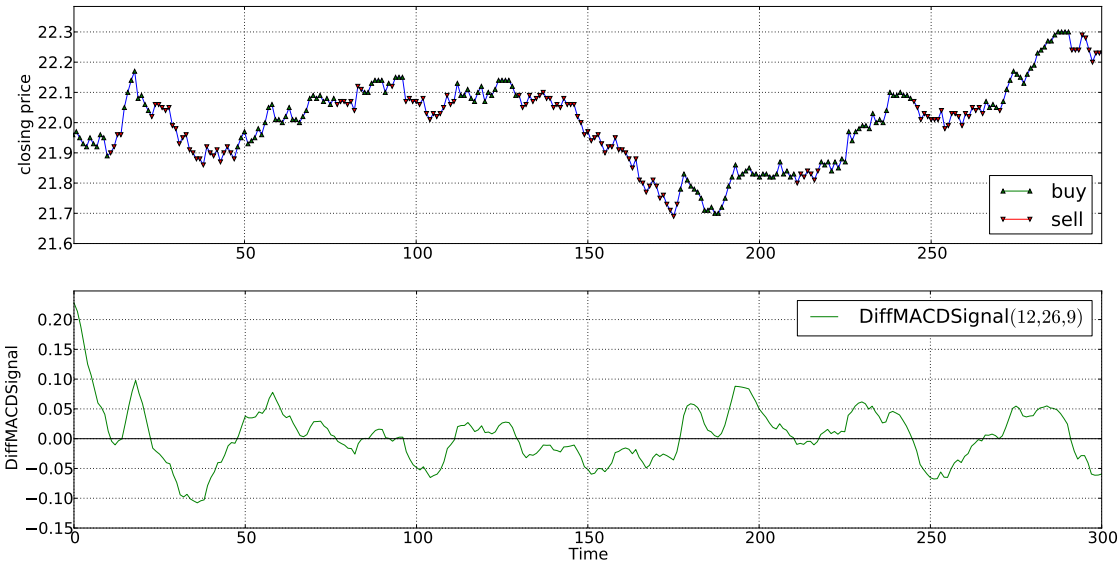


Figure 2.13: Example of using Moving Average Convergence Divergence Oscillator (MACD) as an quantitative technical analysis indicator

creates overbought signals in the top plot when it becomes positive and oversold signals in the top plot when it becomes negative.

Additional examples of over 20 groups of quantitative technical analysis indicators used in this thesis are given in Appendix C.

Research [154, 178, 200, 229] has shown technical analysis to be useful to predict stock direction. In [107] and [218] the authors use both fundamental and technical analysis along with a dimensionality reduced classifiers (a support vector machine and an artificial neural network, respectively) to predict stock prices with good results. Shen et al. [206] uses an RBF neural network optimized heuristically and obtained good results. In addition, Lai et al. [135] uses technical indicators with decision trees to predict the Taiwanese market with ideal results. Chen et al. [48] used Fibonacci numbers, which are commonly used in technical analysis, along with

neural networks and achieved moderate success. Ghandar et al. [90] used technical indicators as attributes in a fuzzy logic system and outperformed in profitability a baseline model. However, all of these papers used daily data. Reboredo et al. [189] uses 5, 10, 30 and 60 minute data and using artificial neural networks found slight predictability in the Standard and Poor's 500 Index (S&P 500). Lippi et al. [148] produced a profitable model using technical analysis with a support vector machine and intra-day data. Wang and Yang [225] also used technical indicators and found intra-day inefficiency within the heating and natural gas markets.

None of these papers however specify if they are able to work in real-time settings and according to [189] there remains a lack of published research using high-frequency data. As noted previously the lack of published results may simply be that existing methods are inadequate or sponsoring companies may not wish to divulge successful applications [245]. This is the goal of this thesis – exploring the use of modern streaming algorithms, such as adaptive and wrapper methods, for use in predicting high-frequency stock prices.

2.5 Conclusion

An efficient market implies that the market can never be consistently beat or predicted and although high rates of return may be achieved, they are on average, proportional to risk. Furthermore a proponent of the Efficient Market Hypothesis would believe research predicting market direction, such as this thesis, is futile.

Our research examined conditional probabilities of stock direction in a popular

stock and found moments of predictability not explained by traditional market dynamics. These high probability events disappeared after several minutes. This gave us confidence to examine stock price predictability further. Additionally we were careful not to make claims that this predictability existed in all stocks or even that this could lead to profitability. It is simply an exploration of and declaration of its existance in some cases.

In Section 2.4 two popular approaches to predicting stock prices were examined – the fundamental and technical. Fundamental approaches have been shown to be useful in predicting stock prices through the use of balance sheets and financial income statements, but this is for longer outlooks (e.g. weeks or months). This provides little help for our problem of predicting higher-frequency events.

The second approach is the technical which examines prior stock prices and volumes as a signal to foresee future changes in prices. This technical approach, with the use of modern machine learning algorithms, is the method we use in this paper and is largely ignored in academic research. The next chapter introduces machine learning.

CHAPTER 3
MACHINE LEARNING INTRODUCTION

3.1 Overview

In Section 2.4.2 of the last chapter we discussed technical analysis indicators used by many traders to predict future market prices. However, the technical analysis indicators shown thus far are simple *if-then* algorithms. For example, *if* the price is above the upper band of the Bollinger Band, *then* the price is expensive, or overbought. These systems have evolved to more sophisticated methods using algorithms from artificial intelligence and machine learning [51].

In this chapter we discuss what is meant by machine learning and how the algorithms can be used to *learn* from data. Then in Section 3.3 we provide an overview of popular machine learning algorithms, like artificial neural networks, support vector machines, and ensembles. Section 3.4 explores different methods of comparing classifiers and lastly Section 3.5 examines different methods to evaluate unseen data (especially streaming data), with holdout sets and sliding windows, using these performance metrics.

3.2 Supervised versus unsupervised learning

Machine learning is a branch of artificial intelligence that uses algorithms, for example, to find patterns in data and make predictions about future events. In machine learning a dataset of observations called *instances* is comprised of a number of variables called *attributes. Supervised learning* is the modeling of these datasets

Table 3.1: An example of a supervised learning dataset

Time	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
09:30	b	n	-0.06	-116.9	-21.7	28.6	0.209	up
09:31	b	b	0.06	-85.2	-61	-21.7	0.261	unchanged
09:32	b	b	0.26	-4.4	-114.7	-61	0.17	down
09:33	n	b	0.11	-112.7	-132.5	-114.7	0.089	unchanged
09:34	n	n	0.08	-128.5	-101.3	-132.5	0.328	down

containing *labeled* instances. In supervised learning, each instance can be represented as (x, y), where x is a set of independent attributes (these can be discrete or continuous) and y is the dependent target attribute. The target attribute y can also be either *continuous* or *discrete*; however the category of modeling is *regression* if it contains a continuous target, but *classification* if it contains a discrete target (which is also called a class label). Table 3.1 demonstrates a dataset for supervised learning with seven independent attributes $x_1, x_2, \ldots, x_7$, and one dependent target attribute y. More specifically, $x_1, x_2 \in \{b, n\}$ and $x_3, \ldots, x_7 \in \mathbb{R}$ and the target attribute $y \in \{\text{up,unchanged,down}\}$. The attribute *time* is used to identify an instance and is not used in the model. Also the training and test datasets are represented in the same way however, where the training set contains a set of vectors of known label (y) values, the labels for the test set is unknown.

In *unsupervised learning* the dataset does not include a target attribute, or a known outcome. Since the class values are not determined a priori, the purpose of this learning technique is to find similarity among the groups or some intrinsic clusters within the data. A very simple two-dimensional (two attributes) demonstration is

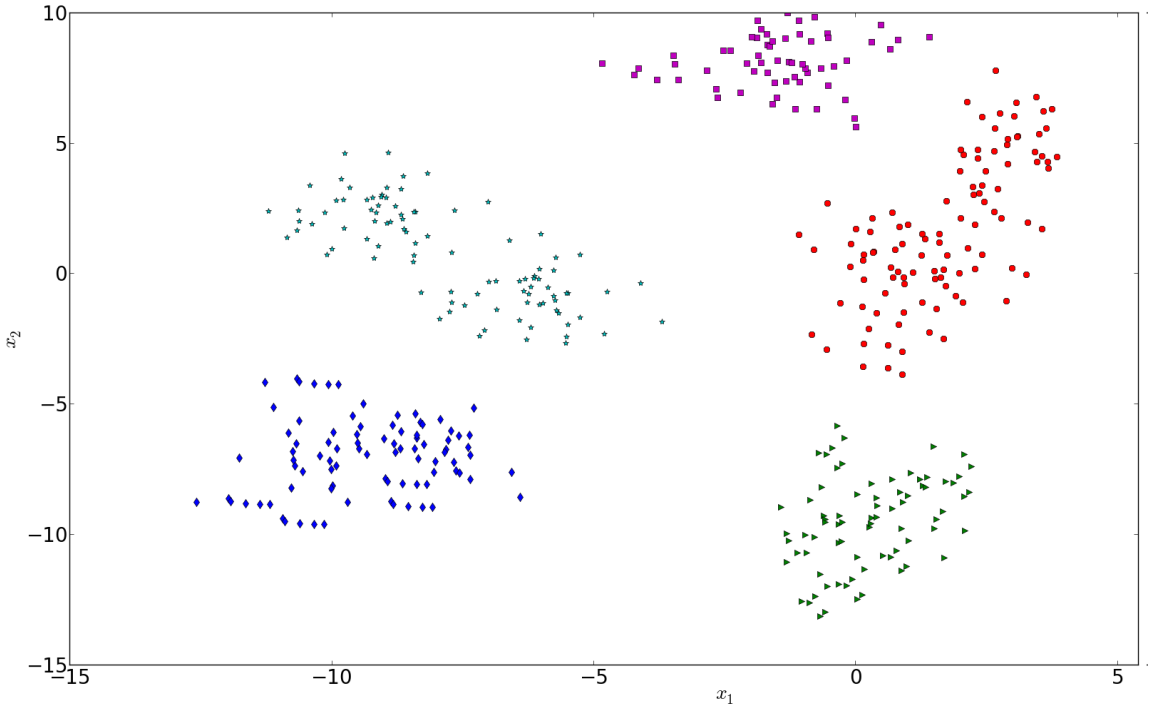


Figure 3.1: An example of an unsupervised learning technique – clustering

shown in Figure 3.1 with the data partitioned into five clusters. A case could be made however that the data should be partitioned into two clusters or three, etc.; the "correct" answer depends on prior knowledge or biases associated with the dataset to determine the level of similarity required for the underlying problem. Theoretically we can have as many clusters as data instances, although that would defeat the purpose of clustering.

Depending on the problem and the data available, the algorithm required can be either a supervised or unsupervised technique. In this thesis, the goal is to predict future price direction of the streaming stock dataset. Since the future direction becomes known after each instance, the training set is constantly expanding

with labeled data as time passes. This requires a supervised learning technique. Additionally, we explore the use of different algorithms since some may be better depending on the underlying data. Care should be taken to avoid, "when all you have is a hammer, everything becomes a nail."

3.3 Supervised learning algorithms

3.3.1 k Nearest-neightbor

The k nearest neighbor (kNN) is one of the simplest machine learning methods and is often referred to as a *lazy* learner because learning is not implemented until actual classification or prediction is required. It takes the most frequent class as measured by the weighted euclidean distance (or some other distance measure) among the k closest training examples in the feature space. In specific problems such as text classification, kNN has been shown to work as well as more complicated models [240]. When nominal attributes are present, it is generally advised to arrive with a "distance" between the different values of the attributes [236]. For our dataset, this could apply to the different trading days, Monday, Tuesday, Wednesday, Thursday, and Friday.

A downside of using this model is the slow classification times, however we can increase speed by using dimensionality reduction algorithms; for example, reducing the number of attributes from 200 to 20. Since the learning is not implemented until the classification phase though, this is an unsuitable algorithm to use when decisions are needed quickly.

3.3.2 Naïve Bayes

The naïve Bayes classifier is an efficient probabilistic model based on the Bayes Theorem that examines the likelihood of features appearing in the predicted classes. Given a set of attributes $X = \{x_1, x_2, \ldots, x_n\}$, the objective is to construct the posterior probability for the event C_k among a set of possible class outcomes $C = \{c_1, c_2, \ldots, c_k\}$. Therefore, with Bayes' rule $P(C_k|x_1, \ldots, x_n) \propto P(C_k)P(x_1, \ldots, x_n|C_k)$, where $P(x_1, \ldots, x_n|C_k)$ is the probability that attribute X belongs to C_j, and assuming independence[1] we can rewrite as $P(C_j|X) \propto P(C_j)\prod_{i=1}^{n} P(x_i|C_j)$. A new instance with a set of attributes X is labeled with the class C_j that achieves the highest posterior probability.

3.3.3 Decision table

A decision table classifier is built on the conceptual idea of a lookup table. The classifier returns the majority class of the training set if the decision table (lookup table) cell matching the new instance is empty. In certain datasets, classification performance has been found to be higher when using decision tables than with more complicated models. A further description can be found in [124, 125, 127].

3.3.4 Support Vector Machines

Support vector machines [221] have long been recognized as being able to efficiently handle high-dimensional data. Originally designed as a two-class classifier, it can work with more classes by making multiple binary classifications (one-versus-

[1]The assumption of independence is the *naïve* aspect of the algorithm.

one between every pair of classes). The algorithm works by classifying instances based on a linear function of the features. Additionally non-linear classification can be performed using a kernel. The classifier is fed with pre-labeled instances and by selecting points as support vectors the SVM searches for a hyperplane that maximizes the margin. More information can be found in [221].

3.3.5 Artificial Neural Networks

An artificial neural network (ANN) is an interconnected group of nodes intended to represent the network of neurons in the brains. They are widely used in literature, because of their ability to learn complex patterns. We present only a short overview of their structure in this section.

The artificial neural network is comprised of nodes (shown as circles in Figure 3.2), an input layer represented as $x_1 \ldots, x_6$, an optional hidden layer, and an output layer y. The objective of the ANN is to determine a set of weights w (between the input, hidden, and output nodes) that minimize the total sum of squared errors. During training these weights w_i are adjusted according to a learning parameter $\lambda \in [0, 1]$ until the outputs become consistent with the output. Large values of λ may make changes to the weights that are too drastic, while values that are too small may require more iterations (called *epochs*) before the model sufficiently learns from the training data.

The difficulty of using artificial neural networks is finding parameters that learn from training data without over fitting (i.e. memorizing the training data) and

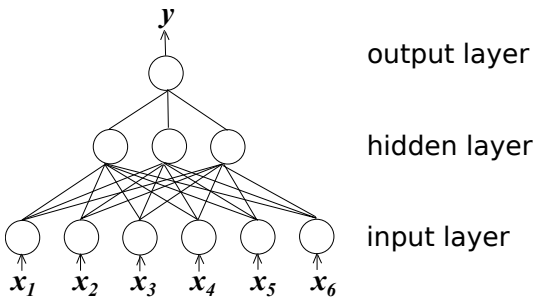


Figure 3.2: Example of a multilayer feed-forward artificial neural network

therefore perform poorly on unseen data. If there are too many hidden nodes, the system may overfit the current data, while if there are too few, it can prevent the system from properly fitting the input values. Also, a choice of stopping criterion has to be chosen. This can include stopping based on when the total error of the network falls below some predetermined error level or when a certain number of epochs (iterations) has been completed [16, 25, 177]. To demonstrate this, see Figure 3.3. This plot represents a segment of our high-frequency trade data that will be used later in this thesis. As the epochs increase (by tens), the number of incorrectly identified training instances decreases, as seen by the decrease in the training error. However, the validation error decreases until 30 epochs, and after 30, starts to increase. Around roughly 80 epochs the validation error begins to decrease again, however we need to make a judgment call since an increase in epochs increases the training times dramatically.

Yu et al. [245] state that with foreign exchange rate forecasting, which is similar to stocks because of the high degree of noise, volatility and complexity, it is advisable to use the sigmoidal type-transfer function (i.e. logistic or hyperbolic tangent). They

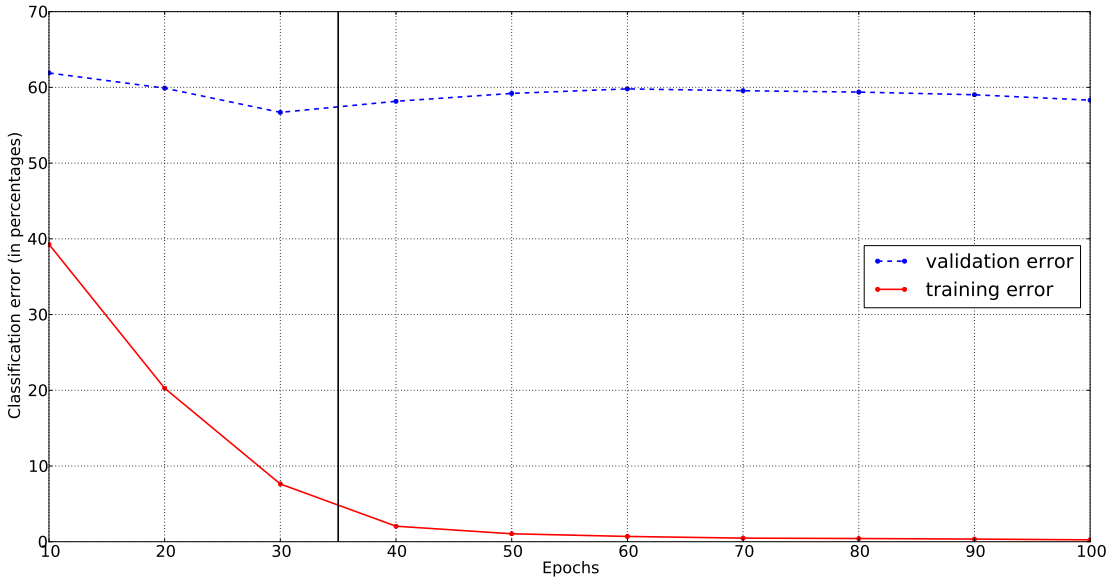


Figure 3.3: Artificial neural network classification error versus number of epochs

base this on the large number of papers that find predictability using this type of function in the hidden layer.

3.3.6 Decision Trees

The decision tree is one of the more widely used classifiers in practice because the algorithm creates rules which are easy to understand and interpret. The version we use in this paper is also one of the most popular forms, the C4.5 [186], which extends the ID3 [185] algorithm. The improvements are: 1) it is more robust to noise, 2) it allows for the use of continuous attribute, and 3) it works with missing data.

The C4.5 begins as a recursive divide-and-conquer algorithm, first by selecting an attribute from the training set to place at the root node. Each value of the attribute

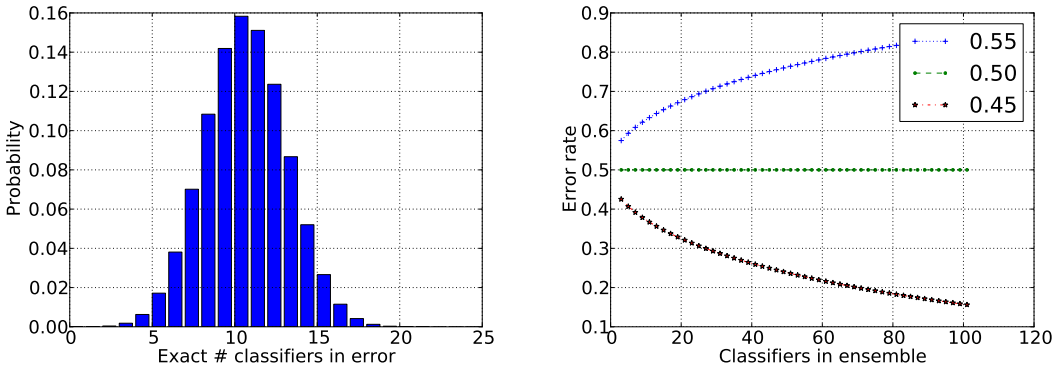
creates a new branch, with this process repeating recursively using all the instances reaching that branch [236]. An ideal node contains all (or nearly all) of one class. To determine the best attribute to choose for a particular node in the tree, the gain in information entropy for the decision is calculated. More information can be found in [186].

3.3.7 Ensembles

An ensemble is a collection of multiple base classifiers that take a new example, pass it to each of its base classifiers, and then combines those predictions according to some method (such as through voting). The motivation is that by combining the predictions, the ensemble is less likely to misclassify. For example, Figure 3.4a demonstrates an ensemble with 25 hypothetical classifiers, each with an independent error rate of 0.45 (assuming a uniform 2 class problem). The probability of getting k incorrect classifier votes is a binomial distribution, $P(k) = \binom{n}{k} p^k (1-p)^{n-k}$. The probability that 13 or more is in error is 0.31, which is less than the error rate of the individual classifier. This is a potential advantage of using multiple models.

This advantage of using multiple models (ensembles) is under the assumption that the individual classifier error rate is less than 0.50. If the independent classifier error rate is 0.55, then the probability of 13 or more in error is 0.69 – it would be better not to use an ensemble of classifiers. Figure 3.4b[2] demonstrates the error rate of the ensemble for three independent error rates, 0.55, 0.50, and 0.45 for ensembles

[2]The idea for the visualization came from [59, 82].



(a) Probability that precisely n of 25 classifiers are in error (assume each has error rate of 0.45)

(b) Error rate versus number of classifiers in the ensemble (employing majority voting) for three independent error rates

Figure 3.4: Ensemble simulation

containing an odd number of classifiers, from 3 to 101. From the figure it can be seen that the smaller the independent classifier error rate is, and the larger the number of classifiers in the ensemble is, the less likely a majority of the classifiers will predict incorrectly [59, 82].

The idea of classifier independence may be unreasonable, given that the classifiers may predict in a similar manner due to the training set. Obtaining a base classifier that generates errors as uncorrelated as possible is ideal. Creating a *diverse* set of classifiers within the ensemble is considered an important property since the likelihood that a majority of the base classifiers misclassify the instance is decreased.

Two of the more popular methods used within ensemble learning is bagging [27] and boosting (e.g. the AdaBoost algorithm [78] described in Subsection 3.3.7.2 is the most common). These methods promote diversity by building base classifiers on different subsets of the training data or different weights of classifiers.

3.3.7.1 Bagging

Bagging, also known as bootstrap aggregation, was proposed by Breiman in 1994 in an early version of [27]. It works by generating k bootstrapped training sets and building a classifier on each (where k is determined by the user). Each training set of size N is created by randomly selecting instances from the original dataset, with each receiving an equal probability of being selected and with replacement. Since every instance has an equal probability of being selected, bagging does not focus on any particular instance of the training data and therefore is less likely to over-fit [177]. Bagging is generally for unstable[3] classifiers such as decision trees and neural networks.

3.3.7.2 Boosting

The AdaBoost (Adaptive Boosting) algorithm of Freud and Schapire [78] in 1995 is synonymous with boosting. The idea however was proposed in 1988 by Michael Kearns [114] in a class project, where he hypothesized that a "weak" classifier, performing slightly better than average, could be "boosted" into a "strong" classifier. In boosting, instances being classified are assigned a weight; instances that were previously incorrectly classified receive larger weights, with the hope that subsequent models correct the mistake of the previous model. In the AdaBoost algorithm the original training set D has a weight w assigned to each of its N instances $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where x_i is a vector of inputs and y_i is the class label of that

[3]By unstable, it is meant that small changes in the training set can lead to large changes in the classifier outcome.

instance. With the weight added the instances become $\{(x_1, y_1, w_1), \ldots, (x_n, y_n, w_n)\}$ and the sum of the w_i must equal 1. The AdaBoost algorithm then builds k base classifiers with an initial weight $w_i = \frac{1}{N}$. Upon each iteration of the algorithm (which is determined by the user), the weight w_i gets adjusted according to the error ϵ_i of the classifier hypothesis[4]. The points that were incorrectly identified receive higher weights, and the ones that were correctly identified receive less. The desire is that on the next iteration, the re-weighting will help to correctly classify the instances that were misclassified by the previous classifier. When implementing the boosting ensemble on test data, the final class is determined by a weighted vote of the classifiers [78, 149].

Boosting does more to reduce bias than variance. This reduction is due to the algorithm adjusting its weight to learn previously misclassified instances and therefore increasing the probability that these instances will be learned correctly in the future. This has had a tendency to correct biases. However, it tends to perform poorly on noisy datasets and therefore the weights become greater, which causes the model to focus on the noisy instances and over-fit the data [195].

3.3.7.3 Combining classifiers for ensembles

The last step in any ensemble-based system is the method used to combine the individual classifiers; this is often referred to as *fusion rules*. Classifiers within an ensemble are most commonly combined using a majority voting algorithm. There

[4]If the error is greater than what would be achieved by guessing the class, then the ensemble is returned to the previously generated base classifier.

are however, different methods of combining, which often depend on the underlying classifiers used. For example, the Naive Bayes algorithm provides continuous valued outputs, allowing a wide range of strategies for combining, while an artificial neural network provides a discrete-valued output, allowing for fewer [133, 134, 247]. A description of each follows:

- Majority voting
  - Plurality majority voting – The class that receives the highest number of votes among classifiers (in literature, majority voting typically refers to version)
  - Simple majority voting – The class that receives one more than fifty percent of all votes among classifiers
  - Unanimous majority voting – The class that all the classifiers unanimously vote on
- Weighted majority voting – If the confidence in among classifiers is not equal, we can weight certain classifiers more heavily. This method is followed in the AdaBoost algorithm.
- Algebraic combiners
  - Mean/Minimum/Maximum/Median rules – The ensemble decision is chosen for the class according to the average/minimum/maximum/median of each classifier's confidence.

Table 3.2: Confusion matrix

		Predicted class	
		+	−
Actual Class	+	TP	FN
	−	FP	TN

While ensembles have shown success in a variety of problems, there are some associated drawbacks. This includes added memory and computation cost in keeping multiple classifiers stored and ready to process. Also the loss of interpretability may be a cause for concern depending on the needs of the problem. For example, a single decision tree can be easily interpreted, while an ensemble of 100 decision trees could be difficult [21].

3.4 Performance metrics

3.4.1 Confusion matrix and accuracy

A *confusion matrix*, also called a *contingency table*, is a visualization of the performance of a supervised learning method. A problem with n classes, requires a confusion matrix of size $n \times n$ with the rows representing the specific actual class and the columns representing the classifiers predicted class. In a confusion matrix, TP (true positive) is the number of positives correctly identified, TN (true negative) is the number of negatives correctly identified, FP (false positive) is the number of negatives incorrectly identified as positive, and FN (false negative) is the number of positives incorrectly identified as negatives. An example of a confusion matrix can be seen in Table 3.2.

From the confusion matrix it is relatively simple to arrive at different measures for comparing models. An example is *accuracy*, which is a widely used metric and is easy to interpret. From Equation 3.1, accuracy is the total number of correct predictions made over the total number of predictions made. While accuracy is a popular metric, it is also not very descriptive when used to measure the performance of a highly imbalanced dataset. A model may have high levels of accuracy, but may not obtain high levels of identification of the class that we are interested in predicting.

For example, if attempting to identify large moves in a stock which is comprised of 99% *small moves* and 1% *large moves*, it is trivial to report a model has accuracy of 99% without additional information. A classifier could also have 99% accuracy by simply reporting the class with the largest number of instances (e.g. the majority class is "small moves"). In an imbalanced dataset, a model may misidentify all positive classes and still have high levels of accuracy; pure randomness is not taken into account with the accuracy metric. Accuracy's complement is the error rate $(1 - \text{Accuracy})$ and can be seen in Equation 3.2.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$\text{Error rate} = \frac{FP + FN}{TP + FP + TN + FN} \tag{3.2}$$

There are several approaches to comparing models with imbalanced datasets. First is the precision and recall metrics and the accompanying harmonic mean, the F-measure. The second metric is based on Cohen's kappa statistic, which takes into account the randomness of the class. The third metric is the receiver operating characteristic which is based on the true positive and false positives rates. The

fourth is a cost-based metric which gives specific "costs" to correctly and incorrectly identifying specific classes. And the last method is based not on the ability of the model to make correct decisions, but instead on the profitability of the classifier as it applies to a trading system. A more detailed description of these metrics follows.

3.4.2 Precision and recall

Precision and recall are both popular metrics for evaluating classifier performance and will be used extensively in this paper. Precision is the percentage that the model correctly predicts positive when making a decision (Equation 3.3). More specifically, precision is the number of correctly identified positive examples divided by the total number of examples that are classified as positive. Recall is the percentage of positives correctly identified out of all the existing positives (Equation 3.4); it is the number of correctly classified positive examples divided by the total number of true positive examples in the test set. From our imbalanced example above with the 99% *small moves* and 1% *large moves*, precision would be how often a large move was correctly identified as such, while recall would be the total number of large moves that are correctly identified out of all the large moves in the dataset.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \tag{3.4}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{3.5}$$

$$\text{F-measure} = \frac{2(\text{precision})(\text{recall})}{\text{precision} + \text{recall}} \tag{3.6}$$

Precision and recall are often achieved at the expense of the other, i.e. high

precision is achieved at the expense of recall and high recall is achieved at the expense of precision. An ideal model would have both high recall and high precision. The F-measure[5], which can be seen in Equation 3.6, is the harmonic measure of precision and recall in a single measurement. The F-measure ranges from 0 to 1, with a measure of 1 being a classifier perfectly capturing precision and recall.

3.4.3 Kappa

The second approach to comparing imbalanced datasets is based on Cohen's kappa statistic. This metric takes into consideration randomness of the class and provides an intuitive result. From [14], the metric can be observed in Equation 3.7 where P_0 is the total agreement probability and P_c is the agreement probability which is due to chance.

$$\kappa = \frac{P_0 - P_c}{1 - P_c} \tag{3.7}$$

$$P_0 = \sum_{i=1}^{I} P(x_{ii}) \tag{3.8}$$

$$P_c = \sum_{i=1}^{I} P(x_{i.})P(x_{.i}) \tag{3.9}$$

The total agreement probability P_0 (i.e. the classifier's accuracy) can be be computed according to Equation 3.8, where I is the number of class values, $P(x_{i.})$ is the row marginal probability and $P(x_{.i})$ is the column marginal probability, with both obtained from the confusion matrix. The probability due to chance, P_c, can be computed according to Equation 3.9. The kappa statistic is constrained to the interval

[5]The F-measure, in the literature is also called the F-score and the F1-score.

Table 3.3: Computing the Kappa statistic from the confusion matrix

(a) Confusion matrix – Numbers

		Predicted class			
		up	**down**	**flat**	$\sum$
Actual class	**up**	139	80	89	308
	down	10	298	13	323
	flat	40	16	313	369
	$\sum$	189	396	4157	1000

(b) Confusion matrix – Probabilities

		Predicted class			
		up	**down**	**flat**	$\sum$
Actual class	**up**	0.14	0.08	0.09	0.31
	down	0.01	0.30	0.01	0.32
	flat	0.04	0.02	0.31	0.37
	$\sum$	0.19	0.40	0.42	1.00

$[-1, 1]$, with a kappa $\kappa = 0$ meaning that agreement is equal to random chance, and a kappa κ equaling 1 and -1 meaning perfect agreement and perfect disagreement respectively.

For example, in Table 3.3a the results of a three-class problem are shown, with the marginal probabilities calculated in Table 3.3b. The total agreement probability, also known as accuracy, is computed as $P_0 = 0.14 + 0.30 + 0.31 = 0.75$, while the probability by chance is $P_c = (0.19 \times 0.31) + (0.40 \times 0.32) + (0.42 \times 0.37) = 0.34$. The kappa statistic is therefore $\kappa = (0.75 - 0.34)/(1 - 0.34) = 0.62$.

3.4.4 ROC

The third approach to comparing classifiers is the **R**eceiver **O**perating **C**haracteristic (ROC) curve. This is a plot of the true positive rate, which is also called recall or

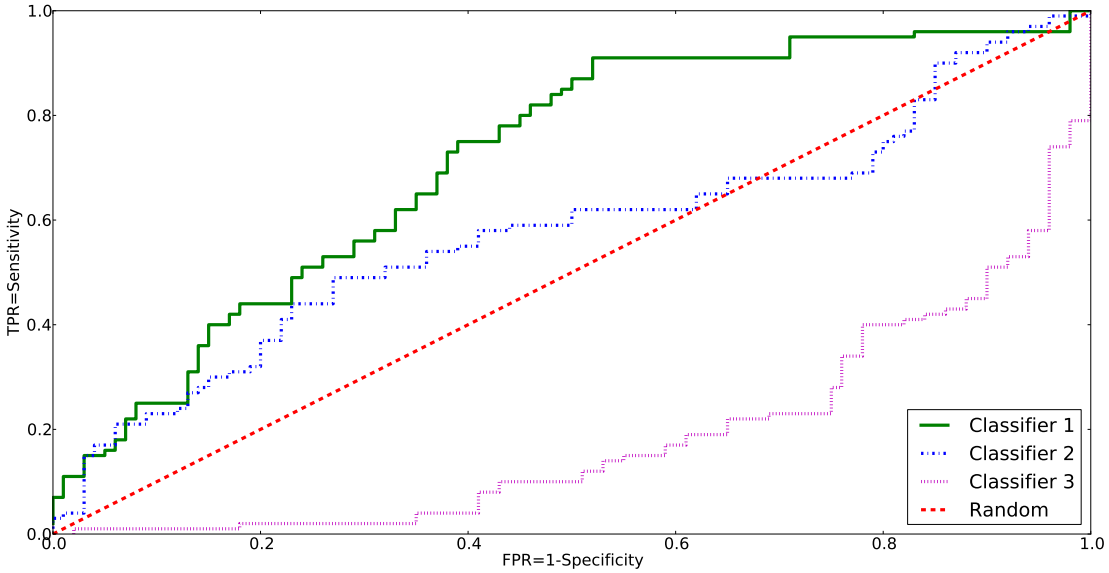


Figure 3.5: ROC curve example

sensitivity (Equation 3.10), against the false positive rate, which is also known as 1-specificity (3.11).

$$TPR = \frac{TP}{TP + FN} \tag{3.10}$$

$$FPR = \frac{FP}{TN + FP} \tag{3.11}$$

The best performance is noted by a curve close to the top left corner (i.e. a small false positive rate and a large true positive rate), with a curve along the diagonal reflecting a purely random classifier. As a demonstration, in Figure 3.5 three ROC curves are displayed for three classifiers. Classifier 1 has a more ideal ROC curve than Classifier 2 or 3. Classifier 2 is slightly better than random, while Classifier 3 is worse. In Classifier 3's case, it would be better to choose as a solution that is opposite of what the classifier predicts.

For single number comparison, the **A**rea **U**nder the ROC **C**urve (AUC) is calculated by integrating the ROC curve. Random would therefore have an AUC of 0.50 and a classifier better and worse than random would have an AUC greater than and less than 0.50 respectively. It is most commonly used with two-class problems although with multi-class examples the AUC can be weighted according to the class distribution. AUC is also equal to the Wilcoxon statistic.

3.4.5 Cost-based

The cost-based method of evaluating classifiers is based on the "cost" associated with making incorrect decisions [61, 65, 102]. The performance metrics seen thus far do not take into consideration the possibility that not all classification errors are equal. For example, an opportunity cost can be associated with missing a large move in a stock. A cost can also be provided for initiating an incorrect trade. A model can be built with a high recall, which misses no large moves in the stock, but the precision would most likely suffer. The cost-based approach gives an associated cost to this decision which can be evaluated to determine the suitability of the model. A cost matrix is used to represent the associated cost of each decision with the goal of minimizing the total cost associated with the model. This can be formalized with a cost matrix C and the entry (i, j) with the actual cost i and the predicted class j. When $i = j$ the prediction is correct and when $i \neq j$ the prediction is incorrect.

An advantage of using a cost-based evaluation metric for trading models is the cost associated with making incorrect decisions is known by analyzing empirical

data. For example all trades incur a cost in the form of a trade commission and money used in a trade is temporarily unavailable, thus incurring an opportunity cost. Additionally, a loss associated with an incorrect decision can be averaged over similar previous losses; gains can be computed similarly. Consider, for example, a trading firm is attempting to predict the directional price move of a stock with the objective to trade on the decision. At time t, the stock can *move up*, *down*, or have *no change* in price; at time $t+n$, the direction is unknown (this can be observed in Figure 3.6). For time $t+1$, a prediction of *up* might result in the firm purchasing the stock. Different errors in classification however would have different associated cost. A firm expecting a move up would purchase the stock in anticipation of the move, but a subsequent move down would be more harmful than no change in price. A actual move down would immediately result in a trading loss, whereas no change in price would result in an temporary opportunity cost with the stock still having the potential to go in the desired direction. Additionally an incorrect prediction of *"no change"* would merely result in an opportunity lost, but no actual money being put to risk since a firm would not trade based on the anticipation of a unchanged market (no change). Table 3.4 represents a theoretical cost matrix of the problem, with three separate error amounts represented: 0.25, 0.50, and 1.25.

3.4.6 Profitability of the model

While the end result of predicting stock price direction is to increase profitability, the performance metrics discussed thus far (with the exception of the cost-based

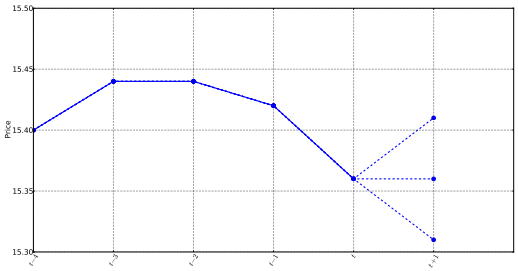


Figure 3.6: Possible directional price moves for our hypothetical example – move up, down, or no change

Table 3.4: Hypothetical cost matrix

		Predicted class		
		Down	**No change**	**Up**
Actual class	**Down**	0	0.25	1.25
	No change	0.50	0	0.50
	Up	1.25	0.25	0

metric) evaluate classifiers based on the ability to correctly classify and not on overall profitability of a trading system. As an example, a classifier may have very high accuracy, kappa, AUC, etc. but this may not necessarily equate to a profitable trading strategy, since profitability of individual trades may be more important than being "right" a majority of times; e.g. making $0.50 on each of one hundred trades is not as profitable as losing $0.05 95 times and then making $12 on each of five trades[6].

Figure 3.7 represents a trading model represented in much of the academic literature, where the classifier is built on the data with a prediction of up, down, or no change in the market price with the outcome passed to a second set of rules. These

[6]An argument can also be made that a less volatile approach is more ideal (i.e. making small sums consistently). This depends on the overall objective of the trader – maximizing stability or overall profitability.

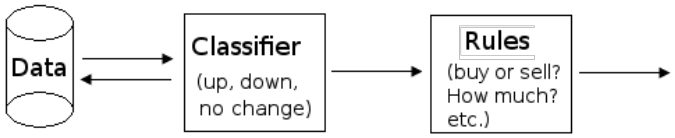


Figure 3.7: Trading algorithm process

rules provide direction if a prediction of "up", for example, should equate to buying stock, buying more stock, or buying back a position that was previously shorted. The rules also address the amount of stock to be purchased, how much to risk, etc.

When considering profitability of a model, the literature generally follows the form of an automated trading model, which is "buy when the model says to buy, then sell after n number of minutes/hours/days [161]" or "buy when the model says to buy, then sell if the position is up $x\%$ or else sell after n minutes/days/hours [138, 164, 202]." Teixeira et al. [214] added another rule (called a "stop loss" within trading), which prevented losses from going past a certain dollar amount during an individual trade.

The goal of this thesis is not to provide an "out of the box" trading system with proven profitability, but to instead help the user make trading decisions with the help of machine learning techniques. Additionally, there are many different rules in the trading literature relating to how much stock to buy or sell, how much money to risk in a position, how often trades should take place, and when to buy and sell; each of these questions are enough for entire dissertations. In practice, trading systems often involve many layers of controls such as forecasting and optimization methodologies

Table 3.5: Importance of using an unbiased estimate of its generalizability – trained using the dataset from Appendix B for January 3, 2012

	January 3, 2012 (training data)	January 4, 2012 (unseen data)
Accuracy	94.713%	37.31%

that are filtered through multiple layers of risk management. This typically involves a human supervisor (risk manager) that can make decisions such as when to override the system [69]. The focus of this paper therefore, will remain on the classifier itself; maximizing predictability when faced with different market conditions.

3.5 Methods of testing

Once a model has been built using a training set, and its parameters are adjusted using a validation set, its performance needs to be evaluated on an unseen subset of the data, the test set. An unseen subset is used since the model is biased toward the training set and may therefore over-fit the data, resulting in an artificially high performance measure. For example, a C4.5 decision tree built on our balanced three class dataset (see Appendix B) using January 3, 2012 data and then tested on our training set results in an accuracy of 94.71%, while more realistically testing it on the following day's data results in an accuracy of 37.31% (see Table 3.5).

The following subsection reviews some of the methods used to evaluate the performance of classifiers.

3.5.1 Holdout

The holdout method is when the labeled dataset D is split into two disjoint sets, a training set D_{train} and a testing set D_{test}, where $D_{train} \cup D_{test}$ and $D_{train} \cap D_{test} = \emptyset$. The split varies from a 50-50 to a two-thirds for training and a one-third split for testing; although this varies widely in the literature and is typically dependent on the underlying problem and the amount of data available.

There are problems associated with the holdout method that needs to be taken into consideration. First, splitting the data into disjointed training and testing sets reduces the amount of data available for learning, with many instances never being accounted for by the model. This can be mostly eliminated by using random subsampling. In this process, the holdout method is repeated several times with different subsets in each dataset. The performance metric is then averaged.

The second potential problem of the holdout method is the performance metric (such as accuracy) can be high, depending on the distribution of the instances in the training and test sets. For example, one class overrepresented in the training set and underrepresented in the test set can result in mediocre model performance [177, 149]. This is especially pertinent when evaluating streaming stock data, where the underlying structure of the data may change over time due to changing market dynamics. As an example, a problem may arise if training a model on an upward moving stock market, where the class "large moves up" dominates "large moves down", and then testing on a downward moving market, where the class "large moves down" dominate "large moves up." This is also known as *concept drift* and will be discussed in detail

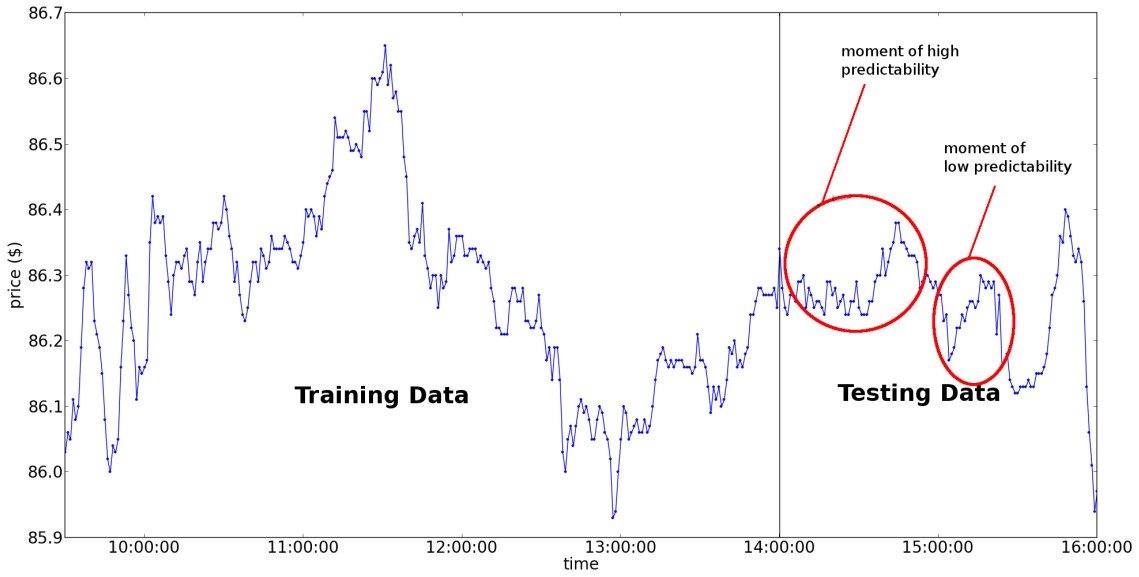


Figure 3.8: Demonstrating a problem with the holdout method – averaging out predictability

in Section 4.2.

The third potential problem when evaluating using the holdout method is inconsistent predictability throughout the test set. This is due to concept drift, or the changing of the underlying concept or target variable and is particularly important when evaluating streaming stock data. For example, in Figure 3.8 both moments of high and low predictability can be seen; this is consistent with concept drift. The results of the testing set would show little predictability because the moments of high predictability are "averaged out" by the moments of low predictability. The use of the sliding window, prequential, and interleavened test-then-train methods can help alleviate this problem.

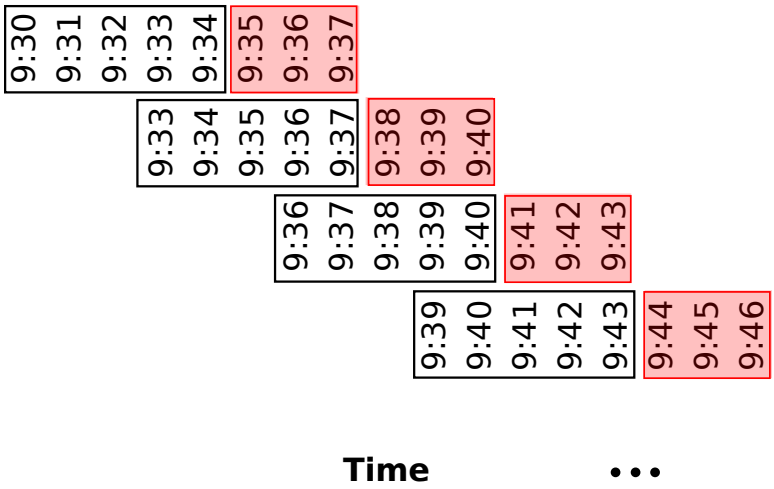


Figure 3.9: An example of the *sliding window* approach to evaluating the data stream performance

3.5.2 Sliding window

The sliding window is a modified holdout method specific to data streams that works by using a window of size w that contains training data at time t to $t-w+1$. The test set is a subset of future instances in a window of size f containing the instances at time $t+1$ to $t+f$. The desired performance metric is then computed for each window.

For example in Figure 3.9 a sliding window ($w = 5$, $f = 3$) is demonstrated with the initial model built using training data from time 9:30 to 9:34. This model is then tested on the unseen data at 9:35 to 9:37. The model then incorporates the (now historical) data into the model and the process continues.

A sliding window approach is perhaps the most intuitive, and it is the main method that we will be using in in our approach. The size of w (the training set size) and f (the testing set size) can be determined *a priori* or via an adaptive algorithm

that takes the level of *concept drift* in the dataset into consideration. This will be discussed later.

3.5.3 Prequential

The prequential approach [55, 56] has become popular in data stream learning [86, 87] because it monitors the error of a model over time by predicting unseen instances one-by-one and then adding those instances into the training set after the observed value is known. The error is computed using an accumulated sum of a loss function between the observed values y_i and predicted values $\hat{y_i}$, $S_i = \sum_{i=1}^{n} L(y_i, \hat{y_i})$, with the mean loss computed as a simple average $M = \frac{1}{n} \times S$.

At the beginning of the dataset, few instances are included in the model, often with high error rates. This lead to the inclusion of a forgetting factor in the model which will give less weight to previously seen examples. The forgetting factor can include either a sliding window of size n or a decay factor. More information can be found in [86].

3.5.4 Interleaved test-then-train

Interleaved test-then-train is an evaluation process for streaming algorithms described in [21] such that every nth incoming instance is used to test the model before it is used for training. The performance metrics are then incrementally updated. Individual instances become less significant as more instances are seen, therefore when plotting the performance a smooth representation is obtained. This is both an advantage and a disadvantage of this metric; it allows for single number comparisons

of the model at the end of the evaluation of the data stream, however it also obscures the performance at any given instance. For this reason, we do not use it in the evaluation of models.

3.5.5 k-fold cross-validation

Cross-validation is the process of splitting the dataset into k equal subsets. Training uses $k-1$ subsets and the remaining subset is then used as the testing set. This is repeated k times so that each subset is used as a testing set once. The total error is computed by summing up the errors obtained during all k runs, with the performance metric (such as accuracy) computed as the average across runs [177].

While cross-validation has been used in the literature for streaming data, we feel that it is not appropriate for stock data since the future market conditions are being "leaked", resulting in sometimes overly optimistic, biased classifiers. Models should be tested only on data that was not available at the time of model creation.

For example in Table 3.6 a hypothetical stream of stock data contains three attributes containing current and past stock prices from time t to $t-3$ and a predicted class (the dataset is also demonstrated in Figure 3.10). The objective of the model is to predict the next one-minute price direction of either "up" or "down." Splitting the data into three subsets for 3-fold cross-validation may result in subsets 1 and 3 being used in the learning model to predict subset 2; subset 3 was not available during subset 2, therefore leaked this data. This also occurs when subsets 2 and 3 are used to predict subset 1. For this reason, we do not use cross-validation in our evaluation

Table 3.6: Data stream with data partitioned into three subsets for cross-validation

Subset	ID	Attribute_t	Attribute_{t-1}	Attribute_{t-2}	Attribute_{t-3}	Class_{t+1}
subset 1	9:30	$13.50_{9:30}$	$13.49_{9:29}$	$13.48_{9:28}$	$13.47_{9:27}$	$\text{up}_{9:31}$
	9:31	$13.52_{9:31}$	$13.50_{9:30}$	$13.49_{9:29}$	$13.48_{9:28}$	$\text{up}_{9:32}$
	9:32	$13.55_{9:32}$	$13.52_{9:31}$	$13.50_{9:30}$	$13.49_{9:29}$	$\text{down}_{9:33}$
subset 2	9:33	$13.54_{9:33}$	$13.55_{9:32}$	$13.52_{9:31}$	$13.50_{9:30}$	$\text{down}_{9:34}$
	9:34	$13.53_{9:34}$	$13.54_{9:33}$	$13.55_{9:32}$	$13.52_{9:31}$	$\text{up}_{9:35}$
	9:35	$13.56_{9:35}$	$13.53_{9:34}$	$13.54_{9:33}$	$13.55_{9:32}$	$\text{up}_{9:36}$
subset 3	9:36	$13.59_{9:36}$	$13.56_{9:35}$	$13.53_{9:34}$	$13.54_{9:33}$	$\text{up}_{9:37}$
	9:37	$13.63_{9:37}$	$13.59_{9:36}$	$13.56_{9:35}$	$13.53_{9:34}$	$\text{up}_{9:38}$
	9:38	$13.64_{9:38}$	$13.63_{9:37}$	$13.59_{9:36}$	$13.56_{9:35}$	$\text{down}_{9:39}$

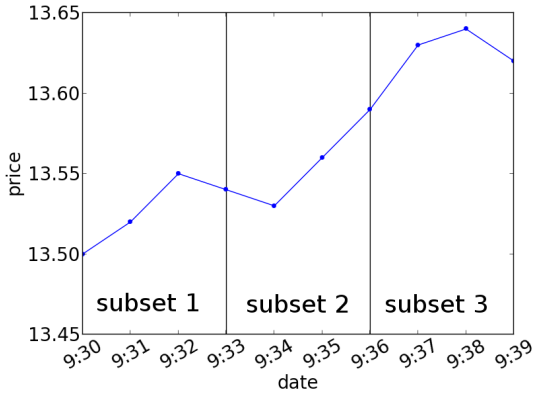


Figure 3.10: Data stream with data partitioned into three subsets for cross-validation

of models.

3.6 Conclusion

This chapter introduced machine learning with a particular emphasis on supervised learning techniques used in later chapters. Methods of comparing classifier performance were also examined such as AUC, which we use to evaluate and compare the performance of our new framework with existing classifiers (more in Chapter 6).

AUC was used both for its popularity within the machine learning community

and its improvement over existing methods such as accuracy or error rate. As explained in this chapter, accuracy is problematic within imbalanced datasets because the result can be high yet provide little predictive ability of the class that is important. AUC provides a nice framework for comparison – above 0.50 is better than random and below 0.50 is worse than random, no matter the class distribution – simple, yet descriptive. Additionally AUC provides, through the ROC curve, for determining alternate cutoffs for class probabilities. By examining the curve, a threshold that maximizes the trade-off between sensitivity and specificity can be determined [73].

This provides for a *confidence* of the prediction. It is this "confidence of decision" that we believe would be particularly important to a trader. As mentioned previously, the idea with this research is not to provide an "out of the box" trading system with proven profitability, but to provide automated price direction predictions so that the trader can make up his/her own mind depending on the objective.

This chapter also examined different approaches to testing, such as hold-out methods and sliding window evaluation, and described why traditional cross-validation methods are problematic when using streaming data with lagged indicators. Incorrect use of cross-validation can lead to *leakers*, which would provide overly optimistic performance evaluations. With an introduction of machine learning techniques and evaluation out of the way, the next chapter introduces predicting streaming data.

CHAPTER 4
DATA STREAM PREDICTION

4.1 Introduction

High-frequency stock data streams require special consideration not often seen when learning from static datasets. Markets do not remain stable, instead they are inherently noisy; technical indicators that show high predictability during one moment may disappear as more traders spot the pattern and implement them in their own trading strategies. Timmermann [215] noted that as traders search for and exploit any market pattern or high-probability event, the predictability disappears, thus making the market constantly evolve. Therefore, classifiers with high initial predictability may decrease in performance as the patterns are discovered and implemented by others.

As market dynamics change, model performance may decrease, requiring an update in the training data and/or change in the quantitative technical analysis indicators used as attributes. This task of keeping models relevant in the face of changing market dynamics, referred to as *concept drift*, is unique to data streams. Ideally, if the concept drifts could be anticipated, then the trader could store models to use with each specific market condition (or concept) and later apply those models to incoming data. The assumption however is the future is uncertain, therefore future concepts are still undecided.

Schulmeister [200] suggests that technical analysis indicators that previously

worked to predict market direction no longer work since widespread adoption of a particular trading approach is enough to drive the price either up or down enough to eliminate the pattern[1]. Instead he finds evidence that predictability has moved to higher-frequency intraday data. This high resolution is more difficult to examine by traders and thereby reduces the number of *eyes viewing the patterns*. Higher predictability with high-frequency data is in line with our result from Section 2.3 in Chapter 2 and in our paper [190].

This focus on high-frequency data to make predictions requires special consideration. As the amount of data increases, limitations in time prevent existing methods from learning; an algorithm that takes 30 minutes to arrive with a prediction is of little value if the goal is to predict price direction just one minute in the future. However, an algorithm that quickly and efficiently produces incorrect decisions is of little value. Thus practical trade-offs between efficiency and performance can occur [80]. This chapter discusses these problems and examines the existing paths researchers have followed to arrive at solutions for predicting future events in data streams.

In Section 4.2 we first provide a formal definition of concept drift and discuss different methodologies for learning when faced with it. For example, in much of the literature about stock predictability, a hold-out method is used. With this method, $\frac{2}{3}$ of the data is used to train the model and $\frac{1}{3}$ of the data is used for testing (see Figure 4.1). While this is easy to implement and requires little programming skill, it

[1]The popularity of technical analysis is evident by doing a simple query on Amazon.com. Using the words "technical analysis, stocks" our query returned over 1600 books (July 31, 2013).

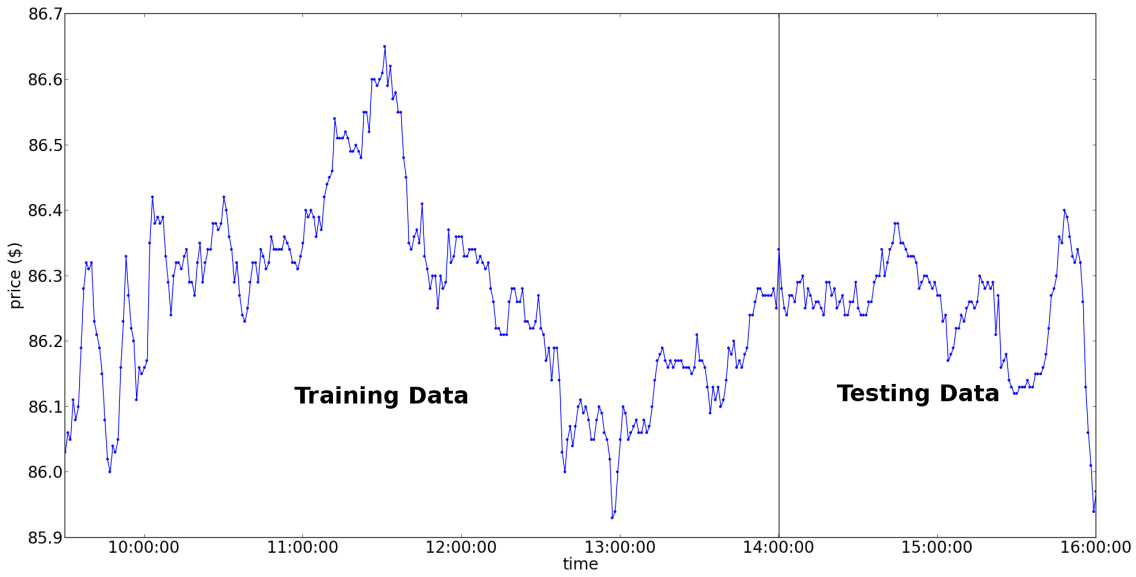
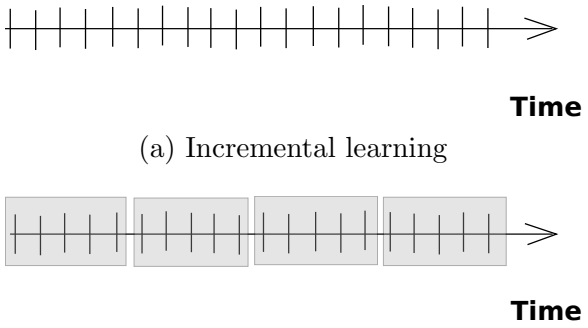


Figure 4.1: Naïve method of learning and testing models using stock data – using $\frac{2}{3}$ data to train and then $\frac{1}{3}$ of the data to test

is a naïve approach to use in a real-time setting since markets rarely remain stable. The most common approach to dealing with this concept drift is with the use of sliding windows; classifiers are trained on a moving subset of the data stream. We propose fixed and adjustable sized sliding windows for our framework and this will be discussed in a later chapter.

We then discuss in Section 4.3 models for use with streaming data. These models can be divided into two methodologies for learning: *adaptive* and *wrapper-based* methods [21, 121]. Adaptive methods are algorithms that have been adapted to work with data streams. They learn data incrementally (Figure 4.2a), as the instances arrive, and learn with one pass through the data; speed is essential to these algorithms. Wrapper methods require the data to be collected into *subsets* (Figure 4.2b) so that a traditional classifier (such as a support vector machine or neural



(a) Incremental learning

(b) Wrapper learning

Figure 4.2: Learning *instance-by-instance* versus by *chunk of instances*

network) can be used to learn.

In Section 4.4 we discuss, as noted by Gaber et al. [80], the trade-off between performance (accuracy) and efficiency (speed) of the algorithm. This infers then that frameworks for high-speed streaming stock data must sacrifice performance for a necessary quick decision. Typically longer training times achieve better results; a brute force method would eventually lead us to a global optimum. Through a demonstration we determine that this is often a more complicated discussion. More data leads to greater training times, which generally leads to better results, but there is a point of diminishing return where more data may lead to worse results.

4.2 Concept drift

4.2.1 Definition and causes

The changing of the underlying concept, or target variable, often complicates the learning of models from streaming data. This *concept drift* is unique to data streams and makes the task of keeping models relevant difficult. As the concept

changes, model performance may decrease, requiring a change or update in the training data. Ideally, if the concept drifts could be anticipated then the trader could store models to use with each specific market condition (or concept) and later apply the model to incoming data. The assumption however is that the concept generating function is unknown and although it can be estimated or predicted, there is no certainty [250]. And while the market periodically displays reoccurring[2] behavior such as economic cycles [12, 85, 103, 198, 237] or behavioral moods [23, 53, 99], rarely are specific market conditions consistently known *a priori*. Additionally, the idea of using the most recent training data may not be valid for all problems, but this is domain specific [3].

Providing a definition for concept drift can be difficult considering that there is not a standard terminology; authors use different names for the same concepts, or use the same names for different concepts [166][3]. Kelly et al. in [115] gives perhaps the most common definition of concept drift explaining it in three forms that concept drift can occur: (1) the class priors, $P(c_i)$, $i = 1, 2, 3, \ldots k$ may change over time, where k is the number of classes; (2) the distribution of the classes may change, $P(\mathbf{X}|c_i)$, where $i = 1, 2, 3, \ldots k$ and $\mathbf{X}$ is a vector of labeled instances; and (3) the

[2]Some of the literature also refers to this as seasonal or cyclical data.

[3]Moreno-Torres et al. [166] provides a list of several terms used in the literature within the past few years. These include "concept shift" or "concept drift", "change of classification", "changing environments", "contrast mining in classification learning", "fracture points" and "fractures between data." Their paper provides yet another term "dataset shift" and the authors provide a reasonable discussion of why it should be named so. However a simple Google Scholar search on June 23, 2013 returns 830 hits for "concept shift" and only 235 for "dataset shift"; we therefore use the most popular term in this thesis.

posterior distribution of the class membership $P(c_i|\mathbf{X})$, $i = 1, 2, 3, \ldots k$ may change. The posterior distribution of the class membership Kelly et al. explains is the only type of change that actually matters; the class priors $P(c_i)$ and distribution of classes $P(\mathbf{X}|c_i)$ may change but the posterior $P(c_i|\mathbf{X})$ will remain constant. However a change in $P(c_i|\mathbf{X})$ will always result in a change in either $P(c_i)$ or $P(\mathbf{X}|c_i)$.

Hoens, Polikar, and Chawla [104] do not differentiate between the three forms of Kelly et al. [115]. Their reasoning is with skewed and imbalanced datasets, changes in the class may require updates or retraining of the model. This occurs when the majority class is underrepresented in the dataset. In streaming datasets, the positive class (the class one is most interested in learning) may occur infrequently, or not at all, during the timespan being used in the training data. For example the classifier may minimize the error rate by predicting everything as the majority class, thus ignoring the small number of positives that we are interested in predicting.

In traditional offline learning, the training and testing data are assumed to be from a stationary distribution with the same concept generating function [104]. This assumption however is often violated in real-world scenarios. In streaming stock data, often known for drastic and irregular movements, a stationary distribution cannot be assumed. Many adaptive algorithms built for streaming data, without modifications, can have difficulty maintaining high performance in the face of concept drift, which is further exacerbated by noise. Overreacting to noise may result in the underlying training data being changed too often and the model loses past knowledge that may be helpful in the future. Not updating training data frequently enough leads to a model

with poor performance. This is often referred to as a *stability-plasticity* dilemma, where stability refers to the ability of the model to maintain existing knowledge and concepts within the model, and plasticity refers to the ability of the model to acquire new data [104, 180]. Elwell and Polikar [67] write about the need of models facing concept drift to strike a balance between prior and new knowledge. They suggest that irrelevant knowledge should be dropped until needed, while relevant knowledge should be reinforced.

Concept drift occurs in the market for a number of reasons. For example, traders preference for stocks change; increases in a stock's value may be followed by decreases. The appearance of trends can cause other traders, not wanting to miss the price increase, to buy, thereby helping to fuel the enthusiasm for the stock and pushing it to higher levels [158, 207]. However as explained previously, a widespread adoption of a particular trading approach is enough to drive the price either up or down enough to eliminate the pattern [215]. This would cause a decrease in the predictability of a particular trading indicator, and thus concept drift would occur.

4.2.2 Approaches to learning with concept drift

Before we discuss approaches to learning with concept drifts, it is first important to demonstrate how concept drift can affect classifier performance. We chose four stocks (symbols: XOM, ANR, APC, BHI) and trained a Support Vector Machine (polynomial degree kernel) on 30,000 minutes and then tested on subsequent intervals of 60 minutes (a full description of the attributes used can be found in Appendix C).

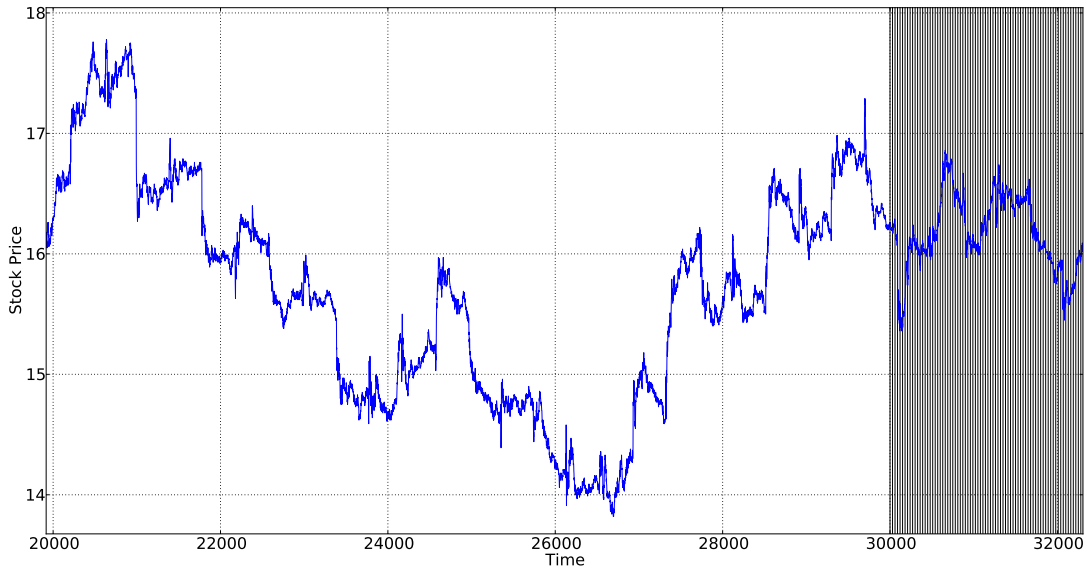
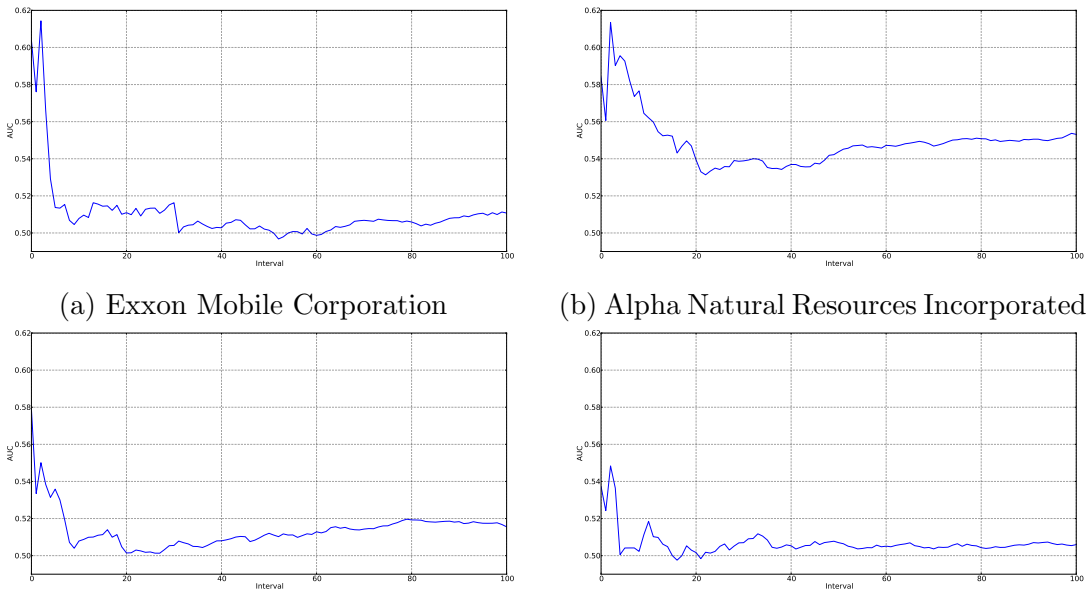


Figure 4.3: Demonstrating train once and test multiple times

This method of training on one subset of data and then testing on multiple subsequent holdout sets can be seen in Figure 4.3. The results of this testing for four random stocks from our dataset can be seen in Figure 4.4. Using a prequential AUC evaluation (see Subsection 3.5.3) the performance decreases the further the testing gets from the data used to train the model. This indicates occurrence of concept drift, signaling a need for an update to the training data.

Through the demonstration, it can be seen (when not using *concept adapting models*), that performance often (although not *always*) decreases when the time between between testing and training data increases. Building an algorithm to learn with drifting concepts generally takes one of two forms. The first is to detect concept drift, such as through the use of novelty detection algorithms, and upon detection, adapt the classifiers to this change of concept [83]. This *adaption* (i.e. retraining)



(a) Exxon Mobile Corporation

(b) Alpha Natural Resources Incorporated

(c) Anadarko Petroleum Coportation

(d) Baker Hughes Incorporated

Figure 4.4: Demonstrating the loss in performance (AUC) as testing gets further away from last training data

of the model to the new concept is often slow, therefore researchers often focus on the use of modified classifiers that learn/train quickly. The second form of adapting to concept change is to *assume* that drift occurs, and to take this assumption into consideration when building the model; the actual level of concept drift or even if it actually *does* occur may not be measured. An example of this is to retain models at fixed intervals, such as through the use of sliding windows. Another solution is through the use of ensembles, such as the work of Street and Kim [210], which uses a pool of previously trained classifiers that are updated according to a heuristic. A discussion of both forms follows.

4.2.2.1 Find evidence of concept drift and then re-train

Klinkenberg et al. [122, 123] describes several methods for the prediction of concept drift. The first method is found by examining the change of the distribution of the classes (i.e. class priors) within the data over time, with the assumption that class imbalances *will* create a need for a new classifier. For example, in Figure 4.5 we visualized the class distribution of the stock Exxon over 30-minute intervals for the first week in 2012[4]. The predicted class is the change in price over a one minute period, $\left(\text{price}_t - \text{price}_{t-1}\right) > \$0.02$ is considered a *large move up*, $\$0.02 \geq \left(\text{price}_t - \text{price}_{t-1}\right) \geq -\$0.02$ is considered an *insignificant move*, and $\left(\text{price}_t - \text{price}_{t-1}\right) < -\$0.02$ is considered a *large move down*. As can be seen, the class distribution does not remain stable over the 30 minute intervals. However, while this is interesting and gives us a better understanding of the dynamics of the stock, such as class imbalance, the information is not necessarily useful. A statistical test may determine that the class distribution has changed, but if the classifier is still performing well, there is no need to change or update the training data or classifier. Instead we use observation and detection of class imbalance to rectify classifier bias toward the majority class, thus misclassifying the minority class instance. The subject of learning under imbalance will be discussed further in Chapter 5.

The second method of determining when concept drift occurs, according to Klinkenberg et al. [122, 123], is by examining the performance measures over time, such as the accuracy, precision, recall, etc. of the classifier. A decrease in performance

[4]Due to an exchange holiday on Monday, the week only included four trading days.

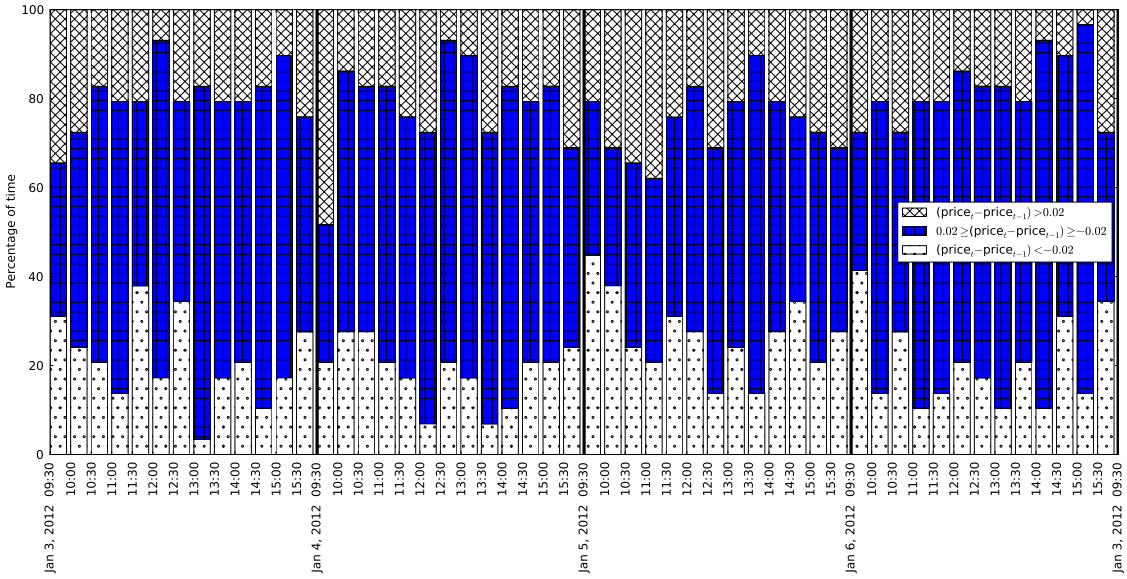


Figure 4.5: Demonstrating via a stacked bar chart the change of class priors for 30 minute/instance periods for the stock Exxon for the first week of 2012

may signal a change in concept, which would indicate new training data should be used within the classifier or older data should be dropped.

One of the first algorithms for determining when concept drift occurred was Kubat and Widmer's **Flo**ating **R**ough **A**pproximation algorithm (FLORA) in 1989 [130, 131, 232, 234]. FLORA was a supervised learning method that determined when to update the classifier based on the appearance of concept drift in a data stream caused by hidden features. The idea behind FLORA is that learning takes place within a fixed-size moving window of instances. Instances that have not been "confirmed" for a while by the FLORA algorithm must therefore be from another concept and will hence be dropped and the classifier retrained. More specifically, the methodology uses sets of disjunctive normal form expressions represented as an input of positive and negative examples of target concepts. These concept descriptions

are stored in three collections of symbolic description sets: (1) ADES contains all positive examples, (2) NDES covers all negative examples, and (3) the PDES concept description contains both positive and negative examples[5]. If any example has been contradicted by the others, then the example is discarded. Each time an example arrives from the stream, it will be added to the window (and therefore learned) and the oldest example will be deleted. As the window moves along the stream of examples, the contents in ADES, NDES, and PDES will change in content.

FLORA2 [232] is an improvement on FLORA and one of the first methods to consider performance when adapting the sliding window size. It uses a heuristic to shrink the size of the window whenever concept drift occurs (by examining accuracy), thereby eliminating older instances, and grows the size of the window when the concept remains stable. This enables the classifiers to train on more instances during moments of stability and decrease during moments of increased concept drift.

FLORA3 [231, 233, 234] is an improvement upon FLORA2. By saving concepts seen thus far for later use, it is able to facilitate learning on cyclical or recurring concepts. When concept change is determined (i.e. by decreasing the size of the window as in FLORA2) the algorithm checks its reserve of concepts for another that might describe the examples currently in the window better. In practice FLORA3 was found to be unstable during periods of slow concept drift and in very noisy periods This lead to the creation of FLORA4, which was an improvement upon FLORA3.

[5]Accepted descriptors (ADES); Disjunctive normal form (DNF); Negative descriptors (NDES); Potential descriptors (PDES)

Similar to the FLORA family of algorithms in using variable-windows, is the work of Bifet and Gavalda [18] with their **Ad**aptive **Win**dow algorithm (ADWIN). Their algorithm adjusts the window size according to the determination of concept drift. The window size increases when no change is apparent (thus increasing the size of the training set) and decreases when change occurs. The idea behind ADWIN is that when two "large enough" subwindows of sliding window W exhibit "distinct enough" averages, the older portion of the window is dropped. This portion is kept until a statistical test is able to reject the null hypothesis that μ_t has remained constant within the sliding window W with a level of confidence δ.

Another relevant work was that of Klinkenberg and Renz [123] who used measurements of accuracy, recall and precision, and by measuring their changes over time, determined when concept change occurred. First the average value and the standard sample error are computed for each of the performance measurements using the last M batches from a sliding window. If the current value of accuracy, recall, and precision were smaller than the confidence interval of α times the standard error around the average value (where $\alpha > 0$), then a concept change occurred.

Gama et al. [83] provided another method of monitoring the error rate to determine when concept drift occurred, appropriately called **D**rift **D**etection **M**ethod (DDM). After n instances, the probability of classifier error (i.e. of getting a *False* and assuming Bernoulli trial) was determined for each instance i; this is calculated after a set number of instances by dividing the number of classifier errors into the total number of instances seen thus far. The standard deviation was given by $s_i =$

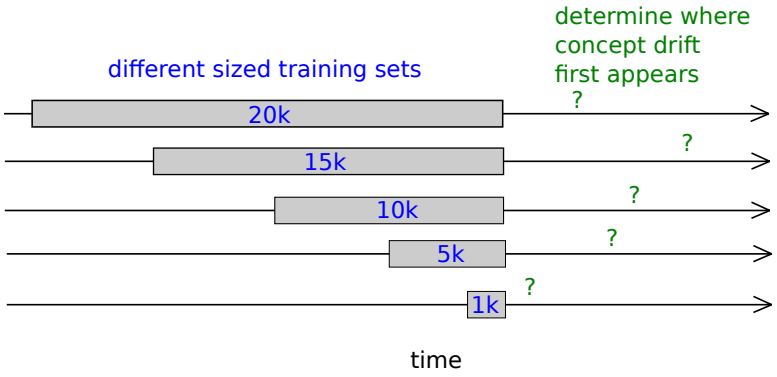


Figure 4.6: Demonstrating the layout of our experiment using Gama et al. [83] concept Drift Detection Method (DDM)

$\sqrt{\frac{p_i(1-p_i)}{i}}$. Additionally, a register of p_{min} and s_{min} was kept. The algorithm gave a *warning* when $p_i + s_i \geq p_{min} + \alpha \cdot s_{min}$ and a determination of *concept drift* when $p_i + s_i \geq p_{min} + \beta \cdot s_{min}$. The paper used $\alpha = 2$ and $\beta = 3$ with the idea that after concept drift has been *flagged*, a new learner is created.

DDM is an often cited method so we demonstrate the algorithm on our stock dataset. Using a support vector machine (with a polynomial degree kernel) we build classifiers using 1000, 5000, 10000, 15000 and 20000 instances. Starting at four different test instances[6], we examine when concept drift is first detected by the algorithm (where $n > 30$). Figure 4.6 demonstrates the layout of the experiment.

The results from Table 4.1 show the average number of instances until a concept drift occurs according to the algorithm for each stock. This is where, according to the algorithm, a new model would need to be built because the concept change has made the model obsolete. For our experiment, we found no discernible pattern in

[6]Random times were chosen to begin testing: May 1, April 16, April 2, and June 15.

Table 4.1: Demonstrating the number of instances (minutes) until the first appearance of concept drift using Gama et al. [83] algorithm for detection of drift

Training set size	ANR	APA	APC	BHI	Average
1000 instances	337±456	177±119	124±79	327±277	241±265
5000 instances	686±753	206±159	138±55	166±98	299±418
10000 instances	2091±2723	129±37	131±67	219±155	642±1494
15000 instances	824±1407	168±172	300±172	2226±3748	879±1981
20000 instances	519±698	101±23	201±242	462±374	320±410

the number of instances used in the training set, with neither a decrease in the first occurrence of concept drift nor an increase. Instead the occurrence of first concept drift appeared to be independent of stock, training set size, and start of the test set. This experiment demonstrates however, that a large range of concept drift was found in the dataset, with a range of *first occurrence of concept drift* appearing after 47 to 7247 instances, with a mean of 476.75 instances and a median of 149 instances (very skewed). Since the dataset uses minutes as instances, this amounts to a median of $2\frac{1}{2}$ hours. This is relevant because it demonstrates that the occurrence of concept drift cannot easily be anticipated. This brings us to our next method of learning with drifting concepts – assuming that drift occurs, without determining if it actually exists.

4.2.2.2 Assuming drift occurs

The second form of learning with changing concepts, is to *assume* that concept drift occurs without determining if it actually does (from the previous experiments, it is probably a safe assumption that stocks drift). This is done most often through

assigning a decreasing weight to older examples or through the use of a sliding window [72]. The sliding window approach constrains the training of classifiers on a moving subset of the data stream (see also Subsection 3.5.2). In a fixed-sized sliding window, a window of size n is determined *a priori* by the user. As time progresses, the model is trained using the most recent data from $t - n$, where t is the current time. The assumption is that the most recent data is most similar to the current and therefore prevents stale or outdated data from influencing the model. A small training window will therefore, in theory, reflect the current distribution better while a larger size window will contain more training instances and have a stronger ability to perform well during moments of stability. This method is not without consequences however, since if the window is too small, the classifier does not contain a large enough number of instances without over fitting; if the window is too large, the classifier may be built using data that contains too many different concepts.

Methods do exist that take into consideration the level of concept drift, by decreasing the size of the training window to allow for fast adaptability, and increasing the window size to provide a larger training set during moments of stability [123]. However, for these "adaptive window management" techniques to work, it is necessary to determine when the concept drift first appears. A nice example of this is the previously described work of Kubat and Widmer [130, 131, 232, 234] with the FLORA family of algorithms or Bifet and Gavalda [18] with the ADWIN algorithm.

There are also ensemble methods that are based on the assumption that concept drift occurs, without explicitly checking for it. The motivation behind this

method is that the most recent data may not always be the most important data to incorporate in the learning algorithms [80]. One of the first uses of ensembles in data streams was the work of Street and Kim [210] with their **S**treaming **E**nsemble **A**lgorithm (SEA) where d instances are read from a sliding window of data and used to build a classifier. This classifier is compared against a pool of previously trained classifiers (from previous sliding windows), and if it improves the ensemble quality it is included at the expense of the worst classifier. This ensemble of classifiers is then used to predict the next d instances. Wang et al. [224] constructed a weighted ensemble of classifiers with one classifier from the most recent sliding window and the other classifiers trained on past sliding windows. The classifiers are assigned a weight according to their performance achieved via cross-validation on the most recent sliding window. Gao et al. [89] provides another framework for learning from drifting concepts where drifts occur as a result of imbalanced class distributions. Their method samples from the previous (sliding window) minority class instances into the current (sliding window) training data to compensate skewed class distribution; this is done multiple times to create multiple training sets. An ensemble of hypothesis is then created from these training sets.

Ensemble methods are an important part of this research and an especially large part of the research within the wrapper framework. This will be discussed in more detail in Subsection 4.3.2 later in this chapter.

4.3 Adaptive models and wrapper frameworks

The importance of concept drift has been explained. As mentioned, there are generally two methods of tackling concept drift. The first is to find evidence of concept drift, such as through the use of novelty detection algorithms. When concept drift has been detected, the learning algorithm (e.g. classifier) is updated with the new training data. The second method is to assume that concept drift occurs and to make updates to the model at intervals with the use of sliding windows. This second method also includes the use of ensembles that combine models trained from different intervals of data.

While we have covered the theoretical methods of learning with concept drift, we have not covered how to rebuild classifiers in a manner that is efficient enough to work with high-frequency data. For example, with the first method of using a novelty detection algorithm to determine concept drift, we have discussed the process of building a new classifier using the new concept, but have not discussed in how this would work in practice. High-frequency data requires a very efficient algorithm (both in terms of speed and accuracy). The second method of learning with concept drift, makes assumptions on the nonstationarity of the data and uses sliding windows to update the learning classifier at specific intervals; however, we have not yet discussed how this is done or provided a framework. This section addresses these factors in detail with exploration of the two approaches to learning from streaming data: adaptive (online) and wrapper methods.

Adaptive models incorporate the instances into the model as they arrive,

instance-by-instance, and efficiently with a single pass through the data. An additional benefit is limited time and memory is required. *Forgetting factors* can be integrated in the model to give less weight to older data, thus *gradually* making older data obsolete. However, this ability to have the most-up-to-data classifier with the most recent data also has the potential downside of not incorporating previously instances that may useful once again in the future [224]. The wrapper-based approach uses traditional classification algorithms, such as the support vector machines or neural networks, that learn on collected batches of data (through the use of sliding windows).

4.3.1 Adaptive Models

4.3.1.1 Overview

Adaptive models, also referred to an *incremental*, *continuous*, *online*, *any-time*, and *real-time learning*, are algorithms that have been adapted to work with data streams in a real-time setting. Two characteristics of adaptive models are that they provide "any-time learning" (the model processes the data as it arrives) and they are efficient, only requiring one pass through the data. This means that the classifier is constantly up-to-date, so if the system is stopped at time t, a solution is available. Also since the data arrives incrementally (instance-by-instance), the model must be computationally efficient, hence the need for one-pass data learning [132].

4.3.1.2 Existing work

4.3.1.2.1 Very Fast Decision Tree

Domingos and Hulten in 2000 [62] introduced the **V**ery **F**ast **D**ecision **T**ree (VFDT) algorithm which is one of the more popular algorithms to use with streaming data. It works by incorporating training instances as the data arrives, versus a decision tree that learns on batches of data (such as the C4.5 algorithm) that have to be rebuilt entirely as each new training instance arrives. The batch decision tree requires the full dataset to be read as part of the learning process. Also, if the streaming data arrives too quickly, it would have to sample data to make-up for the increase in learning times, thereby losing potentially valuable information. The VFDT does not have this problem. Another benefit of the VFDT is that the output is nearly identical to the conventional decision tree.

With the VFDT, the first instances that arrive from the data stream are used to choose the split attribute at the root. Subsequent ones are then passed through the tree until they reach a leaf, and then are used to split an attribute there; this continues recursively [62, 109]. The decision of how many examples are necessary at each node is difficult considering that not all of the data is available from the beginning and are thus infinite. The use of Hoeffding inequality is used that provides an upper and lower bound of the number of examples.

A problem with the VFDT is that it assumes that the data stream is drawn from a stationary distribution (i.e. void of concept drift); this of course is inconsistent with the needs of streaming stock data as was previously discussed. The **C**oncept

Drift **V**ery **F**ast **D**ecision **T**ree (CVFDT) [109] is a modified version of Domingos and Hulten's VFDT algorithm which uses fixed-width sliding windows to adapt the model to concept drift. After building the tree, it adapts by growing new sub-trees and disregarding old sub-trees [223].

VFDT has also been used with adaptive (variable length) sliding windows to learn on data streams with high concept drift, such as Bifet and Gavalda's **Ad**aptive **Win**dowing (ADWIN) algorithm [18]. This algorithm increases the window size during moments of low concept drift to include more training instances and decreases the size of the sliding window during moments of high concept drift to include only the instances in the most recent drift.

4.3.1.2.2 Exponential fading of data

Law and Zaniolo [139] make the assumption of concept drift by weighting the more recent classifiers in an ensemble more than older ones in their **A**daptive **N**earest **N**eighbor **C**lassification **A**lgorithm for **D**atastreams (ANNCAD). Because the nearest neighbor algorithm is computationally expensive and often slow (i.e. it is a *lazy learner* that learns at run time), ANNCAD speeds up the process by dividing the feature space into discretized blocks of equal size. For each class, the number of training instances are counted within a single feature-space block. If the training set is unable to correctly differentiate between the classes according to a pre-determined threshold value, a coarser level of division among the feature space is created (e.g. going from fine 4×4 to a coarser 2×2). To adapt to concept drift, the model then

uses exponential fading to give less weight to older data, thus *gradually* making older data obsolete. In fact, this *gradual* forgetting period is one of the weaknesses of this model; sudden concept drift may go unnoticed.

4.3.1.2.3 Online bagging and boosting

Ensembles often outperform their base models (the classifiers that comprise the ensemble) so it is only natural that an ensemble has been adapted to work real-time with streaming data[7]. However, most ensemble algorithms require batch learning, or the repeatedly reading and processing of the entire dataset. For each base classifier, one pass through the dataset is required, making the ensemble classifier unwieldy to use for large streaming datasets. Oza solves this problem in [175] by creating an online version of the Bagging and Boosting algorithms. This online version requires only one processing of the training set regardless of the number of classifiers in the ensemble; thus eliminating the need to store the training example for reprocessing, since the model contains all of the training instances seen thus far [173].

In batch (non-online) bagging, since sampling with replacement is done, any particular training set instance may be seen multiple times within the bootstrapped samples. This is a binomial distribution, with K copies of each of the N original training examples in each sample. The probability of seeing any specific training example:

$$P(K = k) = \binom{N}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{N-k} \tag{4.1}$$

[7]For a theoretical explanation of why ensembles work, please see Subsection 3.3.7.

The requirement in a batch setting is that the entire dataset of size N is finite and supplied. In a streaming setting, the value of N can be assumed to be infinite and hence the value is unknown. This makes the normal implementation of bagging not possible without modifications.

As can be seen in Equation 4.2, as $N \to \infty$, p is small (i.e. $p = \frac{1}{N}$), and $\lambda = Np$, the binomial distribution, $\text{Bin}(N, p)$, can be approximated by $\text{Poisson}(\lambda = 1)$. The "online" modification of bagging, for each classifier, takes each new labeled data point and samples from the Poisson distribution ($\frac{1}{k!}e^{-1}$) to find the k, or the number of copies of the new data point to place in the training set. The online and batch bagging method continue the same from here with a majority vote determining the label of the new data point.

$$\begin{aligned}
\binom{N}{k}(p)^k(1-p)^{N-k} &= \frac{N(N-1)\dots(N-k+1)}{k!}\left(\frac{\lambda}{N}\right)^k\left(1-\frac{\lambda}{N}\right)^{N-k} \\
&\approx \frac{\lambda^k}{k!}\left(1-\frac{\lambda}{N}\right)^N \\
&\approx \frac{\lambda^k}{k!}e^{-\lambda} \quad \text{when } N \text{ is large} \qquad (4.2)
\end{aligned}$$

Online bagging provides similar results to batch (non-online) bagging when using the same classifier and if the distribution is the same among the training sets [173, 175].

As explained previously in Chapter 3, Section 3.3.7.2, batch boosting re-adjusts the weights (at each iteration) according to the coinciding classifier. Instances incorrectly classified receive larger weights and correctly classified ones receive lesser weights. Oza describes in [175] his online version of boosting, which uses the Poisson

distribution to do sampling with replacement in a similar fashion as was done in his online bagging algorithm [173]. For example, the online AdaBoost method takes a sequence of base classifiers $\{h_1, \ldots, h_M\}$ and the parameters $\{\lambda_1^{\text{correct}}, \ldots, \lambda_M^{\text{correct}}\}$ and $\{\lambda_1^{\text{incorrect}}, \ldots, \lambda_M^{\text{incorrect}}\}$ for the sum of the correctly and incorrectly identified examples for each of the M base classifiers. The algorithm's output is a new classification function that has parameters λ^{correct} and $\lambda^{\text{incorrect}}$ that contain the updated base models h. The training example (x, y), in its first iteration, is given a weight of $\lambda = 1$. At each of the following iterations one of base models is updated by choosing a k according to the Poisson (λ) distribution $\frac{\lambda^k}{k!}e^{-\lambda}$ and is updated k times using (x, y). If h_m is correctly identified then λ^{correct} is increased otherwise $\lambda^{\text{incorrect}}$ is increased. This process is repeated for all M base models.

To illustrate online bagging and boosting and the similar results to the normal implementation of batch bagging and boosting respectively, we examined the algorithms on a simulated dataset called Waveform-21 generator. This dataset contains $100,000$ instances, 21 attributes and one predicted attribute containing 3 classes. All tests were run using the Massive Online Analysis (MOA) [19] and Weka java libraries [96].

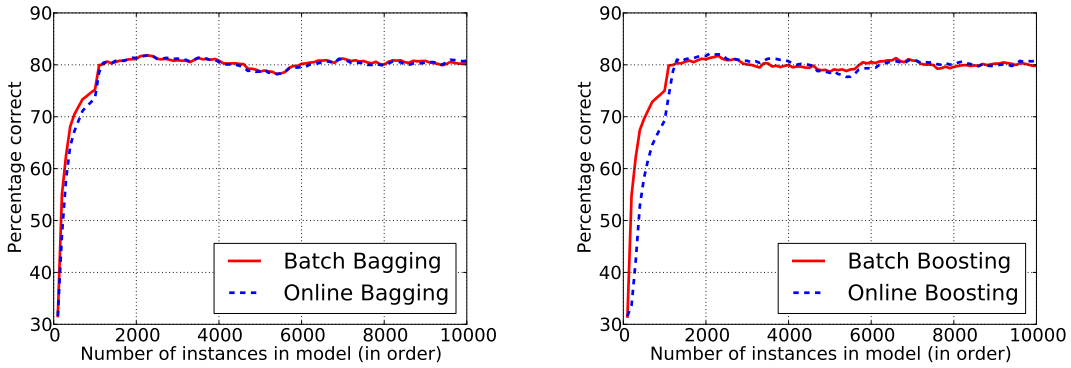
The experiments used five synthetically created, randomly ordered Wavelet datasets (the order of the instances matters to the online implementations) and all ensembles were comprised of ten Naïve Bayes base classifiers. All experiments began after the first 10,000 instances and continued training (using anywhere from 100 to 10,000 instances in the model) and testing using the next 90,000 instances with a

prequential evaluation technique (every 100 instances). Each experiment was conducted on five separate simulated datasets and the results were averaged over the five datasets. The results comparing the online and batch versions of the bagging and boosting algorithms can be seen in Figure 4.7.

For bagging, using 10,000 instances for training in the classifier, the percentage correctly classified [8] was 80.44[±2.12] and 80.78[±2.09] for batch and online respectively; it was not statistically different at the 95% level. Then for boosting, using 10,000 instances for training in the classifier, the average percentage correctly classified was 80.74[±1.99] and 80.86[±2.03] for batch and online respectively; also not statistically different at the 95% level. Additionally, as explained previously, the advantage of the online versions of bagging and boosting is that the algorithms require only one pass through the dataset rather than multiple passes for batch implementations.

For a more thorough analysis (more datasets), please see [20, 174]. The main problem with this online method of boosting and bagging, is its inability to deal with drifting concepts since it assumes a stationary distribution [49]. Bifet et al. [18] provides an additional methods to enable Oza et al.'s ensemble methods to work with concept drift by using sliding windows. Please see that paper for more information.

[8] The results were averaged over the five datasets.



(a) Batch versus online bagging (b) Batch versus online boosting

Figure 4.7: Comparisons of batch and online bagging and boosting using a simulated dataset

4.3.2 Wrapper Frameworks

4.3.2.1 Overview

Whereas adaptive methods are algorithms that have been adapted to work with data streams in a real-time setting, the wrapper framework reuses existing schemes [121]. Instead of increasing the computational speed of the classifier like the adaptive methods (e.g. VFDT), the wrapper method is more of a framework that uses traditional classifiers learned on chunks (i.e. subsets) of collected data. The data must be chosen in such a manner that allows for enough data for generalization, yet not too much so the learning time makes for an impractical algorithm for streaming data. Additionally, a training set that is too large may include too many concepts, thereby reducing predictability. However, if the dataset is too small, it will produce a model too quickly with poor generalizability and therefore, poor performance.

Another difference between adaptive and wrapper methods, is that adaptive

methods alone do not adapt to concept drift. Instead they use exponential fading of data [139] or change detection algorithms based on some form of sliding windows [18, 109]. The use of change detection algorithms to determine when to update models makes for a costly model updating procedure. Once concept drift is detected, old data is eliminated and the model is built from scratch with the new concept's data. This reliance on change detection algorithms often make the use of adaptive models, which are built for speed, slower than they are in batch mode [224].

Wrapper methods offer additional advantages over adaptive ones, including the ability to learn from data streams using traditional classifier types (i.e. artificial neural networks, support vector machines, random forest, etc.) as well as the ability to parallelize easily. A description of existing research on the wrapper framework follows, which sets the stage for our new framework for predicting stock direction in Chapter 6.

4.3.2.2 Existing work

One of the first and most popularly cited wrapper methods in data streams is the **S**treaming **E**nsemble **A**lgorithm (SEA) by Street and Kim [210]. In their algorithm, d instances are read from a recent sliding windows D_t and used to build classifier C_i (the authors use the C4.5 decision tree algorithm). The previous classifier C_{i-1} built on the previous sliding window D_{t-1}, is then evaluated on chunk D_t and this is compared with the accuracy of the existing ensemble E (built with a pool of previously trained classifiers). If the accuracy of classifier C_{i-1} is better than the

existing ensemble E, then the worse base classifier in the ensemble is replaced.

The dynamic nature of the SEA method in updating the classifiers in the ensemble creates diversity, which minimizes the effects of concept drift. This is because the poorest performing classifiers are replaced by higher performing classifiers. This can also be a drawback, since old concepts will gradually be forgotten.

Wang et al. [224] provided another wrapper method consisting of a series of classifiers formed into an ensemble. They construct a weighted ensemble of classifiers with one classifier from the most recent sliding window and the other classifiers trained on past sliding windows. The classifiers are assigned a weight according to their performance achieved via cross-validation with the most recent sliding windows. Their results found that the use of an ensemble worked better than training a single classifier on the same amount of data. The authors rationalize their use of old data in the model, stating that "maintaining a most up-to-date classifier is not necessarily the ideal choice, because potentially valuable information may be wasted by discarding results of previously-trained less-accurate classifiers."

Fan [72] continued where the Wang et al. [224] paper stopped by addressing the question – is it better to train using new data alone *or* new and old data (and how much old data)? Using a series of tests, he concluded that the use of old data "definitely helps" when the data stream contains no concept drift and if the new data is insufficient. When concept drift does exist, old data helps if the new concept and old concept share consistencies. The author discusses a framework for choosing ideal old data for model building.

The framework works by using cross-validation on old data to find the portion of old data that complements the most recent data the most. This data is then used to build a classifier and combined with the most recent classifier to form an ensemble. A critique of the Fan [72] framework observed by [46] is the high level of granularity of cross-validation used. More splits in the validation set (finer granularity) would more accurately provide the desired proportion of old data, however this comes with an increased computation time. The finer the granularity, the more it becomes a brute force method, thus making it undesirable for high-speed learning.

Rushing et al. [194] propose another wrapper method, the **C**overage **B**ased **E**nsemble **A**lgorithm (CBEA). The algorithm is similar to Street and Kim's SEA, but the authors note that SEA has problems when learning from drifting concepts with unevenly distributed datasets. CBEA uses an approach different from SEA for deciding which classifiers to use and which ones to discard. Specifically the CBEA framework keeps information relating to the range of possible values and the age of the classifier as a means of keeping a variety of models covering a larger range of values. Newer classifiers are preferred over older ones and classifiers with the most similar *coverage* overlap (and are oldest) are discarded first. The algorithm chooses the classifiers using k nearest neighbors. For example, the most recent batch of data is compared with the training sets that built previous classifiers. The instances that are nearest to the most recent batch of data (according to k nearest neighbors) are identified. The classifiers that used these instances in their implementations receive one vote; therefore one classifier may get multiple votes or even all of the votes. The

idea is to prevent classifiers from being used on data dissimilar to data on which it trained.

Chu and Zaniolo [49] provide another wrapper method that is both a *fast and light* boosting algorithm that uses ensembles to learn on drifting concepts. The method first works by breaking the data into blocks of equal size; then similar to AdaBoost, the algorithm assigns larger weights to misclassified samples. The weights of the samples are normalized and a classifier is then built on this weighted training block.

Each classifier arrives at a prediction and a mean rule combines the probability of the predictions; the class with the highest probability is the solution (see Subsection 3.3.7.3 for more information on "Combining classifiers for ensembles"). To deal with drifting concepts, this wrapper method actively detects changes and discards old ensembles when found. A downside to this method is old classifiers are deleted; thus (possibly) useful knowledge is eliminated.

An approach by Brzezinski and Stefanowski [31], the **A**ccuracy **U**pdated **E**nsemble algorithm (AUE2), uses incremental Very Fast Decision Trees (VFDT) within a wrapper framework. The VFDT incrementally builds classifiers C_j on evenly sized chunks $B_1, B_2, \ldots, B_n$ each containing d instances. For every incoming chunk B_i the classifiers are evaluated and ranked. The worst performing classifier is replaced with a classifier C_r, deemed "the perfect classifier", that is build on the particular chunk of data that the other classifiers were tested on (the most recent chunk of data). All classifiers are then weighted with a formula that provides a larger weight on the

best performing classifiers and an even heavier weight on "the perfect classifier." The classifiers chosen are now updated incrementally and they become part of a weighted ensemble.

4.4 Performance and efficiency

Changing market dynamics (i.e. concepts) create a need for a methodology that updates training data and/or models. Large amounts of data streaming in at high speeds also require this methodology to be efficient. As mentioned in the introduction to this thesis, an algorithm is of little value to the trader if it takes thirty minutes to arrive at a solution for the prediction of stock price direction one minute in the future. Gaber et al. [80] discusses a tradeoff between speed of the algorithm and the performance, with a shortening of algorithm learning times tending to cause decreases in algorithm performance, such as accuracy. This is generally true, with individual classifiers most often performing worse than ensembles (with longer training times). Additional time also gives us more time to search, often through brute force, for a global optimum. We demonstrate however, that this is often a more complicated discussion; more data leads to greater training times, which generally leads to better results, but there is a point of diminishing returns where more data may lead to worse results.

Increases in training times do not necessarily improve performance. We demonstrate this with the use of a support vector machine (polynomial degree kernel) in an experiment with the stock ANR from our dataset (i.e. our stock problem of predicting

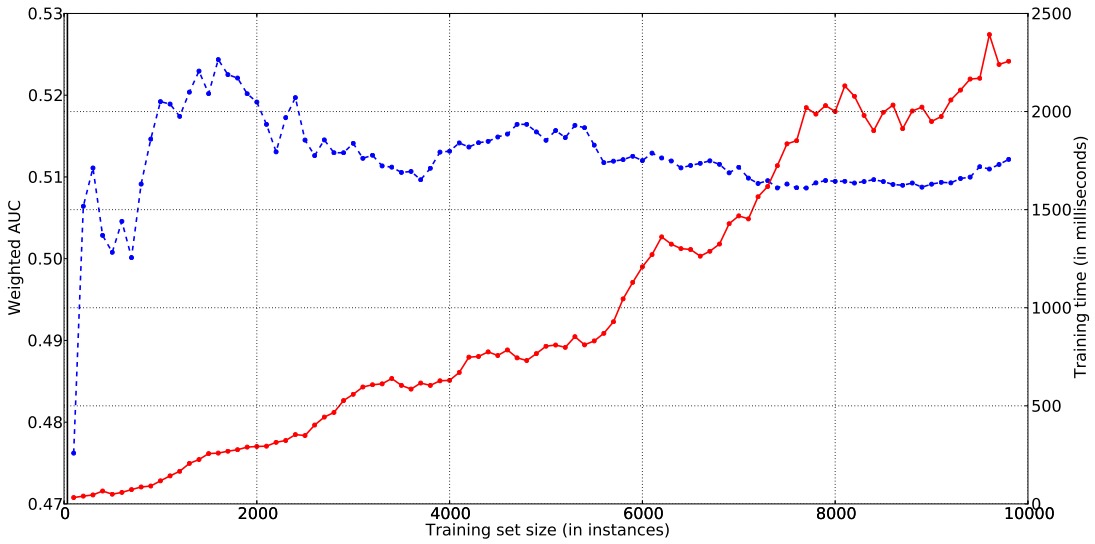


Figure 4.8: Demonstrating performance (blue) and increases in training times (red) due to increases in instances

up, *down*, or *no change*). An increase in the number of instances increase weighted AUC[9] to a point (see Figure 4.8) and longer training times did not necessarily equate to better performance. A possible explanation is that the larger dataset was including too many different concepts, which decreased predictability. We also performed this test multiple times with different stocks and received different cutoffs at which more data began to decrease profitability. This illustrates that more data is not always better when training classifiers for predicting future stock direction.

[9]AUC is generally for two class problems. Weighted AUC [228] however, is the class prior probability weighted AUC. The dataset that we use is a three class problem so therefore we use weighted AUC.

4.5 Conclusion

In Section 4.2.2 the definition of concept drift was discussed and its negative influence on stock price prediction was demonstrated in an experiment. Two methods for learning with drifting concepts are: 1) explicitly detecting concept drift and upon detection adapt the classifiers by retraining and 2) assume that concept drift occurs and build classifiers at fixed width sliding windows or through the use of an ensemble (where base classifiers are updated at specified intervals). Existing research covering both methods were discussed. Our framework to be discussed in Chapter 6, follows this second approach. We also initiated an experiment using the drift detection methods of Gama et al. [83] to determine when drift occurs within our stock dataset.

We next examined two model approaches to adjust for concept drift: the *adaptive* and *wrapper framework*. The adaptive model provides a solution for solving the first method of learning with concept drift (i.e. retraining after detecting concept drift) within the constraints of the time needed for prediction. An example of an adaptive model is the popular Very Fast Decision Tree by Domingos and Hulten [62]. The fast training times of adaptive models provide for a reasonable solution when time constraints require a fast decision. However, adaptive models comprised of single classifiers are generally outperformed by approaches based on ensembles of classifiers (as we showed theoretically in Subsection 3.3.7). Also, adaptive models that work with concept drift are built on the assumption that the most recent data is more valuable for training than earlier data. With stock data, as we will explore in Chapter 5, this assumption is questionable. The second approach is the wrapper

framework which requires lengthier training times than adaptive classifiers (which are built for speed), but this is somewhat offset by the need to only train model periodically. Wrapper frameworks also offer a distinct advantage of adjusting for concept drift automatically. Furthermore, problems pertaining to stock datasets, such as class imbalance and dimensionality reduction, are more easily solved using full subsets of data available with wrapper frameworks. The next chapter discusses this further along with additional solutions for optimizing and decreasing learning times with stock data.

CHAPTER 5
ADDRESSING PROBLEMS SPECIFIC TO STOCK DATA

5.1 Imbalanced data streams

5.1.1 Overview

In the previous chapter we discussed different approaches to learning from data streams and overcoming concept drift; however, this was under the assumption of having a balanced data stream with an equal number of predicted classes. It is more realistic for *stock* datasets to have imbalance, with more instances of a particular class than others (see Figure 4.5). Learning from such a dataset usually results in a classifier having a bias toward the majority class; thus the classifier tends to misclassify the minority class instances. This is due to the rules predicting the larger number of instances being strongly weighted to favor accuracy[1], and therefore cause the instances belonging to the minority class to be misclassified more frequently [155]. In a highly imbalanced dataset, a classifier may have high accuracy while misclassifying most of the minority classes.

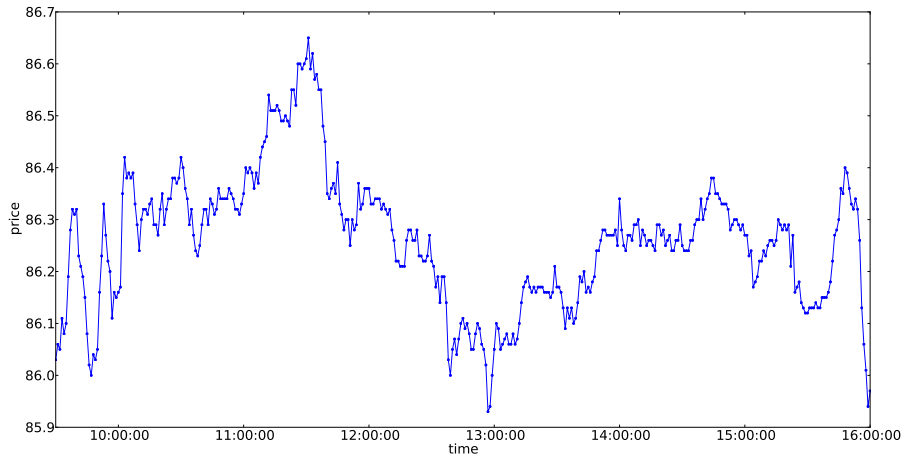
When predicting stock direction we are most interested in the large movements (most often the minority class); moves of a few cents are relatively unimportant in predicting stock direction and less profitable. For example in Figure 5.1, the minute-by-minute stock price of Exxon (symbol: XOM) is shown on January 3, 2012 with the price in Figure 5.1a and the difference of the price at time t from time $t-1$ in

[1]From Section 3.4, accuracy or $\frac{TP+TN}{TP+TN+FP+FN}$, is the total number of correct predictions made over the total number of predictions made.

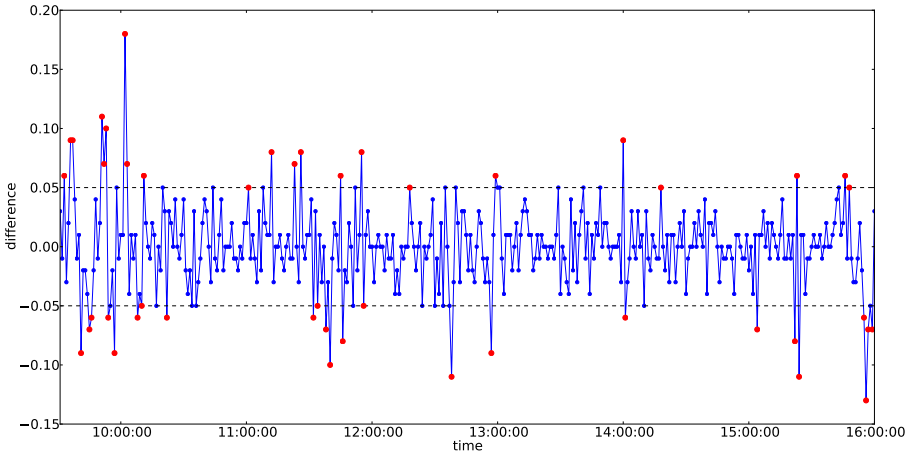
Figure 5.1b. The movements outside of a \$0.05 difference (i.e. $|\text{price}_t - \text{price}_{t-1}| >$ \$0.05) make up only 11.8% of the moves on the particular day, yet those moves are of most interest to a trader. Making this a three class problem, we would have $\left(\text{price}_t - \text{price}_{t-1}\right) >$ \$0.05 comprising of 5.6% of movements, $\left(\text{price}_t - \text{price}_{t-1}\right) <$ $-$\$0.05 comprising of 6.1% , and $0.05 \geq \left(\text{price}_t - \text{price}_{t-1}\right) \geq -$\$0.05 comprising of 88.2% movements. This can be observed in Figure 5.1c.

Stock data streams, as we have previously discussed, suffer from both concept drift and class imbalance. According to [104, 105], one of the advantages of wrapper approaches is their ability to deal with reoccurring concepts. Thus ensembles are often built from past subsets (batches) of data that can be reused to classify new instances from the new concept, similar to the class distribution of the old concept. Another advantage of wrapper methods is, since they are built on batches of data using traditional classification algorithms, proven strategies to overcome imbalance can be used. This includes adding bias along with over- and under-sampling. For example, one of the simplest methods to overcome imbalance is to over-sample the minority class by randomly sampling the minority class instances with replacement. Another method is to under-sample the majority class; however, this results in the loss of knowledge that could have been useful.

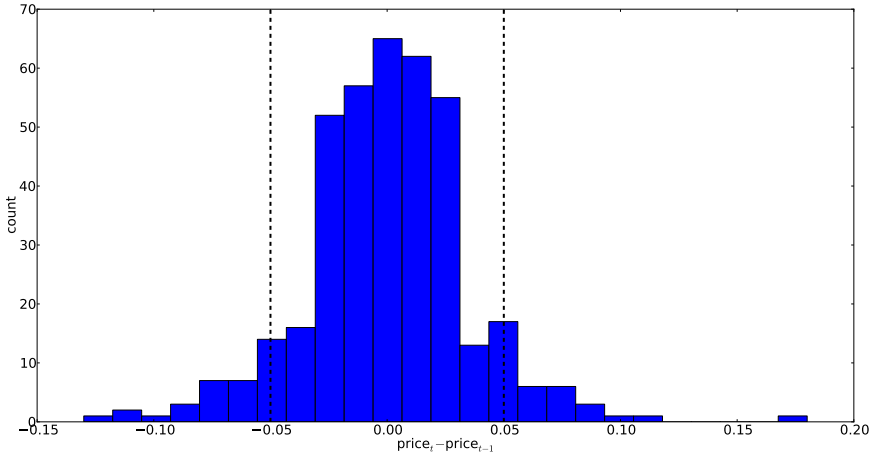
In this section we describe strategies and algorithms for working with imbalanced datasets for streaming data.



(a) Price at time t



(b) Difference of price t from time $t-1$



(c) Histogram of difference of price t from time $t-1$

Figure 5.1: Demonstrating the price change of symbol XOM on January 3, 2012

5.1.2 Strategies

5.1.2.1 Over- and under-sampling and synthetic training generation

Gao et al. in [88, 89] tackles imbalanced data streams (using a wrapper framework) in a relatively simple way. In their method, the incoming data stream arriving in sequential chunks is represented as $S_{t-i}, \ldots, S_{t-1}, S_t, S_{t+1}$ with S_t as the most up-to-date chunk and S_{t+1} representing the next chunk. The arriving set S is then split into the (minority) positives $P_{t-i}, \ldots, P_{t-1}, P_t$ and the (majority) negatives $N_{t-i}, \ldots, N_{t-1}, N_t$. The classifier is trained using the most recent negative instances $\{N_t\}$ and the most recent positive instances going back as far as needed to get a more balanced distribution $\{P_{t-i} \ldots P_t\}$ of positives and negatives. Additionally, the training set comprised of negative instances $\{N_t\}$ is under-sampled.

After the work of Gao et al. [88, 89], other researchers developed modifications to the selection of the minority classes from previous chunks of data. For example, in Chen and He [45] the authors develop **SE**lectively **R**ecursive **A**pproach (SERA) to work on unbalanced datasets with concept drift. Their method is similar to [89], but instead of taking random positive (minority) examples from previous chunks of data, they choose the positive instances based on similarity to the current training data chunk. Chen et al. [47] then expanded their SERA method with **Mu**ltiple **S**electively **R**ecursive **A**pproach (MuSERA) by using a hypothesis built on every training chunk and built over time, as opposed to their original SERA method which kept only one hypothesis.

Another method is used by Liu et al. [151] in their **R**eusing **D**ata **F**or **C**lassifying

Skewed **D**ata **S**treams (RDFCSDS) algorithm to partition the data $\{X_t, y_t\}$ into sequential chunks of equal sizes $S_{t-i}, \ldots, S_{t-1}, S_t, S_{t+1}$ with S_t as the most up-to-date chunk and S_{t+1} representing the next chunk. Sampling is used to create balanced datasets similar to the previously cited Gao et al. [89] and classifiers in each chunk are used to compose an ensemble E_t at the current time t. If the expected value of the ensemble is not equal to the actual value, then concept drift has occurred and only new classifiers are used in the data. If concept drift has not occurred, then classifiers from previous chunks are chosen according to their AUC on the current chunk of data.

An additional method, that is popular for dealing with imbalanced data, artificially creates the minority class based on similarities between existing examples in the dataset. This is the **S**ynthetic **M**inority **O**versampling **T**echniqu**e** (SMOTE) by Chawla et al. [43] (and with an addition of a boosting algorithm is SMOTEBoost [44]). A regular boosting algorithm without synthetically generated minority instances would have a learning bias toward the majority class cases; SMOTEBoost reduces the bias from class imbalance by increasing the weights for the minority class. In each round of boosting, synthetically generated minority class instances are created, and by doing so, increase the probability for the selection of the minority class.

These methods of over- and under-sampling along with synthetic generation approaches can be implemented with data streams through the use of sliding windows (similar to modifying adaptive methods to work with concept drift). In these

approaches, classifiers that are trained on old data are discarded and new classifiers are built on new data as needed. For adaptive (online) learning, the SMOTE algorithm is used by Ditzler et al. [60] with their Learn++.NSE[2] algorithm; more information can be found in that paper.

There are drawbacks to sampling methods however, such as the uncertainty of how much over- and under-sampling to apply. According to Hoens in his dissertation on learning with imbalance [105], it would be ideal to promote the learning of the minority class with over-sampling without over-fitting the model to the data. Additionally, under-sampling might not retain information knowledge about the majority class. One possible solution is through careful use of training sets and validation sets (i.e. build the model on the training set and carefully test on a validation set). There is some evidence however [50], that this is not as effective as building ensembles of classifiers that work without the need for sampling. As we discussed previously, Hoens et al. [104, 105] writes that one of the advantages of batch approaches is their ability to deal with reoccurring concepts – to learn on past batches of data and reuse this information to classify new instances on new concepts with similar class distributions as old concepts.

5.1.2.2 Cost-based solutions

Cost-based learning is another method for adapting to concept drift. With over- and under-sampling, the training set is modified to account for imbalanced

[2]Learn++ [181] is a family of algorithms for incremental, online learning. Learn++.NSE stands for **N**on**s**tationary **e**nvironment.

datasets; however, with *cost-based learning*, the classifier is modified. For example, with traditional classification, the *cost* of error is uniform; this is not very realistic in practice since some errors are more expensive than others (see Section 3.4.5). Through the use of a cost-matrix (see Subsection 3.4.5), penalties can be implemented for misclassifying instances. The classifier minimizes the *cost* of misclassification rather than the misclassification errors (through optimization). Instead of creating an imbalanced data stream through sampling, cost sensitive learning targets the imbalance by using different cost matrices [102]. An imbalanced dataset can therefore be thought of as a cost-dependent problem; the error of minority misclassification can receive higher cost than a misclassification of the majority class.

According to [155], three cost-based solutions have been described. The first is the use of sampling (both over- and under-sampling) according to the weight of the class considered cost matrix. For example, if the bias is toward avoiding errors on the "large move" instances (because the errors are penalized more), the "large move" instances can be sampled at a much higher weight [236]. In addition to sampling, modifying the decision threshold can also be used in proportion to the cost matrix. The second solution is by changing the process by which the classifier learns. For example, a cost matrix can influence an attribute to split data when building a decision tree or can determine if a subtree can be pruned. The third cost-based solution is based on Bayes decision theory that assigns instances to the class with minimum expected cost; an example of this is MetaCost [61, 155].

MetaCost [61] builds separate classifiers on individual bootstrapped training

sets (with replacement) and then modifies the voting threshold for the classes (i.e. using each class' fraction of the total vote as an estimate of its probability given the example). Because of its good results, MetaCost is one of the most popular methods [236].

Overviews for learning from imbalanced data streams can be found in [91, 104, 105].

5.2 Preprocessing of data

5.2.1 Overview

Data preprocessing is the process of preparing the data for use in a classifier to make the model more robust. Electronic trading has increased the reliability of trading data from the past, when trades were executed without the use of computers and clerks inputted the transaction price and volume amount manually. However, preprocessing is still an important task required for the analysis of stock movements to ensure the integrity of the models built. This includes cleaning the data, determining the need for more or less data, transforming the data for use in a specific classifier, de-trending, among others [184].

The New York Stock Exchange (NYSE) has offered ultra high-frequency datasets since the 1990s, which is marketed as TAQ trade data (Trades and Quotes). This is the finest granularity of data with each trade-by-trade transaction being recorded at non-uniform increments. See an example of TAQ data in Table 5.1[3]. Data from

[3]SYMBOL is the stock symbol; PRICE is the traded price; SIZE is the number of shares traded; G127 stands for "Combined G Rule 127" a NYSE rule for *special* trades; COND is

Table 5.1: TAQ trade data

SYMBOL	DATE	TIME	PRICE	SIZE	G127	CORR	COND	EX
IBM	20050103	12:17:28	98.06	100	40	0		N
IBM	20050103	12:17:29	98.07	500	40	0		N
IBM	20050103	12:17:33	98.06	400	40	0		N
IBM	20050103	12:17:37	98.64	1700	0	0	B	T
IBM	20050103	12:17:45	98.06	100	40	0		N
IBM	20050103	12:17:56	98.06	300	40	0		N
IBM	20050103	12:17:58	98.05	400	40	0		N
IBM	20050103	12:17:58	98.05	400	40	0		N

other providers, such as Bloomberg and Reuters, also provide the stock price data in times of 1 second, 1 minute, 5 minute, daily, weekly, etc.

In this section, we discuss the need for preprocessing data. This includes finding and dealing with noise and bad trade data, deciding what to do with too much and too little data, and methods for standardizing and normalizing streaming data.

5.2.2 *Bad* trade data and noise

The importance of preprocessing can be seen in Figure 5.2, which shows every transaction for the stock IBM on January 3, 2005 from 12:04 pm to 12:21 pm. Prices move within a small range until approximately 12:17 at which an outlier can be seen priced \$0.58 away from the previous transaction. This outlier transaction is an example of a conditional bunched trade (shown as a B under COND in Table 5.1). This occurs when a trader combines several small orders into one, resulting in a price

the condition of the sale; and EX is the exchange the trade took place.

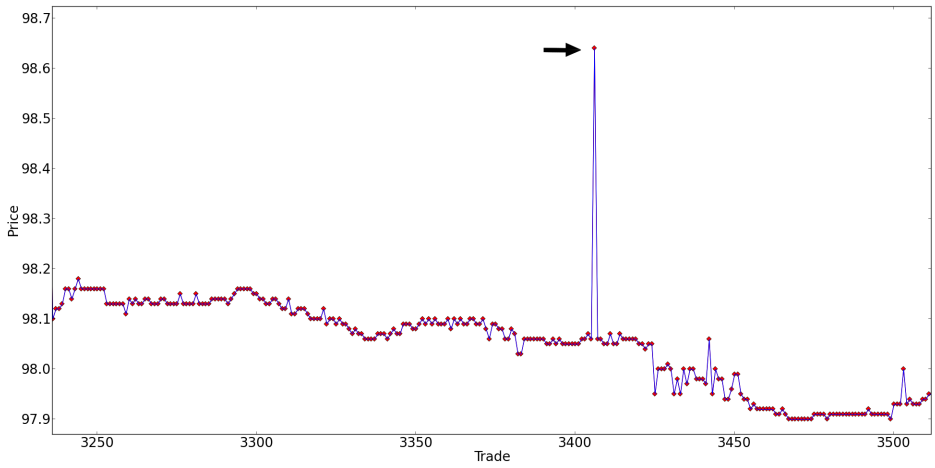


Figure 5.2: Demonstrating IBM stock price with *bad* trade (January 03, 2005)

away from the current traded price (in this example, the bunch trade occurs \$0.58 away from the previous traded price). Many researchers [30, 70, 101, 190] recommend the removal of conditional trades, late-trades, trades reported out-of-sequence, or trades with special settlement-conditions. Not removing these trades (such as the one in Figure 5.2) result in noise and incorrectly calculated signals, many of which are discussed later in this paper. Thus the removal is extremely important, not only to improve model performance, but also (and perhaps more importantly) to prevent an automated trade from activating, based on a false signal.

Brownlees and Gallo [30] find noise using a heuristic shown in Equation 5.1. Price (tick-by-tick) in the formula is represented as $\{p_i\}_{i=1}^{N}$, with $\bar{p}_i(k)$ and $\sigma_i(k)$ the sample mean and standard deviation of the preceding k prices respectively. A granularity parameter γ is a constant. Because prices from preceding days may be very different from the current, the formula only uses observations from the current

day.

$$(|p_i - \bar{p}_i(k)| < 3\sigma_i(k) + \gamma) = \begin{cases} \text{true : observation } i \text{ is kept} \\ \\ \text{false: observation } i \text{ is removed} \end{cases} \tag{5.1}$$

The formula is a heuristic, with the choice of parameter having a significant difference in outcome. The Brownlees and Gallo paper recommends the size of k to be small for lightly traded stocks and larger for more highly traded ones. The choice of γ should be some variation of the price volatility.

Brownlees and Gallo report only the number of noisy trades that are removed but do not compare their results with a benchmark, but instead state "the judgment on the quality of the cleaning can be had only by a visual inspection of the clean tick-by-tick price series graph." While a manual visual inspection of each data point could be valuable, we believe it's helpful to have a quantitative benchmark against which to measure. In our own experiment of Brownlees and Gallos' algorithm, we chose the transactions labeled by the New York Stock Exchange as *out-of-sequence*, or those prices that were transacted at an earlier time as a benchmark. Also, since not all transactions that are labeled *out-of-sequence* are away from the surrounding prices, we test how well the algorithm works when the *out-of-sequence* trades are more than \$0.03 away from the previous non *out-of-sequence* trade. Using data for the stock SPY from the first week of 2005, the results for a subset of the data can be seen in Figure 5.3. Our full experimental results can be found in Table 5.2.

From Table 5.2, as can be expected, the algorithm detects *out-of-sequence* trades that are \$0.03± from the previous trade more readily; however at the al-

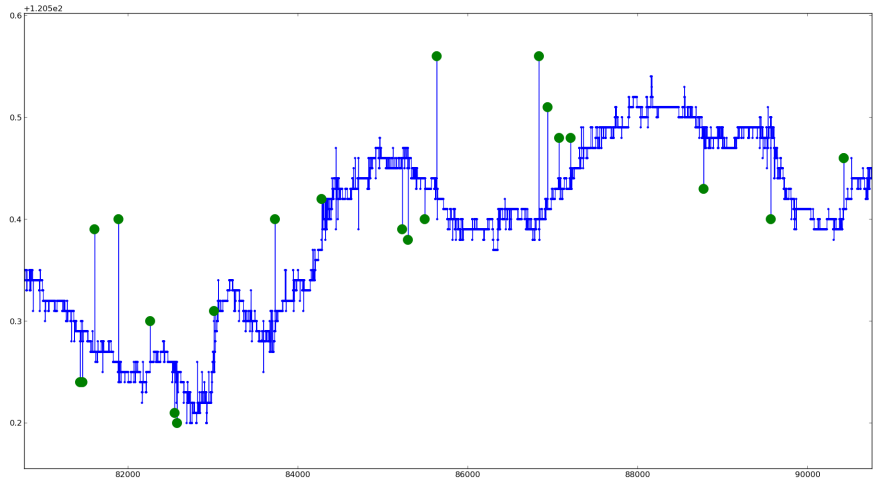


Figure 5.3: A subset of our experimental results of finding noise with the Brownlees and Gallo algorithm (noise as determined by algorithm has a green dot)

Table 5.2: Our experimental results of the Brownlees and Gallo algorithm to detect actual out-of-sequence trades

		out-of-sequence trade			$\lvert$out-of-sequence difference$\rvert \geq \$0.03$		
γ	k	Sensitivity	Precision	Accuracy	Sensitivity	Precision	Accuracy
0.02	25	0.157	0.311	0.998	0.345	0.311	0.999
	50	0.239	0.221	0.998	0.527	0.220	0.998
	100	0.280	0.203	0.998	0.600	0.197	0.998
	150	0.239	0.171	0.998	0.527	0.172	0.998
	200	0.215	0.153	0.998	0.470	0.154	0.998
0.06	25	0.060	0.615	0.999	0.145	0.615	0.999
	50	0.115	0.538	0.999	0.255	0.538	0.999
	100	0.124	0.468	0.998	0.272	0.468	0.999
	150	0.124	0.480	0.998	0.272	0.483	0.999
	200	0.124	0.440	0.998	0.272	0.441	0.999

gorithm's "best" of $k = 100$ and $\gamma = 0.02$, only 60% of the transactions are found (sensitivity) and only 19.7% of the algorithm's predictions are correct. The algorithm would therefore remove many *in-sequence* trades.

Additional methods for reducing noise include increasing the sampling interval from irregular tick-by-tick data to longer intervals such as 1, 2, 5 minute [172], or using volume-weighted averaging [190, 217] over longer intervals of time. This will be discussed more in the next section.

While research has shown the ability of models to learn with greater predictability when noise is removed, Falkenberry [70] indicates the need to be careful not to eliminate actual market conditions which may be highly volatile and chaotic at times. This is a problem: reducing noise while not eliminating volatile market conditions which may distort the remaining data.

5.2.3 Too much and too little trade data

Trade data does not often arrive at regular intervals (see Figure 5.4), which makes it difficult to build classifiers. The abundance of trade data is a concern when the speed of incoming data is higher than the model can efficiently use (i.e. in online adaptive models) and when the size of data becomes a constraint on storage resources. Transactions can also occur sporadically and too little data may result in a model with poor generalizabilty. Both problems of *too much* and *too little data* require efficient solutions.

One solution when faced with too much data is to employ sampling methods or

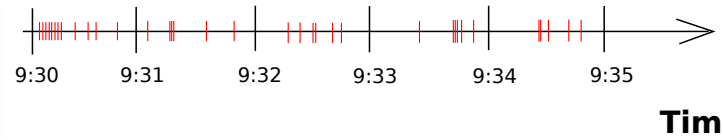
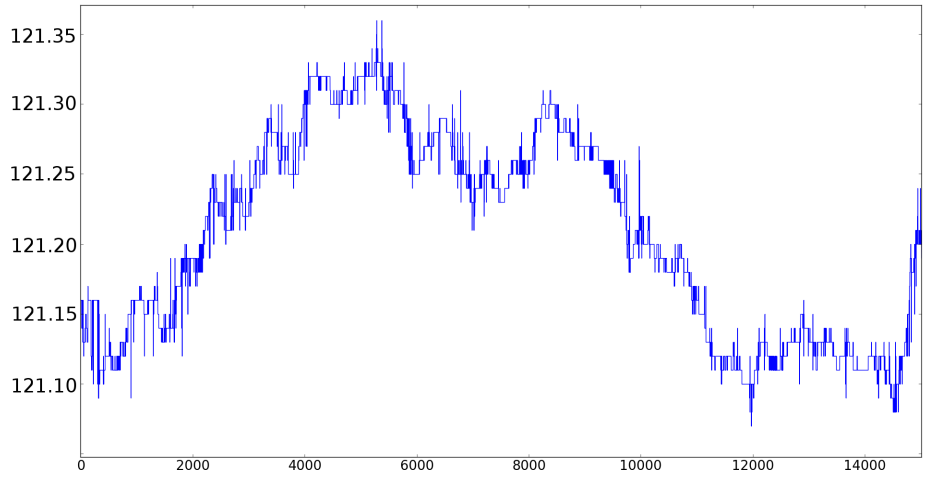


Figure 5.4: Trades (transactions) often arrive at irregular time, thus causing problems when building learning algorithms

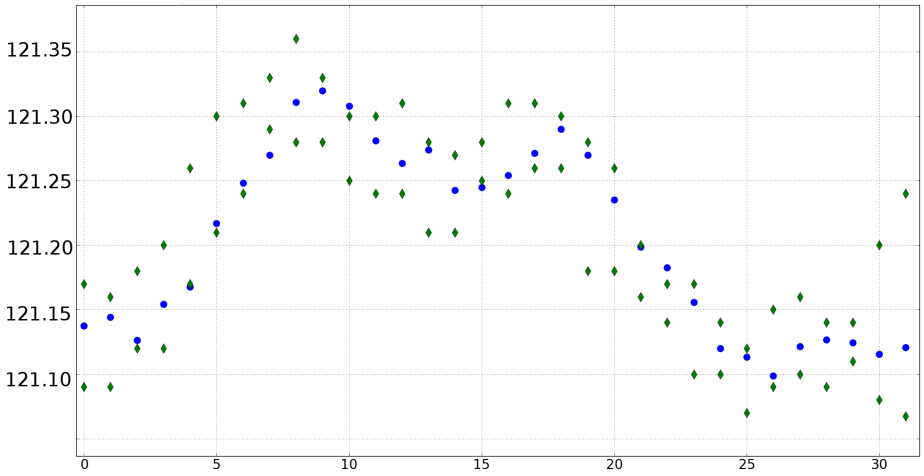
to transform irregularly spaced tick-by-tick observations to regularly spaced intervals such as 1 second, 1 minute, etc. [29, 70]. The latter is perhaps one of the more common methods in literature we have explored and one that we use in this paper[4]. To demonstrate this transformation and reduction of data, in Figure 5.5 the stock SPY is shown with tick data in Figure 5.5a and then below in Figure 5.5b the reduced granularity 1 minute data. Because tick data is irregular, during the first 10 minutes of the trading day nearly 6000 transactions occur, while during 9:50 to 10:00 a.m. fewer than 4000 transaction occur. To keep as much knowledge as possible while reducing the number of data instances, it is common to keep both the trade volumes (i.e. the number of shares traded) and the open, high, low, and closing prices during the interval (see Figure 5.6).

Other advantages of sampling include the reduction of noise and, according to [30, 70], outliers from high-frequency data may be less problematic when the data is sampled or averaged over longer intervals such as 1, 5, 10 minutes [172]. Feature subset reduction can also reduce the size of the dataset while having the advantage of

[4]Another name for reducing the incoming data to increase the learning algorithm's processing speed is *loadshedding*. Additional information can be found in [3, 4].



(a) Tick-by-tick



(b) 1 minute (green triangles represent high and low prices during that interval

Figure 5.5: Demonstration of the reduction of granularity of SPY stock on January 3, 2005 from 9:30 to 10:00 a.m.

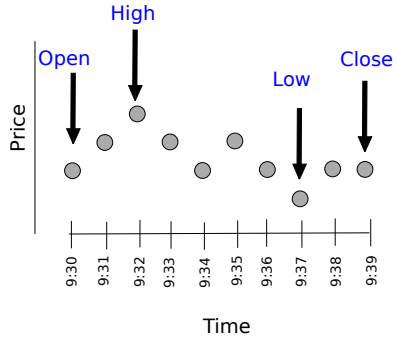


Figure 5.6: Open, high, low and close over a n interval period, where $n = 10$ minutes in this example

increasing generalizability. More information on feature selection and dimensionality reduction is in Subsection 5.2.6.

Depending on the stock traded, problems with sparseness may occur, which is the lack of transactions during a specific timespan. The cause of sparseness may be due to infrequent trading of the particular stock or due to too much data being discarded when cleaned. Sparseness of data can also lead to problems such as class imbalance, or uneven distributions of class. Various strategies have been used to address this problem, such as oversampling (with replacement) of trading data and oversampling with synthetically generated samples (see SMOTE algorithm in [43]). More information on addressing imbalance can be found in [42, 102]. Other solutions include reducing the sparse variables and re-gathering data using additional data sources. In other words, sparsity can sometimes be solved. This however remains a difficult and important problem since lack of data may result in a model with poor generalizability, or the inability to predict future prices.

5.2.4 Transformations

5.2.4.1 Discretization

Some machine learning classification algorithms require data to be in categorical forms. Discretization, also called binning, is the process of transforming a continuous attribute into a categorical one. Binarization is the transforming of discretized attributes into binary form (i.e. dummy variables). Proper use of discretization can also improve the generalizability of the classifier and decrease training times [118].

The two most popular methods of discretization are equal width and equal frequency discretization. In *equal width discretization*, the attributes are divided into k intervals (determined a priori) with the widths determined by $\frac{v_{max}-v_{min}}{k}$, where v_{min} and v_{max} are the minimum and maximum attribute values respectively. In *equal frequency discretization* the values are divided into k categories (again determined a priori) with each category containing the same number of training instances [242]. An additional difference between these two methods is the *equal width discretization* method requires one pass through the data, whereas with the *equal frequency discretization* method it is slightly more computational since the data must first be sorted. Both methods are *unsupervised*; they do not take the class label into consideration. The value of k can be determined by a visual inspection of the training dataset, trial-and-error by building the classifier on the training set and then inspecting on the validation set, or by using Sturges' simple rule of thumb for determining the value of k: $\hat{k} = 1 + \log_2(n)$, where n is the number of training instances [203].

A popular *supervised* method is the often cited work of Fayyad and Irani with their entropy minimization heuristic discretization [74] to determine cut points[5] for continuous attributes in decision trees. An additional *supervised* method is Kerper's ChiMerge [117]. In ChiMerge, within each interval, class frequencies are relatively consistent. If the class frequencies are not consistent within an interval, then the interval is split to express the difference; if two adjacent intervals have similar class

[5]An example of a *cut-point* is a continuous interval $[a, b]$ is partitioned into $[a, c]$ and $[c, d]$; the cut-point then is c since it divides the range into two intervals. Example taken from [150]

frequencies, then they are merged together. A χ^2 test of independence is used to test the hypothesis that two adjacent intervals of an attribute are independent of the class. If found to be independent, then the intervals are combined (otherwise they are kept separate). A description of methods that have built upon ChiMerge can be found in [243]. Surveys of additional discretization methods can be found in [129, 150, 243].

In streaming data, especially stock data, one cannot assume that the past determinations of the discretization parameters will remain the same in the future since the distribution of the underlying data may change (see Section 4.2 on Concept Drift). One solution for this is to discretize over fixed and sliding time windows batch learning classifiers [32, 33, 66, 84, 204]. The idea is that when the underlying concept remains stable, the size of the window increases to allow for more data; when the concept drifts, the window shrinks. If the window is too small, the classifier does not contain enough instances without over fitting; if it's too large, the classifier may be built using too many concepts. Using sliding windows therefore allows for the use of many of the traditional established methods of discretization with streaming data.

Many of the established successful methods for discretization that work well with wrapper learning do not work well with *online* learning. This is due to the need for tighter time constraints in an online setting for the algorithm to adjust to changing (drifting) concepts.

For online streaming data, Domingos and Hulten [62] introduced the **V**ery **F**ast **D**ecision **T**ree (VFDT) and its descendant, which accounts for concept drift with sliding windows, the **C**oncept-adapting **V**ery **F**ast **D**ecision **T**ree (CVFDT)

[109], but both algorithms used categorical attributes. Work by Jin and Agrawal [112] extended the VFDT to use numerical attributes by using **N**umerical **I**nterval **P**runing (NIP) to first partition the range of numerical attributes into intervals of equal-width. These bins were then statistically tested to determine if they would be unlikely to include a split point.

ChiMerge is too slow for use in an online setting with a worse case time of $O(n \log n)$. Elomaa, Lehtinen, and Saarela [66, 141] made modifications to the ChiMerge algorithm to use a balanced binary search tree (BST)[6] to maintain required statistics needed for online computation. By using the BST, the time needed becomes *linear*, thus decreasing its time dramatically.

Gamma and Pinto's **P**artition **I**ncremental **D**iscretization algorithm (PiD) [84] uses histograms for discretization with a two-layer approach. The first layer keeps statistics on the incoming streaming data. The second layer then creates final discretization based on the results from the first layer's statistics. In other words, a large number of initial intervals are created in the first layer without seeing the data and as the data streams in, counters are updated according to the interval it belongs. When the counter of an interval meets a certain threshold, then the interval is split creating a new interval in layer one[7]. The second layer then takes the intervals created by the

[6]Binary search tree is a popular data structure in computer science in which each internal node has two children. The search begins at the root of the tree and then proceeds to one of the two sub-tree below that node. It is efficient in that average search is $O(\log n)$ with a worse-case complexity of $O(n)$.

[7]If the interval that is triggered is the first or last, then a new interval with the same step is created.

first layer and creates either *equal width* or *equal frequency* bins from the intervals.

5.2.4.2 Data normalization and trend correction

Normalization, often synonymous with standardization in machine learning, refers to the process of transforming the data for use in a training model. The most common technique is Equation 5.2 but also used are Equations 5.3 and 5.4. When using large numbers in the classifier such as stock trade volume, Equation 5.4 is commonly used [176, 184].

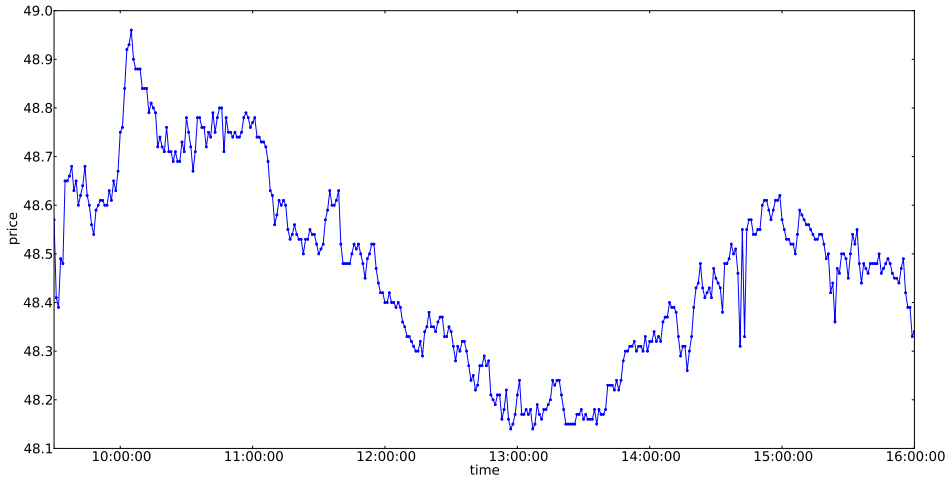
$$x_t = \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{5.2}$$

$$x_t = \frac{x_t}{x_{max}} \tag{5.3}$$

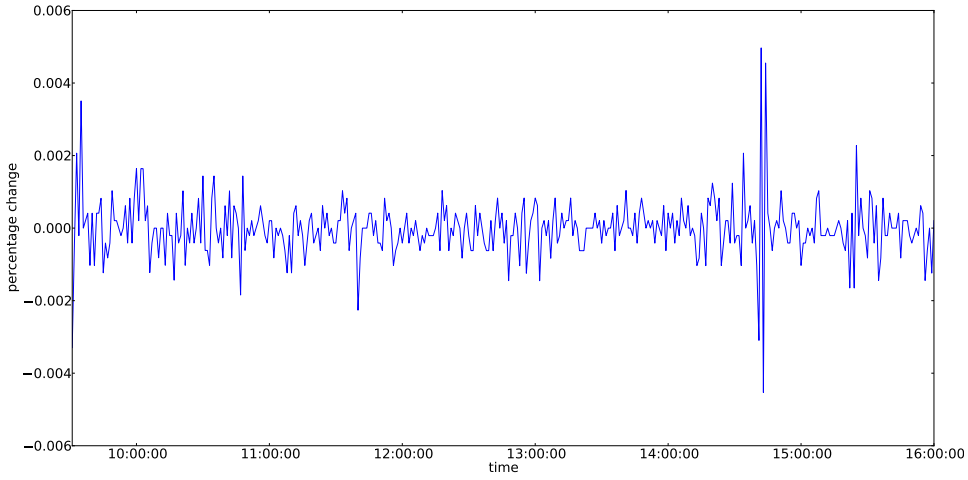
$$x_t = \frac{x_t - \mu}{\sigma} \tag{5.4}$$

$$x_t = \log(x_t) \tag{5.5}$$

There are two main reasons why normalization is done. First, some models such as artificial neural networks are prone to outliers; transforming the data helps to eliminate this problem. Second, trends may be present in the timeseries. This is called nonstationarity. Virili and Freisleben [222] write that the presence of trends often degrades the performance of classifiers severely; transformations are used to stabilize the variability of the series. In addition to the methods in Equations 5.2–5.5, finding percentage changes or differences in the price at time t from time $t - n$ where n can be any number is also commonly used [52, 219]. An example in Figure 5.7 shows the change in variability from the pre- to the post-transformation of the stock AXP for January 3, 2012 (Figures 5.7a and 5.7b respectively).



(a) Price at time t (pre-transformation)



(b) Percentage change of time t from time $t-1$ (post-transformation)

Figure 5.7: Symbol AXP on January 3, 2012

Later in our experiment section, we theorize that using the percentage change of the stock price is ideal since it allows for direct inter-stock comparisons between two stocks, whereas using the difference of price allows for only intra-stock comparison. For example, a $20 move in a $200 stock is more likely than a $20 move in a $50 stock.

Normalization is more difficult in online learning since one cannot assume that the future distribution of the data will remain the same as the past (see Section 4.2 on Concept Drift); the min and max values will not be known until all of the data is available. Gaber et al. [81] recommend discretizing the data (see Subsection 5.2.4.1) or binarizing to eliminate this problem in online learning. This problem does not exist in wrapper learning since the classifiers are built with chunks with the entire training set available at learning. However the minimum and maximum values must still be stored to make similar comparisons with future data.

5.2.5 Attribute creation

5.2.5.1 Overview

Appropriate attributes need to be chosen to predict the direction in the short-term. This is an extremely important step since ill-chosen attributes can either demonstrate no predictability or simply disregard common sense. Leinweber in [142] writes of finding strong statistical evidence between butter production in Bangladesh and the Standard and Poor's 500 (S&P 500); with butter production explaining 75% of the variation in the S&P 500 over a 10 year period. Although Leinweber originally

wrote it as a joke to emphasize the need to perform backtesting (see Section 3.5) and to pick attributes that make sense, the sarcasm of butter production predicting stock price seems to have been lost on some!

In Section 2.4.2 fundamental and quantitative technical approaches were discussed. As mentioned, fundamental approaches examine the economic and financial factors that drive the price of the stock with the aim to reveal the intrinsic value of the stock. Evidence in literature [8, 36, 37, 51, 75, 188, 218] seems to support the predictive nature of these attributes in predicting future events. However, this company-related financial information is of little use in predicting short-term (minute-by-minute) price direction since its release is infrequent; occurring at a minimum of four times per year. Quantitative technical approaches are however useful for short-term prediction. A few are described in this section, with the rest in Appendix C.

5.2.5.2 Sentiment as indicators

Quantitative technical approaches refer to systematic empirical investigation of information to forecast the direction of stock price change. Typically these technical approaches attempt to anticipate what others are thinking based on the price and volume of the stock. However, by analyzing trader sentiment we can attempt to anticipate price movements based on their emotion; as we know from basic psychology, emotion plays a significant role in the decision making process. A message board post or news article may influence an investor's or trader's emotion which may indirectly influence the stock's price. Rechenthin et al. [191] writes of finding slight predictability

when analyzing the sentiment on 80,000 Yahoo Finance message board posts along with historical price information for the next day's stock price. This is further backed by [10, 23, 57, 145, 193, 201, 230, 248, 249].

A problem addressed by Rechenthin et al. is the issue of trust on message boards; message boards can be abused, since users can posts anonymously. DeMarzo et al. [58] argue that people give more weight to the opinions of those with whom they talk and this kind of belief makes it profitable to be an influential participant within the message boards. Short sellers (those who profit when the stock drops in price) trying to frighten others into panic selling are mixed into the boards. Arther Levitt, former Security and Exchange Commission (SEC) Chairman stated "I encourage investors to take what they see over chat rooms not with a grain of salt but with a rock of salt." An act where individuals disseminate false information through message boards and/or email and then sell their stocks at artificially inflated prices is a "pump and dump" scam [94]. This sudden increase in the stock's price entices others to believe the hype and to buy shares as well. When the individuals behind the scheme sell their shares at a profit and stop promoting the stock, the price plummets, and other investors are left holding stocks which are worth significantly less than what they paid for it. A study by Frieder and Zittrain [79] examined stocks where "pumpers" previously sent large quantities of emails to entice others to buy, and found that investors who bought the stocks lost, on average, 5.25% in the two day period following the touting.

While the previously mentioned research points to favorable outcomes when

using sentiment to predict stock direction, we know of no papers that use sentiment to predict stock direction in the short-term (the above papers look at granularity of one day or more). This is most likely because of the (often) infrequent number of posts on message boards. In Rechenthin et al. [191] the number of Yahoo Finance posts averaged from a low of 40 to a high of 445 posts per day – far too little to be used to predict intraday price direction. Due to this sparseness, we therefore determine that solely using sentiment analysis to be of little use in predicting intraday stock direction. Its use however *in addition* to other indicators, such as technical analysis indicators (to be discussed in the next subsection), remains to be seen.

5.2.5.3 Technical analysis indicators

On many days, stocks move erratically without any economic news and are thereby moving by some unobserved stimuli [38]. In Chapter 2, we questioned if traders are instead being affected by the existence of trends within the market; traders seeing stocks trend upward buy, since they don't want to miss the profits, and sell when the market turns around. This further fuels pessimism for the stock and pushes it to even lower levels [158]. Technical analysis indicators are simply a way of attempting to describe and quantify the trend of the stock price.

In Subsection 2.4.2.3, two technical analysis indicators were described, namely Bollinger Bands (BBands) and Moving Average Convergence Divergence Oscillators (MACD). The goal of technical analysis is to identify regularities by extracting patterns from data. Questionnaires given to trading professionals find 80% to 90% of

those polled use some form of technical analysis [157, 162, 163, 213].

The idea behind quantitative technical indicators is to give a numerical description of past stock trend to use as attributes in a machine learning classifier. In addition to the BBands and MACD indicators discussed, we also use what we call the "simple moving average % change" indicator. This formula follows:

$$\begin{aligned} \text{SMA}(n) &= \frac{1}{n}\sum_{i=1}^{n} \text{close}_{t-i} \\ \text{SMA \% change}(n) &= \frac{\text{close}_t - \text{SMA}(n)_t}{\text{SMA}(n)_t} \times 100 \end{aligned}$$

First, a simple moving average (SMA) is calculated over n instances. The closing stock price at time t is compared against the moving average and a percentage change is calculated (see Figure 5.8). By including this as a continuous variable attribute, we can let the classifier algorithm determine ideal splits (if any) for the given dataset. In addition to providing the indicator attribute at time t, we include *lagged* attributes, at time $t-1$, $t-2$, $t-3$, $t-4$, and $t-5$.

In Chapter 6 we explore the use of these quantitative technical analysis indicators along with feature subset selection (Subsection 5.2.6.2) can lead to better predictive algorithms. For a list of additional attributes, see the Appendix C.

5.2.6 Dimensionality reduction and data reduction

5.2.6.1 Overview

Dimensionality reduction serves several purposes including: (1) reducing the number of attributes available to the model may *increase its ability to generalize on unseen data* and therefore *increase predictability.* This is because in many cases,

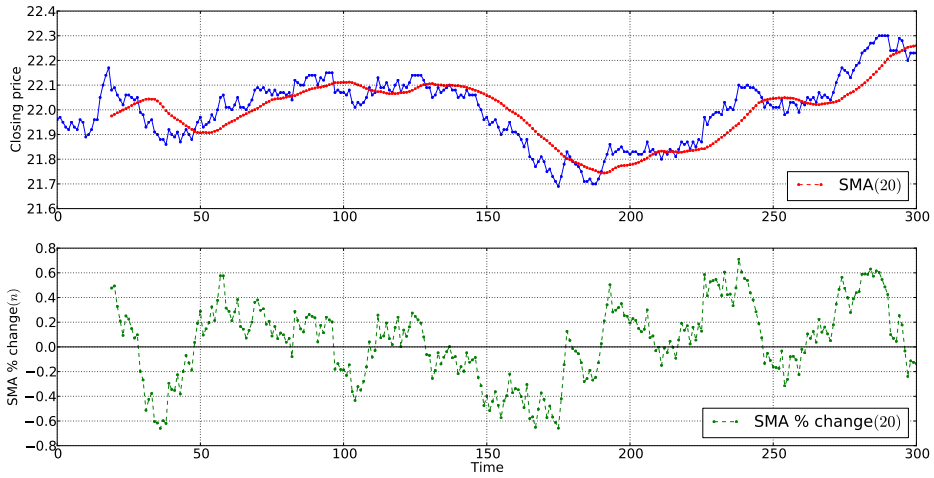


Figure 5.8: Demonstrating the "simple moving average % change" indicator

not all of the attributes are useful in predicting the outcome; many attributes are irrelevant. (2) Reducing the number of features results in *faster learning times*. This is especially relevant for adaptive (online) learning algorithms which rely on a fast learning speed to process streaming data in an appropriate time. (3) Reducing the number of features through data reduction and/or reducing dimensionality results in *smaller training sets, memory and storage requirements*. (4) Results tend to be *easier to interpret* when reducing dimensionality.

Many of the objectives of dimensionality reduction in data streams are similar to that of feature discretization in Section 5.2.4.1. For example, proper use of discretization can also improve generalizability of the classifier and decrease training times [118].

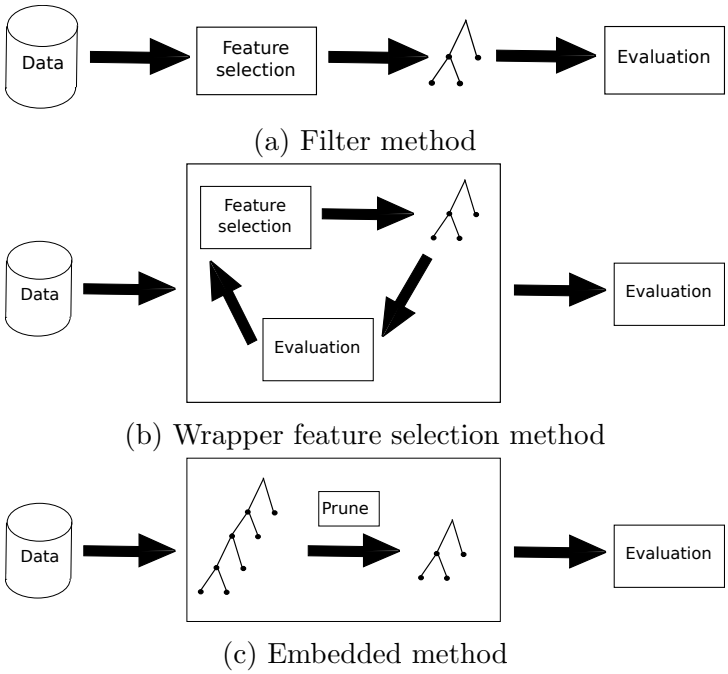
There are multiple methods to reduce dimensionality. The first is by creating new attributes that are a combination of existing attributes, such as combining stock

price and earnings into the price-earnings ratio (e.g. dividing price by earnings thereby creating the common P/E ratio). The second way to reduce dimensionality is through the use of feature (attribute) subset selection [177].

The goal of any classifier should be parsimony, or the simplest explanation of facts using the fewest variables [245]. Feature subset selection is the process of removing as much irrelevant information as possible. Techniques can be divided into three methods: filter, wrapper feature selection, and embedded methods. Filter methods do feature selection as a pre-processing step and is independent of the machine learning algorithm applied. Wrapper feature selection methods (not to be confused with wrapper-based learning methodologies) apply the machine learning algorithm on subsets of the data and use heuristics to search among the space of possible selections to find the optimal subset based on the performance of the model on the tuning set [110]. Lastly, embedded methods do feature selection as a normal process of the algorithm (e.g. pruning a decision tree) [22, 177, 197]. Filter, wrapper feature selection, and embedded methods are represented in Figures 5.9a, 5.9b and 5.9c respectively. A more thorough explanation follows.

5.2.6.2 Filter-based feature selection

Filter methods do feature selection as a pre-processing step before learning begins by examining intrinsic properties of the attributes. The search for features continues until a pre-determined number is found or until another criteria is met [54]. After the selection of features, the classifier is built and evaluated.



(a) Filter method

(b) Wrapper feature selection method

(c) Embedded method

Figure 5.9: Three main divisions of feature selection

Many different filter techniques exists, such as, Information Gain, Gain Ratio, χ^2 Attribute Evaluation and Correlation-based Feature Selection, among others [77]. In Information Gain, entropy [8] is measured when the attribute is given versus removed and a difference is calculated. The attributes with the largest values of information gain (i.e. the greatest reduction in entropy) are then kept since this represents the knowledge gained by including that attribute in the training set. Gain Ratio [186] is a variate of Information Gain resulting from the later having a bias toward selecting attributes with a large number of values. In the Gain Ratio approach, the information

[8]Entropy is the level of uncertainty in a training set due to the presence of more than one possible classification. For example, Entropy can be calculated as follows: $E = -\sum_{i=1}^{C} p_i \log_2 p_i$ where C is the number of classes and $p_i$is the probability of seeing class i out of the total number of instances. An excellent introduction can be found in [26].

gain is adjusted for each attribute to account for the number of and uniformity of values [26]. In χ^2 Attribute Evaluation, the attributes are ranked according to the χ^2 statistic with respect to the class; attributes with larger χ^2 values are kept since higher values mean that the variation is more significant and not merely due to chance. Papers by [241] and [77] found Information Gain, and to a lesser extent χ^2 Attribute Evaluation, performed best among the three filters described thus far; however, in their experiments, no single filter worked well for all datasets.

Advantages of filters include fast computational times (generally much faster than wrapper feature selection methods) and easy scalability. Scalability is of particular importance to high-speed stock data where the selection is needed quickly and the dimensionality of data is high. A disadvantage of filters is that they ignore the dependencies and correlations among the features (i.e. each feature is considered independently) which may lead to poor representation of data and therefore poor performance.

This disregard for dependencies and correlations lead to the creation of multivariate filter techniques, which attempted to incorporate some feature dependencies [197]. An example of this is the **C**orrelation-based **F**eature **S**election (CFS) method Mark Hall described in his dissertation [97]. This method measures the correlation between the attribute and the class, with the hypothesis that an ideal set of features should be highly correlated with the class, yet uncorrelated with other features. This is to ensure that redundancies and numbers of features are minimized (explaining the trend with as few of features as possible while still obtaining high performance). Pair-

wise correlation among all N features results in a time-complexity for CFS of $O(N^2)$, which is considerably larger than $O(N)$, the complexity of the three previously discussed filter methods. Yu and Liu [246] with their **F**ast **C**orrelation-**B**ased **F**ilter (FCBF) method, provide a correlation-based filter without the need for computing a pairwise correlation among all features, thus improving the time-complexity of CFS with similar results[9]. Research has shown that the correlation-based filter methods perform much faster than wrapper feature selection methods. However, according to Hall [97, 98], correlation-based selection methods often perform worse than wrappers when features are highly predictive of only a small part of the dataset. According to Hall, the CFS tended to perform better than the filters previously mentioned.

5.2.6.3 Wrapper feature selection

Wrapper feature selection methods apply the machine learning algorithm on subsets of data chosen with a heuristic(s) and compare the performance among the total space of possible feature selections. This is achieved by creating a subset of features from the training set and learning via an induction algorithm on this subset of attributes. This model is evaluated on a tuning/validation set and compared to previous subsets. The feature subset with the highest evaluation is chosen as the final set, and is then run on the induction algorithm [126]. The argument is that the estimated model performance achieved from using subsets of features is one of the best measures of the value of features. Wrappers also have an advantage that

[9]Additional methods for improving the time-complexity of Correlation-based Feature Selection (CFS) include best-first search and other heuristics.

many filters do not; filters treat features as independent of others, whereas, wrappers use subsets of attributes that explore the dependencies among the features. This can be determined when subsets containing certain combinations of features have higher levels of evaluated performance [54]. Wrappers tend to have superior results over filters [95, 98], although their high computational requirements may make them unsuitable for high-speed equity prediction where efficiency is needed. This however does not exclude their use entirely, as this paper will later discuss.

Iterating over all possible combinations of features is not always ideal. Consider, for example, choosing subsets of 5 attributes out of a total of 100 attributes amounts to a total of $\binom{100}{5} = 75,287,520$ possible combinations. Assuming each learning algorithm takes 5 seconds to evaluate each set, it would take a total of 12 years to evaluate all subsets! If predicting the stock market direction one minute into the future, more than a decade wait would provide little value to the trader[10]. While heuristics can be used to speed up the search by not enumerating through all possible subsets, a cost is still incurred as multiple models must be learned and evaluated. For this reason, wrapper methods are often impractical for large scale problems with many features and a large number of instances when the decision of subset is required quickly [77, 95]. This however does not exclude it use in trading models entirely, especially for predicting timespans further away and in wrapper-based ensemble methods.

Different heuristics exist for searching among the total space of possible fea-

[10]If enumerating over combinations of n features, where the minimum subset is of size 1 to a maximum of size n, the search space is $O(2^n)$. Enumerating then over 100 features would take 2.0×10^{23} years.

tures. Examples include forward selection, backward elimination [126], best-first search, and genetic algorithms [110, 119, 120, 239, 244].

Genetic Algorithms (GA)[106] belong to the much larger family of Evolutionary Algorithms, which also include Genetic programming, Ant Colony optimization[63], Particle Swarm optimization [116, 165], among others [25]. Implementing GAs in feature selection involves a population of chromosomes, each containing a random binary string of bits equal to the number of features, with $x_i = 1$ representing the inclusion and $x_i = 0$ indicating the exclusion of the ith feature (see Figure 5.10). A fitness function for each individual chromosome performance[11] is calculated (each chromosome represents a subset of features) and during each generation, a proportion of chromosomes from the current population is selected, with higher fitness chromosomes having a greater probability of continuing to the next generation (i.e. roulette wheel selection). In the next step, mating is accomplished with a randomly selected crossover point to *fuse* the chromosomes together. To encourage diversity (variation with the population) a mutation with a predetermined probability is added. The process is continued for another generation unless a predefined fitness or generation is reached [176, 244, 245]. A further explanation can be found in [35].

Additionally, hybrid methods have been explored that combine the speed advantage of filters and the subset evaluation ability of wrappers. These methods typically reduce the total problem-space with filters and then further explore subsets by

[11]Performance can be measured using accuracy or AUC or a number of other measures; see Subsection 4.4 for more information.

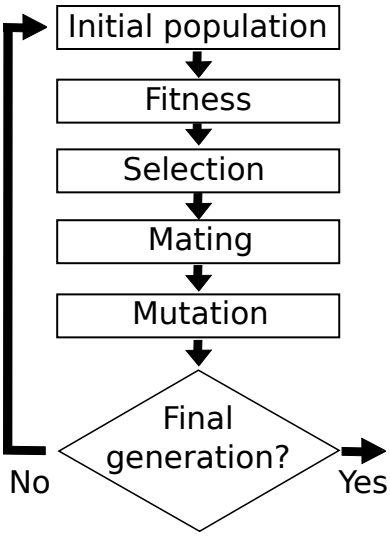


Figure 5.10: Genetic algorithm schema [244]

way of wrappers. Das in [54] create the **B**oosted **D**ecision **S**tump **F**eature **S**election algorithm (BDSFS) which uses information gain to determine the features to choose and then the AdaBoost algorithm with one-level decision trees (Decision Stumps). During each iteration of AdaBoost, the features with the highest information gained that were previously unselected are used. Bermejo et al. in [15] uses a multi-start, randomized subset that is improved with a local search method (hill-climbing had high performance and fast speeds) to reduce the number of evaluations needed by the wrapper. Yang et al. in [238] uses an Information Gain filter to reduce the features and then uses a genetic algorithm for actual feature selection. They theorize that, depending on the dataset, not all features are relevant to prediction. By eliminating these features initially, the subset is reduced therefore increasing the computational speed of the genetic algorithm.

5.2.6.4 Embedded feature selection

Embedded feature selection has the advantage that it is part of (embedded into) the induction algorithm and is less computationally expensive than wrappers. Examples of embedded methods include weighted logistic, naïve Bayes algorithms and decision trees [136, 197]. Decision trees, for example, implicitly split the data according to the importance via information gain of the feature to the classification task.

5.2.6.5 Experiment of time complexity

To demonstrate the time complexity of different feature selection methods, an experiment was taken to reduce five subsets of data containing varying number of instances from 275 attributes down to 30. Three filter methods, Information gain, χ^2 attribute evaluation, and a correlation-based method were examined along with two wrapper methods, a genetic algorithm[12] and a hybrid information gain/genetic algorithm[13]. The dataset used for experimentation is one that will be used throughout this paper (see B). This dataset contains 3-classes and 275 attributes with subsets comprised of 5000, 10000, 2000, 30000, and 40000 instances. The results of the experiment can be found in Table 5.3. Note that this test examines only the time complexity of the algorithm, and is not based on a performance metric on unseen

[12]The genetic algorithm used a decision stump with 20 generations, with each containing 20 populations.

[13]Method used was similar to [238]; Information gain reduce features to 100 and then a genetic algorithm comprised of a decision stump with 20 generations, with each containing 20 populations.

Table 5.3: Time complexity (seconds) of different filter and wrapper methods

Feature selection method used	Number of instances				
	5000	10000	20000	30000	40000
Information gain (filter)	1.4	2.8	8.9	17.5	30.0
χ^2 attribute evaluation (filter)	1.1	2.8	9.3	16.1	28.1
Correlation-based (filter)	0.7	1.8	6.3	10.7	21.0
Genetic algorithm (wrapper)	84.0	560.9	1477.3	1696.6	2226.4
Hybrid IG/GA algorithm (wrapper)	27.3	124.6	439.3	505.9	656.7

data. Analysis of individual feature selection methods on training data and the accompanying classifier performance on future data will be discussed in Chapter 6.

As can be seen in Table 5.3, the filter based methods topped the list with correlation-based feature selection method performing quickest followed by the information gain and χ^2 attribute evaluation methods. The wrapper methods, as expected followed last, however considerable speed increases were found by using the hybrid information gain/genetic algorithm over the genetic algorithm, as was used in [238]. All feature selection experiments were done on a Intel i7-3520M CPU running at 2.90GHz with 8 gigs of ram.

5.3 News and its effect on price

In Subsection 2.4.2 the fundamental analysis approach was discussed, which is the use of financial information to examine the intrinsic value of the company. However, this financial information (e.g. the quarterly financial statement) is released approximately four times per year, thus limiting the use of the fundamental approach for long term prediction. Additionally we discussed the use of past price as a means

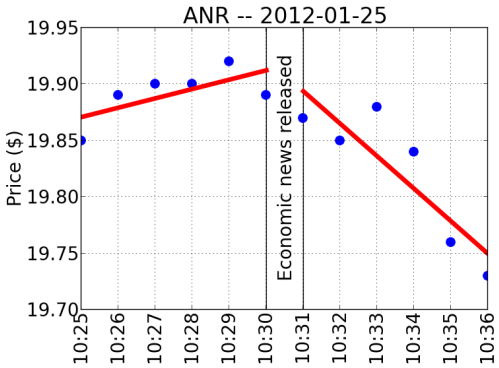
of predicting stock price in the short term via technical analysis. An external event not discussed thus far is the effect that U.S. government reports released through various agencies have on stock prices in both short- and long-term. Unlike blog postings or message board data, the release of the U.S. government reports are usually pre-scheduled at specific times. Additionally the government released statistics are related to overall economic development and not an individual stock. While not all economic reports released by the government move markets, many are barometers that provide an indication of the overall economy, which in turn can move markets [11, 92]. Examples of economic reports released by the government include the natural gas weekly update (released every Thursday at 9:30 a.m CST by the U.S. Energy Information Administration), the employment situation update (released the first Friday of very month at 7:30 a.m. by the U.S. Bureau of Labor Statistics, Department of Labor), the retail sales update (released monthly at 7:30 a.m. by the U.S. Bureau of the Census, Department of Commerce), and the petroleum status report (released every Wednesday at 9:30 CST by the U.S. Energy Information Administration). As discussed in Jiang et al. [111] the interval before the release of announcement, the pre-announcement, are characterized by *information uncertainty*, while the post-annoucements are characterized by *uncertainty resolution*. This difference between uncertainty and resolution was shown by Jiang et al. to also equate to differences in the underlying characteristics of the market or data. Similarly a paper by Beber and Brandt [13] found that higher uncertainty about economic news is associated with higher transaction volumes after the news is released.

5.3.1 Experiments

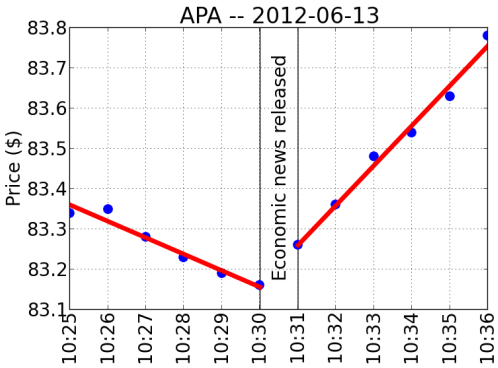
We want to examine the price change in response to the U.S. government petroleum weekly status update which arrives exactly at 9:30 CST (10:30 EST). This economic indicator reports the week-to-week total change in crude oil and gasoline; generally decreases in oil supply from the previous week raises oil prices which in turn benefits oil services companies (i.e. company stock increases). Several questions that we want to explore are as follows: 1) How does this information affect the market with respect to the direction (slope) of the stock price five minutes before the report and the direction five minutes after? 2) Do large gaps in price occur immediately after the reporting of the update and is this larger than normal gaps in the price during the day? 3) Does the volume of trades (number of shares traded) differ before and after the economic news?

To test the first question of market effect, we run a sum of least squares[14] on five prices before and after the 10:30 EST economic number for 34 stocks in the oil and gas services index on each Wednesday for six months from January 2012 to the end of June 2012 for a total of 26 weeks. Four examples can be seen in Figure 5.11. From our results, we find that the slope changes direction for all observations (covering all stocks) 57.5% of the time. Additionally, for 26 out of 34 stocks we observe that for more than 50% of the weeks (at least 14 out of 26), the slope changes direction. We find that this is only statistically significant for three stocks when running a statistical

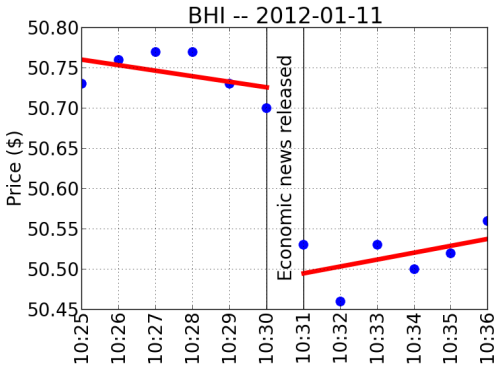
[14]We ignore the assumptions needed to run linear regression from a pure statistical standpoint, but instead use it as a means of approximation.



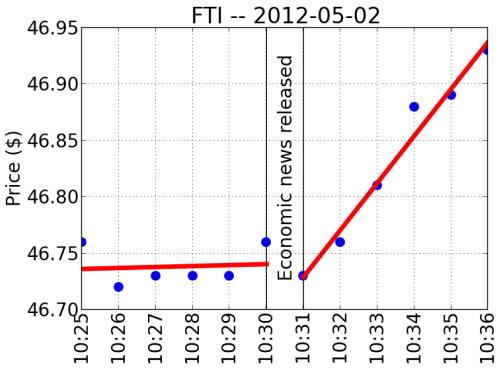
(a) Alpha Natural Resources, Inc. on January 25, 2012



(b) Apache Corp. on June 13, 2012



(c) Baker Hughes Inc. on January 11, 2012



(d) FMC Inc. May 5, 2012

Figure 5.11: Oil services stocks reacting to the U.S. Energy Information Administration release of the petroleum status report

test at the 95% level. However, when the slope is positive *before* the release of the economic number, the slope on average is 0.0163 (i.e. for every 1 minute the price increases by an average of \$0.0163); *after* the economic number when the slope is positive, the slope on average is 0.0232 (i.e. for every 1 minute the price increases by an average of \$0.0232). When the slope is negative *before* the release of the economic number, the slope on average is -0.0181 (i.e. for every 1 minute the price decreases by an average of -\$0.0181); *after* the economic number when the slope is negative, the slope on average is -0.0205 (i.e. for every 1 minute the price decreases by an average of -\$0.0205). These are interesting observations because after the news the stocks on average makes greater moves than before the release of the economic number. This suggests that individuals are strongly reacting to the release of information; further evidence that the market does not always price in information efficiently as suggested by the *strong form* of the Efficient Market Hypothesis.

For our next experiment we test the second question: do large gaps in price occur immediately after release of the petroleum status report, and how does this compare to normal differences in price at time t to the price at time $t-1$ throughout the day? We begin by examining the 34 stocks in the oil and gas services index on each Wednesday throughout the day and find that the (absolute) difference between the current price at time t and the previous price at time $t-1$ is on average \$0.0316. However, the (absolute) price difference immediately after the release of the news is on average \$0.0615; this is statistically significant for 32 out of 34 stocks at the 5% level. We conclude therefore that the release of news does affect the stock price.

For our last experiment we test the third question: does the volume of trades differ before and after the release of petroleum weekly status update. We begin by examining the mean transaction volume (number of shares traded) before and after the news release for all 34 stocks. The mean volume across stocks was found to be 14671[±16632] before and 16135[±16095] after the release of the news. This change in volume was found to be statistically significant in only three of the stocks at the 5% level.

5.3.2 Discussion

From our experiment we discovered that the market is often surprised by these economic releases with the slope of the market price changing directions 57.5% of the time after the release. Increases in volatility were also found as measured by change in stock price before and after the economic news. This provides evidence that time (in some form) should be included in the framework. In Subsection 6.4.6 we include several time attributes and examine its effect on model performance.

5.4 Conclusion

In this chapter we addressed problems specific to stock data and provided suitable solutions. This includes direction on what to do not only with imbalanced data streams, but also when presented with too much, too little, and erroneous trade data. An experiment was run in Subsection 5.2.2 using the algorithm by Brownlees and Gallo [30] to detect noisy prices, which we then compare to prelabeled *out-of-sequence* trades. In Subsection 5.2.4 we covered transforming data in data streams

to improve generalizability. Standardizing was particularly important to allow us to use base classifiers created with sector stocks in a pool of classifiers that we then use to improve our model performance. This will be discussed more in the next chapter.

Another important topic in this chapter was the discussion of attributes in stock data streams. We created as many attributes as possible and then used several approaches discussed in Section 5.2.6, to reduce the attributes to a manageable number.

Lastly, we examined the change in market volatility after the release of pre-scheduled economic news. The next chapter covers our wrapper-based framework to predict stock direction.

CHAPTER 6
OUR WRAPPER FRAMEWORK FOR THE PREDICTION OF STOCK DIRECTION

6.1 Overview

In Chapter 4 we examined the two methods for predicting high-speed data streams, specifically adaptive and wrapper-based approaches. Shifting market conditions such as those discussed in Chapter 5 and also seen in Figure 4.4, can create changes in the underlying concept, which in turn affects the model performance. Two main approaches to learning with concept drift are discussed in Subsection 4.2.2. The first is by either detecting anomalies in the data (which *could* affect classifier performance) or by examining actual decreases in classifier performance which could signal a change in concept (see Subsection 4.2.2.1). Upon detection of a change in concept, the model is updated (i.e. retrained) with new data. This approach is generally used by adaptive learning algorithms (see Subsection 4.3.1). The second approach to learning with concept drift makes the assumption that the underlying data drifts without determining if it actually does (see Subsection 4.2.2.2). This approach makes use of sliding windows by either keeping the classifier up-to-date with recent data or by using wrapper-based ensemble methods (see Subsection 4.3.2) that use classifiers built on prior concepts.

An issue raised with adaptive methods is that the most up-to-date data is not necessarily the most ideal choice; this research (discussed in greater detail in Subsection 4.2) suggests that stocks display reoccurring behavior, such as responding

to economic cycles and behavioral moods[1]. The ability to use traditional classifiers, such as artificial neural networks, support vector machines and decision trees, but also old concepts and knowledge from past days, weeks and years are distinct advantages of wrapper-based learning methods.

In this chapter we describe a new framework for the prediction of stock price direction in the short term. Advantages of this new approach include not only the regular advantages of wrappers (i.e. ability to use traditional classifiers, work in parallel, and work with prior concepts), but also the ability to transfer knowledge contained within additional stocks (Section 6.2.2). Bottlenecks that exists in existing wrapper methods (e.g. having to wait for classifier training to complete before prediction can be implemented) are mostly eliminated in our framework. Furthermore, increases in ensemble diversity are observed using different classifier types, different feature selection methods, and different subsets of data. All will be explained in greater detail in this chapter.

6.2 Our wrapper framework

The basic layout of our wrapper-based framework can be seen in Figure 6.1. The first step is the training of classifiers from random-length chunks of prior data. The chunks are represented as chunk(timestamp$_s$, timestamp$_e$) where timestamp$_s$ is start time and timestamp$_e$ is the end time. For example, chunk$(2500, 9200)$ represents a chunk of data from timestamp $t - 2500$ to timestamp $t - 9200$, where t is the

[1]We demonstrated in an experiment in Section 5.3 that the market reacts (in often predictable ways) to economic news.

current time. Additionally, the size of the chunk is constrained by some minimum and maximum size, where $\text{size}_{\max} > (\text{timestamp}_e - \text{timestamp}_s) \geq \text{size}_{\min}$. The chunk maximum and minimum sizes are chosen in such a manner that allow for enough data for generalization yet are small enough to include as few concepts as possible. Ideally, all data within a chunk are from the same concept and generated by the same distribution, but because the future is unknown, it is unclear which training sets may be useful. For this reason the classifiers are built using random-length (and often overlapping) datasets as an attempt to include as few concepts, yet as much of that particular concept as possible (see Figure 6.2). As the classifiers are built, they are added to a pool. To include additional diversity, we use different classifier types (C4.5 decision tree, non-linear SVM, ANN, etc.) along with different feature subset selection methods to both increase generalization and decrease training times. We also train classifiers using the data not only from the stock of which we are predicting the future price direction, but also from the highly correlated stocks within the same sector. Using training data that is similar to the stock we are predicting gives us more knowledge. Correlation among stocks within the same sector will be discussed in Section 6.2.2.

In step 2 (Figure 6.1) we evaluate the performance of each classifier from the pool by testing on the instances from the most recent sliding window. Classifier performance is determined using a class weighted AUC (see Section 3.4.4).

In step 3 we choose the top k classifiers from the pool (as evaluated on the most recent data) to form an ensemble, and then in step 4 we use this newly formed

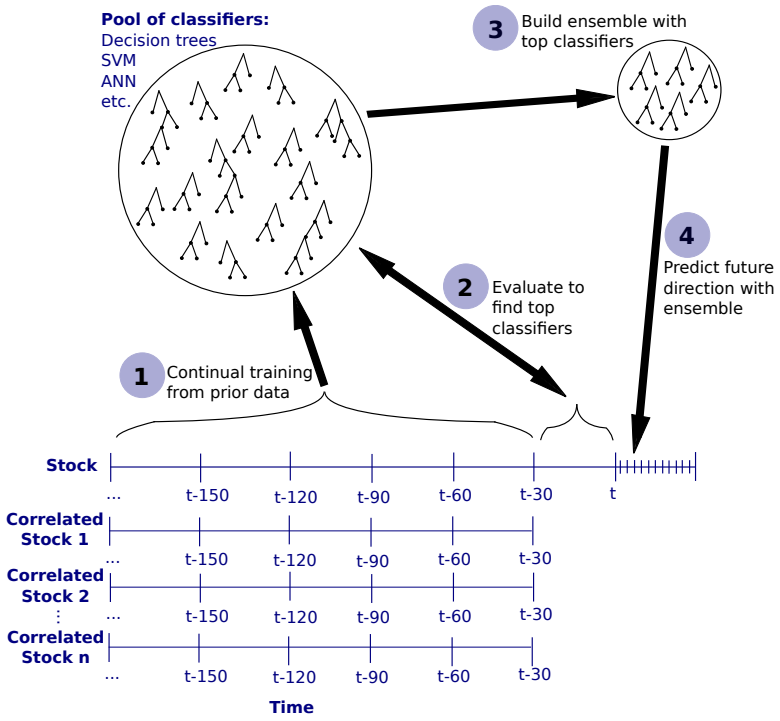


Figure 6.1: Our wrapper-based framework for the prediction of stock direction

ensemble to make a prediction on future stock price direction. The base classifiers in the ensemble are combined using the mean of each classifier's confidence rather than using a simple majority vote (see Subsection 56). After n instances, the process begins again at step 2 (step 1 is continuously training new base classifiers).

Benefits of our new wrapper framework include: 1) the process is easily distributed, specifically in steps 1 and 2, which allows for decreases in computational times; 2) it uses knowledge from highly correlated stocks, thus increasing the amount of data available for training classifiers in the pool; and 3) classifiers used for the pool can include models that are generally slower to train, such as artificial neural networks and those that use computationally expensive feature selection methods. As

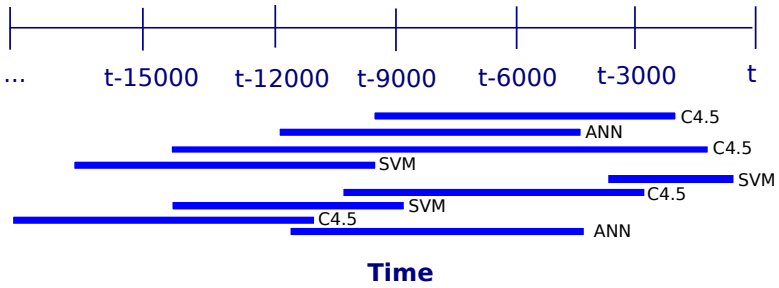


Figure 6.2: Our wrapper-based framework – random starting and ending periods and random classifier types

previously discussed, stocks may be cyclical; therefore, models that may take hours to train, while not available to be included in the pool for the next interval, may still be useful when a similar concept appears.

6.2.1 Slow training versus fast evaluation of classifiers

As discussed in Section 4.3, most learning methods that rely on quick classification for changing, high-speed data streams (such as stock data) take the adaptive learning approach. The choice of an adaptive, incremental classifier approach is largely due to speed – existing wrapper methods such as those discussed in Section 4.3.2, require the base classifiers trained on the last interval of data to be completed before it is evaluated for inclusion in the ensemble.

This creates a bottleneck in *existing* wrapper methods, since the ensemble cannot output a prediction until all base classifiers have been trained and evaluated. This method of training classifiers is costly, and it is the reason why some approaches in classification of high-frequency data use adaptive learning methods instead. Our framework side-steps these problems since classifiers are continuously built for in-

clusion in the framework pool and our ensemble selection heuristic does not rely on the completion of training of any specific classifier for any specific period of time. Our classifiers are built with random start and end times and therefore completion of training of any particular base classifier is relatively unimportant. The pool will contain hundreds or thousands of pre-built classifiers that can be evaluated at any given time. If the classifier is not ready for immediate use, it may still provide value for another future event.

In Table 6.1 the training times in minutes for 25 C4.5 decision trees and 25 non-linear SVM classifiers are displayed (for a visualization see Figure 6.3). Taking into consideration different training subset lengths and number of attributes (selected with an Information Gain feature selection method) the C4.5 decision tree is 1 to 30 times faster than the non-linear SVM. On average, the C4.5 decision tree is 7 times faster than the training time of a comparable non-linear SVM classifier. The time needed to train 25 decision trees with $20,000$ instances and 150 attributes is roughly 13 minutes and for a comparable non-linear SVM the time is roughly 2 hours. All experiments in this section were conducted on a single-core 64 bit i7-3770 Intel processor running at 3.4GHz with 16 gigabytes of ram.

Evaluation of existing classifiers is substantially faster than building new classifiers. For example, to evaluate a pool of 25 pre-built classifiers (such as in our framework) takes roughly 0.5 seconds. This is similar to the C4.5 decision tree and the non-linear SVM classifier for all ranges of attribute counts and training set sizes. More specifically, this includes the time needed to locate the stored classifier from

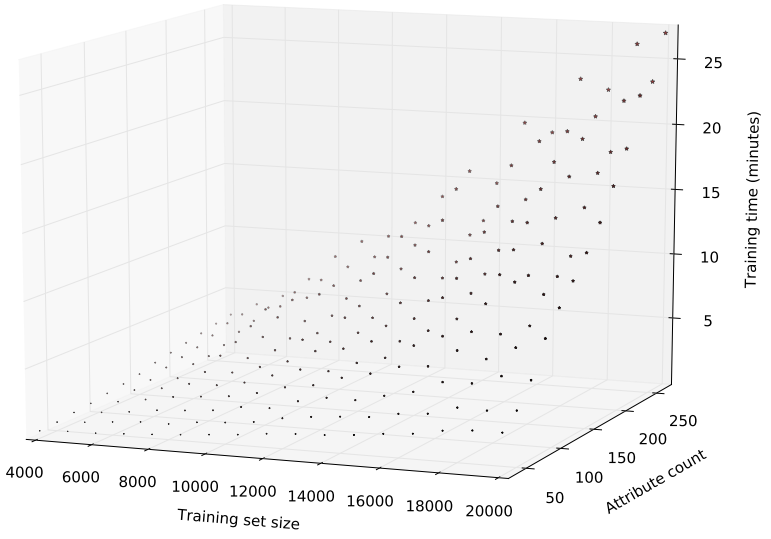
Table 6.1: Classifier (DT and SVM) training times (for 25 classifiers) in minutes for specific training set sizes and attribute counts

(a) C4.5 decision tree classifier results

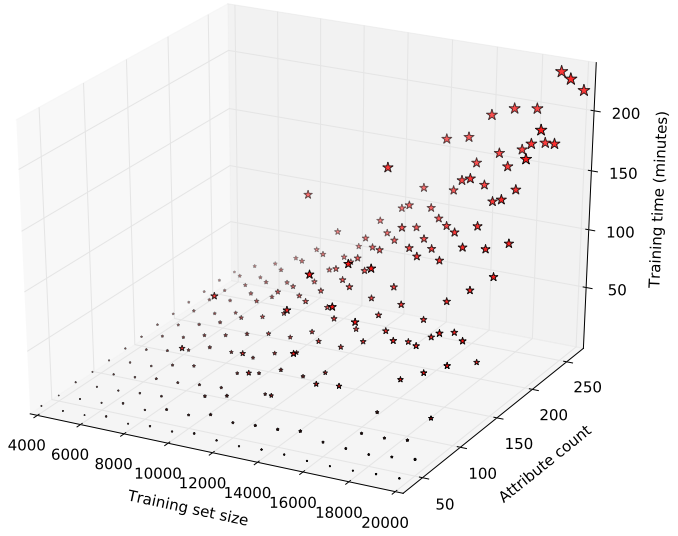
Training set size	Attribute count												
	30	50	70	90	110	130	150	170	190	210	230	250	276 (all)
4000	0.3	0.4	0.6	0.7	0.9	1.1	1.3	1.7	2.0	2.2	2.5	2.8	3.0
5000	0.4	0.6	0.7	1.0	1.4	1.6	2.0	2.3	2.6	3.0	3.4	3.7	4.0
6000	0.5	0.8	1.0	1.3	1.6	1.9	2.4	2.5	2.9	3.4	3.8	4.3	4.8
7000	0.6	0.9	1.2	1.7	2.1	2.5	3.2	3.8	4.5	5.3	5.1	5.7	6.3
8000	0.7	0.9	1.4	1.9	2.2	2.6	3.2	4.3	5.4	6.2	5.9	6.8	8.2
9000	0.9	1.2	1.6	2.1	2.6	3.2	4.0	4.4	5.2	6.8	7.0	8.0	9.6
10000	1.0	1.4	2.0	2.5	3.3	3.8	4.9	4.9	5.8	6.6	7.5	8.9	10.1
11000	1.2	1.6	2.0	3.1	3.9	4.7	4.9	6.3	6.7	8.7	9.5	10.7	11.3
12000	1.4	2.0	2.7	3.4	3.9	5.1	5.8	6.4	7.8	9.2	10.6	11.6	13.5
13000	1.6	2.3	3.1	4.1	4.7	5.8	6.4	7.7	9.1	10.7	12.7	14.6	15.6
14000	1.7	2.5	3.1	4.6	5.2	5.9	7.1	8.0	9.3	10.0	11.6	12.3	14.8
15000	1.9	2.6	3.6	5.2	6.3	6.9	8.9	9.8	10.9	12.4	14.0	16.7	19.6
16000	2.1	3.0	4.1	5.4	7.1	7.9	9.3	10.3	11.6	13.4	14.6	18.7	19.0
17000	2.3	3.2	4.9	6.7	7.6	9.2	10.9	12.3	13.9	16.0	17.6	19.6	23.2
18000	2.7	3.8	5.0	6.3	7.6	10.9	11.0	12.9	14.4	17.0	19.5	20.8	22.5
19000	2.9	3.7	5.7	7.3	8.3	10.2	11.0	12.1	15.3	17.4	18.6	22.1	26.1
20000	2.9	3.9	5.6	8.1	9.8	11.3	12.9	14.7	17.0	19.4	23.0	23.7	27.0

(b) Non-linear SVM classifier results

Training set size	Attribute count												
	30	50	70	90	110	130	150	170	190	210	230	250	276 (all)
4000	2.0	1.2	2.9	2.3	2.7	3.7	4.6	6.0	6.3	7.6	8.0	9.2	7.7
5000	1.7	2.0	2.8	3.9	4.7	6.4	8.8	9.4	12.7	13.5	12.7	16.7	16.5
6000	2.0	2.6	3.4	5.2	5.4	9.4	11.6	13.6	14.0	16.4	21.0	18.9	22.3
7000	4.8	6.1	5.0	7.8	10.0	13.6	17.0	19.1	18.3	26.2	27.3	29.3	30.9
8000	2.5	4.3	6.3	8.2	13.1	19.7	20.9	26.9	32.3	26.8	30.5	40.6	42.1
9000	4.3	8.7	51.2	10.3	77.0	19.4	25.7	31.4	44.0	39.6	111.0	50.2	61.4
10000	3.8	7.9	14.1	11.6	14.4	18.2	29.7	38.1	44.2	47.5	48.9	51.3	51.9
11000	6.2	12.2	18.3	44.9	18.2	25.9	30.3	52.2	52.2	62.3	59.3	71.3	78.2
12000	4.3	9.2	41.7	13.8	25.0	30.3	38.7	50.6	65.2	75.0	75.6	79.4	92.4
13000	3.3	10.8	26.4	88.8	23.2	43.8	53.8	71.8	76.7	85.6	85.2	106.8	113.4
14000	4.6	11.5	65.6	122.0	30.2	42.5	48.5	65.8	167.1	108.1	99.7	107.9	158.4
15000	4.4	14.0	43.9	98.9	125.3	51.1	64.2	78.9	103.1	102.1	110.8	126.3	163.2
16000	3.1	8.7	46.1	89.9	125.0	63.8	76.8	108.3	105.9	116.6	146.4	152.9	185.0
17000	2.7	7.4	10.1	17.9	68.7	58.3	67.7	108.4	123.2	159.5	145.9	164.2	193.4
18000	3.1	6.3	10.1	49.5	68.1	66.0	86.7	122.8	131.7	143.8	164.5	170.4	196.5
19000	4.2	8.3	12.3	58.9	82.3	73.6	99.6	124.9	157.1	156.9	186.5	179.2	230.0
20000	4.9	10.4	34.8	68.7	79.7	52.3	114.3	133.0	193.5	208.9	189.7	234.6	217.6



(a) C4.5 decision tree classifier results – visualization of Table 6.1a



(b) Non-linear SVM classifier results – visualization of Table 6.1b

Figure 6.3: Classifier training times (for 25 classifiers) – visualization of Table 6.1

the pool (previously written to a file), to load it into memory and to evaluate the AUC performance on a 60-instances interval of time. This process is repeated for each classifier in the pool (e.g. for our example it is repeated 25 times for each of the 25 classifiers in the pool). For a pool with 1000 classifiers, the time needed to evaluate all classifiers would take approximately 20 seconds, while the time to train, from start to finish, would take approximately 196 minutes and 1188 minutes for a C4.5 decision tree and non-linear SVM classifier respectively (assuming a training set of 10,000 instances and 150 attributes). Keep in mind that experiments were run on a single-core machine; evaluating classifiers on past data is easily distributable over multiple-cores to increase performance.

The speed of evaluating pre-built classifiers allows our framework to compare the predictability of hundreds of pre-built (and continuously built) classifiers in the same time as is needed to train just one classifier from start to finish using other existing wrapper frameworks. This is a distinct advantage of our approach.

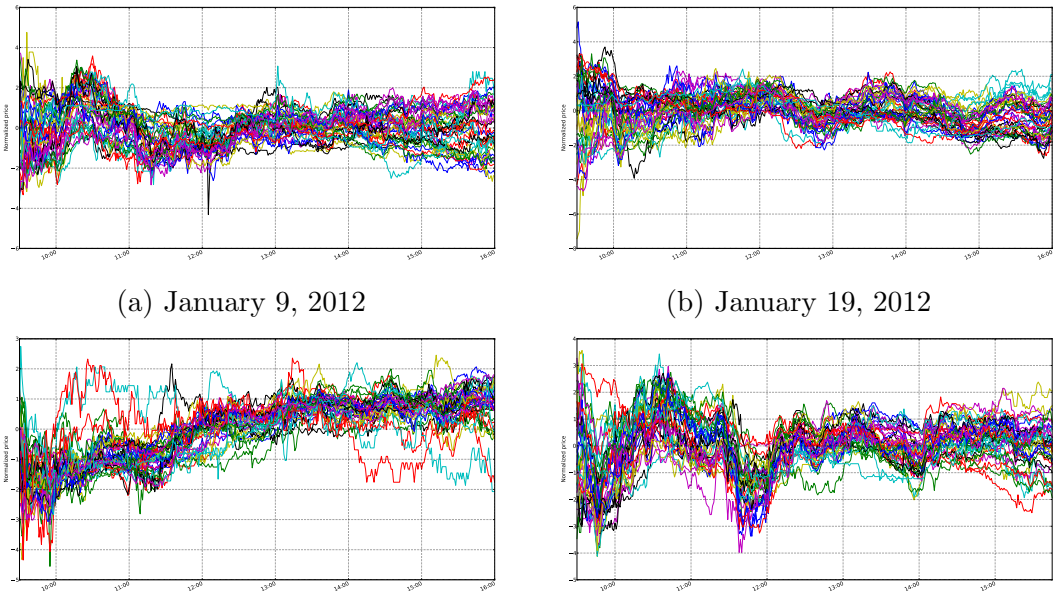
In comparison to a traditional C4.5 classifier, the Very Fast Decision Tree (VFDT) was 5.5 times faster, and 56 times faster than a non-linear SVM classifier. For example, to train a 20,000 instance subset containing all 276 attributes took 11.6 seconds using the VFDT, 64 seconds using the traditional C4.5, and 650 seconds using the non-linear SVM. However, in the same 11.6 seconds needed to build one VFDT, we could have evaluated roughly 580 previously trained classifiers with our method (having been built on different stocks and different timespans). While our framework may not be as fast as some adaptive methods, our results are better than the ones

tested and is *fast enough* for the time span needed.

6.2.2 Use of additional stocks

There is a saying among traders "a rising tide floats all boats" which references the observation that most stocks go up at around the same time; the opposite of this can also be true. This observation was especially evident in the "flash-crash" of May 6, 2010 when almost 8,000 stocks and indices plummeted 5 to 15% within a few minutes only to rebound equally as fast [205]. Additionally, 20,000 trades in 300 stocks traded substantially lower. For example, Procter & Gamble Company (symbol: PG) was trading for $61 a share before the crash and then was momentarily trading for just a penny a share. The initial cause, according to an investigation by the U.S. Securities and Exchange Commission [205], was a mutual fund that (erroneously) sold 75,000 futures contracts (worth $4.1 billion) of the Standard and Poor's 500 Index (S&P 500). Not only did the 500 stocks of which the S&P 500 is composed sell off, but also nearly the entire market plunged. In theory only the stocks within the S&P 500 should have sold, but because of the high level of inter-security correlation, this caused a sell-off of almost the entire market[2]. After the selling subsided, the prices quickly recovered and the day ended at almost the same level as it was before the

[2]According to the investigation [205], the initial plunge was absorbed by high-frequency traders who took the opposite side of the initial drawdown. However, after further decreases in price, many of these initial buyers turned to sellers which further exacerbated the price. As the stocks fell, the liquidity vanished, which caused even further decreases to the prices. Within 15 seconds the S&P 500 dropped 1.7% and within $4\frac{1}{2}$ minutes it had dropped 5%. And while the market had dropped dramatically (extremely uncharacteristic), many market participants were reported to have feared "the occurrence of a cataclysmic event of which they were not yet aware." This lead to a loss of confidence among many participants and an unwillingness to participant in the buying.



(a) January 9, 2012

(b) January 19, 2012

(c) February 16, 2012

(d) March 15, 2012

Figure 6.4: Visually demonstrating the high level of intraday correlation among 34 oil services stocks (each line represents a different normalized stock price)

crash.

The "flash-crash" demonstrated a highly-correlated market, albeit in a very severe way. These high levels of correlation within intraday prices are also observed in Allez et al. [7] and Preis et al. [182] and in Figure 6.4. This figure visually demonstrates the high level of intraday price correlation among 34 stocks in the oil services sector over the course of four random days: January 9, January 19, February 16, and March 15, 2012 (a full six months of correlations for these stocks can be found in the author's YouTube video[3].

In Figure 6.5 we provide an additional view of stock price correlation over 15

[3]http://bit.ly/17Myvv4

minute intervals using Spearman's rank correlation coefficient[4] and a heatmap (red represents strong positive correlation and blue represents strong negative correlation). While Figures 6.4 and 6.5 demonstrates correlation among stocks, it does not quantify the change in correlation throughout the day; we demonstrate this with an animated version of Figure 6.5 in a YouTube video[5].

These visualization demonstrate the high intraday price correlation among the stocks within the sector[6]. With such high levels of correlation, it would therefore seem rational to use these stocks to subsidize our existing data. In Subsection 6.4.3.1, we will compare the inclusion versus exclusion of additional stocks to the model framework and its effect on model performance.

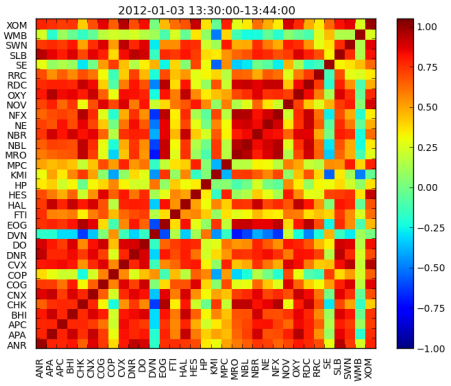
6.3 Benchmarks

To analyze the performance of our wrapper framework, we first compare against several widely used benchmarks. The dataset used for training and testing purposes is the stock dataset used elsewhere in this paper. This 34 stock time-series contains 276 attributes and covers the change in price (as compared against the prior minute) for the first seven months of 2012. The predicted class is the percentage change in price over a one minute period, $\left(\frac{\text{price}_t - \text{price}_{t-1}}{\text{price}_{t-1}}\right) > 0.05\%$ which is considered a *large upward move*, $0.05\% \geq \left(\frac{\text{price}_t - \text{price}_{t-1}}{\text{price}_{t-1}}\right) \geq -0.05\%$ is considered an *insignifi-*

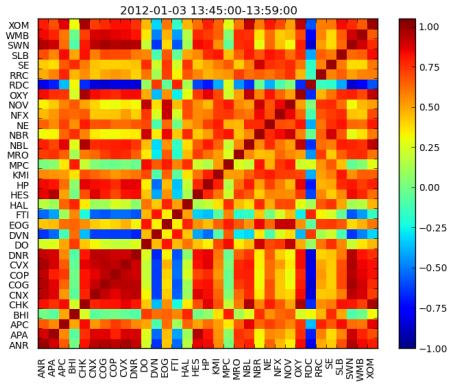
[4]Unlike the Pearson correlation coefficient, Spearman's correlation coefficient was chosen because of its ability to measure the relationships between stock prices that are not necessarily linear. Each stock was normalized.

[5]http://bit.ly/1nkJ5WW

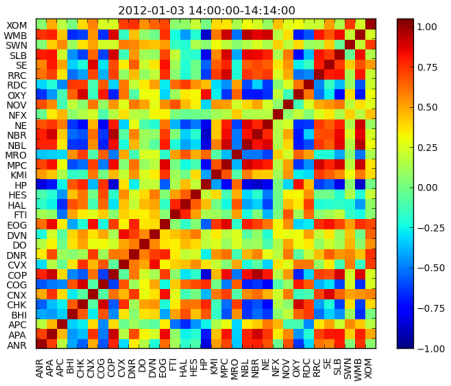
[6]According to a white-paper by J.P. Morgan [128], this is called cross-asset correlation.



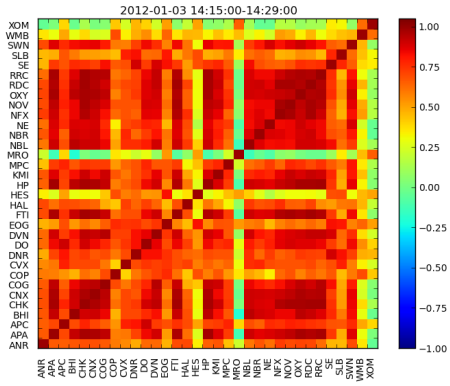
(a) Jan 3, 2013 13:30-13:44



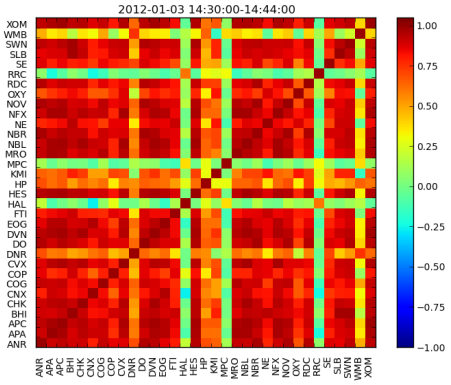
(b) Jan 3, 2013 13:45-13:59



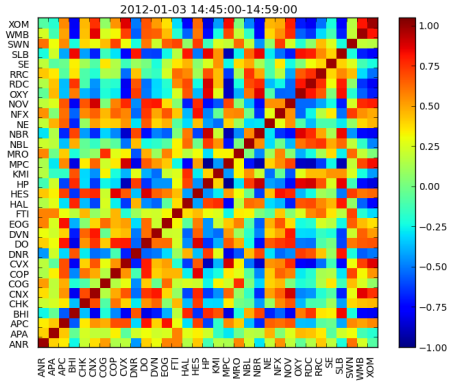
(c) Jan 3, 14:00-14:14



(d) Jan 3, 2013 14:15-14:29



(e) Jan 3, 2013 14:30-14:44



(f) Jan 3, 2013 14:45-14:59

Figure 6.5: Visualization over 15 minutes of the changing nature of the Spearman correlation coefficient matrix over time for stocks within the same sector (oil services)

cant move, and $\left(\frac{\text{price}_t - \text{price}_{t-1}}{\text{price}_{t-1}}\right) < -0.05\%$ is considered a *large downward move*. For a description of the stocks, a preview of the actual price change over the seven months, and a description of attributes along with calculations, please see Appendices B and C.

We evaluate the performance of the baseline models using AUC over a 60 minute sliding window (see Section 3.4.4). AUC is used both for its popularity within the machine learning community and its improvement over other methods such as accuracy or error rate (error rate = 1 − accuracy). Accuracy is problematic within imbalanced datasets, such as ours, because high levels of accuracy can be obtained by reporting the class with the largest number of instances (see Subsection 3.4.1 for more information). AUC also measures the *confidence* of the prediction, whereas with accuracy the confidence of decision is ignored; accuracy simply increases when the class with the largest probability estimate is the same as the actual class [108].

Several baselines have been determined so that our framework can be compared (with results in Table 6.2). The first (and also the most naïve method) is to use the class from the prior interval that is observed the most (the majority class). The second benchmark is an approximation to the Concept Drifting Very Fast Decision Tree (CVFDT) explained in Subsection 4.3.1.2.1 and uses a C4.5 decision tree. As mentioned in Subsection 4.3.1.2.1, the CVFDT uses fixed-width sliding windows to adapt the model to concept drift and because it uses a VFDT, the output is nearly identical to the conventional C4.5 decision tree.

Table 6.2: Benchmarks

Stocks	Majority class	C4.5 (CVFDT approximation)			C4.5 Bagging (Online bagging approximation)			C4.5 Boosting (Online boosting approximation)			Streaming Ensemble Algorithm (SEA)		
	previous interval	5,000 instance sliding window	10,000 instance sliding window	30,000 instance sliding window	5,000 instance sliding window	10,000 instance sliding window	30,000 instance sliding window	5,000 instance sliding window	10,000 instance sliding window	30,000 instance sliding window	5,000 instance sliding window	10,000 instance sliding window	30,000 instance sliding window
ANR	0.500	0.513 [0.030]	0.519 [0.032]	0.517 [0.029]	0.540 [0.034]	**0.543 [0.031]**	0.531 [0.036]	0.526 [0.031]	0.531 [0.033]	0.522 [0.034]	0.532 [0.532]	0.527 [0.026]	0.527 [0.026]
APA	0.500	0.503 [0.035]	0.509 [0.036]	0.509 [0.035]	0.526 [0.039]	0.527 [0.036]	**0.534 [0.037]**	0.519 [0.046]	0.515 [0.041]	0.524 [0.043]	0.508 [0.508]	0.510 [0.018]	0.512 [0.020]
APC	0.500	0.502 [0.029]	0.503 [0.031]	0.505 [0.029]	0.517 [0.043]	**0.520 [0.040]**	**0.520 [0.047]**	0.514 [0.037]	0.512 [0.033]	0.508 [0.037]	0.509 [0.509]	0.510 [0.02]	0.509 [0.018]
BHI	0.500	0.504 [0.031]	0.505 [0.034]	0.510 [0.035]	0.527 [0.040]	**0.534 [0.045]**	**0.534 [0.044]**	0.520 [0.046]	0.524 [0.039]	0.523 [0.036]	0.515 [0.515]	0.514 [0.02]	0.516 [0.020]
CHK	0.500	0.508 [0.027]	0.505 [0.026]	0.515 [0.037]	**0.525 [0.033]**	0.523 [0.031]	0.515 [0.033]	0.515 [0.028]	0.510 [0.033]	0.513 [0.031]	0.520 [0.52]	0.523 [0.03]	0.520 [0.029]
CNX	0.500	0.505 [0.027]	0.501 [0.031]	0.508 [0.030]	0.516 [0.034]	0.518 [0.037]	**0.523 [0.034]**	0.514 [0.033]	0.515 [0.032]	0.517 [0.034]	0.520 [0.52]	0.519 [0.026]	0.516 [0.028]
COG	0.500	0.499 [0.028]	0.508 [0.030]	0.508 [0.033]	0.510 [0.034]	0.510 [0.030]	**0.519 [0.029]**	0.504 [0.035]	0.511 [0.036]	0.513 [0.034]	0.509 [0.509]	0.516 [0.028]	0.513 [0.027]
COP	0.500	0.506 [0.036]	0.504 [0.037]	0.511 [0.035]	0.528 [0.052]	0.532 [0.045]	**0.535 [0.041]**	0.518 [0.043]	0.517 [0.051]	0.523 [0.050]	0.504 [0.504]	0.506 [0.014]	0.507 [0.015]
CVX	0.500	0.505 [0.041]	0.504 [0.043]	0.508 [0.046]	0.533 [0.056]	0.537 [0.064]	**0.547 [0.064]**	0.524 [0.049]	0.526 [0.058]	0.533 [0.060]	0.502 [0.502]	0.504 [0.011]	0.504 [0.009]
DNR	0.500	0.508 [0.026]	0.508 [0.029]	0.506 [0.030]	0.520 [0.032]	**0.525 [0.033]**	0.520 [0.033]	0.512 [0.031]	0.519 [0.031]	0.510 [0.031]	0.518 [0.518]	0.516 [0.025]	0.522 [0.026]
DO	0.500	0.501 [0.039]	0.507 [0.036]	0.504 [0.036]	0.523 [0.039]	0.523 [0.040]	**0.528 [0.036]**	0.518 [0.036]	0.519 [0.038]	0.522 [0.041]	0.510 [0.51]	0.508 [0.014]	0.507 [0.015]
DVN	0.500	0.506 [0.032]	0.508 [0.036]	0.508 [0.036]	0.523 [0.037]	0.521 [0.034]	**0.527 [0.037]**	0.513 [0.037]	0.520 [0.034]	0.520 [0.035]	0.507 [0.507]	0.508 [0.015]	0.508 [0.018]
EOG	0.500	0.505 [0.031]	0.496 [0.031]	0.506 [0.034]	0.517 [0.035]	0.520 [0.032]	**0.523 [0.037]**	0.518 [0.032]	0.521 [0.036]	0.522 [0.039]	0.515 [0.515]	0.513 [0.02]	0.513 [0.019]
FTI	0.500	0.504 [0.034]	0.510 [0.033]	0.502 [0.028]	0.524 [0.043]	**0.525 [0.036]**	0.519 [0.034]	0.512 [0.038]	0.513 [0.037]	0.516 [0.035]	0.512 [0.512]	0.508 [0.021]	0.511 [0.021]
HAL	0.500	0.505 [0.030]	0.508 [0.032]	0.509 [0.030]	0.520 [0.038]	**0.515 [0.035]**	**0.515 [0.034]**	0.513 [0.035]	0.513 [0.037]	**0.515 [0.034]**	0.514 [0.514]	0.512 [0.023]	0.509 [0.023]
HES	0.500	0.503 [0.035]	0.511 [0.033]	0.513 [0.037]	**0.530 [0.034]**	**0.530 [0.038]**	0.528 [0.035]	0.518 [0.036]	0.522 [0.039]	0.515 [0.037]	0.513 [0.513]	0.513 [0.024]	0.515 [0.024]
HP	0.500	0.507 [0.029]	0.502 [0.031]	0.498 [0.033]	0.512 [0.036]	**0.515 [0.038]**	0.514 [0.039]	0.509 [0.036]	0.511 [0.033]	0.511 [0.040]	0.507 [0.507]	0.509 [0.019]	0.510 [0.023]
KMI	0.500	0.512 [0.038]	0.504 [0.041]	0.509 [0.037]	0.540 [0.049]	0.535 [0.048]	**0.549 [0.048]**	0.526 [0.041]	0.527 [0.045]	0.536 [0.048]	0.509 [0.509]	0.507 [0.014]	0.512 [0.019]
MPC	0.500	0.496 [0.031]	0.500 [0.029]	0.502 [0.040]	0.513 [0.035]	0.521 [0.038]	**0.524 [0.043]**	0.510 [0.033]	0.508 [0.036]	**0.524 [0.037]**	0.510 [0.51]	0.508 [0.019]	0.512 [0.021]
MRO	0.500	0.508 [0.036]	0.510 [0.039]	0.497 [0.035]	0.523 [0.042]	**0.528 [0.042]**	0.524 [0.039]	0.516 [0.034]	0.521 [0.039]	0.515 [0.038]	0.506 [0.506]	0.508 [0.018]	0.509 [0.020]
NBL	0.500	0.510 [0.032]	0.502 [0.035]	0.503 [0.030]	0.522 [0.037]	**0.528 [0.037]**	0.527 [0.036]	0.514 [0.038]	0.518 [0.034]	0.518 [0.033]	0.51 [0.51]	0.509 [0.018]	0.513 [0.016]
NBR	0.500	0.504 [0.026]	0.510 [0.028]	0.508 [0.025]	0.518 [0.032]	0.521 [0.033]	**0.524 [0.033]**	0.511 [0.029]	0.513 [0.032]	0.512 [0.034]	0.519 [0.519]	0.519 [0.025]	0.514 [0.024]
NE	0.500	0.504 [0.030]	0.508 [0.031]	0.510 [0.031]	0.513 [0.038]	0.510 [0.033]	0.514 [0.031]	0.512 [0.031]	0.515 [0.032]	0.512 [0.032]	0.512 [0.512]	**0.516 [0.023]**	0.510 [0.021]
NFX	0.500	0.502 [0.031]	0.503 [0.029]	0.510 [0.031]	0.510 [0.034]	0.514 [0.035]	**0.515 [0.037]**	0.513 [0.033]	0.504 [0.034]	0.504 [0.034]	0.513 [0.513]	0.510 [0.025]	0.508 [0.025]
NOV	0.500	0.506 [0.033]	0.514 [0.036]	0.506 [0.029]	0.510 [0.036]	0.516 [0.038]	**0.526 [0.034]**	0.512 [0.038]	0.521 [0.036]	0.523 [0.040]	0.511 [0.511]	0.516 [0.027]	0.513 [0.022]
OXY	0.500	0.506 [0.040]	0.506 [0.032]	0.505 [0.036]	0.522 [0.040]	0.523 [0.038]	**0.528 [0.041]**	0.522 [0.043]	0.516 [0.034]	0.516 [0.039]	0.510 [0.51]	0.509 [0.015]	0.514 [0.021]
RDC	0.500	0.507 [0.032]	0.501 [0.030]	0.500 [0.035]	0.515 [0.032]	0.511 [0.033]	0.511 [0.032]	0.512 [0.033]	0.499 [0.029]	**0.520 [0.032]**	0.508 [0.508]	0.511 [0.018]	0.508 [0.022]
RRC	0.500	0.509 [0.032]	0.509 [0.027]	0.505 [0.030]	**0.528 [0.042]**	0.524 [0.032]	**0.528 [0.036]**	0.515 [0.035]	0.513 [0.033]	0.520 [0.038]	0.518 [0.518]	0.511 [0.022]	0.510 [0.021]
SE	0.500	0.505 [0.039]	0.514 [0.038]	0.513 [0.038]	0.519 [0.058]	0.525 [0.047]	0.524 [0.055]	0.522 [0.047]	**0.526 [0.047]**	0.510 [0.047]	0.501 [0.501]	0.500 [0.004]	0.501 [0.005]
SLB	0.500	0.507 [0.034]	0.506 [0.032]	0.504 [0.039]	0.521 [0.044]	**0.528 [0.037]**	0.527 [0.038]	0.515 [0.034]	0.521 [0.035]	0.522 [0.037]	0.511 [0.511]	0.514 [0.02]	0.514 [0.021]
SUN	0.500	0.509 [0.031]	0.503 [0.035]	0.499 [0.050]	0.528 [0.061]	**0.533 [0.068]**	0.529 [0.071]	0.523 [0.080]	0.518 [0.061]	0.526 [0.066]	0.501 [0.501]	0.501 [0.008]	0.502 [0.009]
SWN	0.500	0.501 [0.032]	0.499 [0.035]	0.505 [0.031]	0.514 [0.034]	0.513 [0.031]	**0.521 [0.034]**	0.513 [0.033]	0.501 [0.030]	0.509 [0.039]	0.514 [0.514]	0.511 [0.021]	0.511 [0.021]
WMB	0.500	0.499 [0.037]	0.509 [0.037]	0.497 [0.040]	0.516 [0.040]	0.515 [0.042]	**0.524 [0.044]**	0.504 [0.040]	0.515 [0.041]	0.511 [0.040]	0.502 [0.502]	0.502 [0.014]	0.504 [0.015]
XOM	0.500	0.506 [0.043]	0.511 [0.044]	0.527 [0.050]	**0.568 [0.064]**	0.566 [0.069]	0.563 [0.071]	0.527 [0.076]	0.546 [0.067]	0.552 [0.063]	0.502 [0.502]	0.503 [0.009]	0.504 [0.011]
Avg.	0.500	0.505	0.506	0.507	0.523	0.524	0.526	0.516	0.517	0.519	0.511	0.511	0.511

Table 6.3: Benchmarks visualized with darker shades of green representing the particular classifier is outperforming the average of all the classifiers performances on the stock and darker shades of red represents varying levels of underperformance (same as Table 6.2)

Stocks	Majority class previous interval	C4.5 (CVFDT approximation)			C4.5 Bagging (Online bagging approximation)			C4.5 Boosting (Online boosting approximation)			Streaming ensemble algorithm (SEA)		
		5,000 instances sliding window	10,000 instance sliding window	30,000 instances sliding window	5,000 instances sliding window	10,000 instance sliding window	30,000 instances sliding window	5,000 instances sliding window	10,000 instance sliding window	30,000 instances sliding window	5,000 instances sliding window	10,000 instances sliding window	30,000 instances sliding window
ANR	0.5	0.513	0.519	0.517	0.54	**0.543**	0.531	0.526	0.531	0.522	0.532	0.527	0.527
APA	0.5	0.503	0.509	0.509	0.526	0.527	**0.534**	0.519	0.515	0.524	0.508	0.510	0.512
APC	0.5	0.502	0.503	0.505	0.517	**0.52**	**0.52**	0.514	0.512	0.508	0.509	0.510	0.509
BHI	0.5	0.504	0.505	0.51	0.527	**0.534**	**0.534**	0.52	0.524	0.523	0.515	0.514	0.516
CHK	0.5	0.508	0.505	0.515	**0.525**	0.523	0.515	0.515	0.51	0.513	0.520	0.523	0.520
CNX	0.5	0.505	0.501	0.508	0.516	0.518	**0.523**	0.514	0.515	0.517	0.520	0.519	0.516
COG	0.5	0.499	0.508	0.508	0.51	0.51	**0.519**	0.504	0.511	0.513	0.509	0.516	0.513
COP	0.5	0.506	0.504	0.511	0.528	0.532	**0.535**	0.518	0.517	0.523	0.504	0.506	0.507
CVX	0.5	0.505	0.504	0.508	0.533	0.537	**0.547**	0.524	0.526	0.533	0.502	0.504	0.504
DNR	0.5	0.508	0.508	0.506	0.52	**0.525**	0.52	0.512	0.519	0.51	0.518	0.516	0.522
DO	0.5	0.501	0.507	0.504	0.523	0.523	**0.528**	0.518	0.519	0.522	0.510	0.508	0.507
DVN	0.5	0.506	0.508	0.508	0.523	0.521	**0.527**	0.513	0.52	0.52	0.507	0.508	0.508
EOG	0.5	0.505	0.496	0.506	0.517	0.52	**0.523**	0.518	0.521	0.522	0.515	0.513	0.513
FTI	0.5	0.504	0.51	0.502	0.524	**0.525**	0.519	0.512	0.513	0.516	0.512	0.508	0.511
HAL	0.5	0.505	0.508	0.509	**0.52**	0.515	0.515	0.513	0.513	0.515	0.514	0.512	0.509
HES	0.5	0.503	0.511	0.513	**0.53**	**0.53**	0.528	0.518	0.522	0.515	0.513	0.513	0.515
HP	0.5	0.507	0.502	0.498	0.512	**0.515**	0.514	0.509	0.511	0.511	0.507	0.509	0.510
KMI	0.5	0.512	0.504	0.509	0.54	0.535	**0.549**	0.526	0.527	0.536	0.509	0.507	0.512
MPC	0.5	0.496	0.5	0.502	0.513	0.521	**0.524**	0.51	0.508	**0.524**	0.510	0.508	0.512
MRO	0.5	0.508	0.51	0.497	0.523	**0.528**	0.524	0.516	0.521	0.515	0.506	0.508	0.509
NBL	0.5	0.51	0.502	0.503	0.522	**0.528**	0.527	0.514	0.518	0.518	0.510	0.509	0.513
NBR	0.5	0.504	0.51	0.508	0.518	0.521	**0.524**	0.511	0.513	0.512	0.519	0.519	0.514
NE	0.5	0.504	0.508	0.51	0.513	0.51	0.514	0.512	0.515	0.512	0.512	**0.516**	0.510
NFX	0.5	0.502	0.503	0.51	0.51	0.514	**0.515**	0.513	0.504	0.504	0.513	0.510	0.508
NOV	0.5	0.506	0.514	0.506	0.51	0.516	**0.526**	0.512	0.521	0.523	0.511	0.516	0.513
OXY	0.5	0.506	0.506	0.505	0.522	0.523	**0.528**	0.522	0.516	0.516	0.510	0.509	0.514
RDC	0.5	0.507	0.501	0.5	0.515	0.511	0.511	0.512	0.499	**0.52**	0.508	0.511	0.508
RRC	0.5	0.509	0.509	0.505	**0.528**	0.524	**0.528**	0.515	0.513	0.52	0.518	0.511	0.510
SE	0.5	0.505	0.514	0.513	0.519	0.525	0.524	0.522	**0.526**	0.51	0.501	0.500	0.501
SLB	0.5	0.507	0.506	0.504	0.521	**0.528**	0.527	0.515	0.521	0.522	0.511	0.514	0.514
SUN	0.5	0.509	0.503	0.499	0.528	**0.533**	0.529	0.523	0.518	0.526	0.501	0.501	0.502
SWN	0.5	0.501	0.499	0.505	0.514	0.513	**0.521**	0.513	0.501	0.509	0.514	0.511	0.511
WMB	0.5	0.499	0.509	0.497	0.516	0.515	**0.524**	0.504	0.515	0.511	0.502	0.502	0.504
XOM	0.5	0.506	0.511	0.527	**0.568**	0.566	0.563	0.527	0.546	0.552	0.502	0.503	0.504
	0.500	0.505	0.506	0.507	0.523	0.524	0.526	0.516	0.517	0.519	0.511	0.511	0.511

The third comparison uses an approximation to another adaptive method, Oza's online bagging and boosting algorithms (see Subsection 4.3.1.2.3). As demonstrated in an experiment from that subsection, the solutions from a traditional C4.5 bagging and boosting algorithm should be very similar to the online bagging and boosting method respectively, although the time will be much slower with our conventional learner. As an example, it took over 30 hours to run a 30,000 instance boosted decision tree with sliding windows on an 8-core machine. Both bagging and boosting algorithms used 25 base classifiers in the ensemble.

The last benchmark comparison uses the Streaming Ensemble Algorithm (SEA) of Street and Kim [210], which is covered in detail in Subsection 4.2.2.2. Similar to the results obtained by Street and Kim, we found that 20 base classifiers obtained the highest levels of AUC when compared with using 10 and 50 base classifiers.

The results from the baseline test can be seen in Table 6.2 and are visualized in Table 6.3. In the visualization, darker shades of green represent that the particular classifier is outperforming the average of all the classifier's performances on the stock, and darker shades of red represent varying levels of underperformance. As mentioned, all baselines contain $5,000$, $10,000$ and $30,000$ instances updating with a sliding window every 60 instances and tested on future intervals of 60-instance length. By updating these models using sliding windows, we are in effect accounting for concept drift, thus making for a more realistic baseline.

Building a classifier using the majority class of the previous interval performed worst. This is partially because of our inability to get a probability estimate of our

Table 6.4: Average baseline classifier rank covering all 34 stocks used in the study

Baseline		Avg. Rank
Majority class previous interval		12.65
C4.5 (CVFDT approximation)	5,000 instance sliding window	10.74
	10,000 instance sliding window	10.09
	30,000 instance sliding window	10.15
C4.5 Bagging (Online bagging approximation)	5,000 instance sliding window	3.35
	10,000 instance sliding window	2.62
	30,000 instance sliding window	2.03
C4.5 Boosting (Online boosting approximation)	5,000 instance sliding window	6.21
	10,000 instance sliding window	5.59
	30,000 instance sliding window	5.06
Streaming Ensemble Algorithm (SEA)	5,000 instance sliding window	7.56
	10,000 instance sliding window	7.44
	30,000 instance sliding window	7.53

class prediction and therefore are unable to rank our predicted probability decisions in order to get AUC. The best baseline was the online bagging approximation (C4.5 bagging algorithm) which outperformed (or performed as well as) the other baselines in 33 of the 34 stocks. From the visualization in Table 6.3, it is obvious that the ensemble methods greatly outperform individual classifiers.

To extend the analysis provided in Tables 6.2 and 6.3, we compare the average rank of the benchmark classifiers over the 34 stocks in Table 6.4 (the lower the rank, the better). From this table, the Online bagging approximation benchmark with 30,000 instances is clearly the highest performing classifier (in terms of AUC) and the majority class benchmark is the worst. Generalizing this, Online bagging performs best, followed by Online boosting, Streaming Ensemble Algorithm, Concept Drifting Very Fast Decision Trees, and then majority class.

6.4 Model choices

6.4.1 Overview

Ensemble diversity has received much attention in the machine learning literature, where a diverse set of classifiers within the ensemble is considered an important property. Breiman [28] for example, found that including a diverse set of decision trees increased the performance of the Random Forest algorithm ("...better random forests have lower correlation between classifiers and higher strength"). Dietterich [59] also found that stronger performing ensembles have a large degree of dispersion among the base classifiers (this was also evident from our demonstration in Subsection 3.3.7 and in Figure 3.4). We increase diversity in our framework in four ways: using different classifier types (Subsection 6.4.2), different stocks (Subsection 6.4.3), different feature selection methods (Subsection 6.4.4), and different subsets of data for training base classifiers (Subsection 6.4.5). A description of these and rationale behind our decisions follow.

6.4.2 Classifiers types

For our ensemble we chose three different base classifiers, specifically the artificial neural networks (ANN), support-vector machine (SVM), and the C4.5 decision tree (DT). Different base classifiers created a variety of outcomes from which to choose for the ensemble. Because we are choosing the best classifiers (according to AUC performance) from the last time interval, we let the model decide which base classifiers are important. We therefore may have an ensemble with only one type of base clas-

sifier, since our pool includes 1020 classifiers with 340 of each type, yet we choose 50 base classifiers to include in the ensemble.

For the ANN, we experimented with a number of parameters using a process similar to the one we used in Subsection 3.3.5, which was to examine when changes to parameters either increased or decreased the classification error. Specifically this included a learning rate of 0.30, momentum of 0.20, 1 hidden layer with 100 nodes, and 300 epochs[7]. The SVM used was nonlinear with a polynomial kernel (exponent of 2.0). The DT was of the C4.5 variant and was pruned.

To demonstrate the predictability of an individual classifier type in an ensemble compared to an ensemble composed of all three, we implemented an experiment. We chose the top 50 models[8] from among 1020 base classifiers (34 stocks $\times$ 30 base classifiers) based on the performance of the most recent interval of 60 minutes. The ensemble composed of all three base classifiers included an equal proportion of each base classifier (34 stocks $\times$ 10 DT, 10 SVM, 10 ANN = 1020 base classifiers) in a pool. Additionally the ensemble composed of the best individual performing classifiers, the SVM and the ANN, were of equal proportion to reach 1020 base classifiers in the pool (34 stocks $\times$ 15 SVM, 15 ANN). The results can be found in Table 6.5.

As can be seen from the table, the diverse base classifier ensembles outperformed all three of the ensembles comprised of just one type of classifier for all stocks,

[7]To determine these parameters, we examined different parameters on a small holdout set similar to the method discussed in Subsection 3.3.5.

[8]Fifty base classifiers in the ensemble were found to be ideal; beyond 50 lessened over AUC performance.

Table 6.5: Comparison of ensembles composed of pools comprised only of decision trees (DT), nonlinear support vector machines (SVM), artificial neural networks (ANN), equal combination of all three (DT, SVM, ANN) and a combination of the two individual best classifiers (SVM, ANN)

Stock	DT	SVM	ANN	[DT, SVM, ANN]	[SVM, ANN]
ANR	0.527 [0.032]	0.569 [0.037]	0.557 [0.037]	0.573 [0.037]	**0.575 [0.037]**
APA	0.519 [0.036]	0.522 [0.044]	0.544 [0.041]	0.549 [0.044]	**0.551 [0.044]**
APC	0.521 [0.036]	0.525 [0.041]	0.532 [0.043]	0.544 [0.043]	**0.546 [0.044]**
BHI	0.529 [0.039]	0.531 [0.042]	0.557 [0.042]	0.557 [0.045]	**0.558 [0.043]**
CHK	0.516 [0.033]	0.551 [0.034]	0.538 [0.034]	0.553 [0.033]	**0.554 [0.034]**
CNX	0.518 [0.035]	0.525 [0.038]	0.545 [0.036]	0.545 [0.040]	**0.546 [0.038]**
COG	0.514 [0.030]	0.521 [0.037]	0.530 [0.040]	**0.534 [0.039]**	0.533 [0.040]
COP	0.524 [0.042]	0.526 [0.035]	0.547 [0.045]	0.546 [0.048]	**0.548 [0.044]**
CVX	0.525 [0.052]	0.519 [0.051]	0.554 [0.060]	0.555 [0.065]	**0.557 [0.060]**
DNR	0.518 [0.028]	0.550 [0.038]	0.542 [0.035]	0.553 [0.041]	**0.556 [0.040]**
DO	0.518 [0.042]	0.522 [0.038]	0.542 [0.042]	0.546 [0.046]	**0.547 [0.043]**
DVN	0.525 [0.037]	0.526 [0.042]	0.545 [0.038]	**0.555 [0.041]**	0.552 [0.040]
EOG	0.517 [0.032]	0.524 [0.041]	0.534 [0.040]	0.538 [0.041]	0.537 [0.043]
FTI	0.521 [0.035]	0.524 [0.037]	0.544 [0.040]	0.544 [0.043]	**0.545 [0.044]**
HAL	0.519 [0.040]	0.523 [0.038]	0.538 [0.038]	0.540 [0.040]	**0.541 [0.04]**
HES	0.521 [0.036]	0.525 [0.042]	0.550 [0.046]	0.548 [0.045]	**0.552 [0.046]**
HP	0.517 [0.032]	0.524 [0.039]	0.533 [0.038]	0.538 [0.039]	**0.540 [0.040]**
KMI	0.533 [0.048]	0.544 [0.051]	0.556 [0.053]	0.562 [0.050]	**0.564 [0.050]**
MPC	0.528 [0.031]	0.521 [0.042]	0.533 [0.034]	**0.540 [0.039]**	0.538 [0.042]
MRO	0.517 [0.037]	0.538 [0.039]	0.545 [0.039]	0.549 [0.042]	**0.552 [0.041]**
NBL	0.517 [0.035]	0.515 [0.039]	0.540 [0.040]	0.544 [0.038]	**0.545 [0.040]**
NBR	0.515 [0.029]	0.550 [0.033]	0.548 [0.030]	0.557 [0.032]	**0.558 [0.031]**
NE	0.515 [0.033]	0.518 [0.041]	0.535 [0.036]	**0.536 [0.038]**	0.535 [0.040]
NFX	0.516 [0.032]	0.520 [0.036]	0.521 [0.036]	0.529 [0.036]	**0.53 [0.038]**
NOV	0.524 [0.037]	0.524 [0.031]	0.540 [0.039]	**0.547 [0.037]**	**0.547 [0.036]**
OXY	0.526 [0.043]	0.520 [0.038]	0.544 [0.041]	**0.546 [0.042]**	0.545 [0.039]
RDC	0.520 [0.032]	0.524 [0.037]	0.535 [0.040]	0.534 [0.039]	**0.537 [0.043]**
RRC	0.518 [0.038]	0.527 [0.036]	0.539 [0.039]	0.544 [0.040]	**0.546 [0.037]**
SE	0.521 [0.048]	0.532 [0.051]	0.537 [0.055]	0.542 [0.058]	**0.543 [0.054]**
SLB	0.519 [0.037]	0.519 [0.033]	0.542 [0.041]	0.544 [0.041]	**0.546 [0.039]**
SUN	0.530 [0.071]	0.532 [0.071]	0.550 [0.070]	0.555 [0.072]	**0.558 [0.070]**
SWN	0.512 [0.032]	0.520 [0.038]	0.537 [0.034]	0.536 [0.036]	**0.537 [0.036]**
WMB	0.511 [0.040]	0.531 [0.039]	0.536 [0.042]	0.540 [0.044]	**0.543 [0.040]**
XOM	0.527 [0.063]	0.531 [0.064]	0.574 [0.069]	**0.579 [0.067]**	**0.579 [0.067]**
Average	0.521	0.529	0.542	0.547	0.548

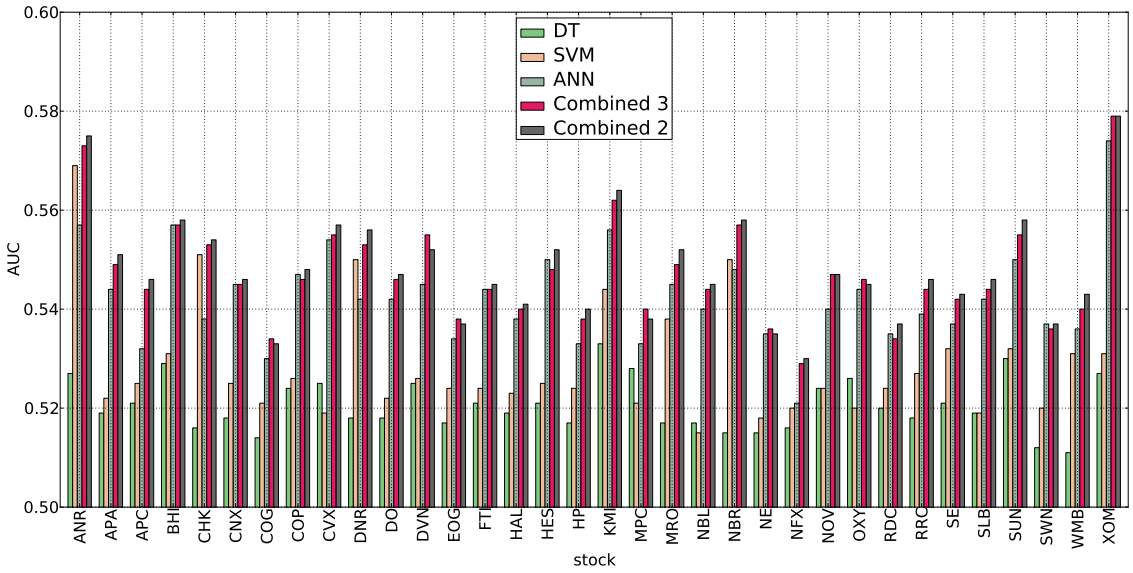


Figure 6.6: Comparison of ensembles composed of pools comprised only of decision trees (DT), nonlinear support vector machines (SVM), artificial neural networks (ANN), an equal combination of all three (Combined 3) and an equal combination of the two best performing base classifiers (Combined 2)

even when accounting for the same number of base classifiers. The ensemble comprised of the two best individual classifiers, the SVM and the ANN, slightly outperformed the ensemble comprised of the DT, SVM, and ANN base classifiers. The combined DT, SVM, and ANN ensemble outperformed the ensemble comprised of a single base classifier type (but containing the same number of base classifiers in the pool as the combined) in 28 stocks and performed the same in 2 stocks. Therefore, for a total of 30 out of 34 stocks the combined diverse model performed at least as well. These results are also shown in Figure 6.6. Using a combination of the SVM and ANN worked the best the majority of the time.

We also questioned the proportion of classifier types included within the 50 classifiers chosen at each interval (among the 1020 base classifiers in the pool evaluated

on the previous interval) and how, if at all, the proportion of classifier type included in the ensemble affected performance. Using results from testing 34 stocks over 89 intervals (each interval is 60 instances in length) we fit a model using least squares. Holding everything else constant, an increase in the proportion of the ANN or SVM in the ensemble (which decrease the proportion of DT), increases the AUC slightly. Both are found to be significant at the 0.05 level and the overall model is found to be significant with an F-statistic of 5.84 and a p-value of < 0.0001 (albeit with a low R^2 model fit of 0.06).

Next we analyzed the proportion of base classifier types chosen in the ensemble based on the base classifier's performance on the previous interval. The objective was to determine how and if the proportion of base classifiers changed over stocks, especially since our results previously found that a higher proportion of ANN and SVM led to slightly greater AUC performance. We ran an experiment using our framework with a total of 1020 base classifiers and an even split between DT, SVM and ANN base classifiers (340 each). We also used an even split between the SVM and ANN base classifiers (510 each). The proportion of base classifier types in the ensemble aggregated over the experiment (88 intervals of 60 minutes) can be seen in Table 6.6. Overall the proportion, when choosing between DT, SVM and ANN in the pool, is relatively stable across stocks with an average proportion of 0.24, 0.31 and 0.45 respectively (Table 6.6a). The proportion, when choosing between SVM and ANN in the pool, is also relatively stable across stocks with an average proportion of 0.34 and 0.66 respectively (Table 6.6b).

Table 6.6: Aggregate base classifier proportion in the ensemble when base classifier was chosen by evaluating on the sliding window $t-1$ over the length of the experiment (largest proportion in bold)

(a) Ensemble pool contains equal proportions of three base classifiers: DT, SVM, and ANN

Stock	DT	SVM	ANN
ANR	0.12	**0.58**	0.30
APA	0.24	0.26	**0.50**
APC	0.25	0.29	**0.45**
BHI	0.22	0.28	**0.50**
CHK	0.17	0.49	**0.35**
CNX	0.24	0.28	**0.48**
COG	0.27	0.31	**0.43**
COP	0.26	0.26	**0.48**
CVX	0.26	0.23	**0.51**
DNR	0.17	**0.47**	0.36
DO	0.25	0.27	**0.48**
DVN	0.24	0.26	**0.49**
EOG	0.25	0.30	**0.45**
FTI	0.24	0.29	**0.46**
HAL	0.24	0.30	**0.46**
HES	0.24	0.28	**0.48**
HP	0.25	0.32	**0.44**
KMI	0.23	0.28	**0.49**
MPC	0.26	0.27	**0.46**
MRO	0.23	0.32	**0.45**
NBL	0.26	0.27	**0.46**
NBR	0.16	**0.50**	0.34
NE	0.26	0.29	**0.45**
NFX	0.26	0.33	**0.40**
NOV	0.26	0.27	**0.47**
OXY	0.25	0.27	**0.48**
RDC	0.27	0.29	**0.44**
RRC	0.23	0.31	**0.46**
SE	0.26	0.31	**0.44**
SLB	0.25	0.27	**0.48**
SUN	0.26	0.30	**0.44**
SWN	0.24	0.32	**0.44**
WMB	0.25	0.31	**0.44**
XOM	0.26	0.21	**0.53**
Average	0.24	0.31	**0.45**

(b) Ensemble pool contains equal proportions of two base classifiers: SVM and ANN

Stock	SVM	ANN
ANR	**0.63**	0.37
APA	0.28	**0.72**
APC	0.34	**0.66**
BHI	0.27	**0.73**
CHK	**0.51**	0.49
CNX	0.29	**0.71**
COG	0.33	**0.67**
COP	0.28	**0.72**
CVX	0.24	**0.76**
DNR	0.49	**0.51**
DO	0.29	**0.71**
DVN	0.27	**0.73**
EOG	0.34	**0.66**
FTI	0.33	**0.67**
HAL	0.33	**0.67**
HES	0.29	**0.71**
HP	0.37	**0.63**
KMI	0.29	**0.71**
MPC	0.30	**0.70**
MRO	0.34	**0.66**
NBL	0.31	**0.69**
NBR	**0.53**	0.47
NE	0.32	**0.68**
NFX	0.39	**0.61**
NOV	0.30	**0.70**
OXY	0.30	**0.70**
RDC	0.33	**0.67**
RRC	0.32	**0.68**
SE	0.35	**0.65**
SLB	0.30	**0.70**
SUN	0.35	**0.65**
SWN	0.35	**0.65**
WMB	0.35	**0.65**
XOM	0.21	**0.79**
Average	0.34	**0.66**

6.4.3 Additional stocks in the classifier pool

6.4.3.1 Inclusion versus exclusion

In Section 6.2.2 we theorized that the inclusion of base classifiers trained on a diverse set of stocks may help increase performance, so in this subsection we ran an experiment to test for this. Training on multiple stocks has the potential to not only increase performance, but to also reduce the total number of base classifiers needed to test all the stocks in the sector. For example, if we set the pool of classifiers in the framework to include a total of 1020 classifiers from multiple stocks, then for each of our 34 sector stocks, 30 base classifiers would be trained. These 1020 base classifiers could then be used to test and predict all the stocks in the sector. However, if we excluded multiple stocks in our framework to predict individual stocks, we would need 34,680 base classifiers (or 1020 for each stock's pool) to test the same number of 34 sector stocks. Therefore, the inclusion of base classifiers from multiple stocks in the pool has the potential to decreases training times (and therefore computational times) for predicting all the stocks in the sector, but also to decreases the amount of physical space needed on the hard drive to save trained models.

Next, we ran an experiment to test if the inclusion of additional sector stocks in the framework pool (34 stocks $\times$ 30 classifiers = 1020 base classifiers) increased stock direction predictability. This experiment tested 88 intervals, each covering 60 instances (minutes), with the best 50 base classifiers from the previous interval being used in the ensemble[9]. The results from including additional stocks in the pool are

[9]Fifty base classifiers in the ensemble were found to be ideal; beyond 50 lessened over

compared with results from excluding additional sector stocks in the framework pool; for a fair comparison we kept the number of base classifiers the same (1 stock $\times$ 1020 classifiers = 1020 base classifiers).

The results from the experiment can be found in Table 6.7. From this table it can be observed that the addition of sector stocks increased predictability even when keeping the number of classifiers in the framework pool stable (1020 base classifiers) in 33 out of the 34 stocks tested. The average AUC across stocks with the inclusion of sector stocks is 0.548 and without the additional stocks is 0.531. In 27 out of the 34 stocks the results are statistically significantly different when examining using a t-test at the critical value for $\alpha = 0.05$. This experiment demonstrates that including additional stocks increases ensemble diversity and increases the average AUC.

Furthermore, when comparing our framework results from Table 6.7 to the baseline results in Table 6.2, it is clear that with the additional stocks added to the pool, our framework outperforms all of the baseline methods for every stock. This is statistically significant at the critical value for $\alpha = 0.05$ for all the stocks except Chevron Corporation (CVX) and Exxon Mobile Corporation (XOM).

6.4.3.2 Change in ensemble stock proportions

In Subsection 6.4.3.1 we concluded that the inclusion of multiple stocks in the pool increases ensemble performance. In our framework each classifier in the pool is trained on one stock; these classifiers are examined and the best (as judged on the

---

AUC performance.

Table 6.7: Including within our ensemble pool, classifiers from the stock we are predicting only (exclusion) or also adding classifiers from stocks within the same sector also (inclusion)

Stock	[SVM, ANN] Exclusion of additional stocks	[SVM, ANN] Inclusion of additional stocks	Stat. Sig. Different
ANR	0.571 [0.037]	**0.575 [0.037]**	no
APA	0.519 [0.041]	**0.551 [0.044]**	yes
APC	0.526 [0.043]	**0.546 [0.044]**	yes
BHI	0.527 [0.036]	**0.558 [0.043]**	yes
CHK	**0.558 [0.038]**	0.554 [0.034]	no
CNX	0.529 [0.042]	**0.546 [0.038]**	yes
COG	0.521 [0.037]	**0.533 [0.040]**	yes
COP	0.532 [0.045]	**0.548 [0.044]**	yes
CVX	0.526 [0.059]	**0.557 [0.060]**	yes
DNR	0.551 [0.037]	**0.556 [0.040]**	no
DO	0.521 [0.038]	**0.547 [0.043]**	yes
DVN	0.528 [0.039]	**0.552 [0.040]**	yes
EOG	0.526 [0.038]	**0.537 [0.043]**	no
FTI	0.522 [0.034]	**0.545 [0.044]**	yes
HAL	0.525 [0.033]	**0.541 [0.040]**	yes
HES	0.525 [0.041]	**0.552 [0.046]**	yes
HP	0.521 [0.037]	**0.540 [0.040]**	yes
KMI	0.549 [0.047]	**0.564 [0.050]**	yes
MPC	0.524 [0.042]	**0.538 [0.042]**	yes
MRO	0.539 [0.037]	**0.552 [0.041]**	yes
NBL	0.522 [0.041]	**0.545 [0.040]**	yes
NBR	0.556 [0.034]	**0.558 [0.031]**	no
NE	0.519 [0.042]	**0.535 [0.040]**	yes
NFX	0.518 [0.034]	**0.530 [0.038]**	yes
NOV	0.530 [0.033]	**0.547 [0.036]**	yes
OXY	0.519 [0.042]	**0.545 [0.039]**	yes
RDC	0.518 [0.040]	**0.537 [0.043]**	yes
RRC	0.527 [0.033]	**0.546 [0.037]**	yes
SE	0.531 [0.052]	**0.543 [0.054]**	no
SLB	0.530 [0.032]	**0.546 [0.039]**	yes
SUN	0.535 [0.071]	**0.558 [0.070]**	yes
SWN	0.526 [0.029]	**0.537 [0.036]**	yes
WMB	0.532 [0.039]	**0.543 [0.040]**	no
XOM	0.553 [0.064]	**0.579 [0.067]**	yes
Average	0.531	0.548	–

previous interval) are selected for inclusion in the pool. With a maximum of 1020 classifiers in the pool at any given time, or 30 classifiers for each of the 34 stocks, a specific stock will comprise roughly $\frac{30}{1020} \approx 3\%$ of the pool at any given time (on average)[10]. In this subsection we examine the composition of this classifier pool to determine if any particular stocks appear more frequently in the ensembles.

In Table 6.8 the average of the stock (over 88 intervals of each 60 instances) is shown as a proportion of the pool and also as a proportion of the classifier selected ensemble. On average, in 24 stocks, the stock makes up a larger proportion of the ensemble than of the entire pool; in 2 stocks it is the same; and in 8 stock the classifier make up a smaller proportion of the ensemble than it does of the pool. Running a one-tailed statistical test at the 0.05 significance level, such that:

$$H_0 : p_{\text{pool}} = p_{\text{ensemble}}$$

$$H_a : p_{\text{pool}} \leq p_{\text{ensemble}}$$

we find that in 18 stocks the stock makes up a statistically larger proportion of its ensemble than it does of the pool (i.e. we reject H_0).

We also checked the performance of the ensemble (as measured by AUC) when the stock we are trying to predict made up a statistically larger proportion of the ensemble; this is compared against the AUC when the stock makes up a statistically smaller proportion of the ensemble. When the stock is a larger proportion of the

[10]This number does vary slightly over the intervals because we use a sliding window covering only the last 30,000 instances. For more information on this technique, please see Subsection 6.4.5 and in particular Figure 6.12.

ensemble than it is of the pool, the AUC of the ensemble is 0.5624, and when the stock is a smaller proportion, the AUC of the ensemble is 0.5506. Increasing the size of the proportion of the stock we are trying to predict in the ensemble would however have a limit, since in the previous Subsection 6.4.3.1 we concluded that more diversity in the pool (and therefore in the ensemble) increased ensemble performance. This needs further study to determine the proportion at which the AUC dropped and which (if any) of these stocks share commonalities.

Also examined is the number of times (over 88 intervals) an individual stock appears as the largest proportion of the ensemble; this amounts to an average of roughly 5 times as the largest proportion of the 88 intervals, or roughly 5.7% of the intervals. The number of times a particular stock appears as the largest proportion can be found in Table 6.9. From this table, it can also be seen that the stock BHI represents the largest proportion of the ensemble the most number of times across the intervals for all but 11 of the stocks. The total number of times BHI is the largest proportion of the ensemble is 276 out of 2992 total intervals, or 9.2%; the remaining stocks are the largest proportion of the ensemble on average 82 out of 2992 intervals, or 2.4% of the total. By simply looking at the chart of BHI from the Appendix B it is difficult to determine exactly why this stock is so prevalent as the largest proportion of the ensemble. This remains an open area for further study.

Table 6.8: Stock n and its average proportion (over all interval) for both the pool and its selection in the ensemble

Stock	Stock as proportion of pool	Stock as proportion of ensemble	Ensemble proportion is statistically larger	Average of AUC
ANR	0.028	0.070	reject H_o	0.578
APA	0.026	0.033	reject H_o	0.556
APC	0.030	0.038	reject H_o	0.550
BHI	0.026	0.048	reject H_o	0.559
CHK	0.030	0.054	reject H_o	0.557
CNX	0.032	0.040	reject H_o	0.549
COG	0.029	0.031	can't reject H_o	0.537
COP	0.031	0.033	can't rejectH_o	0.550
CVX	0.028	0.039	reject H_o	0.566
DNR	0.029	0.045	reject H_o	0.560
DO	0.028	0.030	can't rejectH_o	0.551
DVN	0.033	0.038	reject H_o	0.555
EOG	0.031	0.038	reject H_o	0.548
FTI	0.029	0.028	can't rejectH_o	0.546
HAL	0.029	0.038	reject H_o	0.544
HES	0.031	0.027	can't reject H_o	0.558
HP	0.029	0.031	can't reject H_o	0.547
KMI	0.029	0.054	reject H_o	0.559
MPC	0.026	0.023	can't rejectH_o	0.538
MRO	0.033	0.031	can't rejectH_o	0.555
NBL	0.025	0.028	can't rejectH_o	0.545
NBR	0.030	0.047	reject H_o	0.559
NE	0.032	0.032	can't rejectH_o	0.541
NFX	0.029	0.027	can't rejectH_o	0.533
NOV	0.029	0.040	reject H_o	0.555
OXY	0.026	0.026	can't rejectH_o	0.553
RDC	0.031	0.028	can't rejectH_o	0.542
RRC	0.029	0.039	reject H_o	0.551
SE	0.029	0.027	can't rejectH_o	0.555
SLB	0.033	0.046	reject H_o	0.548
SUN	0.031	0.037	reject H_o	0.555
SWN	0.028	0.023	can't rejectH_o	0.546
WMB	0.027	0.023	can't rejectH_o	0.543
XOM	0.030	0.054	reject H_o	0.582
Average	0.029	0.037	–	0.552

Table 6.9: The number of times (out of 88 intervals) a particular stock makes up the largest proportion of the ensemble (i.e. has the most trained classifiers in the ensemble)

Stock	ANR	APA	APC	BHI	CHK	CNX	COG	COP	CVX	DNR	DO	DVN	EOG	FTI	HAL	HES	HP	KMI	MPC	MRO	NBL	NBR	NE	NFX	NOV	OXY	RDC	RRC	SE	SLB	SUN	SWN	WMB	XOM	Sum of rows
ANR	25		4	3	7	2	2			3	2	2	1	1	3	2	1			2	1	4	1	1		4		3	3	2	3	5	1		88
APA	5	5	6	12	1	3	1	2	5	1	1	3	3	3	3	3	1	2	2	2	2	2			4	3	1	3	2	3	1		2	1	88
APC	2	2	3	11	3	2	2	1	7	1	4	7	1	1	2		2	3	4	1	1		2	2	2	1		4	2	4	2	3	3	3	88
BHI	2	5	2	12	4	3		4	4	2	2	4	1	2	8	1	3	7	2	2	1			2	5	2		2		3	1	1	1		88
CHK	21	3	2	2	8	2	3	1	1	5	1	2			3	2	1	3	1	1			2	4	1	3	1	4		1	1	5	2	2	88
CNX	4	2	4	11	2	3	5	5	1	1	3	1	2	1	5	1	2	6	5	4	2		2	1		2	1	5	1	2	3			1	88
COG	2	2	1	5	3	5	4	3	6	3	4	2	1		4	3	7	3	2	4	1		1	2	3	4	2	6		1	1	2	1		88
COP	2	4	3	11	2	1	1	2	4	2	2	5	1	2	2	2	3	6			1		2	1	5	4	1	3	4		1			11	88
CVX	1	4	7	8	1	3		3	7	1	3	2		2	2	3	3	7	3	1	3			1	4	7	3	1	1	2	1	1		3	88
DNR	13	1	1	3	3	2	2			5	1	4		2	15	2		6		3		3	1	3	3	2	1	3	3	1			2	3	88
DO	4	3	4	10	1	3	2	2	5		3	1	1	1	1	6	1	4	2	4	2	1	1	2	8		1	2		3	1	3	1	5	88
DVN	3	1	4	10	3	3	2	6	2	3	2	1	2	2	6	1	5	2	2	2	2	1	3	1	3	1	1	2		4	3	1	1	3	88
EOG	6	4	5	11	4	1		2	3	1	4	1	2		3	2	2	4	6		3	1	2	2	4	1	1	4	3	2		1	1	2	88
FTI	2	6	3	11		1	2	2	6		3	3		1	5	1	2	3	1	3	2	3	4	1	4	4	3	2	1	2	2		1	4	88
HAL	4	3	3	14	3	2	4	3	2	4	4		1	4	5		2	5	1	5	2	2		2		2		2	1	1	3	1	1	2	88
HES	3	7	3	6		3	2	3	1		3	5	1	4	2		3	4	5	3	2		2	2	2	1	2	7	3	1	2		5	1	88
HP	3	4	3	8	2	3	3		4	2	2	1	3	2	5		4	5	1	4	4		3	3	3	3		3	1	2	3		1	3	88
KMI	3	1	5	3	3	2	1	3	5	1	1		2	1	4	1	1	20	1		1	1	3	1	1	2	3	1	3	1	6	1	2	4	88
MPC	1	1	1	4	2	3	3	2	7	1	3		3	1	2	3	5	8	3	4	3	2	1	1	1	4	2	2		3	5		2	5	88
MRO	12	1	1	3	3	3	2	2	3	5	3	4	2	1	4	1	2	1	7	3	2		2	3	3	2	1	5			2	1	2	2	88
NBL	4	4	4	16		2	4	1	2	5	6	1	3	1	3	2	2	4	1	1	1	4	2		1	2		1	1	4		2	2	2	88
NBR	19	1		1	11		3	1		5	3	4	1	1	8	3	3	2		2	1	3		2	4	2	1	2	2		2	1			88
NE	6	4	6	11	2	5	2	2	4	1		2	2	2	2		2	3	2	9	2		2	5	2	1	3	2			2			2	88
NFX	8	2	3	7	2	2	3	1	2	3	6	1	1	1	2	2	2	3	1	3	3	1	4	3	2	1	3	3		3	3	2	1	4	88
NOV	3	8	8	9	1	1	1	1	3		5	5	3	2	2	1	1	8	3		4	1	1		5	3	2	1		4	2				88
OXY	1	5	7	11	2	3	1	2	5	1	1	3	3	2	5	2		3	4	3		2	2	1	3	4	1	5	1	1	2		1	1	88
RDC	1	3	4	13	2	2	2	2	4	1	1	2		1	2	2	3	6	4	2	1	1	3	3	5	3		2	1	3	2	2		5	88
RRC	5	1	2	7	3	3			2	3	4	7	1	3	6	2	2	4	1	3	5	2		1	2	4	4	2	1	4		2	1	1	88
SE	3	1		8	3	5	1	3	4		1	2	1		5	3	1	4	3	1	1	3	4	2	5	6	1	3	1	2	4	1	1	5	88
SLB	6	4	5	9	3	5		4	5		1	2	4		4		2	6			2	1	2	3	2	4		5		4	2		1	2	88
SUN	1	1	6	2	2			4	7	4	5	1	1		4		2	14	6	3	4	2	2	1	1	2			2	1	2		2	6	88
SWN	5	6	2	10	3	4	1	2	3		1	3	1	1	3	5		2	2	4	1	1	6		2	3		5	2	2	2	2	1	3	88
WMB	4	2	2	7	5	1		5	2	1	1	3	3	1	4	2		6	1	2	5	2	1	4	4	3	1	2	3		4	2	4	1	88
XOM	1		6	7		2	1	1	12	2	3	4		2		1	3	6	2		3	2	1	3	3	3	1	1	1	2	2		3	10	88
Sum of columns	185	101	120	276	94	85	60	75	128	67	89	88	51	48	134	59	73	170	78	81	68	45	62	63	97	93	41	98	43	68	70	39	46	97	2992

'Sum of columns' represents the number of times the column stock was the largest proportion of the ensemble for all the stocks

'Sum of rows' represents the number of times the column stock was the largest proportion of the ensemble for the row stock

6.4.4 Feature reduction analysis

From Appendix C, the total number of attributes used in this thesis is 276. However as explained in that section, these 276 attributes are actually 20 technical analysis indicators (groups) with different parameters. These twenty groups are: lines, rates of changes, moving averages, moving variance ratios, moving average ratios, Aroon indicators, Bollinger bands, commodity channel index, Chaikin volatility, close location value, Chaikin money flow, Chande momentum oscillators, MACD, trend detection index, triple smoothed exponential oscillator, volatility, Williams %, relative strength index, stochastic, and lag correlations. For information on the calculation of each along with the parameter settings, please see the Appendix C.

With such a large number of attributes, Section 5.2.6 discussed the three types of methods for reducing the number of attributes. First were filter-based attribute selection methods (Subsection 5.2.6.2) which do feature-selection as a pre-processing step and is independent of the machine learning algorithm chosen. Second were wrapper feature selection methods (Subsection 5.2.6.3) that apply the machine learning classifier on subsets of data chosen with a heuristic and compare the performance among the total space of possible feature selections. Third were embedded feature selection methods (Subsection 5.2.6.4) that are part of the induction algorithm. An example of an embedded feature selection method is a pruned decision tree.

For our problem we chose two filter-based methods: Information Gain and the Correlation-based filter feature selection methods, for two reasons. The first benefit is speed; in an experiment run previously (Table 5.3) we showed that the

computational speeds are fast. Although we did not include wrapper-based feature selection methods in our experiment, our framework does not exclude the use of these slower computation methods. Using wrapper-based feature selection methods, the base classifier may not be available to use immediately (because of slow feature selection times due to repeated trainings to fit the given dataset), but the classifier may still include important concepts that may obtain significant results in a future period. This will remain an area of further future research. The second reason for choosing filters is their popularity in literature. For example Enke and Thawornwong [68] achieved ideal results using Information Gain feature selection in predicting stock prices. Furthermore, Hall and Holmes [98] found correlation-based selection methods outperformed alternative filter selection methods. In our framework we decided to use both Information Gain and Correlation-based filter feature selection methods; the goal is to use both feature selection methods alternatively on training data before building base classifiers in attempt to create a diverse set of classifiers for the pool.

Preliminary research found reducing the 276 attributes to the top 30 with a filter-based feature subset selection provided ideal results. The process used in our experiments is demonstrated in Figure 6.7. This method shows training data retrieved (with random starting and ending times), reduced with a filter, and this reduced subset is then passed to the next step where a classifier is built with the data. The finished classifier is then placed in the framework pool. Specifically we found that Information Gain and Correlation-based filter methods provided a diverse set of features (as we demonstrate later in this section) and a nice trade-off between AUC

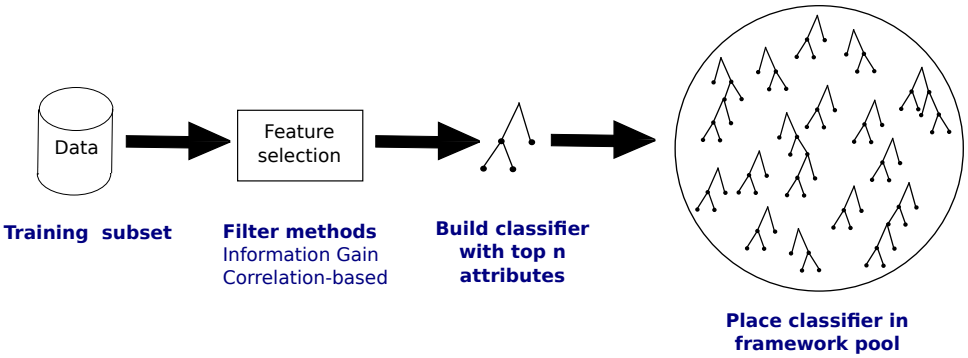


Figure 6.7: Process of implementing a filter-based feature subset selection procedure in our framework

and speed of performance. As mentioned in Section 5.2.6 the goal of any classifier should be the simplest explanation of facts using the fewest variables. This helps minimize model over fitting.

An experiment was conducted to understand the attributes being chosen by the two different feature selection methods and how/if the attributes were similar across the 34 stocks. We first divided our dataset of 47,000 instances into intervals of 1,000 instances each and then ran the two filters (Information gain and correlation-based methods) on each interval while choosing the top 30 attributes[11]. The results for the stock Exxon Corp. can be seen in a raster plot in Figure 6.8; selected attributes are in black. To reduce the figure to one page, we reduced the number of viewable attributes to 100 down from 276 attributes. A dark band existing in either the Correlation-based feature selection filter from Figure 6.8a and the Information Gain feature selection filter from Figure 6.8b signals attributes chosen for the particular

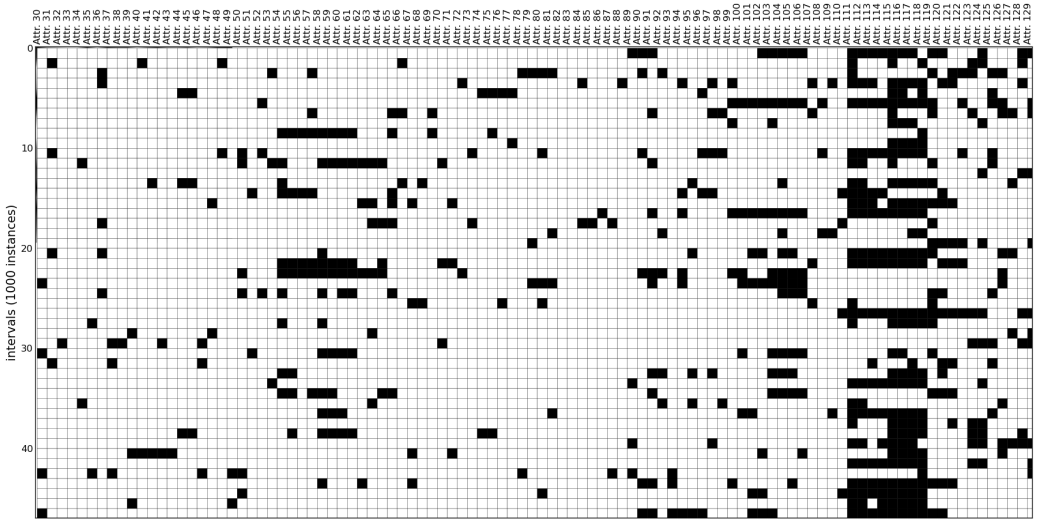
[11]Our preliminary research found 30 attributes provided favorable results. Additional numbers of features tended to overfit and increase training times.

interval. The attributes (marked 111 to 119) are the Chaikin Money Flow and the Chande momentum oscillator with different parameters. More information about the dataset can be found in Appendix B and the attributes can be found in the Appendix C.

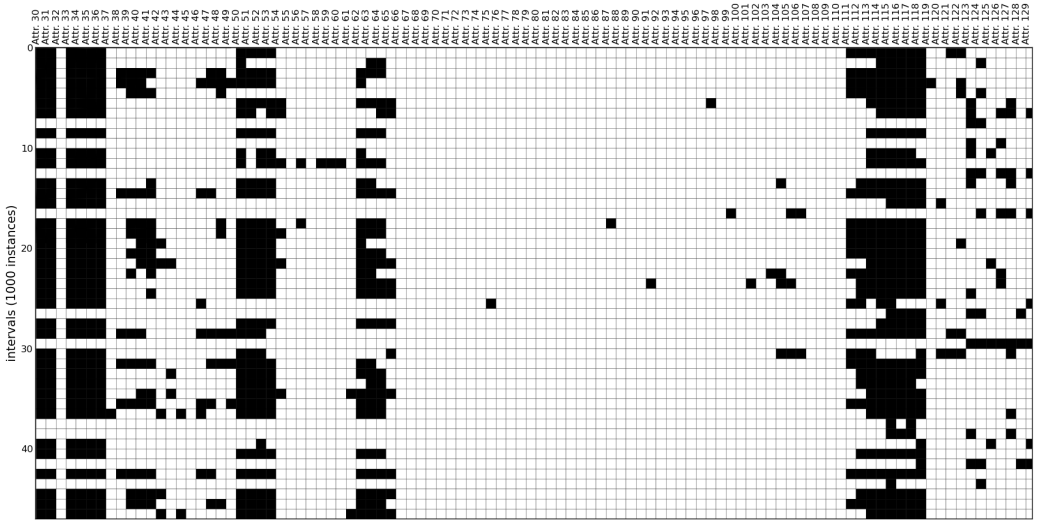
We next examined the technical analysis indicators that were most commonly chosen by each filter. To do so, we followed the same procedure as above by dividing our dataset into intervals of 1,000 instances and running both filters on the dataset. We then counted the number of features from each of the twenty groups of technical analysis indicators chosen over the intervals. Figure 6.9 shows the results of our experiment using both filters. The five most common technical analysis indicator attribute groups chosen by the Correlation-based filter and their average proportion of overall attributes are (in order from highest to lowest): correlation attribute[12] (14.0%), trend detection index (9.3%), moving average rate of change (8.3%), MACD (6.7%), and the Aroon indicator (6.6%). The five least common attributes chosen by the Correlation-based filter and their average proportion of overall attributes chosen are (in order from low to lowest): Bollinger bands (2.7%), commodity channel index (2.6%), close location value (1.6%), volatility (1.0%), and moving average variance (1.0%). The attribute groups chosen by the Correlation-based feature selection method remains relatively stable across stocks can also be seen in Figure 6.9a.

The results are quite different when examining the attributes chosen by the Information Gain feature selection method. From Figure 6.9b the five most pop-

[12]The selection of the correlation attribute by the correlation-based filter is not surprising.

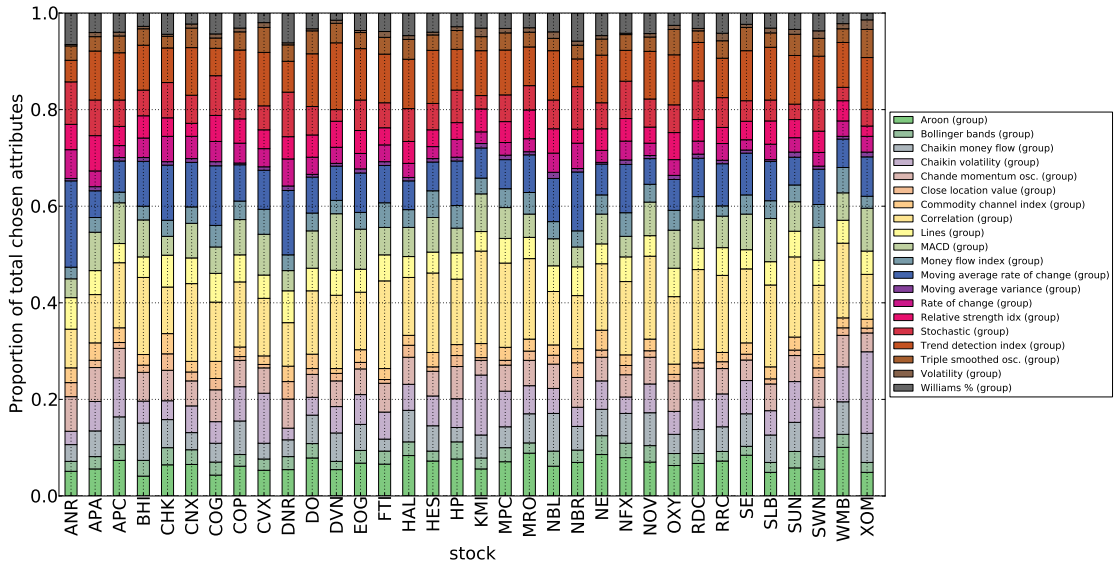


(a) Correlation feature selection filter

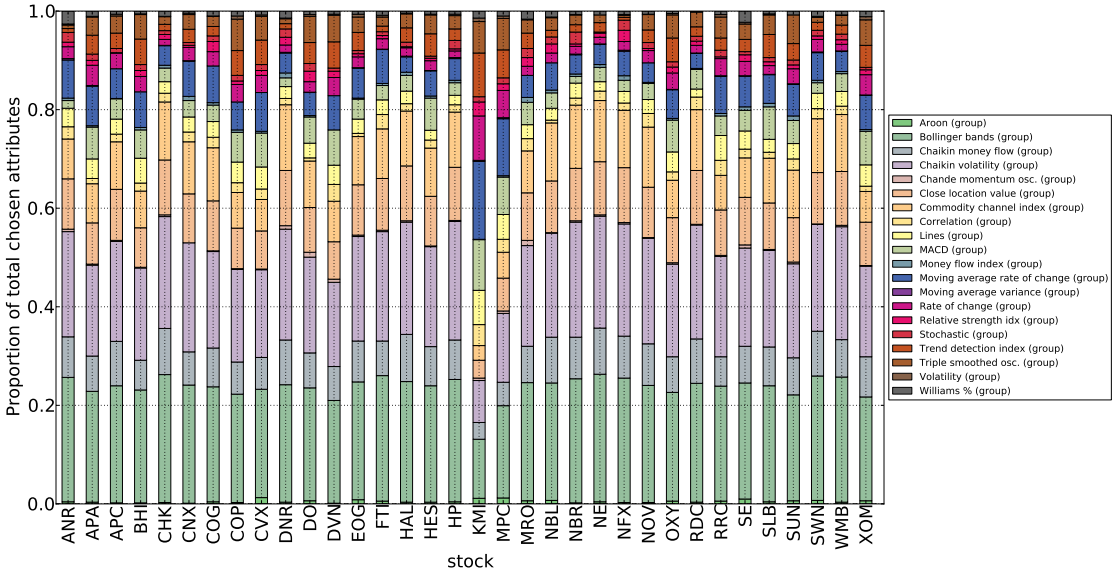


(b) Information Gain feature selection filter

Figure 6.8: Comparison of two different feature selection filters on the stock Exxon (symbol: XOM) using a sliding window of 1000 instances (showing only attributes 30 through 129 to save space)



(a) Correlation-based feature selection



(b) Information Gain feature selection

Figure 6.9: Visualizing the different groups of attributes as a proportion of total attributes chosen by the different filter feature selection methods 46 intervals of 1000

ular attribute groups chosen by the Information Gain filter and their average proportion of overall attributes are (in order from highest to lowest): Bollinger bands (23.2%), Chaikin volatility (20.2%), close location value (9.6%), commodity channel index (9.5%), Chaikin money flow (7.7%). The five least common attributes chosen by the Information Gain filter and their average proportion of over attributes chosen are (in order from low to lowest): Aroon (0.5%), Money flow index (0.5%), Chande momentum oscillator (0.3%), volatility (0.2%), and moving average variance (0.1%). Additionally, three of the top five most common groups across the stocks chosen by the Information Gain filter are in the bottom five (least chosen attribute) of the Correlation-based filter. These are Bollinger bands, commodity channel index and close location value. However, both volatility and moving average variance are within the bottom, or the least chosen, attribute group for both Information Gain and Correlation-based filters.

It is important to note that although *groups* of attributes may remain relatively stable across stocks, *individual* attributes chosen by the filter models may vary greatly. These individual attributes are composed of alternative technical analysis indicator parameters and/or lagged indicators (see Appendix C). As can be seen from the example for Exxon (Figure 6.8a), the attributes chosen during individual intervals can vary from one interval to the next.

We next analyzed if one particular filter was more common among the 50 base classifiers chosen for the ensemble, based on its performance on the previous interval. An experiment was run using our framework with a total of 1020 base

classifiers, with 510 using Information Gain and 510 using Correlation-based feature selection to reduce features. The aggregate proportion of base classifier chosen for the ensemble when trained using either the Correlation-based or Information Gain feature selection filters can be seen in Table 6.10. In 30 out of 34 stocks Information Gain filtered classifiers were most common in the ensemble, with on average 52% of the base classifiers having been filtered using this method. No discernible difference was found in the AUC when using entirely Correlation-based feature selection or Information Gain feature selection.

6.4.5 Subsets for training

Determining the correct or ideal training set size is often accomplished by trial-and-error and can be problematic when using a sliding-window approach to learning with concept drift. A small training set may contain specific concepts but may overfit and not generalize well, while a large dataset may contain too many conflicting concepts which would decrease classifier performance. Our framework's solution to this problem is to train the base classifiers on subsets of data, ranging from a minimum of 5,000 instances to a maximum of 20,000 instances. The objective is to place all instances in the framework pool and evaluate each on the previous interval and find the top performing n base classifiers for the ensemble. Figure 6.10 visualizes 100 base classifiers created with random start and end times within the constraint training set size between 5,000 and 20,000 instances.

In this subsection we compare the classifiers *in the pool* to the classifiers *chosen*

Table 6.10: Aggregate proportion of base classifiers chosen for the ensemble trained using either the Correlation-based or Information Gain filters (base classifiers were chosen by evaluating on the sliding window $t-1$ over the length of the experiment)

Stock	Correlation-based filter	Information gain filter
ANR	**0.56**	0.44
APA	0.47	**0.53**
APC	0.46	**0.54**
BHI	0.45	**0.55**
CHK	**0.54**	0.46
CNX	0.48	**0.52**
COG	0.49	**0.51**
COP	0.46	**0.54**
CVX	0.44	**0.56**
DNR	**0.53**	0.47
DO	0.46	**0.54**
DVN	0.46	**0.54**
EOG	0.48	**0.52**
FTI	0.47	**0.53**
HAL	0.48	**0.52**
HES	0.46	**0.54**
HP	0.48	**0.52**
KMI	0.48	**0.52**
MPC	0.48	**0.52**
MRO	0.49	**0.51**
NBL	0.47	**0.53**
NBR	**0.54**	0.46
NE	0.49	**0.51**
NFX	0.49	**0.51**
NOV	0.48	**0.52**
OXY	0.47	**0.53**
RDC	0.49	**0.51**
RRC	0.49	**0.51**
SE	0.48	**0.52**
SLB	0.46	**0.54**
SUN	0.48	**0.52**
SWN	0.49	**0.51**
WMB	0.49	**0.51**
XOM	0.42	**0.58**
Average	0.48	**0.52**

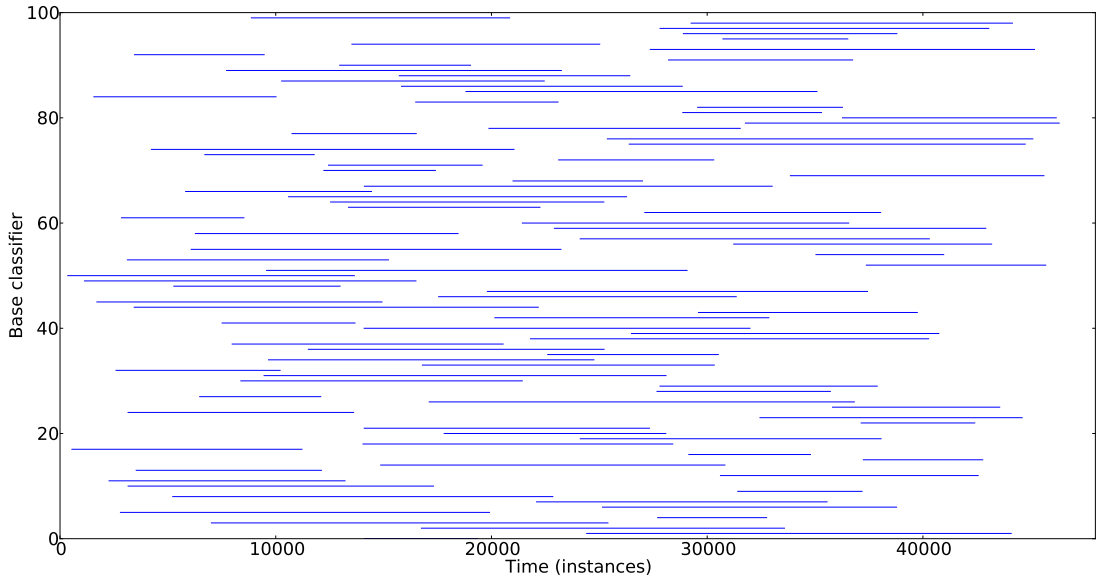


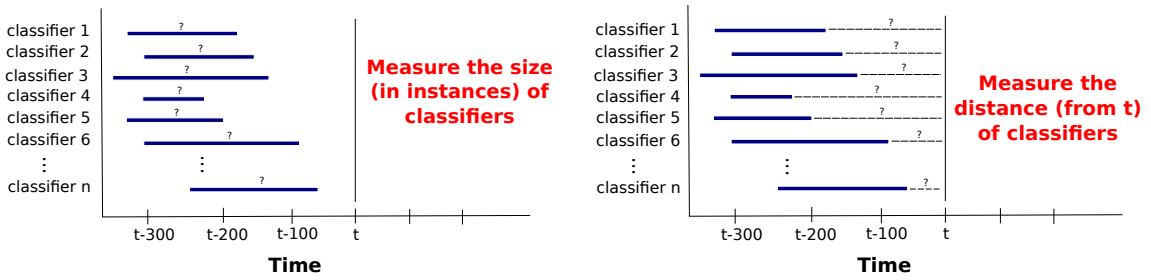
Figure 6.10: Visualization of 100 base classifiers and their random start and end times over 47,000 instances

from the pool for inclusion in the ensemble[13]. Recall from Step 2 in Figure 6.1, the ensemble classifiers are chosen from the pool according to their performance on the last 60 minutes. We also attempt to determine if (1) the size of the training set selected and (2) the age of the training set selected for inclusion in the ensemble differs from the distribution from which it is drawn (i.e. the classifier pool); both are visualized in Figure 6.11.

To explain the question of training set size further, each classifier in the pool is built with a training set of n instances (minimum of 5,000 to maximum of 20,000 instances) with the pool limited to a total of 1020 classifiers[14]. The mean value of

[13]The experiment used the top fifty base classifiers from the pool (across multiple stocks) for inclusion in the ensemble. Beyond fifty base classifiers were found to lessen overall ensemble performance. This is discussed in Subsection 6.4.3.1.

[14]The 1020 total base classifiers are comprised of 30 classifiers for each of the 34 stocks.



(a) Measure the size of the classifiers

(b) Measure the distance from t of the classifiers

Figure 6.11: Each classifier is build with a training subset of size n instances and a distance (or age) of length k from the current time t

the training set size n is calculated for the 50 classifiers in the ensemble and the 1020 classifiers in the pool to determine if they differ; this value of n is visualized in Figure 6.11a.

The question of training set age examines the distance of the last instance in the training set from the current evaluated time; this is visualized in Figure 6.11b as value k. A value of 0 for k represents that the last instance in the classifier is nearest the evaluated instance at time t.

For a fair comparison of classifiers distance (Figure 6.11b), we create an experimental setup[15] similar to Figure 6.12. By using a moving window of size n, where n

See Subsection 6.4.3.1 for more information on this decision.

[15]In this experiment we limit the experiment to using subsets from the past 30,000 instances (which is roughly 60 trading days) to achieve a fair comparison of classifier distance from time t as the series extends. This size of 30,000 was due to constraints with the size of our dataset. In practice however, there might not be a need to limit the distance a classifier can come back. The only constraint would need to be on the number of classifiers built, which would become quite large as classifiers are continuously built and included in the pool. One potential method of doing this is to eliminate classifiers that have not been chosen for a specific number of intervals. Another potential method is to eliminate classifiers that share too much *overlap* (in the time subset)

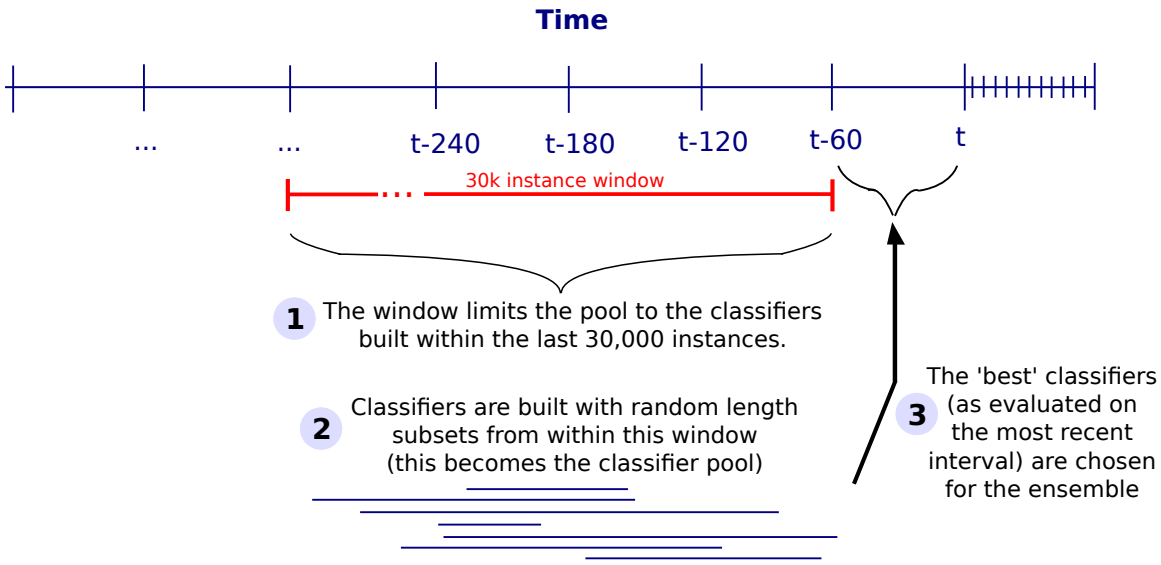
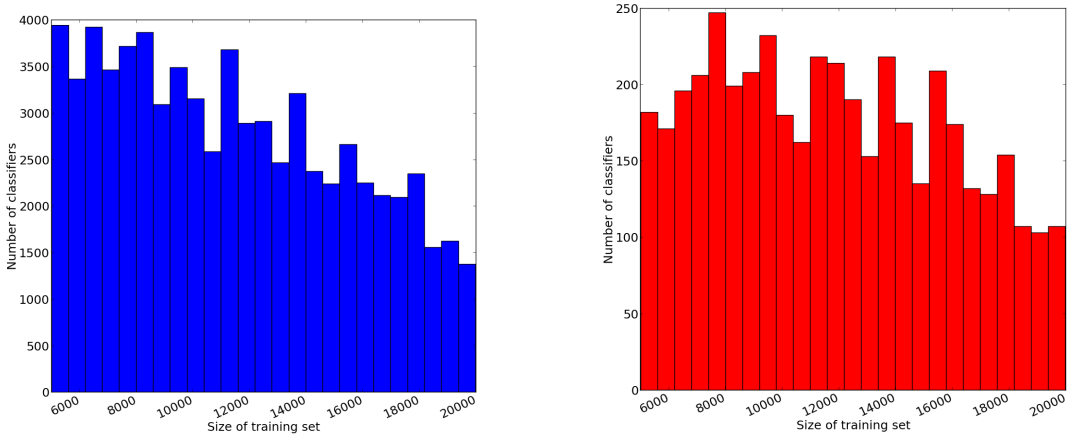


Figure 6.12: The moving window of size n (where n is $30,000$ in our approach) limits the classifiers used in the ensemble

is $30,000$, we limit any bias the experiment may have. Without this sliding window, at the beginning of the experiment we would have a greater number of models in the classifier pool close to time t, and then as the series extends we would have a far greater number of classifiers further away from time t in the pool. The creation of the sliding window of size n therefore allows for an unbiased exploration of classifier distance (from current time t) over time.

This distribution of the sizes of the classifier training sets for the stock WMB (Williams Corporation) can be seen in Figure 6.13 for both the pool (Figure 6.13a) and those chosen from the pool for inclusion in the ensemble (Figure 6.13b). We quantify this difference in distributions using a non-parametric chi-square test of independence to determine if the observed distribution of the classifiers in the ensemble differed significantly from the distribution of the classifiers in the pool (i.e. the expected



(a) Classifiers in pool (b) Classifiers in ensemble

Figure 6.13: Distribution of base classifier training sets for the stock WMB (over entire experiment)

ensemble distribution) at the critical value for $\alpha = 0.05$. This was done to test the distributions of the classifiers in the pool versus the ensemble for both 1) the size of the training sets and 2) the age or distance from time t of the chosen classifiers. Additionally, both a paired t-test and a non-parametric Wilcoxon test[16] at $\alpha = 0.05$ was used to examine the mean size and age of the classifier over the 88 intervals (with each containing 60 instances) to determine if they differ significantly.

The results from the *training set size* test can be seen in Table 6.11a and the *distance from time t* test in Table 6.11b. Additional tables condensing the results can be found in Tables 6.12a and 6.12b respectively.

The average size of the training set (see Figure 6.11a), where an instance represents one minute of time, is slightly larger for the classifiers in the ensemble (i.e.

[16]A case could be made that the paired t-test is not appropriate, hence the addition of its nonparametric equivalent, the Wilcoxon test.

Table 6.11: In minutes, average size of the training set and the average distance of the classifiers from time t for the classifiers in the pool versus the classifiers in the ensemble (larger number in bold) – test statistic at $\alpha = 0.05$

(a) Average size of the training set (across all intervals) for the classifiers in the pool versus those in ensemble (in minutes)

Stock	Pool	Ensemble	χ^2 test	paired t-test
ANR	11439	**11792**	yes	yes
APA	11462	**11863**	yes	yes
APC	11444	**11850**	yes	yes
BHI	11436	**11806**	yes	yes
CHK	11453	**11638**	yes	yes
CNX	11458	**11891**	yes	yes
COG	11445	**11722**	yes	yes
COP	11448	**11981**	yes	yes
CVX	11456	**11933**	yes	yes
DNR	11456	**11757**	yes	yes
DO	11452	**11951**	yes	yes
DVN	11452	**11849**	yes	yes
EOG	11442	**11783**	yes	yes
FTI	11448	**11764**	yes	yes
HAL	11459	**11788**	yes	yes
HES	11445	**11795**	yes	yes
HP	11454	**11681**	yes	yes
KMI	11436	**11710**	yes	yes
MPC	11450	**11782**	yes	yes
MRO	11442	**11645**	yes	yes
NBL	11559	**11804**	yes	yes
NBR	11428	**11599**	yes	yes
NE	11439	**11707**	yes	yes
NFX	11460	**11620**	no	yes
NOV	11443	**11877**	yes	yes
OXY	11457	**11933**	yes	yes
RDC	11444	**11633**	no	no
RRC	11559	**11931**	yes	yes
SE	11450	**11781**	yes	yes
SLB	11452	**11799**	yes	yes
SUN	11447	**11749**	yes	yes
SWN	11452	**11740**	yes	yes
WMB	11442	**11823**	yes	yes
XOM	11444	**11985**	yes	yes
$\text{Average}_{\text{all}}$	11454	11793	–	

(b) Average distance from time t (across all intervals) for the classifiers in the pool versus those in ensemble (in minutes)

Stock	Pool	Ensemble	χ^2 test	paired t-test
ANR	**8731**	8683	yes	no
APA	**8707**	8678	yes	no
APC	**8727**	8598	no	no
BHI	**8740**	8721	no	no
CHK	8728	**8767**	no	no
CNX	**8711**	8681	no	no
COG	8721	**8742**	no	no
COP	**8736**	8566	no	no
CVX	**8718**	8682	no	no
DNR	8710	**8865**	no	no
DO	**8721**	8699	no	no
DVN	**8727**	8657	no	no
EOG	8726	**8947**	no	yes
FTI	8715	**8803**	no	no
HAL	**8705**	8620	no	no
HES	8729	**8980**	no	yes
HP	8735	**8826**	no	no
KMI	8726	**8854**	yes	no
MPC	**8727**	8684	no	no
MRO	8773	**9000**	no	yes
NBL	8537	**8694**	no	no
NBR	8736	**9049**	no	yes
NE	8728	**8786**	yes	no
NFX	8719	**8840**	yes	no
NOV	8721	**8773**	no	no
OXY	**8714**	8618	no	no
RDC	**8720**	8711	yes	no
RRC	8537	**8586**	no	no
SE	**8734**	8709	no	no
SLB	**8712**	8638	yes	no
SUN	8710	**8915**	no	yes
SWN	**8747**	8692	no	no
WMB	**8726**	8683	no	no
XOM	**8723**	8521	yes	yes
$\text{Average}_{\text{all}}$	8714	8743	–	

the best performing classifiers chosen from the pool) than the classifiers in the pool for all 34 stocks examined. From Table 6.11a, the average size of the ensemble classifiers is $11,793$ instances and the average size of the pool classifiers is $11,454$ – a paired t-test covering the 88 intervals (each containing 60 instances) finds statistical difference between the sizes (of the ensemble versus pool) in 33 of the 34 stocks[17]. This is a small difference in the average (roughly 3%), yet a non-parametric chi-square test (testing at the 95% confidence interval) found that the distribution of the training set was statistically different for the classifiers in the ensemble than from the classifiers in the pool for 32 of the 34 stocks. The distributions of the training set sizes of only two stocks, NFX and RDC, were the same.

In judging the distance (i.e. age) of the classifiers from time t (see Figure 6.11b), the distance of the classifier in the pool is slightly closer to the time being evaluated (i.e. time t), than the classifiers in the ensemble. From Table 6.12b, the average pool distance from time t is $8,714$ and the average ensemble distance from time t is $8,743$ – less than one third of 1% difference. The non-parametric chi-square test finds the ensemble comes from a statistically different distribution than the pool in 8 of the 34 stocks. Additionally, the paired t-test finds the age of the classifiers differ over the 88 intervals in 6 of the 34 stocks[18].

Both of the experimental results are combined in Tables 6.12a and 6.12b re-

[17]The nonparametric t-test equivalent, the Wilcoxon test, came to the same conclusion for the difference in training set sizes.

[18]The Wilcoxon test, also found a difference in 6 of the 34 stock's training set ages over the 88 intervals.

Table 6.12: Comparison of classifiers from the pool versus the classifiers chosen for the ensemble

(a) Comparison of classifiers in the pool and ensemble in judging the size of the average classifiers training set size

	Pool	**Ensemble**
Avg. training size	11,454	11,793
Count of stocks, where training size is larger (pool vs. ensemble)	0 stocks pool larger	34 stocks ensemble larger
Paired t-test (sig.)	33 stocks significant	
Chi-square test (sig.)	32 stocks significant	

(b) Comparison of classifiers in the pool and ensemble in judging the average classifiers distance from time t

	Pool	**Ensemble**
Avg. dist. from t	8,714	8,743
Count of stocks, where training size is older (pool vs. ensemble)	18 stocks pool older	16 stocks ensemble older
Paired t-test (sig.)	6 stocks significant	
Chi-square test (sig.)	8 stocks significant	

spectively. These experiments suggest on average, larger training set sizes outperform smaller ones and rarely is there much difference in the average distance of the classifier from the current time t in the performance. This needs to be explored further in future research.

6.4.6 Incorporating time into the predictive model

In Section 5.3, the release of pre-scheduled economic reports was found to have an effect on the volatility of the stock price. The reaction of stocks to economic reports makes sense since the data released often gives an indication of the overall economic health. Through an experiment, we found that market participants often

appeared to be "surprised" at the release of the petroleum weekly status update since the 34 energy stocks examined changed direction immediately after the release of the news. Since these economic reports are pre-scheduled, using time as an indicator may increase the predictability of the market direction. To further discover how time might affect the stock direction, we conduct an experiment to empirically examine the change in the probability of large market moves throughout the day, where a large move is determined to be a move greater than 0.05% in either direction, or $|\frac{price_t - price_{t-1}}{price_{t-1}}| \geq 0.0005$.

Our experiment analyzed the same stocks (as the previous experiment) and divide the days into 30 minute increments (e.g. 9:30-9:59, 10:00-10:29, 10:30-10:59, etc.). The percentage of large price moves greater than 0.05% (in either direction) for each increment is measured and from anecdotal examination, a sum of least squares regression line is computed for 9:30 a.m. E.S.T. (the market opening) to 12:30 p.m. Another is computed for 1 p.m. to 4 p.m. E.S.T. (the market close). The slopes are computed and we observe in 98.333% of the days, the slope decreases from the opening of the trading day to 12:30 p.m. and in 52.778% of the days, the slope increases from 1 p.m. to the closing of the trading day. An example of the test for the stock ConocoPhillips can be seen in Figure 6.14. The blue bars in this illustration represent the percentage of time that the interval is comprised of moves greater than 0.05% and the red line represents the regression line from the market opening to 12:30 and from 1:00 p.m. to the market close.

The increased volatility at the beginning of the day is often attributed to

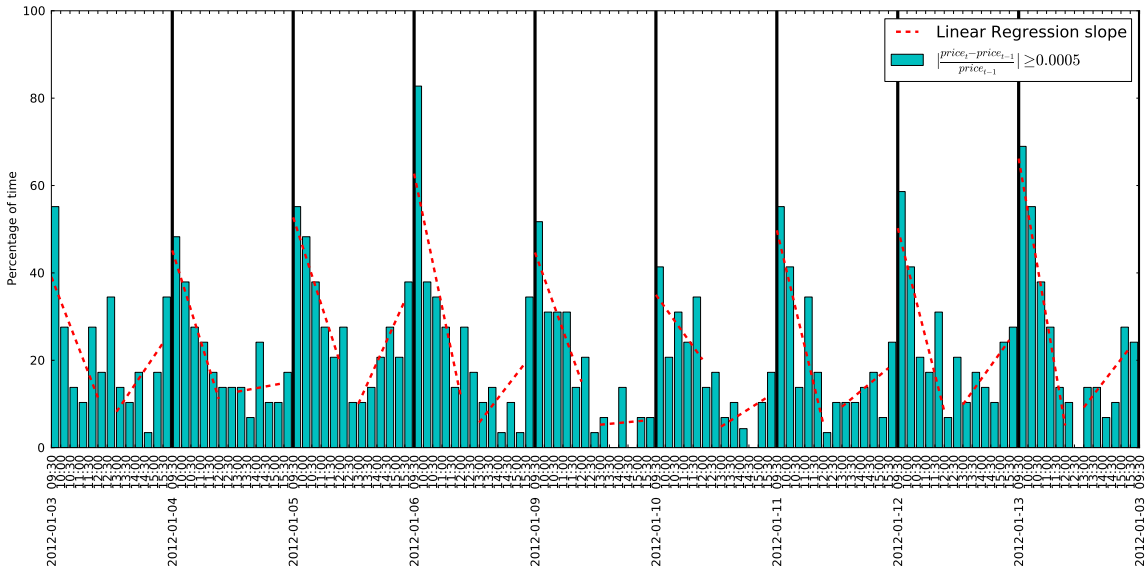


Figure 6.14: Demonstrating a decrease in slope before 12:30 p.m. and an increase in slope after 1:00 p.m. in the level of price moves over 0.05% (either up or down) throughout the trading day (stock: ConocoPhillips)

normal market behavior. Normal market hours begin at 9:30 E.S.T. and it is at this time that the *specialist* at the New York Stock Exchange reviews the queue of orders that have arrived across the country from individuals wanting to buy and sell the stock to try and determine where to open the stock price[19]. An imbalance of buyers and sellers may cause volatility of the stock price.

In both experiments time affects the distribution of class; this could be considered a form of *concept drift* according to the first of the three definitions of concept

[19]For example, a stock closes (ends) Monday at $61.00. Overnight bad news occurs and individuals across the world send in orders to sell the stock. On Tuesday morning the specialists may decide, because of the imbalance of orders (more sellers than buyers), to open the stock down $0.25 to $60.75. The specialist must take the other side of the order if there is an imbalance and therefore wants to mitigate their risk. By opening the price at a lower level, the specialist intends to entice buyers back in the marketplace and (hopefully) sell the excess inventory at a higher price.

drift provided by Kelly et al. [115] (see Subsection 4.2). In this definition, concept drift is defined as a change in the probability of class priors over time (i.e. $P(c_i)$, $i = 1, 2, 3, \ldots, k$). The problem, as explained in that subsection, is that change in the class priors may change over time but may not affect the overall predictability of the market. However, Hoens et al. [104] reason that, while model predictability may not suffer as a result of a change in the class priors, often the change creates a class imbalance (see Section 5.1) which *may* require retraining.

Our framework deals with this change in the class distribution (and therefore the potential of a decrease in model predictability) by modifying the base classifiers in the ensemble (by evaluating individual base classifiers on the most recent interval of data). However, can additional improvements be made by including a time attribute? In an experiment we test this hypothesis to determine if the inclusion of a time-base feature increases the predictability of the ensemble. The new attributes include: month, day of week, hour of day, and minute of day. The base classifiers used in the experiment were SVM and ANN and the inclusion of base classifiers trained on additional sector stocks were included in the pool (as was determined from Subsection 6.4.2 and 6.4.3.1 respectively). Results of this experiment can be found in Table 6.13.

From the table, it can be seen that the inclusion of the four time attributes slightly increases the AUC, although for none of the stocks is the results statistically significant when testing using a t-test. However, in 28 out of the 34 stocks the inclusion of time in model training slightly outperforms not including it. Including the stocks where it does *at least as well* is 31 out of 34 stocks; only in 3 stocks does

Table 6.13: Including base classifiers in the pool with and without four new time attributes (best in bold) – test statistic at $\alpha = 0.05$

Stock	[SVM, ANN] Exclusion of time attributes	[SVM, ANN] Inclusion of time attributes	Stat. Sig. Different
ANR	0.575 [0.037]	**0.579 [0.038]**	no
APA	0.551 [0.044]	**0.557 [0.054]**	no
APC	0.546 [0.044]	**0.551 [0.050]**	no
BHI	0.558 [0.043]	0.558 [0.054]	no
CHK	0.554 [0.034]	**0.558 [0.037]**	no
CNX	0.546 [0.038]	**0.548 [0.040]**	no
COG	0.533 [0.040]	**0.536 [0.046]**	no
COP	0.548 [0.044]	**0.550 [0.050]**	no
CVX	0.557 [0.060]	**0.564 [0.065]**	no
DNR	0.556 [0.040]	**0.557 [0.039]**	no
DO	0.547 [0.043]	**0.550 [0.055]**	no
DVN	0.552 [0.040]	**0.554 [0.047]**	no
EOG	0.537 [0.043]	**0.546 [0.044]**	no
FTI	0.545 [0.044]	**0.548 [0.041]**	no
HAL	0.541 [0.040]	**0.545 [0.042]**	no
HES	0.552 [0.046]	**0.557 [0.050]**	no
HP	0.540 [0.040]	**0.546 [0.039]**	no
KMI	**0.564 [0.050]**	0.559 [0.047]	no
MPC	0.538 [0.042]	**0.539 [0.041]**	no
MRO	0.552 [0.041]	**0.555 [0.040]**	no
NBL	0.545 [0.040]	0.545 [0.044]	no
NBR	0.558 [0.031]	**0.560 [0.035]**	no
NE	0.535 [0.040]	**0.542 [0.045]**	no
NFX	0.530 [0.038]	**0.535 [0.039]**	no
NOV	0.547 [0.036]	**0.555 [0.036]**	no
OXY	0.545 [0.039]	**0.551 [0.046]**	no
RDC	0.537 [0.043]	**0.543 [0.041]**	no
RRC	0.546 [0.037]	**0.551 [0.044]**	no
SE	0.543 [0.054]	**0.552 [0.054]**	no
SLB	0.546 [0.039]	**0.548 [0.049]**	no
SUN	**0.558 [0.070]**	0.556 [0.078]	no
SWN	0.537 [0.036]	**0.545 [0.039]**	no
WMB	**0.543 [0.040]**	0.542 [0.045]	no
XOM	0.579 [0.067]	0.579 [0.079]	no
Average	0.548	**0.552**	–

the inclusion of time not increase model predictability.

CHAPTER 7
CONCLUSION AND FUTURE RESEARCH

7.1 Summary of prediction of stock market direction

This thesis explored the difficult problem of attempting to predict stock market direction in the near term – one minute or less in the future. While this is a popular problem in both the finance industry and in academia, there remains minimal published methods in machine learning for doing this.

This thesis first tackles the argument that stock prices are purely efficient and do not display trends. In Chapter 2 we examine 22 million transactions in the Standard and Poor's 500 over the course of a year and find reoccurring trends. While many of these inefficiencies can be explained by traditional market dynamics, we found that on average the market escapes the confines of the bid and ask after several seconds, and that high probability events could be observed until several minutes. This gave us confidence that our research area was not futile and allowed us to examine predictability of the market further.

Chapter 3 provided an introduction to some of the machine learning classifiers and performance evaluation methods used in this thesis. While we used AUC (**A**rea **U**nder the ROC **C**urve) to examine classifier performance, a common question asked is why we did not examine model profitability. While this would be the obvious end goal of predicting stock price direction, this adds considerable complexity that is best left to another dissertation. The goal, as we discuss in this chapter, is not to provide

an "out of box" trading system, but to instead help the user make more informed trading decision with the help of machine learning techniques.

Two main methods of learning from concept drift were discussed in Chapter 4. These two methods are: 1) detect concept drift and upon finding it, re-learn using new data in the classifier and 2) assume that concept drift occurs and build this assumption into a framework. The first method, which uses novelty detection algorithms, detects either changes in the distribution of the underlying data (with the assumption that distribution change signal a change in the underlying concept) or a decrease in the performance in the model. Upon detecting a change, the model eliminates or uses *forgetting factors* to give less weight to older data. This approach may be problematic since it requires a fast implementation of a classifier, often called an adaptive or online learning method, to work with streaming high-frequency data; speed of classification often comes at the expense of optimized performance (such as AUC or accuracy). Furthermore, it makes the assumption that older data will not be useful again in the future. This is a difficult argument to make since the stock market periodically displays reoccurring behavior such as economic cycles or behavioral moods. Also, eliminating old data from the previous concept requires a new model to be built; enough new data would need to arrive before building could begin.

This thesis takes the second approach, which assumes that concept drift occurs in high-frequency stock data. This theory is validated through an experiment in Subsection 4.2.2 using the **D**rift **D**etection **M**ethod (DDM) algorithm [83]. Using the

DDM we demonstrate that decreases in model predictability did occur, thus signaling occurrence of concept drift. Advantages of using the wrapper approach to predicting data streams includes the ability to use traditional classifiers, the ability to work in parallel on a multi-core machine, and the use of older data which may still provide value. Maintaining a model with the most up-to-date data is not necessarily the most ideal choice since markets may stabilize and old knowledge may become useful again. In general, wrapper frameworks work by building classifiers on chunks of data and heuristically combining those decisions into a prediction.

Before the discussion of our new wrapper framework, in Chapter 5 we cover challenges in learning from high-frequency stock data. One problem identified in this chapter is stock data stream class imbalance. This can cause problems with classifier performance; the classifier may have high levels of accuracy while misclassifying most of the minority class (often the most revealing part of the algorithm). This is an advantage of our framework since over-, under- and synthetic-sampling of data can be easily applied when the data is learned on chucks of data. Another difficulty with stock data is the data streams often need preprocessing to eliminate *bad* trade data and noise (we run an experiment in this chapter using a heuristic by Brownlees and Gallo [30] to eliminate out-of-sequence trades with moderate success).

Chapter 5 also examines attribute creation for high-frequency stock prediction. This includes the use of sentiment analysis and also technical analysis indicators. More examples of technical analysis along with descriptions can be found in Appendix C. Lastly this chapter explores different methods of dimensionality reduction along

with its ability to not only increase classifier predictability, but also decrease the time spent in training.

Chapter 6 describes our new wrapper framework for the prediction of high-frequency stock direction. In addition to the regular advantages of wrappers, which includes the ability to use traditional classifiers, work in parallel, and work with prior concepts, our method also has the ability to transfer knowledge contained within additional stocks. Our framework builds thousands of classifiers using random subsets of past data (and from correlated sector stocks) and heuristically combines the best of these classifiers as determined by evaluating on the most recent interval of data. The knowledge transfer from the correlated sector stocks in the pool increases the overall framework's predictability even while maintaining the same number of classifiers in the pool. For example, we achieve better performance when including a combined 1020 classifiers in the pool spread across all 34 stocks (i.e. 34 stocks $\times$ 8 classifiers = 1020 base classifiers in the pool) rather than including all classifiers from one stock (i.e. 1 stock $\times$ 1020 classifiers = 1020 base classifiers in the pool). This diversity of stocks increased the AUC for 33 of the 34 stocks, while reducing the total number of classifiers needed to test the same number of 34 sector stocks, from $34,680$ (or 1020 for each stock's pool) to 1020.

While our new wrapper framework is slower at training than many adaptive methods, such as Very Fast Decision Trees, it is *fast enough* to work with higher-frequency data, even though the classifiers used in the framework may take several minutes to build (as demonstrated in Subsection 6.2.1. This is because the emphasis of

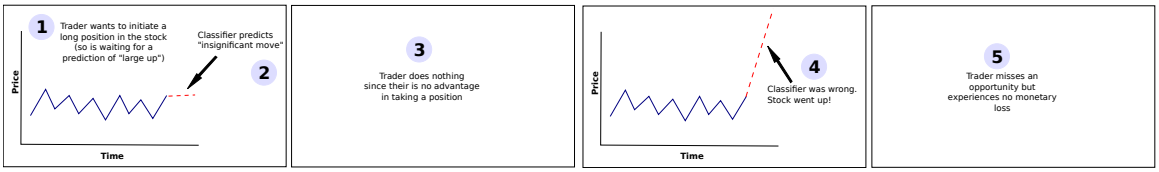
our approach is on evaluating many previously built classifiers (often trained on older concepts). This allows our approach to evaluate hundreds or thousands of classifiers, covering many different concepts and stocks, quickly and in the time needed. Our approach can compete with faster adaptive methods and achieve higher levels of predictability.

7.2 Future research

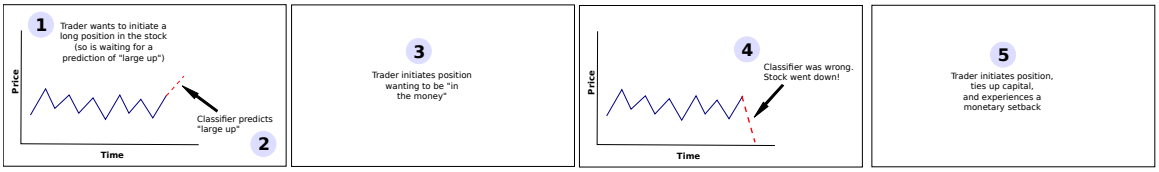
Currently AUC is used as an evaluation metric, but this assumes that all misidentification are the same. When trading, not all mistakes are the same and this may change depending on the position of the trader (shares held). For example, an opportunity cost can be associated with missing a large move in a stock. Additionally an actual monetary loss could be incurred if a trader acts on the prediction of a large price increase, but the stock actually decreased in price. Different cost could be associated with both and these costs can change depending on the needs of the trader.

In Figure 7.1 two examples are given explaining different costs associated with different misclassifications. For example, in Figure 7.1a the trader wants to initiate a long position[1] in a stock with the intentions of selling at a higher price. Therefore, the trader is hopes to buy the stock in anticipation of a large upward move. The

[1]A trader can take one of two positions in a stock: a long position or a short position. A *long position* is when the trader buys the stock with the hopes of the stock going up in price. A *short position* is when the traders sells the stock (without actually owning it), with the intentions of buying it back at a lower price. When a trader is short a stock, a drop in price is favorable.



(a) Incorrect prediction with opportunity costs



(b) Incorrect prediction with monetary costs

Figure 7.1: Incorrect predictions have different costs depending on the objective

classifier in this example predicts that the stock will move an "insignificant amount." The trader therefore does nothing since their is no advantage of taking a long position. It turns out however that the classifier misidentifies the next move in stock price and the stock goes up. In this example, the trader misses the opportunity, but experiences no monetary costs.

In Figure 7.1b the trader once again wants to initiate a long position in the stock with the intentions of selling at a higher price. The classifier in this example predicts a "large up" (a large upward price in the stock), therefore the trader initiates a position with the hope of selling at a higher price. However the classifier was incorrect and the stock price actually decreases by a large amount. In this example the classifier was also wrong, but in this situation the trader actually initiated a position in the stock. This ties up capital (which may prevent future transactions/trades until more money is available) and the trader experiences a *paper loss* (an actual loss is not incurred until the trader actually sells the stock).

Instead of using AUC to evaluate classifiers where the cost or error is uniform, a cost-sensitive approach could be used with explicitly defined cost-matrices. Depending on the situation or objective of the trader, such as if precision or sensitive (recall) is more important in the decision, different penalties could be implemented for misclassifying instances. The classifier therefore minimizes the cost of misclassification rather than the misclassification errors. In Subsection 5.1.2.2 three cost-based solutions are described. In particular MetaCost [61] has achieved interesting results in our preliminary research.

In Table 7.1a, the cost-matrix emphasis is on precision. For example, predicting {Up | Down} when the price is actually {Down | Up} would result in a monetary loss which we estimate at $1.25. When predicting {Up | Down} when the price is actually {No Change} would result in a trade but may still have the possibility of a correct movement in the future, otherwise transactions would have incurred (which we estimate at $0.25). Lastly, correct decisions results in no costs.

In Table 7.1b, the cost-matrix emphasis is on sensitivity (recall) of "up" moves. For example, misidentifying a downward moves as up results in a loss of $1.25; this is better than misidentifying an upward move by predicting as down (loss of $2.50).

Adding cost-matrices or other elements to address cost implications would boost the complexity of our framework and optimize performance.

Table 7.1: Hypothetical cost matrix

(a) Cost-matrix 1

		Predicted class		
		Down	**No change**	**Up**
Actual class	**Down**	0	0.25	1.25
	No change	0.50	0	0.50
	Up	1.25	0.25	0

(b) Cost-matrix 2

		Predicted class		
		Down	**No change**	**Up**
Actual class	**Down**	0	0	1.25
	No change	0.25	0	0.25
	Up	2.50	2.50	0

APPENDIX A
PROBABILITY TABLES

The following tables are the full prior conditional probabilities of market direction movements for trade-by-trade, 1, 3, 5, 10, 20, and 30 second interval timespans, followed by 1 and 5 minute timespans. Thirty-minute timespans were not included because of the lack of priors at extended depths. Also included is the number of weeks out of the year that are statistically significant when compared against the probability of an uptick (Pr(+)).

Table A.1: Conditional probabilities of market directional movements for trade-by-trade (tick) data

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.500	0.001	n/a
1	Pr(+—-)	0.790	0.014	52
	Pr(+—+)	0.209	0.014	52
2	Pr(+—-,-)	0.846	0.018	52
	Pr(+—-,+)	0.776	0.014	52
	Pr(+—+,-)	0.224	0.014	52
	Pr(+—+,+)	0.153	0.017	52
3	Pr(+—-,-,-)	0.848	0.025	52
	Pr(+—-,-,+)	0.846	0.017	52
	Pr(+—-,+,-)	0.790	0.016	52
	Pr(+—-,+,+)	0.725	0.009	52
	Pr(+—+,-,-)	0.274	0.009	52
	Pr(+—+,-,+)	0.210	0.016	52
	Pr(+—+,+,-)	0.153	0.017	52
	Pr(+—+,+,+)	0.153	0.025	52
4	Pr(+—-,-,-,-)	0.843	0.035	52
	Pr(+—-,-,-,+)	0.849	0.025	52
	Pr(+—-,-,+,-)	0.845	0.019	52
	Pr(+—-,-,+,+)	0.848	0.016	52
	Pr(+—-,+,-,-)	0.770	0.019	52
	Pr(+—-,+,-,+)	0.795	0.016	52
	Pr(+—-,+,+,-)	0.722	0.010	52
	Pr(+—-,+,+,+)	0.741	0.018	52
	Pr(+—+,-,-,-)	0.256	0.018	52
	Pr(+—+,-,-,+)	0.277	0.010	52
	Pr(+—+,-,+,-)	0.204	0.016	52
	Pr(+—+,-,+,+)	0.230	0.019	52
	Pr(+—+,+,-,-)	0.151	0.015	52
	Pr(+—+,+,-,+)	0.153	0.019	52
	Pr(+—+,+,+,-)	0.151	0.025	52
	Pr(+—+,+,+,+)	0.166	0.034	52
5	Pr(+—-,-,-,-,-)	0.838	0.073	50
	Pr(+—-,-,-,-,+)	0.843	0.040	52
	Pr(+—-,-,-,+,-)	0.845	0.026	52
	Pr(+—-,-,-,+,+)	0.857	0.031	52
	Pr(+—-,-,+,-,-)	0.829	0.024	52
	Pr(+—-,-,+,-,+)	0.849	0.018	52
	Pr(+—-,-,+,+,-)	0.847	0.018	52
	Pr(+—-,-,+,+,+)	0.853	0.023	52
	Pr(+—-,+,-,-,-)	0.754	0.030	52
	Pr(+—-,+,-,-,+)	0.773	0.017	52
	Pr(+—-,+,-,+,-)	0.797	0.017	52
	Pr(+—-,+,-,+,+)	0.787	0.013	52
	Pr(+—-,+,+,-,-)	0.704	0.013	52
	Pr(+—-,+,+,-,+)	0.729	0.011	52
	Pr(+—-,+,+,+,-)	0.739	0.019	52
	Pr(+—-,+,+,+,+)	0.750	0.044	52
	Pr(+—+,-,-,-,-)	0.249	0.043	52
	Pr(+—+,-,-,-,+)	0.257	0.019	52
	Pr(+—+,-,-,+,-)	0.269	0.011	52
	Pr(+—+,-,-,+,+)	0.297	0.014	52
	Pr(+—+,-,+,-,-)	0.213	0.013	52
	Pr(+—+,-,+,-,+)	0.202	0.017	52
	Pr(+—+,-,+,+,-)	0.227	0.017	52
	Pr(+—+,-,+,+,+)	0.245	0.032	52
	Pr(+—+,+,-,-,-)	0.147	0.031	52
	Pr(+—+,+,-,-,+)	0.152	0.017	52
	Pr(+—+,+,-,+,-)	0.149	0.018	52
	Pr(+—+,+,-,+,+)	0.166	0.022	52
	Pr(+—+,+,+,-,-)	0.146	0.035	52
	Pr(+—+,+,+,-,+)	0.153	0.026	52
	Pr(+—+,+,+,+,-)	0.162	0.033	52
	Pr(+—+,+,+,+,+)	0.184	0.077	48

Table A.2: Conditional probabilities of market directional movements for 1 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.500	0.002	n/a
1	Pr(+—-)	0.644	0.012	52
	Pr(+—+)	0.356	0.011	52
2	Pr(+—-,-)	0.653	0.016	52
	Pr(+—-,+)	0.639	0.012	52
	Pr(+—+,-)	0.359	0.010	52
	Pr(+—+,+)	0.351	0.016	52
3	Pr(+—-,-,-)	0.634	0.018	52
	Pr(+—-,-,+)	0.663	0.016	52
	Pr(+—-,+,-)	0.640	0.015	52
	Pr(+—-,+,+)	0.636	0.010	52
	Pr(+—+,-,-)	0.360	0.010	52
	Pr(+—+,-,+)	0.358	0.012	52
	Pr(+—+,+,-)	0.338	0.016	52
	Pr(+—+,+,+)	0.374	0.019	52
4	Pr(+—-,-,-,-)	0.621	0.025	52
	Pr(+—-,-,-,+)	0.642	0.019	52
	Pr(+—-,-,+,-)	0.654	0.019	52
	Pr(+—-,-,+,+)	0.679	0.015	52
	Pr(+—-,+,-,-)	0.622	0.017	52
	Pr(+—-,+,-,+)	0.651	0.015	52
	Pr(+—-,+,+,-)	0.630	0.010	52
	Pr(+—-,+,+,+)	0.649	0.016	52
	Pr(+—+,-,-,-)	0.344	0.016	52
	Pr(+—+,-,-,+)	0.368	0.010	52
	Pr(+—+,-,+,-)	0.347	0.013	52
	Pr(+—+,-,+,+)	0.378	0.014	52
	Pr(+—+,+,-,-)	0.320	0.016	52
	Pr(+—+,+,-,+)	0.348	0.018	52
	Pr(+—+,+,+,-)	0.366	0.020	52
	Pr(+—+,+,+,+)	0.388	0.027	52
5	Pr(+—-,-,-,-,-)	0.609	0.039	51
	Pr(+—-,-,-,-,+)	0.629	0.026	52
	Pr(+—-,-,-,+,-)	0.635	0.022	52
	Pr(+—-,-,-,+,+)	0.655	0.027	52
	Pr(+—-,-,+,-,-)	0.640	0.021	52
	Pr(+—-,-,+,-,+)	0.662	0.021	52
	Pr(+—-,-,+,+,-)	0.674	0.016	52
	Pr(+—-,-,+,+,+)	0.690	0.022	52
	Pr(+—-,+,-,-,-)	0.600	0.021	52
	Pr(+—-,+,-,-,+)	0.633	0.019	52
	Pr(+—-,+,-,+,-)	0.647	0.016	52
	Pr(+—-,+,-,+,+)	0.657	0.018	52
	Pr(+—-,+,+,-,-)	0.613	0.016	52
	Pr(+—-,+,+,-,+)	0.640	0.011	52
	Pr(+—-,+,+,+,-)	0.640	0.019	52
	Pr(+—-,+,+,+,+)	0.664	0.022	52
	Pr(+—+,-,-,-,-)	0.335	0.023	52
	Pr(+—+,-,-,-,+)	0.350	0.018	52
	Pr(+—+,-,-,+,-)	0.358	0.013	52
	Pr(+—+,-,-,+,+)	0.385	0.015	52
	Pr(+—+,-,+,-,-)	0.340	0.015	52
	Pr(+—+,-,+,-,+)	0.350	0.014	52
	Pr(+—+,-,+,+,-)	0.364	0.015	52
	Pr(+—+,-,+,+,+)	0.405	0.018	52
	Pr(+—+,+,-,-,-)	0.310	0.026	52
	Pr(+—+,+,-,-,+)	0.325	0.021	52
	Pr(+—+,+,-,+,-)	0.339	0.022	52
	Pr(+—+,+,-,+,+)	0.363	0.018	52
	Pr(+—+,+,+,-,-)	0.353	0.025	52
	Pr(+—+,+,+,-,+)	0.372	0.023	52
	Pr(+—+,+,+,+,-)	0.384	0.030	52
	Pr(+—+,+,+,+,+)	0.395	0.033	50

Table A.3: Conditional probabilities of market directional movements for 3 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.499	0.003	n/a
1	Pr(+—-)	0.593	0.016	52
	Pr(+—+)	0.405	0.015	52
2	Pr(+—-,-)	0.580	0.022	52
	Pr(+—-,+)	0.603	0.013	52
	Pr(+—+,-)	0.395	0.014	52
	Pr(+—+,+)	0.419	0.019	52
3	Pr(+—-,-,-)	0.555	0.019	51
	Pr(+—-,-,+)	0.599	0.024	52
	Pr(+—-,+,-)	0.595	0.017	52
	Pr(+—-,+,+)	0.615	0.014	52
	Pr(+—+,-,-)	0.383	0.014	52
	Pr(+—+,-,+)	0.403	0.017	52
	Pr(+—+,+,-)	0.399	0.019	52
	Pr(+—+,+,+)	0.447	0.021	50
4	Pr(+—-,-,-,-)	0.540	0.020	31
	Pr(+—-,-,-,+)	0.567	0.024	50
	Pr(+—-,-,+,-)	0.590	0.025	52
	Pr(+—-,-,+,+)	0.612	0.028	52
	Pr(+—-,+,-,-)	0.578	0.020	52
	Pr(+—-,+,-,+)	0.606	0.017	52
	Pr(+—-,+,+,-)	0.609	0.015	52
	Pr(+—-,+,+,+)	0.623	0.022	52
	Pr(+—+,-,-,-)	0.373	0.021	52
	Pr(+—+,-,-,+)	0.390	0.015	52
	Pr(+—+,-,+,-)	0.391	0.017	52
	Pr(+—+,-,+,+)	0.419	0.020	52
	Pr(+—+,+,-,-)	0.389	0.020	52
	Pr(+—+,+,-,+)	0.406	0.022	52
	Pr(+—+,+,+,-)	0.436	0.023	50
	Pr(+—+,+,+,+)	0.462	0.024	31
5	Pr(+—-,-,-,-,-)	0.533	0.027	12
	Pr(+—-,-,-,-,+)	0.546	0.025	27
	Pr(+—-,-,-,+,-)	0.562	0.025	43
	Pr(+—-,-,-,+,+)	0.575	0.033	43
	Pr(+—-,-,+,-,-)	0.584	0.029	52
	Pr(+—-,-,+,-,+)	0.596	0.027	52
	Pr(+—-,-,+,+,-)	0.613	0.029	52
	Pr(+—-,-,+,+,+)	0.610	0.034	51
	Pr(+—-,+,-,-,-)	0.569	0.021	49
	Pr(+—-,+,-,-,+)	0.583	0.026	52
	Pr(+—-,+,-,+,-)	0.605	0.018	52
	Pr(+—-,+,-,+,+)	0.609	0.021	52
	Pr(+—-,+,+,-,-)	0.601	0.021	52
	Pr(+—-,+,+,-,+)	0.615	0.019	52
	Pr(+—-,+,+,+,-)	0.623	0.024	52
	Pr(+—-,+,+,+,+)	0.623	0.028	52
	Pr(+—+,-,-,-,-)	0.374	0.028	52
	Pr(+—+,-,-,-,+)	0.373	0.023	52
	Pr(+—+,-,-,+,-)	0.387	0.018	52
	Pr(+—+,-,-,+,+)	0.395	0.018	52
	Pr(+—+,-,+,-,-)	0.389	0.020	52
	Pr(+—+,-,+,-,+)	0.393	0.018	52
	Pr(+—+,-,+,+,-)	0.408	0.022	52
	Pr(+—+,-,+,+,+)	0.435	0.024	47
	Pr(+—+,+,-,-,-)	0.389	0.029	52
	Pr(+—+,+,-,-,+)	0.388	0.023	52
	Pr(+—+,+,-,+,-)	0.401	0.027	52
	Pr(+—+,+,-,+,+)	0.414	0.024	52
	Pr(+—+,+,+,-,-)	0.431	0.030	42
	Pr(+—+,+,+,-,+)	0.439	0.027	43
	Pr(+—+,+,+,+,-)	0.460	0.028	26
	Pr(+—+,+,+,+,+)	0.464	0.027	20

Table A.4: Conditional probabilities of market directional movements for 5 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.498	0.003	n/a
1	Pr(+—-)	0.524	0.024	40
	Pr(+—+)	0.472	0.023	45
2	Pr(+—-,-)	0.494	0.028	36
	Pr(+—-,+)	0.551	0.021	48
	Pr(+—+,-)	0.444	0.020	51
	Pr(+—+,+)	0.502	0.025	33
3	Pr(+—-,-,-)	0.475	0.025	34
	Pr(+—-,-,+)	0.513	0.032	30
	Pr(+—-,+,-)	0.551	0.022	48
	Pr(+—-,+,+)	0.551	0.024	45
	Pr(+—+,-,-)	0.445	0.021	49
	Pr(+—+,-,+)	0.444	0.023	47
	Pr(+—+,+,-)	0.481	0.028	29
	Pr(+—+,+,+)	0.523	0.024	29
4	Pr(+—-,-,-,-)	0.466	0.026	30
	Pr(+—-,-,-,+)	0.486	0.029	17
	Pr(+—-,-,+,-)	0.514	0.033	26
	Pr(+—-,-,+,+)	0.511	0.037	20
	Pr(+—-,+,-,-)	0.541	0.024	35
	Pr(+—-,+,-,+)	0.560	0.025	46
	Pr(+—-,+,+,-)	0.558	0.024	41
	Pr(+—-,+,+,+)	0.544	0.028	36
	Pr(+—+,-,-,-)	0.451	0.025	39
	Pr(+—+,-,-,+)	0.439	0.022	48
	Pr(+—+,-,+,-)	0.439	0.024	47
	Pr(+—+,-,+,+)	0.450	0.026	41
	Pr(+—+,+,-,-)	0.482	0.030	23
	Pr(+—+,+,-,+)	0.481	0.030	23
	Pr(+—+,+,+,-)	0.516	0.026	15
	Pr(+—+,+,+,+)	0.529	0.025	29
5	Pr(+—-,-,-,-,-)	0.459	0.027	29
	Pr(+—-,-,-,-,+)	0.473	0.033	19
	Pr(+—-,-,-,+,-)	0.483	0.033	16
	Pr(+—-,-,-,+,+)	0.489	0.034	12
	Pr(+—-,-,+,-,-)	0.503	0.036	13
	Pr(+—-,-,+,-,+)	0.523	0.037	22
	Pr(+—-,-,+,+,-)	0.517	0.043	17
	Pr(+—-,-,+,+,+)	0.505	0.037	10
	Pr(+—-,+,-,-,-)	0.530	0.026	16
	Pr(+—-,+,-,-,+)	0.550	0.031	30
	Pr(+—-,+,-,+,-)	0.564	0.029	46
	Pr(+—-,+,-,+,+)	0.554	0.026	39
	Pr(+—-,+,+,-,-)	0.556	0.031	37
	Pr(+—-,+,+,-,+)	0.560	0.030	38
	Pr(+—-,+,+,+,-)	0.550	0.028	31
	Pr(+—-,+,+,+,+)	0.539	0.034	21
	Pr(+—+,-,-,-,-)	0.454	0.031	29
	Pr(+—+,-,-,-,+)	0.448	0.029	30
	Pr(+—+,-,-,+,-)	0.431	0.026	46
	Pr(+—+,-,-,+,+)	0.448	0.030	31
	Pr(+—+,-,+,-,-)	0.439	0.031	42
	Pr(+—+,-,+,-,+)	0.438	0.023	47
	Pr(+—+,-,+,+,-)	0.441	0.033	37
	Pr(+—+,-,+,+,+)	0.461	0.025	26
	Pr(+—+,+,-,-,-)	0.487	0.036	12
	Pr(+—+,+,-,-,+)	0.477	0.031	18
	Pr(+—+,+,-,+,-)	0.475	0.034	18
	Pr(+—+,+,-,+,+)	0.489	0.033	12
	Pr(+—+,+,+,-,-)	0.516	0.029	11
	Pr(+—+,+,+,-,+)	0.517	0.029	12
	Pr(+—+,+,+,+,-)	0.526	0.023	16
	Pr(+—+,+,+,+,+)	0.532	0.031	26

Table A.5: Conditional probabilities of market directional movements for 10 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.496	0.005	n/a
1	Pr(+—-)	0.463	0.022	43
	Pr(+—+)	0.529	0.020	41
2	Pr(+—-,-)	0.451	0.021	47
	Pr(+—-,+)	0.477	0.028	33
	Pr(+—+,-)	0.515	0.023	21
	Pr(+—+,+)	0.542	0.020	46
3	Pr(+—-,-,-)	0.445	0.020	49
	Pr(+—-,-,+)	0.457	0.027	38
	Pr(+—-,+,-)	0.485	0.028	17
	Pr(+—-,+,+)	0.469	0.030	32
	Pr(+—+,-,-)	0.524	0.025	22
	Pr(+—+,-,+)	0.506	0.024	14
	Pr(+—+,+,-)	0.535	0.024	30
	Pr(+—+,+,+)	0.548	0.020	44
4	Pr(+—-,-,-,-)	0.445	0.021	44
	Pr(+—-,-,-,+)	0.446	0.027	39
	Pr(+—-,-,+,-)	0.460	0.032	29
	Pr(+—-,-,+,+)	0.454	0.030	32
	Pr(+—-,+,-,-)	0.482	0.027	12
	Pr(+—-,+,-,+)	0.488	0.035	14
	Pr(+—-,+,+,-)	0.485	0.035	13
	Pr(+—-,+,+,+)	0.455	0.032	32
	Pr(+—+,-,-,-)	0.530	0.027	21
	Pr(+—+,-,-,+)	0.517	0.033	11
	Pr(+—+,-,+,-)	0.499	0.030	10
	Pr(+—+,-,+,+)	0.512	0.029	8
	Pr(+—+,+,-,-)	0.536	0.027	22
	Pr(+—+,+,-,+)	0.535	0.026	19
	Pr(+—+,+,+,-)	0.544	0.028	30
	Pr(+—+,+,+,+)	0.552	0.020	41
5	Pr(+—-,-,-,-,-)	0.443	0.027	37
	Pr(+—-,-,-,-,+)	0.447	0.030	30
	Pr(+—-,-,-,+,-)	0.444	0.034	29
	Pr(+—-,-,-,+,+)	0.447	0.034	30
	Pr(+—-,-,+,-,-)	0.457	0.038	22
	Pr(+—-,-,+,-,+)	0.463	0.042	15
	Pr(+—-,-,+,+,-)	0.460	0.040	18
	Pr(+—-,-,+,+,+)	0.449	0.031	26
	Pr(+—-,+,-,-,-)	0.483	0.034	8
	Pr(+—-,+,-,-,+)	0.481	0.031	8
	Pr(+—-,+,-,+,-)	0.492	0.049	12
	Pr(+—-,+,-,+,+)	0.484	0.033	3
	Pr(+—-,+,+,-,-)	0.482	0.044	14
	Pr(+—-,+,+,-,+)	0.489	0.042	9
	Pr(+—-,+,+,+,-)	0.463	0.043	17
	Pr(+—-,+,+,+,+)	0.447	0.033	31
	Pr(+—+,-,-,-,-)	0.537	0.030	17
	Pr(+—+,-,-,-,+)	0.522	0.038	10
	Pr(+—+,-,-,+,-)	0.509	0.039	3
	Pr(+—+,-,-,+,+)	0.523	0.036	9
	Pr(+—+,-,+,-,-)	0.504	0.035	5
	Pr(+—+,-,+,-,+)	0.494	0.043	10
	Pr(+—+,-,+,+,-)	0.511	0.039	6
	Pr(+—+,-,+,+,+)	0.513	0.032	4
	Pr(+—+,+,-,-,-)	0.538	0.032	17
	Pr(+—+,+,-,-,+)	0.533	0.037	15
	Pr(+—+,+,-,+,-)	0.532	0.033	11
	Pr(+—+,+,-,+,+)	0.537	0.031	16
	Pr(+—+,+,+,-,-)	0.539	0.030	17
	Pr(+—+,+,+,-,+)	0.549	0.036	22
	Pr(+—+,+,+,+,-)	0.550	0.028	25
	Pr(+—+,+,+,+,+)	0.553	0.027	32

Table A.6: Conditional probabilities of market directional movements for 20 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.495	0.007	n/a
1	Pr(+—-)	0.431	0.015	52
	Pr(+—+)	0.561	0.013	52
2	Pr(+—-,-)	0.435	0.013	52
	Pr(+—-,+)	0.425	0.024	50
	Pr(+—+,-)	0.569	0.021	49
	Pr(+—+,+)	0.554	0.014	51
3	Pr(+—-,-,-)	0.436	0.019	50
	Pr(+—-,-,+)	0.434	0.019	48
	Pr(+—-,+,-)	0.437	0.032	37
	Pr(+—-,+,+)	0.417	0.026	49
	Pr(+—+,-,-)	0.572	0.023	48
	Pr(+—+,-,+)	0.565	0.028	39
	Pr(+—+,+,-)	0.552	0.024	33
	Pr(+—+,+,+)	0.556	0.018	46
4	Pr(+—-,-,-,-)	0.437	0.024	39
	Pr(+—-,-,-,+)	0.435	0.029	38
	Pr(+—-,-,+,-)	0.434	0.027	36
	Pr(+—-,-,+,+)	0.432	0.025	42
	Pr(+—-,+,-,-)	0.439	0.037	28
	Pr(+—-,+,-,+)	0.433	0.042	24
	Pr(+—-,+,+,-)	0.429	0.035	35
	Pr(+—-,+,+,+)	0.407	0.03	48
	Pr(+—+,-,-,-)	0.578	0.029	44
	Pr(+—+,-,-,+)	0.563	0.035	30
	Pr(+—+,-,+,-)	0.572	0.038	34
	Pr(+—+,-,+,+)	0.56	0.033	28
	Pr(+—+,+,-,-)	0.551	0.028	28
	Pr(+—+,+,-,+)	0.552	0.031	19
	Pr(+—+,+,+,-)	0.558	0.024	30
	Pr(+—+,+,+,+)	0.555	0.023	33
5	Pr(+—-,-,-,-,-)	0.439	0.033	29
	Pr(+—-,-,-,-,+)	0.434	0.031	26
	Pr(+—-,-,-,+,-)	0.431	0.039	22
	Pr(+—-,-,-,+,+)	0.439	0.038	25
	Pr(+—-,-,+,-,-)	0.431	0.037	22
	Pr(+—-,-,+,-,+)	0.438	0.043	17
	Pr(+—-,-,+,+,-)	0.434	0.042	23
	Pr(+—-,-,+,+,+)	0.431	0.03	27
	Pr(+—-,+,-,-,-)	0.442	0.042	15
	Pr(+—-,+,-,-,+)	0.435	0.054	21
	Pr(+—-,+,-,+,-)	0.436	0.053	12
	Pr(+—-,+,-,+,+)	0.431	0.049	19
	Pr(+—-,+,+,-,-)	0.434	0.043	20
	Pr(+—-,+,+,-,+)	0.422	0.044	24
	Pr(+—-,+,+,+,-)	0.416	0.042	34
	Pr(+—-,+,+,+,+)	0.4	0.038	41
	Pr(+—+,-,-,-,-)	0.577	0.034	32
	Pr(+—+,-,-,-,+)	0.58	0.035	29
	Pr(+—+,-,-,+,-)	0.567	0.049	19
	Pr(+—+,-,-,+,+)	0.56	0.042	19
	Pr(+—+,-,+,-,-)	0.576	0.046	21
	Pr(+—+,-,+,-,+)	0.566	0.062	17
	Pr(+—+,-,+,+,-)	0.566	0.048	20
	Pr(+—+,-,+,+,+)	0.554	0.041	16
	Pr(+—+,+,-,-,-)	0.552	0.035	22
	Pr(+—+,+,-,-,+)	0.551	0.039	12
	Pr(+—+,+,-,+,-)	0.552	0.052	12
	Pr(+—+,+,-,+,+)	0.552	0.047	11
	Pr(+—+,+,+,-,-)	0.562	0.033	23
	Pr(+—+,+,+,-,+)	0.553	0.031	9
	Pr(+—+,+,+,+,-)	0.56	0.037	23
	Pr(+—+,+,+,+,+)	0.552	0.026	20

Table A.7: Conditional probabilities of market directional movements for 30 second timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.495	0.009	n/a
1	Pr(+—-)	0.422	0.013	52
	Pr(+—+)	0.570	0.013	52
2	Pr(+—-,-)	0.434	0.017	51
	Pr(+—-,+)	0.407	0.023	52
	Pr(+—+,-)	0.587	0.019	52
	Pr(+—+,+)	0.557	0.015	48
3	Pr(+—-,-,-)	0.437	0.021	45
	Pr(+—-,-,+)	0.429	0.024	42
	Pr(+—-,+,-)	0.414	0.031	43
	Pr(+—-,+,+)	0.402	0.026	50
	Pr(+—+,-,-)	0.587	0.024	49
	Pr(+—+,-,+)	0.586	0.029	45
	Pr(+—+,+,-)	0.558	0.023	40
	Pr(+—+,+,+)	0.556	0.021	42
4	Pr(+—-,-,-,-)	0.440	0.029	30
	Pr(+—-,-,-,+)	0.434	0.031	33
	Pr(+—-,-,+,-)	0.433	0.036	24
	Pr(+—-,-,+,+)	0.426	0.031	34
	Pr(+—-,+,-,-)	0.414	0.036	36
	Pr(+—-,+,-,+)	0.413	0.045	28
	Pr(+—-,+,+,-)	0.409	0.041	38
	Pr(+—-,+,+,+)	0.395	0.029	49
	Pr(+—+,-,-,-)	0.593	0.034	42
	Pr(+—+,-,-,+)	0.580	0.029	34
	Pr(+—+,-,+,-)	0.594	0.045	32
	Pr(+—+,-,+,+)	0.581	0.035	34
	Pr(+—+,+,-,-)	0.561	0.026	27
	Pr(+—+,+,-,+)	0.554	0.035	16
	Pr(+—+,+,+,-)	0.560	0.031	28
	Pr(+—+,+,+,+)	0.553	0.028	26
5	Pr(+—-,-,-,-,-)	0.441	0.034	19
	Pr(+—-,-,-,-,+)	0.438	0.043	18
	Pr(+—-,-,-,+,-)	0.437	0.047	13
	Pr(+—-,-,-,+,+)	0.431	0.041	24
	Pr(+—-,-,+,-,-)	0.438	0.052	18
	Pr(+—-,-,+,-,+)	0.429	0.054	10
	Pr(+—-,-,+,+,-)	0.426	0.039	21
	Pr(+—-,-,+,+,+)	0.426	0.049	24
	Pr(+—-,+,-,-,-)	0.427	0.050	17
	Pr(+—-,+,-,-,+)	0.397	0.050	21
	Pr(+—-,+,-,+,-)	0.414	0.067	14
	Pr(+—-,+,-,+,+)	0.414	0.060	13
	Pr(+—-,+,+,-,-)	0.423	0.054	21
	Pr(+—-,+,+,-,+)	0.389	0.062	31
	Pr(+—-,+,+,+,-)	0.406	0.047	27
	Pr(+—-,+,+,+,+)	0.387	0.040	41
	Pr(+—+,-,-,-,-)	0.599	0.039	34
	Pr(+—+,-,-,-,+)	0.585	0.052	24
	Pr(+—+,-,-,+,-)	0.589	0.044	20
	Pr(+—+,-,-,+,+)	0.574	0.041	23
	Pr(+—+,-,+,-,-)	0.607	0.060	29
	Pr(+—+,-,+,-,+)	0.575	0.059	12
	Pr(+—+,-,+,+,-)	0.583	0.058	21
	Pr(+—+,-,+,+,+)	0.579	0.045	20
	Pr(+—+,+,-,-,-)	0.557	0.035	13
	Pr(+—+,+,-,-,+)	0.565	0.045	12
	Pr(+—+,+,-,+,-)	0.558	0.055	12
	Pr(+—+,+,-,+,+)	0.551	0.049	11
	Pr(+—+,+,+,-,-)	0.555	0.037	11
	Pr(+—+,+,+,-,+)	0.567	0.043	13
	Pr(+—+,+,+,+,-)	0.562	0.046	17
	Pr(+—+,+,+,+,+)	0.546	0.040	15

Table A.8: Conditional probabilities of market directional movements for 1 minute timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.494	0.015	n/a
1	Pr(+—-)	0.422	0.019	51
	Pr(+—+)	0.568	0.017	50
2	Pr(+—-,-)	0.444	0.022	38
	Pr(+—-,+)	0.391	0.030	51
	Pr(+—+,-)	0.599	0.027	52
	Pr(+—+,+)	0.544	0.022	30
3	Pr(+—-,-,-)	0.449	0.028	21
	Pr(+—-,-,+)	0.438	0.030	28
	Pr(+—-,+,-)	0.398	0.042	36
	Pr(+—-,+,+)	0.386	0.037	46
	Pr(+—+,-,-)	0.608	0.031	48
	Pr(+—+,-,+)	0.584	0.039	27
	Pr(+—+,+,-)	0.547	0.037	18
	Pr(+—+,+,+)	0.542	0.027	16
4	Pr(+—-,-,-,-)	0.459	0.037	15
	Pr(+—-,-,-,+)	0.435	0.043	16
	Pr(+—-,-,+,-)	0.435	0.054	13
	Pr(+—-,-,+,+)	0.440	0.039	17
	Pr(+—-,+,-,-)	0.407	0.050	28
	Pr(+—-,+,-,+)	0.387	0.069	24
	Pr(+—-,+,+,-)	0.404	0.049	31
	Pr(+—-,+,+,+)	0.371	0.041	44
	Pr(+—+,-,-,-)	0.614	0.038	41
	Pr(+—+,-,-,+)	0.600	0.047	26
	Pr(+—+,-,+,-)	0.604	0.071	20
	Pr(+—+,-,+,+)	0.570	0.049	14
	Pr(+—+,+,-,-)	0.544	0.044	13
	Pr(+—+,+,-,+)	0.550	0.057	11
	Pr(+—+,+,+,-)	0.563	0.041	13
	Pr(+—+,+,+,+)	0.524	0.039	6
5	Pr(+—-,-,-,-,-)	0.465	0.050	6
	Pr(+—-,-,-,-,+)	0.452	0.063	11
	Pr(+—-,-,-,+,-)	0.443	0.062	6
	Pr(+—-,-,-,+,+)	0.430	0.057	17
	Pr(+—-,-,+,-,-)	0.449	0.064	6
	Pr(+—-,-,+,-,+)	0.418	0.094	0
	Pr(+—-,-,+,+,-)	0.450	0.064	9
	Pr(+—-,-,+,+,+)	0.433	0.057	11
	Pr(+—-,+,-,-,-)	0.415	0.067	14
	Pr(+—-,+,-,-,+)	0.397	0.067	11
	Pr(+—-,+,-,+,-)	0.392	0.109	10
	Pr(+—-,+,-,+,+)	0.383	0.084	21
	Pr(+—-,+,+,-,-)	0.415	0.070	19
	Pr(+—-,+,+,-,+)	0.387	0.073	17
	Pr(+—-,+,+,+,-)	0.387	0.070	23
	Pr(+—-,+,+,+,+)	0.359	0.052	39
	Pr(+—+,-,-,-,-)	0.629	0.054	31
	Pr(+—+,-,-,-,+)	0.596	0.050	14
	Pr(+—+,-,-,+,-)	0.612	0.085	17
	Pr(+—+,-,-,+,+)	0.591	0.055	16
	Pr(+—+,-,+,-,-)	0.618	0.090	20
	Pr(+—+,-,+,-,+)	0.585	0.115	9
	Pr(+—+,-,+,+,-)	0.585	0.076	11
	Pr(+—+,-,+,+,+)	0.558	0.071	8
	Pr(+—+,+,-,-,-)	0.544	0.059	10
	Pr(+—+,+,-,-,+)	0.546	0.062	5
	Pr(+—+,+,-,+,-)	0.575	0.098	13
	Pr(+—+,+,-,+,+)	0.533	0.068	4
	Pr(+—+,+,+,-,-)	0.566	0.051	12
	Pr(+—+,+,+,-,+)	0.560	0.073	9
	Pr(+—+,+,+,+,-)	0.538	0.067	8
	Pr(+—+,+,+,+,+)	0.513	0.055	2

Table A.9: Conditional probabilities of market directional movements for 5 minute timespan

Depth	Event	Mean	SD	# weeks that are stat. sig. from Pr(+)
0	Pr(+)	0.493	0.030	n/a
1	Pr(+—-)	0.443	0.039	21
	Pr(+—+)	0.546	0.030	9
2	Pr(+—-,-)	0.471	0.054	6
	Pr(+—-,+)	0.408	0.054	19
	Pr(+—+,-)	0.591	0.051	19
	Pr(+—+,+)	0.508	0.044	2
3	Pr(+—-,-,-)	0.493	0.072	3
	Pr(+—-,-,+)	0.446	0.085	12
	Pr(+—-,+,-)	0.429	0.086	10
	Pr(+—-,+,+)	0.392	0.073	16
	Pr(+—+,-,-)	0.581	0.069	10
	Pr(+—+,-,+)	0.609	0.092	12
	Pr(+—+,+,-)	0.523	0.077	4
	Pr(+—+,+,+)	0.496	0.056	1
4	Pr(+—-,-,-,-)	0.515	0.091	2
	Pr(+—-,-,-,+)	0.468	0.102	4
	Pr(+—-,-,+,-)	0.479	0.125	4
	Pr(+—-,-,+,+)	0.425	0.104	10
	Pr(+—-,+,-,-)	0.450	0.119	6
	Pr(+—-,+,-,+)	0.392	0.142	11
	Pr(+—-,+,+,-)	0.395	0.123	10
	Pr(+—-,+,+,+)	0.384	0.096	15
	Pr(+—+,-,-,-)	0.582	0.103	9
	Pr(+—+,-,-,+)	0.577	0.100	7
	Pr(+—+,-,+,-)	0.637	0.139	13
	Pr(+—+,-,+,+)	0.583	0.126	8
	Pr(+—+,+,-,-)	0.547	0.105	4
	Pr(+—+,+,-,+)	0.490	0.104	2
	Pr(+—+,+,+,-)	0.538	0.085	2
	Pr(+—+,+,+,+)	0.455	0.081	1
5	Pr(+—-,-,-,-,-)	0.535	0.134	2
	Pr(+—-,-,-,-,+)	0.498	0.118	3
	Pr(+—-,-,-,+,-)	0.474	0.211	13
	Pr(+—-,-,-,+,+)	0.466	0.130	5
	Pr(+—-,-,+,-,-)	0.506	0.150	5
	Pr(+—-,-,+,-,+)	0.427	0.204	9
	Pr(+—-,-,+,+,-)	0.414	0.136	5
	Pr(+—-,-,+,+,+)	0.432	0.139	10
	Pr(+—-,+,-,-,-)	0.459	0.154	7
	Pr(+—-,+,-,-,+)	0.453	0.220	11
	Pr(+—-,+,-,+,-)	0.429	0.258	12
	Pr(+—-,+,-,+,+)	0.370	0.172	9
	Pr(+—-,+,+,-,-)	0.406	0.148	10
	Pr(+—-,+,+,-,+)	0.380	0.166	11
	Pr(+—-,+,+,+,-)	0.393	0.147	9
	Pr(+—-,+,+,+,+)	0.371	0.143	13
	Pr(+—+,-,-,-,-)	0.586	0.142	7
	Pr(+—+,-,-,-,+)	0.575	0.149	7
	Pr(+—+,-,-,+,-)	0.616	0.166	11
	Pr(+—+,-,-,+,+)	0.545	0.142	4
	Pr(+—+,-,+,-,-)	0.621	0.179	13
	Pr(+—+,-,+,-,+)	0.636	0.288	22
	Pr(+—+,-,+,+,-)	0.596	0.210	14
	Pr(+—+,-,+,+,+)	0.580	0.149	5
	Pr(+—+,+,-,-,-)	0.545	0.146	5
	Pr(+—+,+,-,-,+)	0.547	0.159	8
	Pr(+—+,+,-,+,-)	0.502	0.141	1
	Pr(+—+,+,-,+,+)	0.483	0.159	6
	Pr(+—+,+,+,-,-)	0.536	0.122	2
	Pr(+—+,+,+,-,+)	0.538	0.160	4
	Pr(+—+,+,+,+,-)	0.498	0.105	0
	Pr(+—+,+,+,+,+)	0.396	0.150	7

APPENDIX B
DESCRIPTION OF DATASET

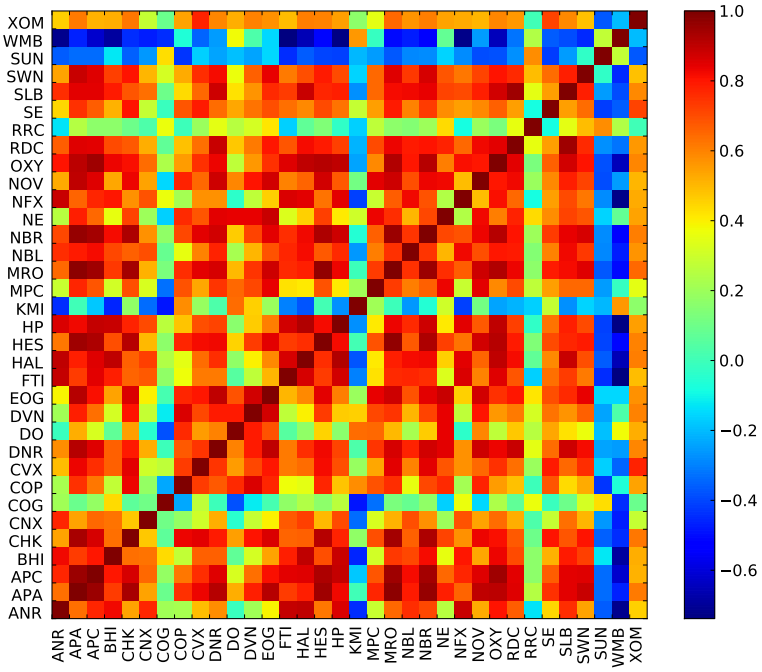
The dataset is comprised of 34 stocks from the energy sector listed on the New York Stock Exchange (Table B.1). The available data is from January 3, 2013 to July 31, 2013 excluding the New York Stock Exchange holidays[1]. Overall these 34 stocks display a strong level of intraday correlation which can be seen in Figure B.1a – most positively correlated with several stocks displaying negative correlation. For a comparison, this is compared against 34 random stocks in Figure B.1b – most not displaying much correlation at all.

The stock intraday price data (1-minute intervals) is displayed in Figures B.2, B.3, and B.4.

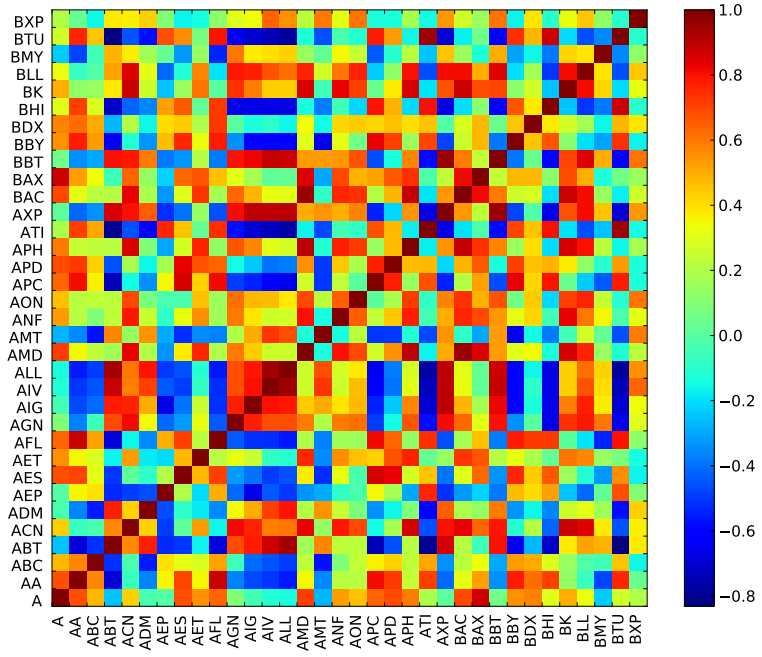
The experiment's objective (unless otherwise stated in the text) is to predict the stock price direction 1 minute into the future ($t + 1$, where t is time). We divide this into a three class problem: *move up*, *down*, or *no change* (insignificant change). A *move up* is defined as the stock moving up 0.05% from the last price, a *move down* is defined as a move down of 0.05% from the last price, and a *no change* in price is defined as an insignificant move between a move up of 0.05% and a move down of 0.05%. The proportion of each class for each stock over the first seven months of 2012 can be seen in Figures B.5, B.6, and B.7.

---

[1]During our seven month period these holidays included: New Year's Day (January 2), Martin Luther King Day (January 16), Presidents Day (February 20), Good Friday (April 6), Memorial Day (May 28), and Independence Day (July 4).

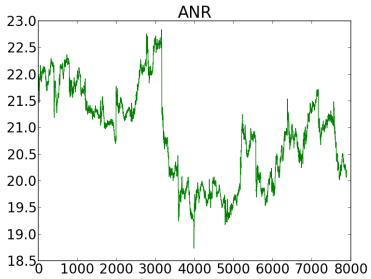


(a) Correlation among our 34 sector stocks

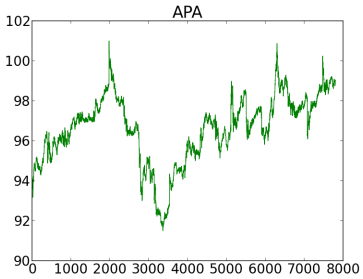


(b) Correlation among 34 random stocks

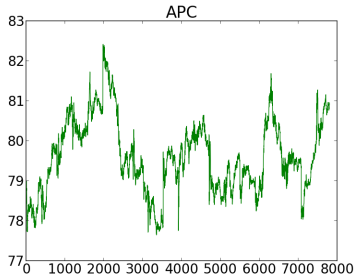
Figure B.1: Intraday Spearman Rank correlation over 7 months for our sector and (as a comparison) random stocks



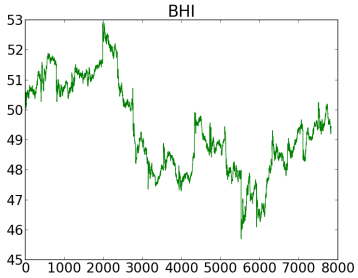
(a) Alpha Natural Resources Inc.



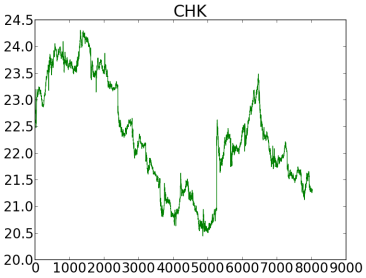
(b) Apache Corp.



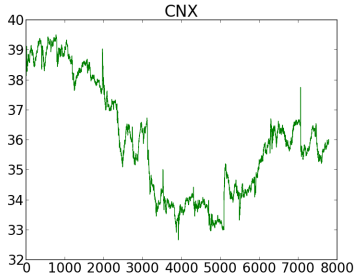
(c) Anadarko Petroleum Corp.



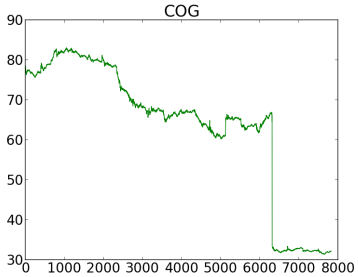
(d) Baker Hughes Inc.



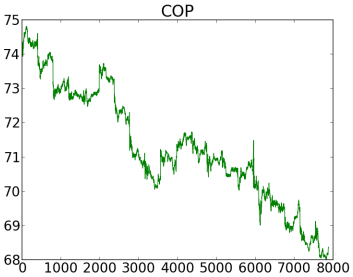
(e) Chesapeake Energy



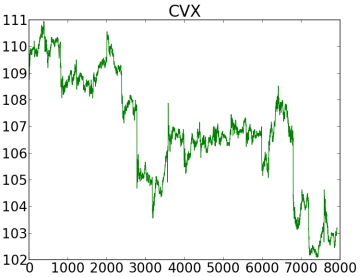
(f) CONSOL Energy Inc.



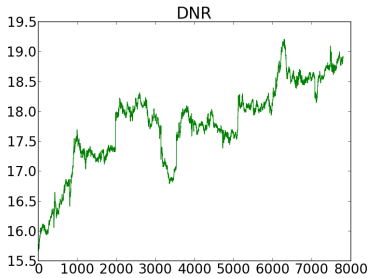
(g) Cabot Oil & Gas



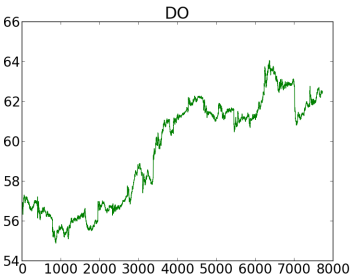
(h) ConocoPhillips



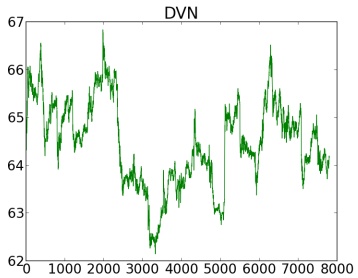
(i) Chevron Corp.



(j) Denbury Resources Inc.

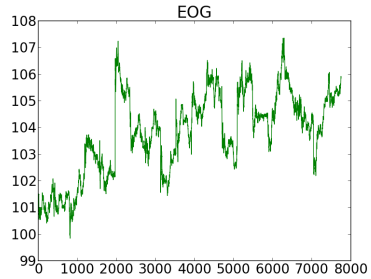


(k) Diamond Offshore Drilling

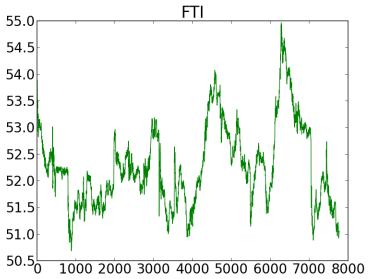


(l) Devon Energy Corp.

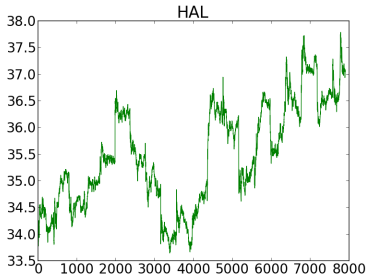
Figure B.2: January through July stock data (symbols: ANR – DVN)



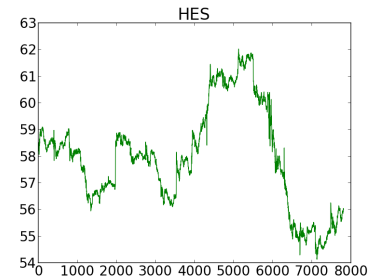
(a) EOG Resources



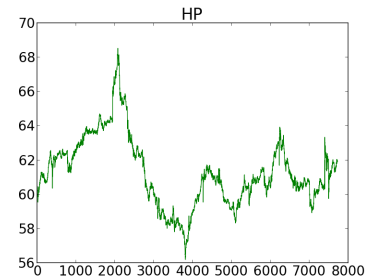
(b) FMC Technologies Inc.



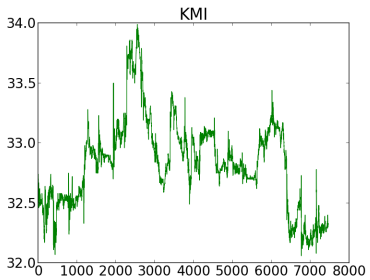
(c) Halliburton Co.



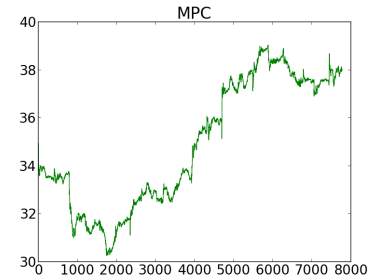
(d) Hess Corp.



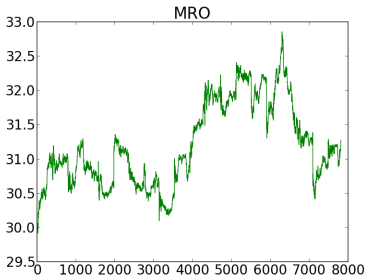
(e) Helmerich & Payne



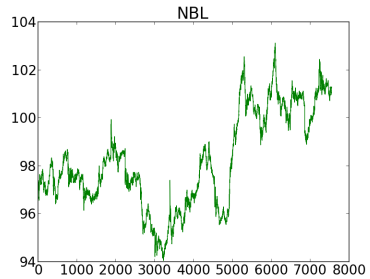
(f) Kinder Morgan



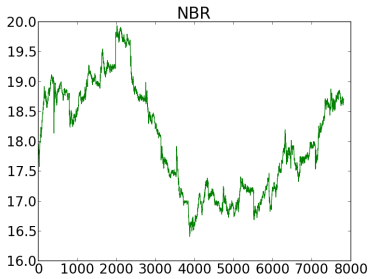
(g) Marathon Petroleum



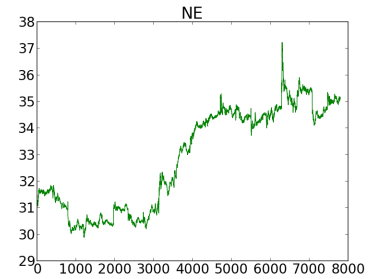
(h) Marathon Oil Corp.



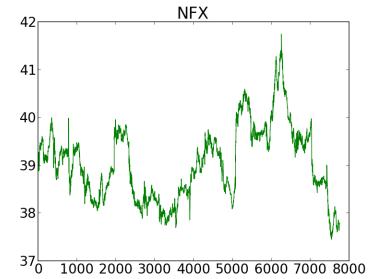
(i) Noble Energy Inc.



(j) Nabors Industries Ltd

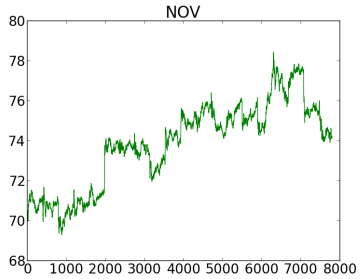


(k) Noble Corp.

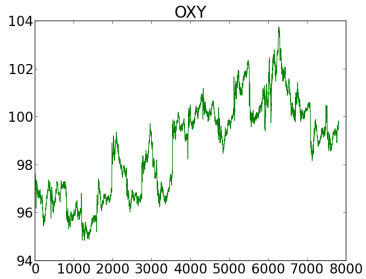


(l) Newfield Exploration Co.

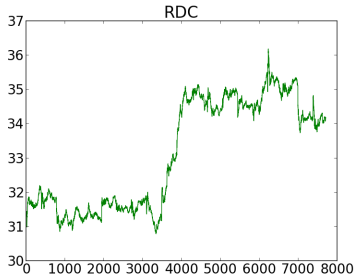
Figure B.3: January through July stock data (symbols: EOG – NFX)



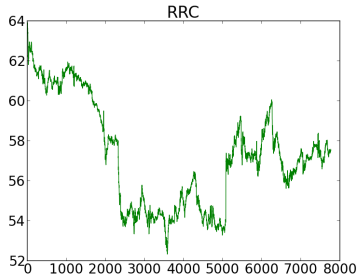
(a) National Oilwell Varco Inc.



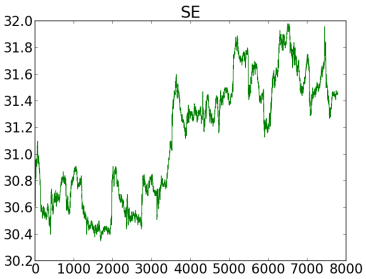
(b) Occidental Petroleum



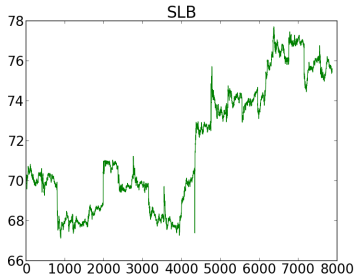
(c) Rowan Cos



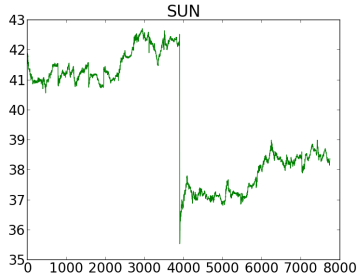
(d) Range Resources Corp.



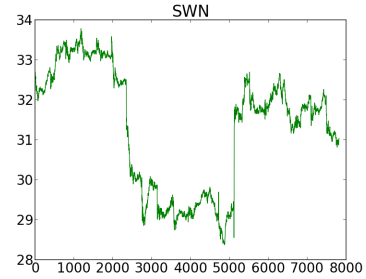
(e) Spectra Energy Corp.



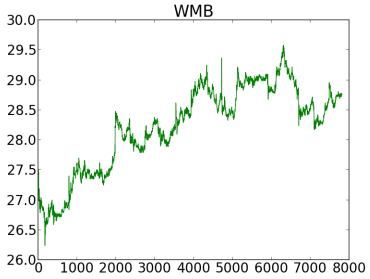
(f) Schlumberger Ltd



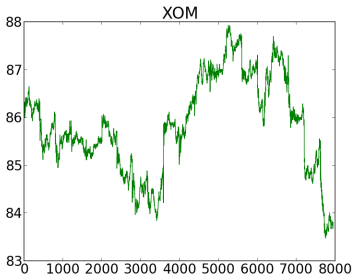
(g) Sunoco Inc.



(h) Southwestern Energy

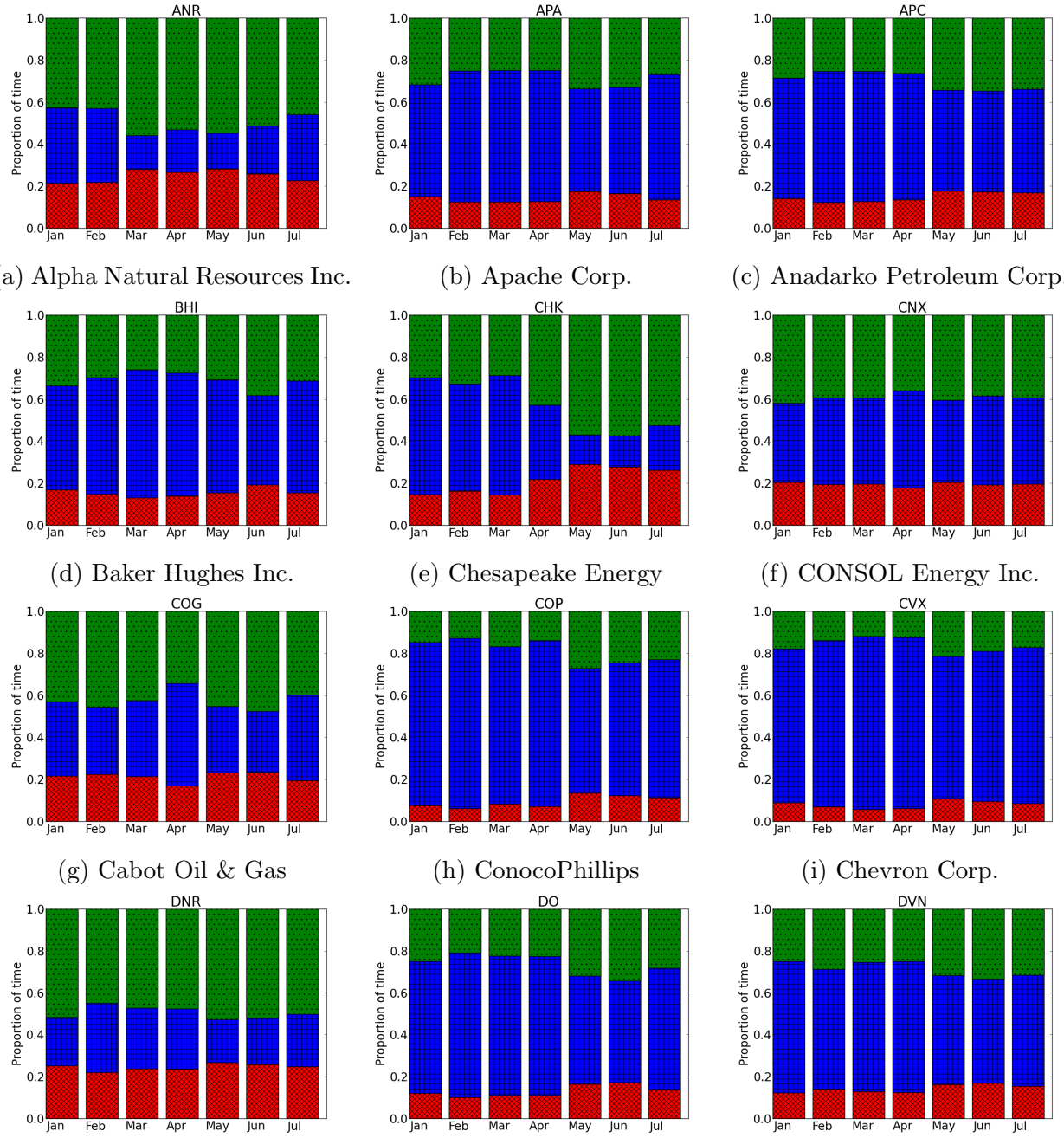


(i) Williams Corp.



(j) Exxon Mobil Corp.

Figure B.4: January through July stock data (symbols: NOV – XOM)



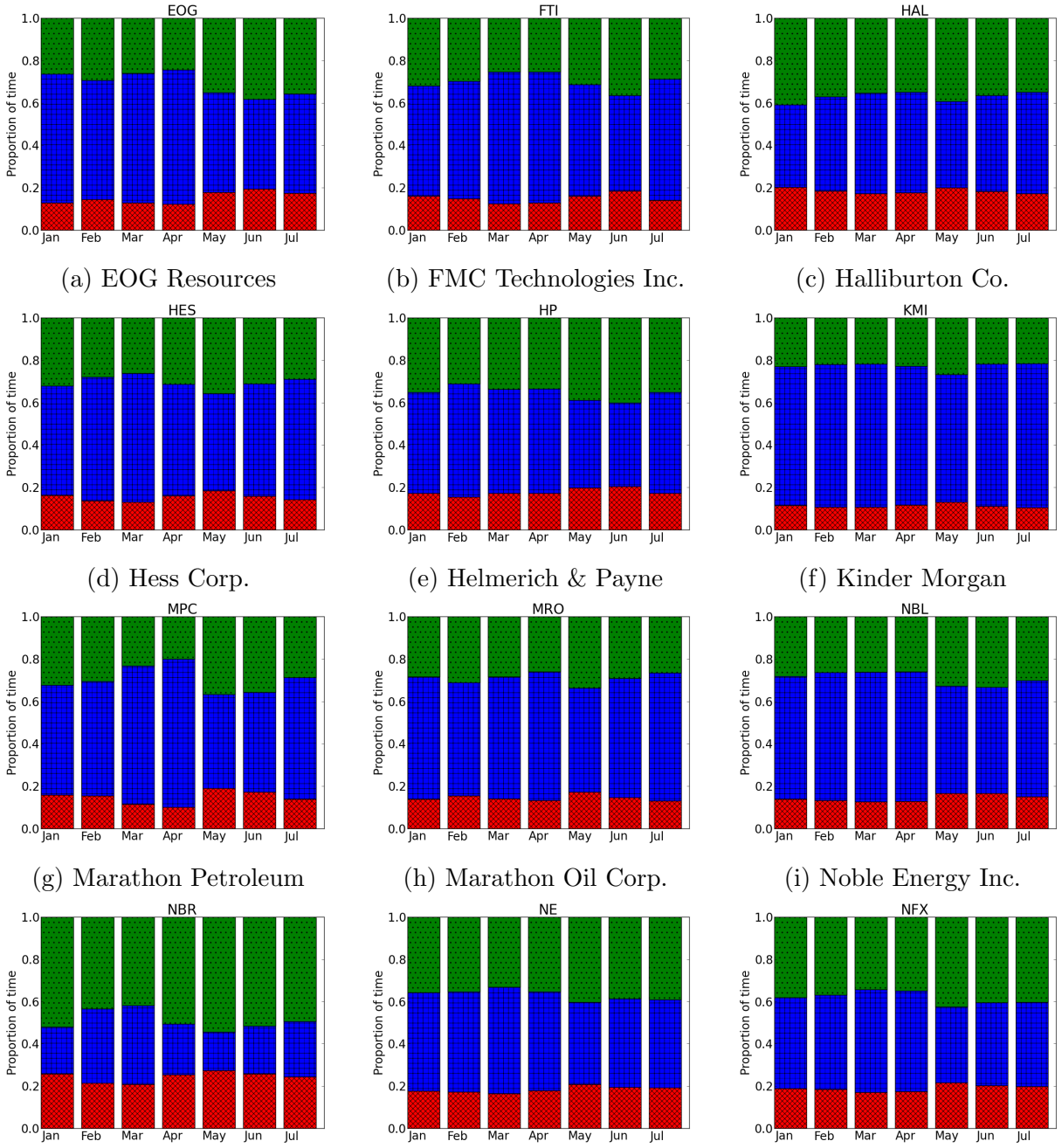
(a) Alpha Natural Resources Inc. (b) Apache Corp. (c) Anadarko Petroleum Corp.

(d) Baker Hughes Inc. (e) Chesapeake Energy (f) CONSOL Energy Inc.

(g) Cabot Oil & Gas (h) ConocoPhillips (i) Chevron Corp.

(j) Denbury Resources Inc. (k) Diamond Offshore Drilling (l) Devon Energy Corp.

Figure B.5: Proportion of time defined as "move down" (red), "no change" (blue), or "move up" (green) over the course of 7 months (symbols: ANR – DVN)



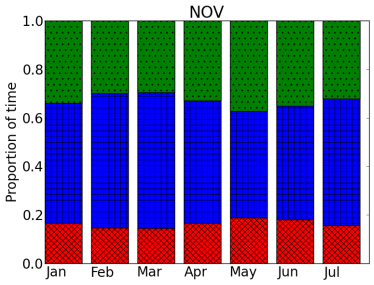
(a) EOG Resources (b) FMC Technologies Inc. (c) Halliburton Co.

(d) Hess Corp. (e) Helmerich & Payne (f) Kinder Morgan

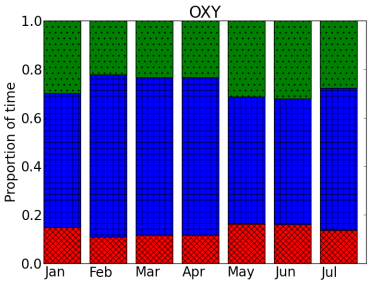
(g) Marathon Petroleum (h) Marathon Oil Corp. (i) Noble Energy Inc.

(j) Nabors Industries Ltd (k) Noble Corp. (l) Newfield Exploration Co.

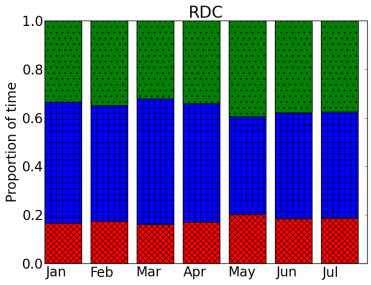
Figure B.6: Proportion of time defined as "move down" (red), "no change" (blue), or "move up" (green) over the course of 7 months (symbols: EOG – NFX)



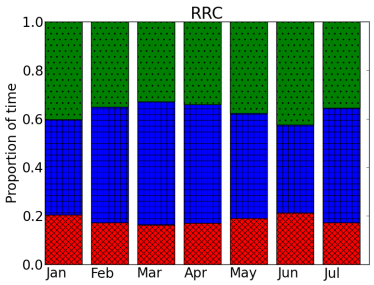
(a) National Oilwell Varco Inc.



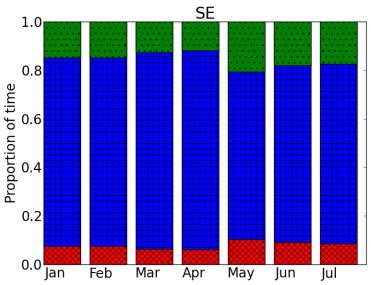
(b) Occidental Petroleum



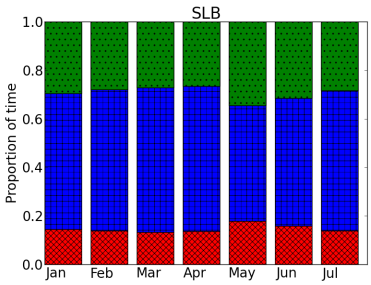
(c) Rowan Cos



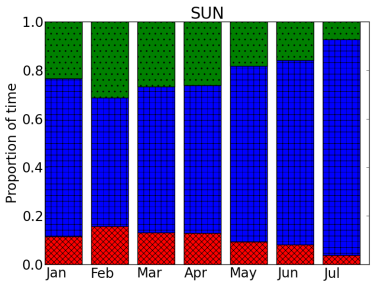
(d) Range Resources Corp.



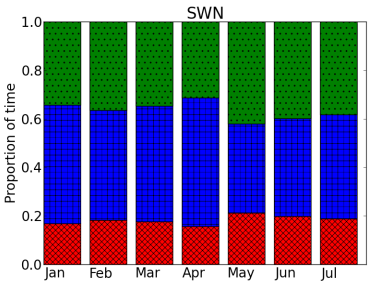
(e) Spectra Energy Corp.



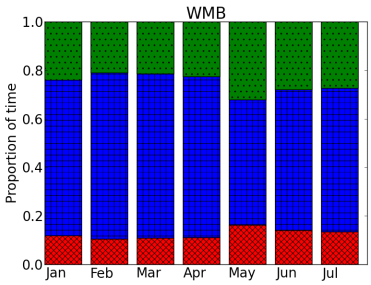
(f) Schlumberger Ltd



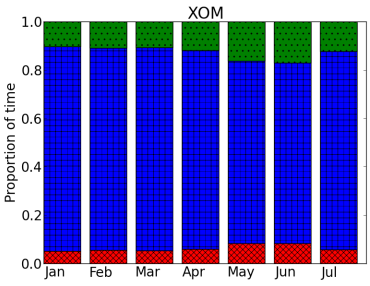
(g) Sunoco Inc.



(h) Southwestern Energy



(i) Williams Corp.



(j) Exxon Mobil Corp.

Figure B.7: Proportion of time defined as "move down" (red), "no change" (blue), or "move up" (green) over the course of 7 months (symbols: NOV – XOM)

Table B.1: List of stocks used in the experiments

Symbol	Company Name	Symbol	Company Name
ANR	Alpha Natural Resources Inc.	KMI	Kinder Morgan
APA	Apache Corp.	MPC	Marathon Petroleum
APC	Anadarko Petroleum Corp.	MRO	Marathon Oil Corp.
BHI	Baker Hughes Inc.	NBL	Noble Energy Inc.
CHK	Chesapeake Energy	NBR	Nabors Industries Ltd
CNX	CONSOL Energy Inc.	NE	Noble Corp.
COG	Cabot Oil & Gas	NFX	Newfield Exploration Co.
COP	ConocoPhillips	NOV	National Oilwell Varco Inc.
CVX	Chevron Corp.	OXY	Occidental Petroleum
DNR	Denbury Resources Inc.	RDC	Rowan Cos
DO	Diamond Offshore Drilling	RRC	Range Resources Corp.
DVN	Devon Energy Corp.	SE	Spectra Energy Corp.
EOG	EOG Resources	SLB	Schlumberger Ltd
FTI	FMC Technologies Inc.	SUN	Sunoco Inc.
HAL	Halliburton Co.	SWN	Southwestern Energy
HES	Hess Corp.	WMB	Williams Corp.
HP	Helmerich & Payne	XOM	Exxon Mobil Corp.

APPENDIX C
CREATION OF ATTRIBUTES WITH TECHNICAL ANALYSIS INDICATORS

This appendix explains the twenty groups of technical analysis indicators used in this thesis which, with parameters amounts to a total of 276 attributes. These twenty groups are: lines, rates of changes, moving averages, moving variance ratios, moving average ratios, Aroon indicators, Bollinger bands, commodity channel index, Chaiken volatility, close location value, Chaikin money flow, Chande momentum oscillators, MACD, trend detection index, triple smoothed exponential oscillator, volatility, Williams %, relative strength index, stochastic, and lag correlations. This appendix will explain each. In addition to the citations included in this appendix, the author would like to thank the authors and hosts behind the following websites [76, 146, 208] and R packages [196, 220] – all of which facilitated the understanding of the technical analysis indicators.

The stock data used to create the indicators/attributes contains an open, high, low, close and share volume during each one minute time interval. A description of the data is below and also visualized in Figure C.1. The calculations for the indicators follows, along with the parameter settings used to create the attributes used in this

thesis:

$$\begin{aligned} \text{open}_t &= \text{the opening/beginning price} \\ \text{close}_t &= \text{the closing/ending price} \\ \text{high}_t &= \text{the highest price reached during the interval} \\ \text{low}_t &= \text{the lowest price reached during the interval} \\ \text{volume}_t &= \text{the total number of shares traded during the interval} \end{aligned}$$

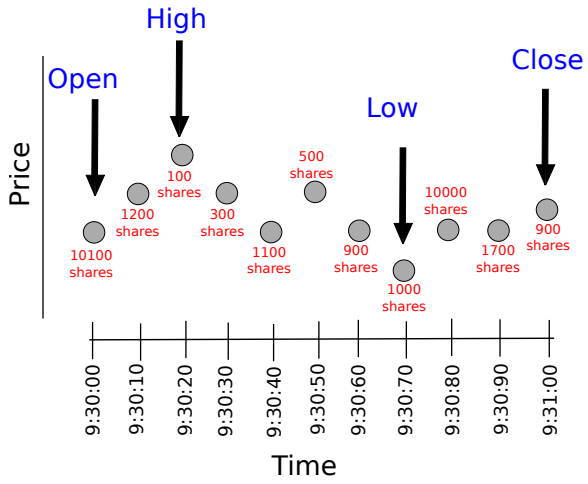


Figure C.1: Demonstrating open, high, low, close and share volumes during an interval of 1 minute

C.1 Rate of change

The rate of change is computed as the percentage change of the current closing price at time t to the closing price from $t-n$, where $n = 1, 2, 3, 4, 5, 10, 20$ in the paper.

The formula follows:

$$\text{ROC}(n) = \frac{close_t - close_{t-n}}{close_{t-n}} \times 100$$

C.2 Moving averages

C.2.1 Simple moving average % change

The simple moving average uses a moving window of n past prices in its calculation, where $n = 3, 4, 5, 10, 20$ in the paper (see Figure C.2). Some technical analysts view increases of the closing price above the moving average as bullish (positive) and closing prices below the moving average as bearish (negative). As an output, we calculate the percentage change of the closing price from the SMA. The formula follows:

$$\begin{aligned} \text{SMA}(n) &= \frac{1}{n}\sum_{i=1}^{n} \text{close}_{t-i} \\ \text{SMA \% change}(n) &= \frac{\text{close}_t - \text{SMA}(n)_t}{\text{SMA}(n)_t} \times 100 \end{aligned}$$

Analysis of moving averages is popular among technical analysts. As the number of n instances increase, the moving average becomes less responsive to short-term fluctuations. Many practitioners use multiple moving averages in their analysis.

C.2.2 Exponential moving average % change

The exponential moving average percentage change is the change of the current price t compared against the exponential moving average covering the past n prices, where $n = 3, 4, 5, 10, 20$ in the paper. The difference between the exponential and the simple is that recent observations receive more weight.

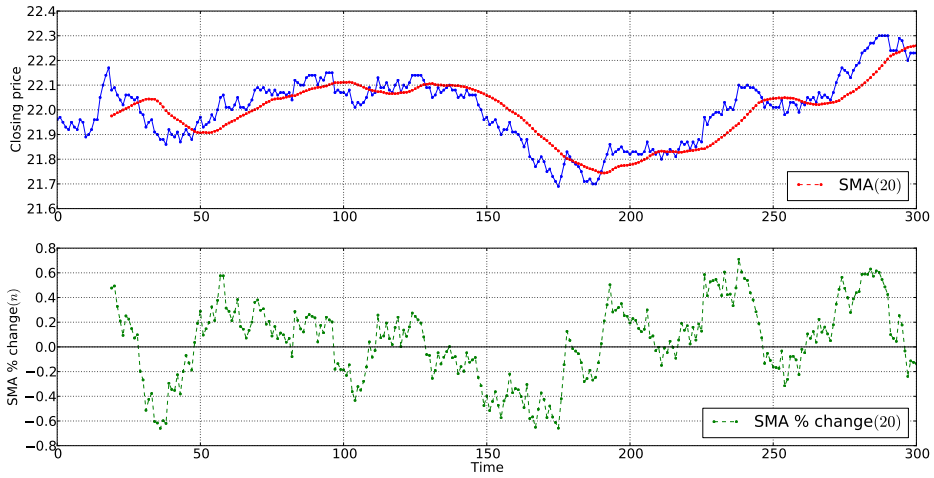


Figure C.2: Demonstrating price with SMA % change(20)

C.2.3 Exponential moving average volume weighted % change

The exponential moving average volume weighted change is the percentage change of the current price t compared against the exponential moving average of the past n prices, where $n = 3, 4, 5, 10, 20$. The dampening factor is weighted though by the volume traded instead of recency of trade price.

C.3 Regression

A simple sum of least squares calculation is used as an indicator, with the closing price used to fit a straight line through the set of n points, where $n = 2, 5, 10, 20$. The output is the percentage difference between the estimator and the actual price. Additionally, the coefficient of determination R^2 is output to describe how well the regression line fits the set of past n prices. See Figure C.3.

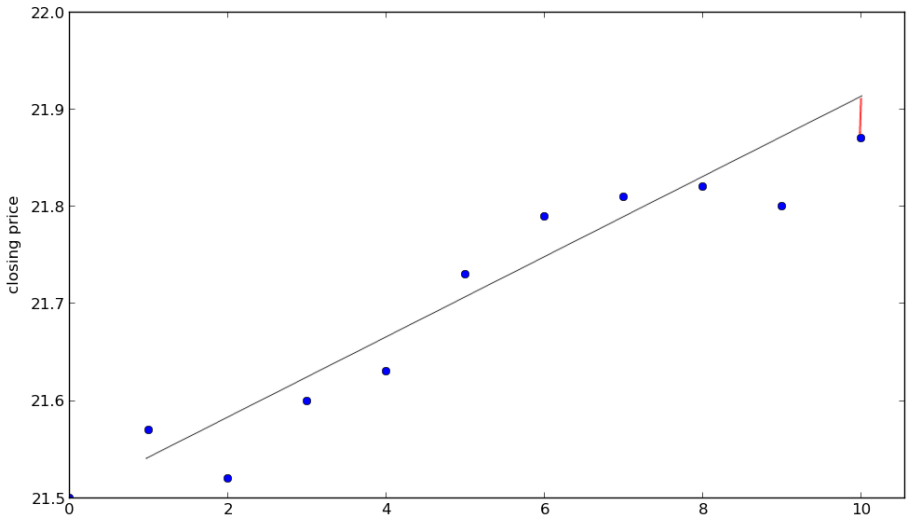


Figure C.3: Price with regression(10). Red line on last price at time t represents the distance (percentage change) between the current price and the predicted.

C.4 Moving average of variance ratio

The moving average of variance is the variance of the *close* over the past n periods. A ratio is then computed by dividing the moving average of variance of size n_1 by n_2. In our paper we use n_1 and n_2 as 5 and 10 respectively; we also use 5 and 20 respectively. The formula follows (see also Figure C.4):

$$
\begin{aligned}
\text{SMA}(n_1) &= \frac{1}{n_1}\sum_{i=1}^{n_1} close_{t-i} \\
\text{SMA}(n_2) &= \frac{1}{n_2}\sum_{i=1}^{n_2} close_{t-i} \\
\text{Moving Average of Variance}(n_1) &= \frac{1}{n_1}\sum_{i=1}^{n_1} (\text{close}_{t-i} - \text{SMA}(n_1))^2 \\
\text{Moving Average of Variance}(n_2) &= \frac{1}{n_2}\sum_{i=1}^{n_2} (\text{close}_{t-i} - \text{SMA}(n_2))^2 \\
\text{MovAvgVar}(n_1, n_2) &= \frac{\text{Moving Average of Variance}(n_1)}{\text{Moving Average of Variance}(n_2)}
\end{aligned}
$$

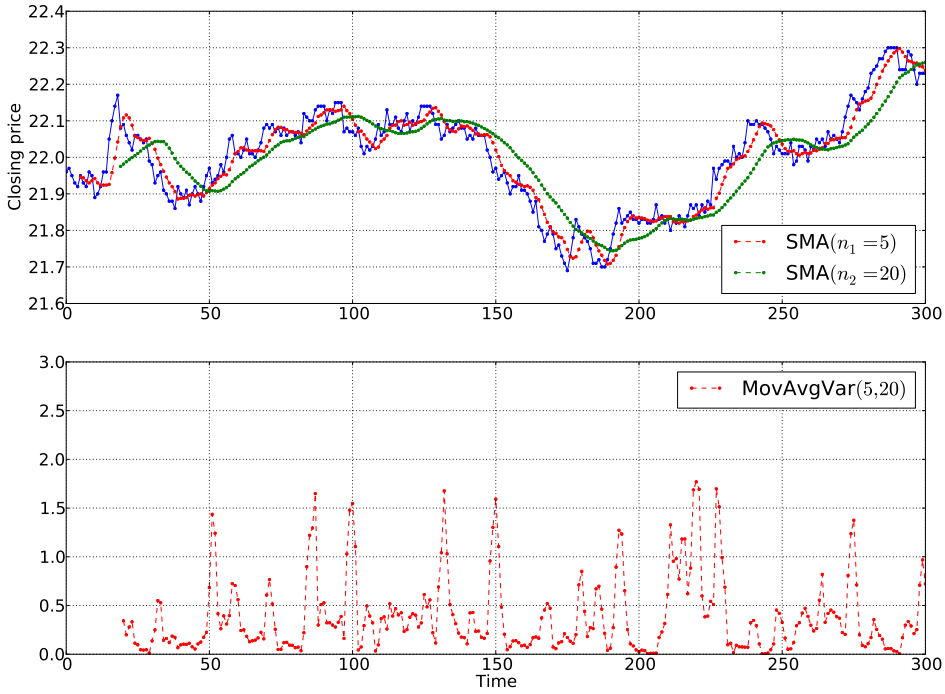


Figure C.4: Demonstrating price with MovAvgVar(5,20)

C.5 Relative strength index

The Relative Strength Index is credited to J. Welles Wilder, Jr. [235] and is intended to formalize the strength or weakness of a stock. To calculate the RSI the Relative Strength (RS) calculation is required; this is a ratio of the average gain over the past n periods, divided by the average loss over the past n periods. For example, the average gain over the past 5 periods is the total price gained during the up days divided by 5. The average loss over the past 5 days is the total price lost during the down days divided by 5. The RS is then inserted into the formula for the RSI seen

below:

$$\begin{aligned} \mathrm{RS}(n) &= \frac{\text{average gain over } t-n \text{ periods}}{\text{average loss over } t-n \text{ periods}} \\ \mathrm{RSI}(n) &= 100 - \frac{100}{1+\mathrm{RS}(n)} \end{aligned}$$

The RSI is on a scale from 0 to 100, with values of 70 generally considered overbought (sell) and values below 30 considered oversold (buy) [167]; in the paper we use $n = 5, 10, 20$.

To demonstrate the RSI as a technical analyst may use it, in Figure C.5 the closing price along with RSI($n = 5$) is shown. When the RSI crosses below 30, a green triangle is drawn on the price data, representing a "buy", and when the RSI crosses above the 70, a red triangle is drawn representing a "sell."

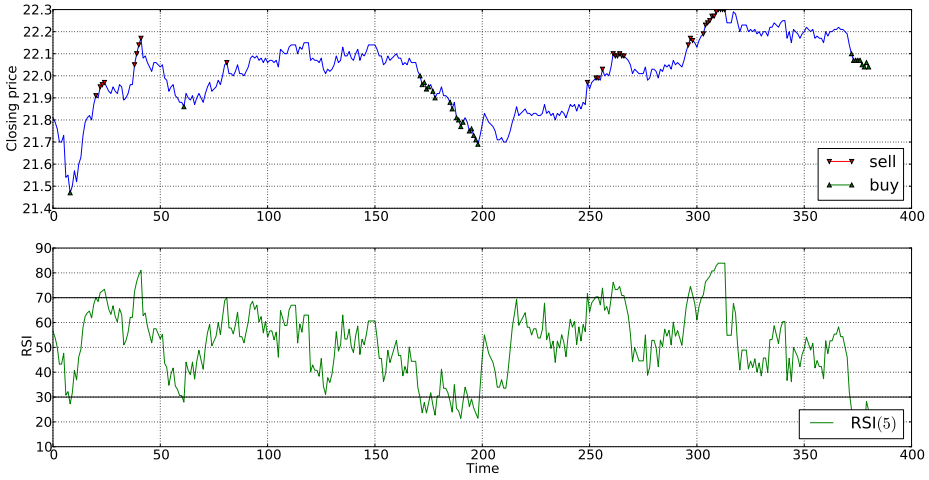


Figure C.5: Demonstrating price with RSI(5)

C.6 Chande momentum oscillator

Developed by Tushar S. Chande [39], the Chande Momentum Oscillator (CMO) is an attempt to identify overbought and oversold conditions in the market. Generally, technical traders view conditions over 50 as overbought conditions, while values under -50 indicate oversold conditions. Additionally, extreme values of CMO values indicate strong trends according to Chande. The formula folows (e.g. in the paper we use $n = 5, 10, 20$):

$$\begin{aligned} \text{up}(n) &= \sum_{i=1}^{n} \left(\text{close}_t - \text{close}_{t-i}\right), \text{ on "up" intervals} \\ \text{down}(n) &= \sum_{i=1}^{n} \left(\text{close}_{t-i} - \text{close}_t\right), \text{ on "down" intervals} \\ \text{CMO}(n) &= \frac{\text{up}(n) - \text{down}(n)}{\text{up}(n) + \text{down}(n)} \times 100 \end{aligned}$$

To demonstrate the CMO as technical analyst may use it, in Figure C.6 the closing price along with CMO($n = 5$) is shown. When the CMO crosses below -50, a green triangle is drawn on the price data, representing an oversold (buy), and when the CMO crosses above the 50 threshold, a red triangle is drawn representing a overbought (sell) indicator.

C.7 Aroon indicator

The Aroon indicator was developed by Tushar S. Chande and consists of two indicators, both an *Up* and a *Down* indicator, that work jointly to form a trading

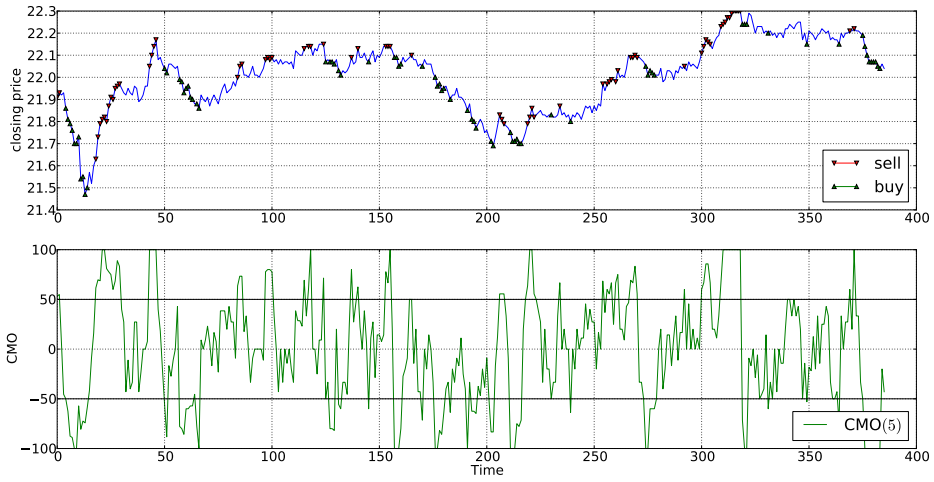


Figure C.6: Demonstrating price with CMO(5)

decision. The following shows the calculations for the indicators:

$$\begin{aligned}
\text{SinceHighest}(n) &= \text{periods since highest } \textit{high} \text{ covering } t-n \text{ instances} \\
\text{SinceLowest}(n) &= \text{periods since lowest } \textit{low} \text{ covering } t-n \text{ instances} \\
\text{UpIndicator}(n) &= 100 \times \left(\frac{n - \text{SinceHighest}(n)}{n} \right) \\
\text{DownIndicator}(n) &= 100 \times \left(\frac{n - \text{SinceLowest}(n)}{n} \right)
\end{aligned}$$

The indicator measures the number of periods back the peaks and valleys are during a time of n instances (in the paper we use $n = 5, 10, 20$). The UpIndicator(n) is intended to measure the strength of the up trend, while the DownIndicator(n) measures the strength of the down trend. Because they are reported as percentages of total time, the indicators fluctuate between zero and 100; values close to 0 and 100, represent weak and strong trends respectively. For example, if the highest high over a 10 instance period occurred 2 instances ago, then SinceHighest(10) = 2, and the UpIndicator(n) = 80. The thought is that since this movement recently occurred,

the upward trend is strong. A visual demonstration of this indicator can be found in Figure C.7.

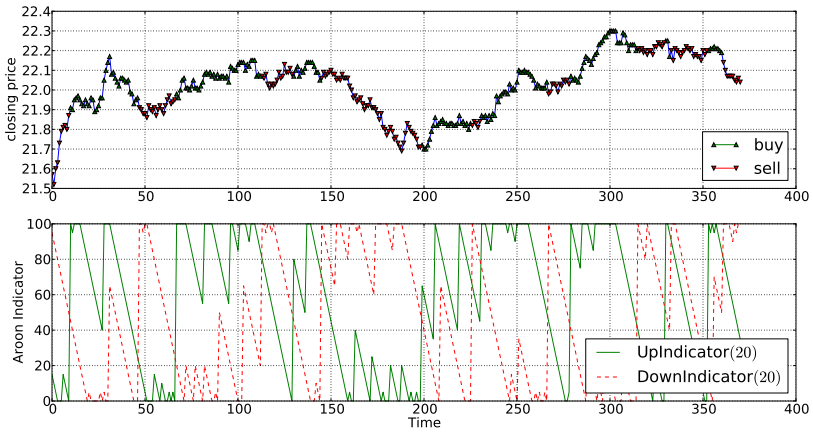


Figure C.7: Demonstrating Aroon UpIndicator(20) and DownIndicator(20)

C.8 Bollinger Bands

The Bollinger Band was developed and popularized in trading by John Bollinger [24, 143]. The indicator is comprised of a middle, upper, and lower band. The middle band is a moving average of size n of the average of the high, low, and closing prices. The upper band is d standard deviations (generally two) above the middle band, and

the lower is d standard deviations below. The formalization follows:

$$
\begin{aligned}
\text{SMA}(n) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\text{high}_{t-i}+\text{low}_{t-i}+\text{close}_{t-i}}{3} \\
\sigma &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\text{high}_{t-i}+\text{low}_{t-i}+\text{close}_{t-i}}{3}-\text{SMA}(n)\right)} \\
\text{MiddleBand}(n) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\text{high}_{t-i}+\text{low}_{t-i}+\text{close}_{t-i}}{3} \\
\text{UpperBand}(d,n) &= \text{MiddleBand}(n)+(d\times\sigma) \\
\text{LowerBand}(d,n) &= \text{MiddleBand}(n)-(d\times\sigma) \\
\text{BBand}(d,n) &= \begin{cases}
\text{close}_t > \text{UpperBand}(d,n) & : \text{“overbought”} \\
\text{close}_t < \text{LowerBand}(d,n) & : \text{“oversold”} \\
\text{else} & : \text{“do nothing”}
\end{cases}
\end{aligned}
$$

The upper and lower bands widen and narrow when the volatility of the price is higher or lower, respectively. They are used as an indicator of overbought or oversold conditions. When the price is near the upper or lower band, it indicates that a reversal is imminent [167]. In our paper we used parameters $d = 2, n = 5$ and $d = 2, n = 10$ and use as features whether the closing price of the stock is within the lower and upper bounds, above the upper band, or below the lower band.

Traders often visualize Bollinger Bands, similar to Figure C.8. In this example, the closing price of stock ANR is displayed with Bollinger Bands added with parameters MiddleBand($n = 5$), UpperBand($d = 2, n = 5$), and LowerBand($d = 2, n = 5$). Additionally, green triangles represent oversold areas (buy) below the lower bound, and red triangles represent areas that are overbought (sell).

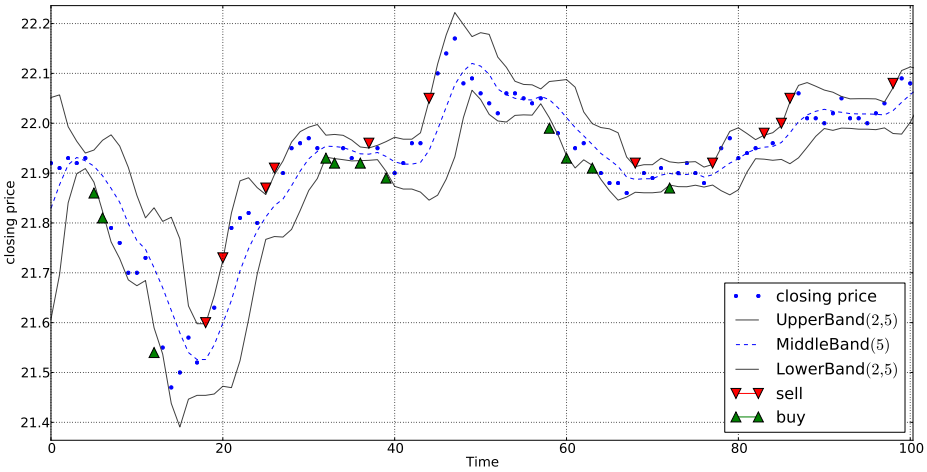


Figure C.8: Demonstrating price with Bollinger Bands

C.9 Commodity channel index

The Commodity channel index (CCI) [137] represents the mean of the current high, low, and closing price relative to the average of price over a recent period. It is designed to detect beginning and ending trends. It is comprised of a typical price, TP, which is a mean of the current high, low and closing prices. A moving average of the typical price (MATP) is computed over an interval of size n. Next, a mean deviation of typical price (MDTP) is computed and from here the Commodity channel index. The full calculation follows:

$$\begin{aligned}
\mathrm{TP} &= \frac{\mathrm{high}_t + \mathrm{low}_t + \mathrm{close}_t}{3} \\
\mathrm{MATP}(n) &= \frac{1}{n}\sum_{i=1}^{n} \mathrm{TP}_{t-i} \\
\mathrm{MDTP}(n) &= \frac{1}{n}\sum_{i=1}^{n} |\mathrm{TP}_t - \mathrm{MATP}(n)_{t-i}| \\
\mathrm{CCI}(n) &= \frac{\mathrm{TP}(n)_t - \mathrm{MATP}(n)_t}{\mathrm{MDTP}(n)_t \times 0.015}
\end{aligned}$$

According to Donald Lambert [137], the developer of CCI, a range of 100 to -100 is considered a normal trading range with values outside of this range indicating an overbought or oversold condition. In the paper we use $n = 5, 10, 20$.

A demonstration of CCI as technical analyst may use it, with a parameter $n = 20$, can be seen in Figure C.9. In this example, the closing price of stock ANR is displayed with the CCI pinpointing trading decisions. When the CCI crosses below -100, a green triangle is drawn on the price data, representing a "buy", and when the CCI crosses above 100, a red triangle is drawn representing a "sell."

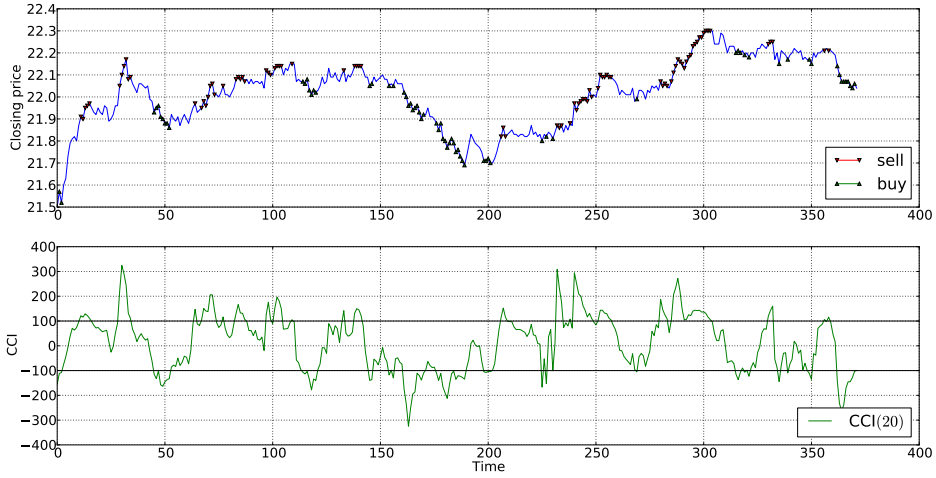


Figure C.9: Price with CCI(20)

C.10 Chaikin volatility

Chaikin volatility [64] is obtained by first finding the exponential moving average of the difference between the high and low prices over s instances. The exponential

moving average (EMA) at time t is then compared against the EMA at time $t-n$ and a percentage change is calculated.

$$\begin{aligned} \text{EMA}(s) &= ((\text{high}_t - \text{low}_t) - \text{EMA}(s)_{t-1}) \times \frac{2}{1+s} + \text{EMA}(s)_{t-1} \\ \text{ChaikinVolatility}(s,n) &= \frac{\text{EMA}(s)_t - \text{EMA}(s)_{t-n}}{\text{EMA}(s)_{t-n}} \times 100 \end{aligned}$$

According to Marc Chaikin, who this indicator is named for, an increase in the Chaikin volatility indicator over a short period of time indicates that traders are getting nervous (and thus the spread between the high and low price is widening) and therefore a market decrease is expected.

In the paper, we use the parameters $\text{ChaikinVolatility}(s,n) = \{5,5\}, \{10,10\}, \{20,20\}$.

C.11 Chaikin money flow

The Chaikin money flow indicator (CMF) takes the high, low, close and volume into account. It was developed by Marc Chaikin and is intended to measure the buying and selling pressures over a period of size n [113]. The formula follows:

$$\begin{aligned} \text{CLV} &= \frac{(\text{close}_t - \text{low}_t) - (\text{high}_t - \text{close}_t)}{\text{high}_t - \text{low}_t} \\ \text{CMF}(n) &= \frac{\sum_{i=1}^{n} (\text{CLV}_{t-i} \times \text{volume}_{t-i})}{\sum_{i=1}^{n} \text{volume}_{t-i}} \end{aligned}$$

The Chaikin money flow is used to confirm a market direction. For example, when the indicator is positive above a level y_1 it is suppose to confirm positive market direction. When the indicator is negative below a level y_2 it is suppose to confirm a negative market direction. In the paper we use $n = 5, 10, 20$.

C.12 Chaikin Accumulation/Distribution

The Chaikin accumulation/distribution (ChaikinAD) is a simplified version of the Chaikin money flow indicator (CMF). It can be seen below (the paper uses $n = 5, 10, 20$):

$$\text{ChaikinAD} = \left(\frac{(\text{close}_t - \text{low}_t) - (\text{high}_t - \text{close}_t)}{\text{high}_t - \text{low}_t} \right) \times \text{volume}_t$$

C.13 Close location value

The close location value (CLV) relates the closing price to its trading range, which is the period's high minus low price. The formula follows:

$$\text{CLV} = \frac{(\text{close}_t - \text{low}_t) - (\text{high}_t - \text{close}_t)}{\text{high}_t - \text{low}_t}$$

A visual representation of the CLV can be seen in Figure C.10.

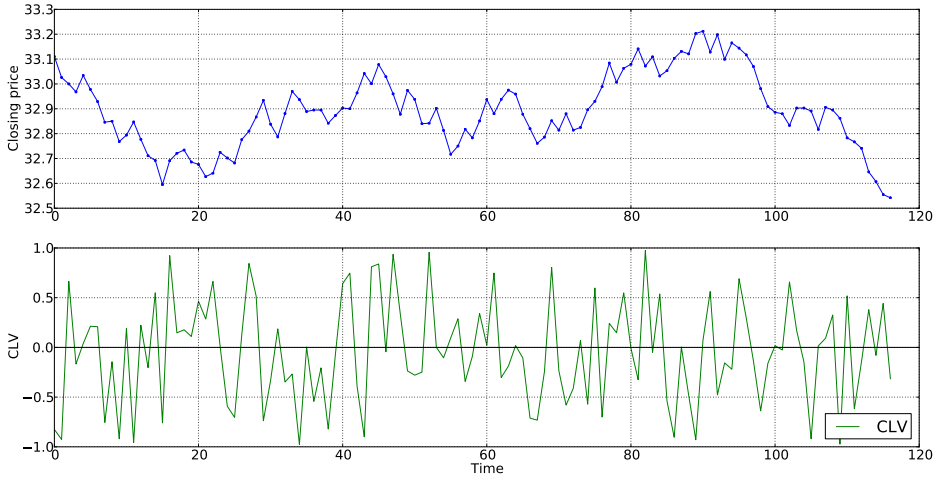


Figure C.10: Demonstrating the CLV

C.14 Moving average convergence divergence oscillator

The moving average convergence divergence oscillator (MACD) is the difference between two exponential moving averages, a short-term and a longer-term moving average and was developed by Gerald Appel [9]. A signal line, which is an exponential moving average of the MACD determines instances at which to buy and sell when used in conjunction with the MACD. The calculation follows:

$$\begin{aligned}
\text{ShortEMA}(n_1) &= \text{exponential moving average of size } n_1 \\
\text{LongEMA}(n_2) &= \text{exponential moving average of size } n_2 \\
\text{MACD}(n_1, n_2) &= \text{ShortEMA}(n_1)_t - \text{LongEMA}(n_2)_t \\
\text{SignalLine}(n_1, n_2, n_3) &= \text{exponential moving average of MACD}(n_1, n_2) \text{ of size } n_3 \\
\text{DiffMACDSignal}(n_1, n_2, n_3) &= \text{MACD}(n_1, n_2)_t - \text{SignalLine}(n_1, n_2, n_3)_t
\end{aligned}$$

For example, in the literature, high-values of the MACD indicate an overbought condition and low values indicate an oversold condition. When the MACD crosses below the signal line a buy signal is generated and likewise, a sell signal is generated when the MACD crosses above the signal line. This can also be interpreted by examining the DiffMACDSignal(n_1, n_2, n_3). Positive values would be considered bullish (oversold) and negative values would be considered bearish (overbought). In Figure C.11 the stock ANR is plotted along with the accompanying DiffMACDSignal indicator using the common parameters of 12, 26, and 9. Green triangles represent buy levels, and red triangles represent sell levels.

In our paper, we use two sets of parameters for the ShortEMA, LongEMA,

and SignalLine. First are parameters of 12, 26, and 9 and second are parameters of 6, 13, 4.

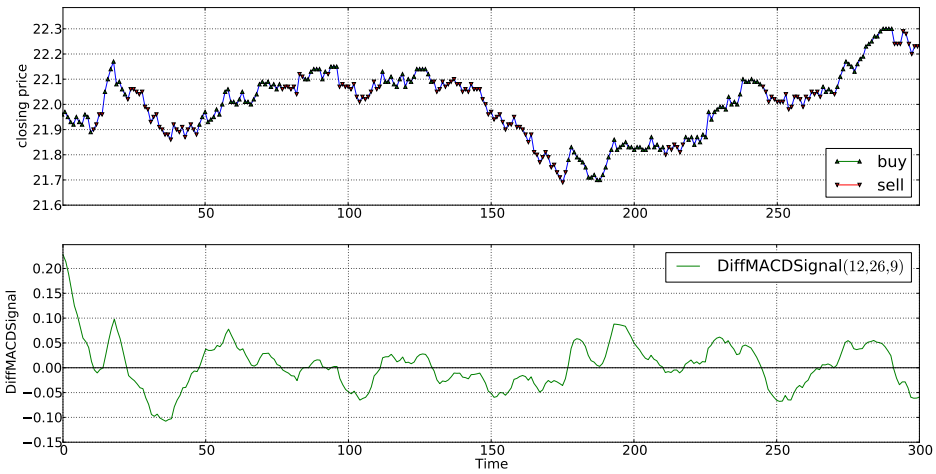


Figure C.11: Price with DiffMACDSignal$(12, 26, 9)$

C.15 Money flow index

The Money Flow index (MFI) uses price and volume to determine buying and selling pressure [64]. First, moving averages of prices and volumes are calculated and multiplied together to determine a dollar value of the amount of stock traded; this is called "money flow." A money ratio is then created by dividing positive money flows over negative money flows. From this an index is created to create a value of 0 to 100. For example, a value of 80 is generally considered overbought and a value of 20

is considered oversold. The formula follows:

$$\begin{aligned}
\text{TypicalAvgPrice}(n) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\text{high}_{t-i}+\text{low}_{t-i}+\text{close}_{t-i}}{3} \\
\text{TypicalAvgVolume}(n) &= \frac{1}{n}\sum_{i=1}^{n}\text{volume}_{t-i} \\
\text{MoneyFlow}(n) &= \text{TypicalAvgPrice}(n)_t \times \text{TypicalAvgVolume}(n)_t \\
\text{PositiveMoneyFlow}(n) &= \sum_{i=1}^{n}(\text{when MoneyFlow}(n)_{t-i} \geq \text{MoneyFlow}(n)_{t-i-1}) \\
\text{NegativeMoneyFlow}(n) &= \sum_{i=1}^{n}(\text{when MoneyFlow}(n)_{t-i} < \text{MoneyFlow}(n)_{t-i-1}) \\
\text{MoneyRatio}(n) &= \frac{\text{PositiveMoneyFlow}(n)_t}{\text{NegativeMoneyFlow}(n)_t} \\
\text{MoneyFlowIndex}(n) &= 100 - \frac{100}{1+\text{MoneyRatio}(n)_t}
\end{aligned}$$

In the paper, we use $n = 5, 10, 15$. A demonstration of the money flow index indicator as a technical analyst may use it can be seen in Figure C.12.

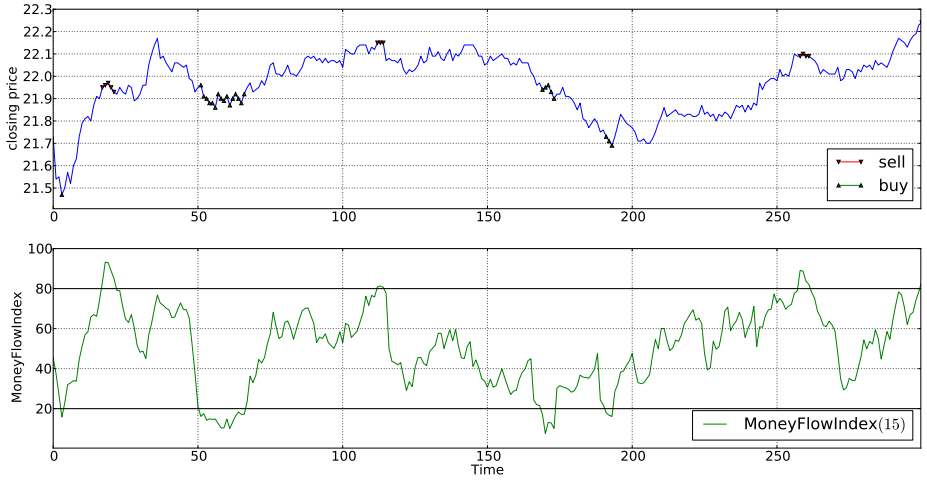


Figure C.12: Demonstrating price with MoneyFlowIndex(15)

C.16 Trend detection index

The Trend Detection Index (TDI) was developed by Pee [179] and it attempts to identify the beginning and ending of trends. The formalization follows [147]:

$$\begin{aligned}
\text{Momentum}(n) &= \text{close}_t - \text{close}_{t-n} \\
\text{MomentumAbs}(n) &= |\text{Momentum}(n)| \\
\text{MomentumSum}(n) &= \sum_{i=1}^{n} \text{Momentum}(n)_{t-i} \\
\text{MomentumSumAbs}(n) &= |\text{MomentumSum}(n)_t| \\
\text{MomentumAbsSumDiff}(n,k) &= \left(\sum_{i=1}^{n\times k} \text{MomentumAbs}(n)_{t-i}\right) - \left(\sum_{i=1}^{n} \text{MomentumAbs}(n)_{t-i}\right) \\
\text{TDI}(n,k) &= \text{MomentumSumAbs}(n)_t - \text{MomentumAbsSumDif}(n,k)_t
\end{aligned}$$

According to the developer, upward trends are signaled by a positive TDI value. Additionally, to obtain a better indication, the TDI and the MomentumSum are to be observed at time t. A buy signal is generated when both the TDI and the MomentumSum are positive, and a downtrend is observed when the TDI and the MomentumSum are negative. We demonstrate TDI on a stock in Figure C.13 with buy and sell signals added to the plot based on the indicators. Values of $n = 5, 10, 20$ and $k = 1, 2, 3$.

C.17 Williams %R Relative strength index

The Williams %R Relative strength index was developed by Larry Williams and attempts to indicate overbought (sell) and oversold (buy) conditions. In the original formula the Williams %R indicator is from 0 to 100 and is plotted on an inverted index. However, for ease of plotting, it is now common to multiply the

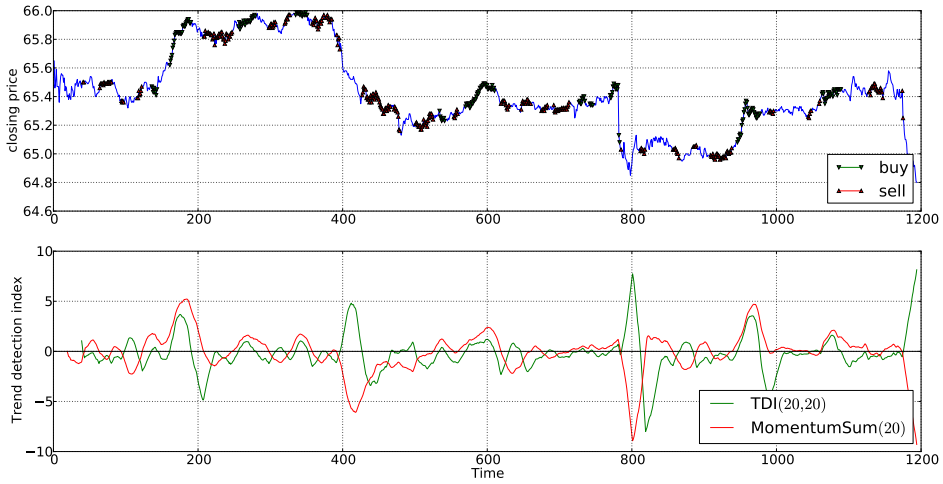


Figure C.13: Demonstrating price with TDI$(20, 20)$

Williams %R by -100 instead of by 100. Therefore readings of 0 to -20 are considered overbought and readings of -80 to -100 are considered oversold [209]. The calculation follows:

$$\begin{aligned} \text{HighestHigh}(n) &= \text{highest high covering } t-n \text{ instances} \\ \text{LowestLow}(n) &= \text{lowest low covering } t-n \text{ instances} \\ \text{WilliamsRSI}(n) &= -100 \times \frac{\text{HighestHigh}(n) - \text{close}_t}{\text{HighestHigh}(n) - \text{LowestLow}(n)} \end{aligned}$$

A visual demonstration of the Williams %R relative strength index can be found in Figure C.14. In our paper we use $n = 5, 10, 20$ and use the value of WilliamsRSI(n).

C.18 Stochastic Momentum Oscillator

The Stochastic Momentum Oscillator (Stoch) displays the location of each day's close relative to the high/low range over the past n instances [209]. Extreme

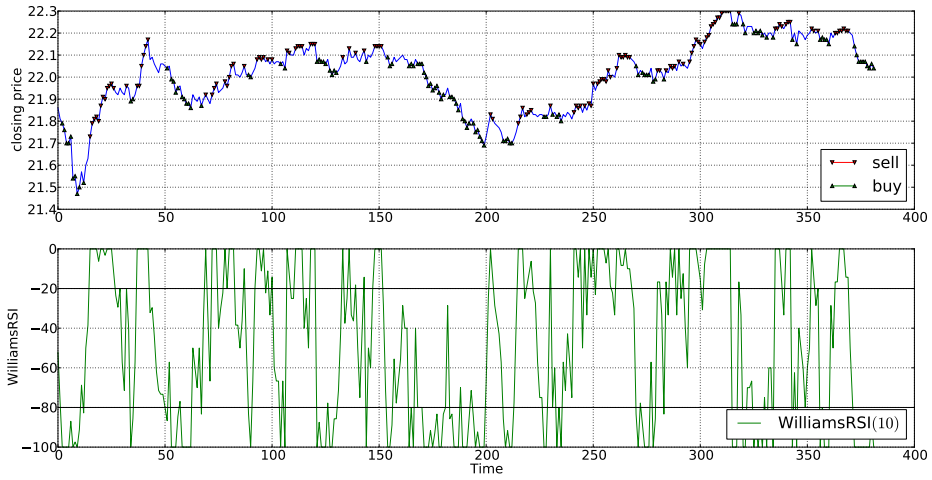


Figure C.14: Demonstrating price with WilliamsRSI(10)

values of high and low typically represent overbought and oversold conditions respectively. The formalization follows:

$$\begin{aligned}
\text{LowestLow}(n) &= \text{the lowest low over } t-n \text{ periods} \\
\text{HighestHigh}(n) &= \text{the highest high over } t-n \text{ periods} \\
\text{K}(n) &= 100 \times \frac{\text{close}_t - \text{LowestLow}(n)_t}{\text{HighestHigh}(n)_t - \text{LowestLow}(n)_t} \\
\text{Stoch}(n) &= \frac{1}{n}\sum_{i=1}^{n} \text{K}(n)_{t-i}
\end{aligned}$$

C.19 Correlation analysis

Another indicator used in the paper was correlation analysis of the current price at time t to the price at time $t-k$ covering n instances. Additionally the relationship between the high and low prices were computed. See Figure C.15 for a visualization of the closing price along with the correlation. In our paper we use $n = 5, 10, 20$ for corrHighLow(n) and $k = 1, 2, 3$; $n = 5, 10, 20$ for corrClose(k, n). See

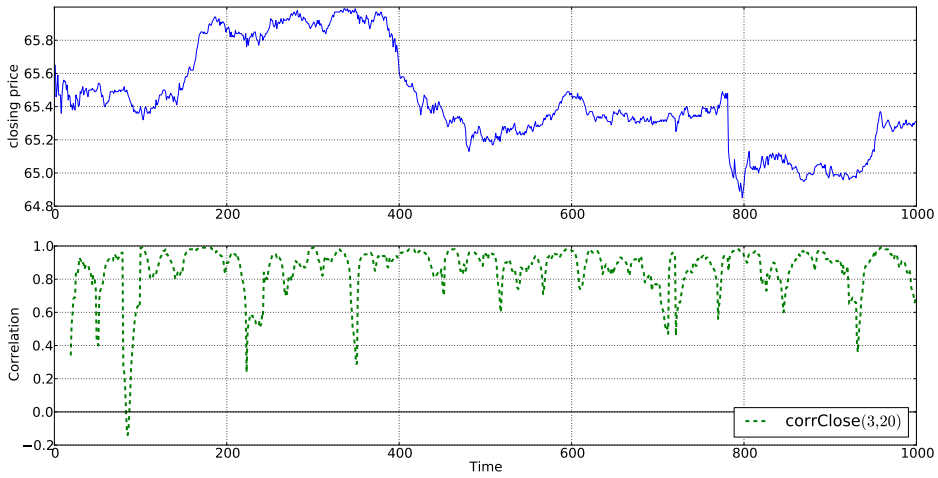


Figure C.15: Demonstrating price with corrClose(3,20)

the following calculation:

$$\text{corrHighLow}(n) = \frac{n\left(\sum_{i=1}^{n} low_{t-i}high_{t-i}\right) - \left(\sum_{i=1}^{n} low_{t-i}\right)\left(\sum_{i=1}^{n} high_{t-i}\right)}{\left[n\sum_{i=1}^{n} low_{t-i}^2 - \left(\sum_{i=1}^{n} low_{t-i}\right)^2\right]\left[n\sum_{i=1}^{n} high_{t-i}^2 - \left(\sum_{i=1}^{n} high_{t-i}\right)^2\right]}$$

$$\text{corrClose}(k,n) = \frac{n\left(\sum_{i=1}^{n} close_{t-i}close_{t-i-k}\right) - \left(\sum_{i=1}^{n} close_{t-i}\right)\left(\sum_{i=1}^{n} close_{t-i-k}\right)}{\left[n\sum_{i=1}^{n} close_{t-i}^2 - \left(\sum_{i=1}^{n} close_{t-i}\right)^2\right]\left[n\sum_{i=1}^{n} close_{t-i-k}^2 - \left(\sum_{i=1}^{n} close_{t-i-k}\right)^2\right]}$$

REFERENCES

[1] Steven B Achelis. *Technical Analysis from A to Z*. McGraw Hill New York, 2001.

[2] Jerry Adler. Raging bulls: How Wall Street got addicted to light-speed trading. Wired.com, August 2012; accessed October 2012.

[3] Charu C Aggarwal. *Data Streams: Models and Algorithms*. Springer, 2007.

[4] Charu C Aggarwal. Data streams: An overview and scientific applications. In *Scientific Data Mining and Knowledge Discovery*, pages 377–397. Springer, 2010.

[5] Colin Alexander. *Capturing Full-trend Profits in the Commodity Futures Markets: Maximizing Reward and Minimizing Risk with the Wellspring System*. Windsor Books, 1992.

[6] S. S. Alexander. Price movement in speculative markets: Trend or random walks, no. 2. *The Random Character of Stock Market Prices, MIT Press, Cambridge, MA*, pages 338–372, 1964.

[7] Romain Allez and Jean-Philippe Bouchaud. Individual and collective stock dynamics: Intra-day seasonalities. *New Journal of Physics*, 13(2):025010, 2011.

[8] Andrew Ang and Geert Bekaert. Stock return predictability: Is it there? *Review of Financial Studies*, 20(3):651–707, 2007.

[9] Gerald Appel. *Technical Analysis: Power Tools for Active Investors*. Financial Times/Prentice Hall, 2005.

[10] Ramji Balakrishnan, Xin Ying Qiu, and Padmini Srinivasan. On the predictive ability of narrative disclosures in annual reports. *European Journal of Operational Research*, 202(3):789–801, May 2010.

[11] Bernard Baumohl. *The Secrets of Economic Indicators: Hidden Clues to Future Economic Trends and Investment Opportunities*. FT Press, 2012.

[12] Marianne Baxter and Robert G King. Measuring business cycles: Approximate band-pass filters for economic time series. *Review of Economics and Statistics*, 81(4):575–593, 1999.

[13] Alessandro Beber and Michael W Brandt. Resolving macroeconomic uncertainty in stock and bond markets*. *Review of Finance*, 13(1):1–45, 2009.

[14] Arie Ben-David. About the relationship between ROC curves and Cohen's kappa. *Engineering Applications of Artificial Intelligence*, 21(6):874–882, 2008.

[15] Pablo Bermejo, Jose A Gámez, and Jose M Puerta. A GRASP algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. *Pattern Recognition Letters*, 32(5):701–711, 2011.

[16] Michael J Berry and Gordon Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. John Wiley & Sons, Inc., 2004.

[17] H. Bessembinder. Bid-ask spreads in the interbank foreign exchange markets. *Journal of Financial Economics*, 35(3):317–348, 1994.

[18] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

[19] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *The Journal of Machine Learning Research*, 99:1601–1604, 2010.

[20] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, and Ricard Gavalda. Improving adaptive bagging methods for evolving data streams. *Advances in Machine Learning*, pages 23–37, 2009.

[21] Albert Bifet and Richard Kirkby. Data stream mining a practical approach. 2009.

[22] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1):245–271, 1997.

[23] J. Bollen, H. Mao, and X. J Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1), 2011.

[24] John Bollinger. Using bollinger bands. *Technical Analysis of Stocks & Commodities*, 10, 1992.

[25] Anthony Brabazon and Michael O'Neill. *Biologically Inspired Algorithms for Financial Modelling*, volume 20. Springer Berlin, 2006.

[26] Max Bramer. *Principles of Data Mining*. Springer, 2007.

[27] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[28] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[29] Christian T Brownlees and Giampiero M Gallo. Ultra high-frequency data management. *Modelli Complessi e Metodi Computazionali Intensivi per la Stima e la Previsione (ed. C. Provasi), Atti del Convegno S. Co., CLUEP, Padova*, 2005.

[30] Christian T Brownlees and Giampiero M Gallo. Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51(4):2232–2245, 2006.

[31] Dariusz Brzezinski and Jerzy Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. 2013.

[32] Francesco Buccafurri and Gianluca Lax. Reducing data stream sliding windows by cyclic tree-like histograms. In *Knowledge Discovery in Databases: PKDD 2004*, pages 75–86. Springer, 2004.

[33] Francesco Buccafurri and Gianluca Lax. Approximating sliding windows by cyclic tree-like histograms for efficient range queries. *Data & Knowledge Engineering*, 69(9):979–997, 2010.

[34] Warren E Buffett. The superinvestors of Graham-and-Doodsville. *Hermes, Columbia Business School magazine*, 1984.

[35] Edmund Burke and Graham Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer Science+ Business Media, 2005.

[36] John Y. Campbell and Samuel B. Thompson. Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies*, 21(4):pp. 1509–1531, 2008.

[37] John Y Campbell and Motohiro Yogo. Efficient tests of stock return predictability. *Journal of Financial Economics*, 81(1):27–60, 2006.

[38] Wesley S Chan. Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260, 2003.

[39] Tushar S Chande. *Beyond technical analysis: How to develop and implement a winning trading system*, volume 101. Wiley. com, 2001.

[40] P. H.K Chang and C. L Osler. Methodical madness: Technical analysis and the irrationality of exchange rate forecasts. *The Economic Journal*, 109(458):636–661, 1999.

[41] Amar Kumar Chaudhary. Impact of behavioral finance in investment decisions and strategies – a fresh approach. *International Journal of Management Research and Business Strategy*, 2013.

[42] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, pages 875–886. Springer, 2010.

[43] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, June 2002.

[44] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

[45] Sheng Chen and Haibo He. Sera: Selectively recursive approach towards nonstationary imbalanced stream data mining. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 522–529. IEEE, 2009.

[46] Sheng Chen and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

[47] Sheng Chen, Haibo He, Kang Li, and Sachi Desai. Musera: Multiple selectively recursive approach towards imbalanced stream data mining. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[48] Tai-Liang Chen, Ching-Hsue Cheng, and Hia Jong Teoh. Fuzzy time-series based on Fibonacci sequence for stock price forecasting. *Physica A: Statistical Mechanics and its Applications*, 380:377–390, 2007.

[49] Fang Chu and Carlo Zaniolo. Fast and light boosting for adaptive mining of data streams. *Advances in Knowledge Discovery and Data Mining*, pages 282–292, 2004.

[50] David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

[51] Iván Contreras, J Ignacio Hidalgo, Laura Nuñez-Letamendía, and Yiyi Jiang. Parallel architectures for improving the performance of a GA based trading system. In *Parallel Architectures and Bioinspired Algorithms*, pages 189–218. Springer, 2012.

[52] Jonathan D Cryer and Kung-Sik Chan. *Time Series Analysis: With Applications in R*. SpringerVerlag New York, 2008.

[53] S. R Das and M. Y Chen. Yahoo! for Amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[54] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Machine Learning International Workshop and Conference*, pages 74–81. Citeseer, 2001.

[55] A Philip Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, pages 278–292, 1984.

[56] A Philip Dawid and Vladimir G Vovk. Prequential probability: Principles and properties. *Bernoulli*, 5(1):125–162, 1999.

[57] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Can blog communication dynamics be correlated with stock market activity? In *Proceedings of the nineteenth ACM Conference on Hypertext and Hypermedia*, HT '08, pages 55–60, New York, NY, USA, 2008. ACM.

[58] P. M DeMarzo, D. Vayanos, and J. Zwiebel. Persuasion bias, social influence, and unidimensional opinions. *The Quarterly Journal of Economics*, 118(3):909, 2003.

[59] Thomas Dietterich. Ensemble methods in machine learning. *Multiple Classifier Systems*, pages 1–15, 2000.

[60] Gregory Ditzler, Robi Polikar, and Nitesh Chawla. An incremental learning algorithm for non-stationary environments and class imbalance. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2997–3000. IEEE, 2010.

[61] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM, 1999.

[62] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM, 2000.

[63] Marco Dorigo and Gianni Di Caro. Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2. IEEE, 1999.

[64] Robert Davis Edwards, John Magee, and WHC Bassetti. *Technical Analysis of Stock Trends*. CRC Press, 2012.

[65] Charles Elkan. The foundations of cost-sensitive learning. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 973–978. Citeseer, 2001.

[66] Tapio Elomaa and Petri Lehtinen. Maintaining optimal multi-way splits for numerical attributes in data streams. In *Advances in Knowledge Discovery and Data Mining*, pages 544–553. Springer, 2008.

[67] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on*, 22(10):1517–1531, 2011.

[68] David Enke and Suraphan Thawornwong. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4):927–940, 2005.

[69] Frank J Fabozzi, Sergio Focardi, and Caroline Jonas. Trends in quantitative equity management: survey results. *Quantitative Finance*, 7(2):115–122, 2007.

[70] Thomas N Falkenberry. High frequency data filtering. *Tick Data*, 2002.

[71] E. F. Fama. The behavior of stock-market prices. *Journal of Business*, 38(1), 1965.

[72] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 128–137. ACM, 2004.

[73] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[74] Usama M Fayyad and Keki B Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.

[75] Miguel A. Ferreira and Pedro Santa-Clara. Forecasting stock market returns: The sum of the parts is more than the whole. *Journal of Financial Economics*, 100(3):514 – 537, 2011.

[76] FM Labs. Indicator reference. Retrieved from http://www.fmlabs.com, October 2013.

[77] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.

[78] Yoav Freund and Robert Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[79] L. Frieder and J. Zittrain. Spam works: Evidence from stock touts and corresponding market activity. *Hastings Communications and Entertainment Law Journal*, 30:479, 2007.

[80] Mohamed Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. A survey of classification methods in data streams. *in Data Streams: Models and Algorithms*, pages 39–59, 2007.

[81] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[82] João Gama. *Knowledge Discovery from Data Streams*. Citeseer, 2010.

[83] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004*, pages 286–295. Springer, 2004.

[84] João Gama and Carlos Pinto. Discretization from data streams: Applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 662–667. ACM, 2006.

[85] João Gama and Pedro Rodrigues. Stream-based electricity load forecast. *Knowledge Discovery in Databases: PKDD 2007*, pages 446–453, 2007.

[86] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 329–338. ACM, 2009.

[87] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[88] Jing Gao, Bolin Ding, Wei Fan, Jiawei Han, and Philip S Yu. Classifying data streams with skewed class distributions and concept drifts. *Internet Computing, IEEE*, 12(6):37–49, 2008.

[89] Jing Gao, Wei Fan, Jiawei Han, and Philip S Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proc 7th International Conference on Data Mining, SIAM, Philadelphia, PA*, 2007.

[90] Adam Ghandar, Zbigniew Michalewicz, Martin Schmidt, Thuy-Duong To, and Ralf Zurbruegg. A computational intelligence portfolio construction system for equity market trading. In *IEEE Congress on Evolutionary Computation, 2007.*, pages 798–805. IEEE, 2007.

[91] Abhijeet Godase and Vahida Attar. Classification of data streams with skewed distribution. In *Proceedings of the CUBE International Information Technology Conference*, pages 284–289. ACM, 2012.

[92] Linda S Goldberg and Christian Grisse. Time variation in asset price responses to macro announcements. Technical report, 2013.

[93] Sanford J Grossman and Joseph E Stiglitz. On the impossibility of informationally efficient markets. *The American Economic Review*, 70(3):393–408, 1980.

[94] B. Gu, P. Konana, A. Liu, B. Rajagopalan, and J. Ghosh. Identifying information in stock message boards and its implications for stock market efficiency. In *Workshop on Information Systems and Economics*, 2006.

[95] Martin Gutlein, Eibe Frank, Mark Hall, and Andreas Karwath. Large-scale attribute selection using wrappers. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 332–339. IEEE, 2009.

[96] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[97] Mark A Hall. *Correlation-Based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[98] Mark Andrew Hall and Geoffrey Holmes. Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 15(6):1437–1447, 2003.

[99] Michael Bonnell Harries, Claude Sammut, and Kim Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

[100] Y. Hashimoto, T. Ito, T. Ohnishi, M. Takayasu, H. Takayasu, and T. Watanabe. *Random walk or a run: Market microstructure analysis of the foreign exchange rate movements based on conditional probability*. National Bureau of Economic Research Cambridge, Mass., USA, 2008.

[101] Nikolaus Hautsch. *Econometrics of Financial High-Frequency Data*. SpringerVerlag Berlin Heidelberg, 2012.

[102] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[103] Anthony F Herbst and Craig W Slinkman. Political-economic cycles in the US stock market. *Financial Analysts Journal*, pages 38–44, 1984.

[104] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

[105] Thomas Ryan Hoens. *Living in an Imbalanced World*. PhD thesis, University of Notre Dame, 2012.

[106] John H Holland. Genetic algorithms. *Scientific American*, 267(1):66–72, 1992.

[107] Cheng-Lung Huang and Cheng-Yi Tsai. A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting. *Expert Systems with Applications*, 36(2, Part 1):1529 – 1539, 2009.

[108] Jin Huang and C.X. Ling. Using AUC and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299–310, 2005.

[109] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106. ACM, 2001.

[110] Jacek Jarmulak and Susan Craw. Genetic algorithms for feature selection and weighting. In *in Proceedings of the IJCAI*, volume 99, pages 28–33. Citeseer, 1999.

[111] George J Jiang, Ingrid Lo, and Giorgio Valente. High frequency trading in the US treasury market: Evidence around macroeconomic news announcements. In *Proceedings of the Northern Finance Association*, Quebec City, Canada, 2013.

[112] Ruoming Jin and Gagan Agrawal. Efficient decision tree construction on streaming data. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 571–576. ACM, 2003.

[113] K Senthamarai Kannan, P Sailapathi Sekar, M Mohamed Sathik, and P Arumugam. Financial stock market forecast using data mining techniques. In *proceedings of International Multi Conference of Engineers and Computer Scientist (IMECS2010)*, volume 1, 2010.

[114] Michael Kearns. Thoughts on hypothesis boosting. *Unpublished manuscript, December*, 1988.

[115] Mark G Kelly, David J Hand, and Niall M Adams. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–371. ACM, 1999.

[116] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[117] Randy Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth National Conference on Artificial Intelligence*, pages 123–128. AAAI Press, 1992.

[118] Kyoung-jae Kim and Ingoo Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, 19(2):125–132, 2000.

[119] YongSeog Kim and W Nick Street. An intelligent system for customer targeting: A data mining approach. *Decision Support Systems*, 37(2):215–228, 2004.

[120] YongSeog Kim, W Nick Street, and Filippo Menczer. Optimal ensemble construction via meta-evolutionary ensembles. *Expert Systems with Applications*, 30(4):705–714, 2006.

[121] Richard Brendon Kirkby. *Improving Hoeffding Trees*. PhD thesis, The University of Waikato, 2007.

[122] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

[123] Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering: Learning drifting concepts. In *Proc. of AAAI-98/ICML-98 workshop Learning for Text Categorization*, pages 33–40, 1998.

[124] Arno J Knobbe and Eric KY Ho. Pattern teams. In *Knowledge Discovery in Databases: PKDD 2006*, pages 577–584. Springer, 2006.

[125] Ron Kohavi. The power of decision tables. In *Machine Learning: ECML-95*, pages 174–189. Springer, 1995.

[126] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997.

[127] Ron Kohavi and Daniel Sommerfield. Targeting business users with decision table classifiers. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 249–253, 1998.

[128] Marko Kolanovic. Rise of cross-asset correlations. *J.P. Morgan – Delta One Strategy*, 2011.

[129] Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[130] Miroslav Kubat. Floating approximation in time-varying knowledge bases. *Pattern Recognition Letters*, 10(4):223–227, 1989.

[131] Miroslav Kubat. Conceptual inductive learning: The case of unreliable teachers. *Artificial Intelligence*, 52(2):169–182, 1991.

[132] Ludmila Kuncheva. Classifier ensembles for changing environments. *Multiple Classifier Systems*, pages 1–15, 2004.

[133] Ludmila I Kuncheva. A theoretical study on six classifier fusion strategies. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):281–286, 2002.

[134] Ludmila I Kuncheva, James C Bezdek, and Robert PW Duin. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[135] Robert K Lai, Chin-Yuan Fan, Wei-Hsiu Huang, and Pei-Chann Chang. Evolving and clustering fuzzy decision tree for financial time series data forecasting. *Expert Systems with Applications*, 36(2):3761–3773, 2009.

[136] Thomas Navin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. Embedded methods. In *Feature Extraction*, pages 137–165. Springer, 2006.

[137] Donald R Lambert. Commodity channel index: Tool for trading cyclic trends. *Technical Analysis of Stocks & Commodities*, 1, 1983.

[138] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Language models for financial news recommendation. In *Proceedings of the ninth International Conference on Information and Knowledge Management*, pages 389–396. ACM, 2000.

[139] Yan-Nei Law and Carlo Zaniolo. An adaptive nearest neighbor classification algorithm for data streams. *Knowledge Discovery in Databases: PKDD 2005*, pages 108–120, 2005.

[140] C. M. C. Lee and M. J Ready. Inferring trade direction from intraday data. *Journal of Finance*, pages 733–746, 1991.

[141] Petri Lehtinen, Matti Saarela, and Tapio Elomaa. Online chimerge algorithm. In *Data Mining: Foundations and Intelligent Paradigms*, pages 199–216. Springer, 2012.

[142] David J Leinweber. Stupid data miner tricks: Overfitting the s&p 500. *The Journal of Investing*, 16(1):15–22, 2007.

[143] Camillo Lento, Nikola Gradojevic, and CS Wright. Investment information content in bollinger bands? *Applied Financial Economics Letters*, 3(4):263–267, 2007.

[144] Richard M. Levich and Lee R. Thomas. The significance of technical trading-rule profits in the foreign exchange market: A bootstrap approach. *Journal of International Money and Finance*, 12(5):451–474, 1993.

[145] X. Li, C. Wang, J. Dong, F. Wang, X. Deng, and S. Zhu. Improving stock market prediction by integrating both market news and stock prices. In *Database and Expert Systems Applications*, pages 279–293. Springer, 2011.

[146] Linn Software Inc. Investor/RT. Retrieved from http://www.linnsoft.com, October 2013.

[147] Linn Software Inc. Trend detection index. Retrieved from http://www.linnsoft.com/tour/techind/tdi.htm, October 2013.

[148] Marco Lippi, Lorenzo Menconi, and Marco Gori. Balancing recall and precision in stock market predictors using support vector machines. In *Neural Nets and Surroundings*, pages 51–58. Springer, 2013.

[149] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer Verlag, 2011.

[150] Huan Liu, Farhad Hussain, Chew Lim Tan, and Manoranjan Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.

[151] Peng Liu, Yong Wang, Lijun Cai, and Longbo Zhang. Classifying skewed data streams based on reusing data. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 4, pages V4–90. IEEE, 2010.

[152] Andrew Lo. Efficient markets hypothesis. 2007.

[153] Andrew Lo and Mark Mueller. Warning: Physics envy may be hazardous to your wealth! 2010.

[154] Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4):1705–1770, 2002.

[155] Victoria López, Alberto Fernández, Jose G Moreno-Torres, and Francisco Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.

[156] Roger Lowenstein. *Buffett: The Making of an American Capitalist*. Random House Digital, Inc., 1995.

[157] Yu-Hon Lui and David Mole. The use of fundamental and technical analyses by foreign exchange dealers: Hong Kong evidence. *Journal of International Money and Finance*, 17(3):535–545, 1998.

[158] Burton G. Malkiel. *A Random Walk Down Wall Street*. W. W. Norton & Company, 2nd edition, January 2003.

[159] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[160] Gerald Martin and John Puthenpurackal. Imitation is the sincerest form of flattery: Warren Buffett and Berkshire Hathaway. *Available at SSRN 806246*, 2008.

[161] Saulius Masteika and Rimvydas Simutis. Stock trading system based on formalized technical analysis and ranking technique. In *Computational Science–ICCS 2006*, pages 332–339. Springer, 2006.

[162] Lukas Menkhoff. The use of technical analysis by fund managers: International evidence. *Journal of Banking & Finance*, 34(11):2573–2586, 2010.

[163] Lukas Menkhoff and Mark P Taylor. The obstinate passion of foreign exchange professionals: Technical analysis. *Journal of Economic Literature*, 45(4):936–972, 2007.

[164] M-A Mittermayer. Forecasting intraday stock price trends with text mining techniques. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2004.

[165] Alberto Moraglio, Cecilia Di Chio, and Riccardo Poli. Geometric particle swarm optimisation. In *Genetic Programming*, pages 125–136. Springer, 2007.

[166] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

[167] John Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.com, 1999.

[168] John J Murphy. *Intermarket Technical Analysis: Trading Strategies for the Global Stock, Bond, Commodity, and Currency Markets*, volume 6. Wiley, 1991.

[169] C. Neely, P. Weller, and R. Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *Journal of Financial and Quantitative Analysis*, 32(04):405–426, 1997.

[170] V. Niederhoffer and M. F. M. Osborne. Market making and reversal on the stock exchange. *Journal of the American Statistical Association*, 61(316):897–916, 1966.

[171] Toru Ohira, Naoya Sazuka, Kouhei Marumo, Tokiko Shimizu, Misako Takayasu, and Hideki Takayasu. Predictability of currency market exchange. *Physica A: Statistical Mechanics and its Applications*, 308(1-4):368–374, May 2002.

[172] R. C. A. Oomen. Properties of realized variance under alternative sampling schemes. *Journal of Business and Economic Statistics*, 24(2):219–237, 2006.

[173] Nikunj C Oza. Online bagging and boosting. In *2005 IEEE International Conference Systems on Man and Cybernetics*, volume 3, pages 2340–2345. IEEE, 2005.

[174] Nikunj C Oza and Stuart Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 359–364. ACM, 2001.

[175] Nikunj Chandrakant Oza. *Online Ensemble Learning*. PhD thesis, 2001.

[176] Ajoy K Palit and Dobrivoje Popovic. *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications (Advances in Industrial Control)*. Springer-Verlag New York, Inc., Secaucus, NJ, 2005.

[177] Tan Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[178] Cheol-Ho Park and Scott H Irwin. What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826, 2007.

[179] M.H. Pee. Trend detection index. *Technical Analysis of Stocks & Commodities*, 19(10):54–55, 2001.

[180] Robi Polikar. Ensemble learning. In *Ensemble Machine Learning*, pages 1–34. Springer, 2012.

[181] Robi Polikar, L Upda, SS Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(4):497–508, 2001.

[182] Tobias Preis, Dror Y Kenett, H Eugene Stanley, Dirk Helbing, and Eshel Ben-Jacob. Quantifying the behavior of stock correlations under market stress. *Scientific Reports*, 2, 2012.

[183] John Price and Edward Kelly. Warren buffett: Investment genius or statistical anomaly? *Intelligent finance: A convergence of mathematical finance with technical and fundamental analysis*, 2004.

[184] Dorian Pyle. *Data Preparation for Data Mining*, volume 1. Morgan Kaufmann Publishers, Inc., 1999.

[185] John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[186] John Ross Quinlan. *C4.5: Programs for Machine Learning*, volume 1. Morgan Kaufmann Publishers, Inc., 1993.

[187] E. Rahm and H. H Do. Data cleaning: Problems and current approaches. *Bulletin of the Technical Committee*, page 3, 2000.

[188] David E. Rapach, Jack K. Strauss, and Guofu Zhou. Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *Review of Financial Studies*, 23(2):821–862, 2010.

[189] Juan Reboredo, José Matías, and Raquel Garcia-Rubio. Nonlinearity in forecasting of high-frequency stock returns. *Computational Economics*, 40(3):245–264, 2012.

[190] Michael Rechenthin and W. Nick Street. Using conditional probability to identify trends in intra-day high-frequency equity pricing. *Physica A: Statistical Mechanics and its Applications*, 392(24):6169 – 6188, 2013.

[191] Michael Rechenthin, W. Nick Street, and Padmini Srinivasan. Stock chatter: Using stock sentiment to predict price direction. *Journal of Algorithmic Finance*, Volume 2(3-4), 2013.

[192] Richard Roll. A simple implicit measure of the effective Bid-Ask spread in an efficient market. *The Journal of Finance*, 39(4):1127–1139, September 1984.

[193] Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 513–522, New York, NY, USA, 2012. ACM.

[194] John Rushing, Sara Graves, Evans Criswell, and Amy Lin. A coverage based ensemble algorithm (cbea) for streaming data. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 106–112. IEEE, 2004.

[195] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial Intelligence: A Modern Approach*, volume 2. Prentice hall Englewood Cliffs, NJ, 1995.

[196] Jeffrey A Ryan. quantmod: Quantitative financial modelling framework. *R package version 0.3-17*, 2011.

[197] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[198] Pedro Santa-Clara and Rossen Valkanov. The presidential puzzle: Political cycles and the stock market. *The Journal of Finance*, 58(5):1841–1872, 2003.

[199] N. Sazuka. Analysis of binarized high frequency financial data. *The European Physical Journal B-Condensed Matter and Complex Systems*, 50(1):129–131, 2006.

[200] Stephan Schulmeister. Profitability of technical stock trading: Has it moved from daily to intraday data? *Review of Financial Economics*, 18(4):190–201, 2009.

[201] Robert P Schumaker and Hsinchun Chen. A quantitative stock prediction system based on financial news. *Information Processing & Management*, 45(5):571–583, 2009.

[202] Robert P Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12, 2009.

[203] David W Scott. Sturges' rule. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):303–306, 2009.

[204] Raquel Sebastião and João Gama. Change detection in learning histograms from data streams. In *Progress in artificial intelligence*, pages 112–123. Springer, 2007.

[205] Securities, Exchange Commission, et al. Findings regarding the market events of may 6, 2010. *Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues*, 2010.

[206] Wei Shen, Xiaopen Guo, Chao Wu, and Desheng Wu. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems*, 24(3):378–385, 2011.

[207] Kenneth O Stanley. Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[208] StockCharts.com. Chart school. Retrieved from http://www.stockcharts.com, October 2013.

[209] StockCharts.com. William percent r. Retrieved from http://tinyurl.com/3yorqr, October 2013.

[210] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM, 2001.

[211] Richard J. Sweeney. Some new filter rule tests: Methods and results. *The Journal of Financial and Quantitative Analysis*, 23(3):285–300, 1988.

[212] M. Tanaka-Yamawaki. Stability of Markovian structure observed in high frequency foreign exchange data. *Annals of the Institute of Statistical Mathematics*, 55(2):437–446, 2003.

[213] Mark P Taylor and Helen Allen. The use of technical analysis in the foreign exchange market. *Journal of international Money and Finance*, 11(3):304–314, 1992.

[214] Lamartine Almeida Teixeira and Adriano Lorena Inacio De Oliveira. A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, 37(10):6885–6890, 2010.

[215] A. Timmermann. Elusive return predictability. *International Journal of Forecasting*, 24(1):1–18, 2008.

[216] A. Timmermann and C. W. J. Granger. Efficient market hypothesis and forecasting. *International Journal of Forecasting*, 20(1):15–27, 2004.

[217] Christopher Ting. Which daily price is less noisy? *Financial Management*, 35(3):81–95, 2006.

[218] Chih-Fong Tsai and Yu-Chieh Hsiao. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, 50(1):258 – 269, 2010.

[219] Ruey S Tsay. *Analysis of Financial Time Series*, volume 543. Wiley-Interscience, 2005.

[220] J Ulrich. TTR: Technical trading rules. *R package version 0.20-1*, 2009.

[221] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[222] Francesco Virili and Bernd Freisleben. Nonstationarity and data preprocessing for neural network predictions of an economic time series. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 129–134. IEEE, 2000.

[223] Periasamy Vivekanandan and Raju Nedunchezhian. Mining data streams with concept drifts using genetic algorithm. *Artificial Intelligence Review*, 36(3):163–178, 2011.

[224] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM, 2003.

[225] Tao Wang and Jian Yang. Nonlinearity and intraday efficiency tests on energy futures markets. *Energy Economics*, 32(2):496–503, 2010.

[226] Jordan Wathen. Why berkshire can beat the market. The Motley Fool, May 2013.

[227] Paul A Weller, Geoffrey C Friesen, and Lee M Dunham. Price trends and patterns in technical analysis: A theoretical and empirical examination. *Journal of Banking & Finance*, 33(6):1089–1100, 2009.

[228] Cheng G Weng and Josiah Poon. A new evaluation measure for imbalanced datasets. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 27–32. Australian Computer Society, Inc., 2008.

[229] Frank H Westerhoff. Technical analysis based on price-volume signals and the power of trading breaks. *International Journal of Theoretical and Applied Finance*, 9(02):227–244, 2006.

[230] Felix Wex, Natascha Widder, Michael Liebmann, and Dirk Neumann. Early warning of impending oil crises using the predictive power of online news stories. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 1512–1521. IEEE, 2013.

[231] Gerhard Widmer. Combining robustness and flexibility in learning drifting concepts. In *ECAI*, pages 468–468. Citeseer, 1994.

[232] Gerhard Widmer and Miroslav Kubat. Learning flexible concepts from streams of examples: Flora2. 1992.

[233] Gerhard Widmer and Miroslav Kubat. Effective learning in dynamic environments by explicit context tracking. In *Machine Learning: ECML-93*, pages 227–243. Springer, 1993.

[234] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[235] J Welles Wilder. *New Concepts in Technical Trading Systems*. Trend Research McLeansville, NC, 1978.

[236] Ian H Witten, Eibe Frank, and Mark A Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Elsevier, 2011.

[237] Andrew C Worthington. Political cycles in the Australian stock market since federation. 2006.

[238] Cheng-Huei Yang, Li-Yeh Chuang, and Cheng-Hong Yang. IG-GA: A hybrid filter/wrapper method for feature selection of microarray data. *Journal of Medical and Biological Engineering*, 30(1):23–28, 2010.

[239] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.

[240] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.

[241] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Machine Learning International Workshop then Conference*, pages 412–420. Morgan Kaufmann Publishers, Inc., 1997.

[242] Ying Yang and Geoffrey I Webb. Discretization for naive-Bayes learning: Managing discretization bias and variance. *Machine Learning*, 74(1):39–74, 2009.

[243] Ying Yang, Geoffrey I Webb, and Xindong Wu. Discretization methods. In *Data Mining and Knowledge Discovery Handbook*, pages 101–116. Springer, 2010.

[244] Lean Yu, Shouyang Wang, and Kin Keung Lai. Mining stock market tendency using GA-based support vector machines. In *Internet and Network Economics*, pages 336–345. Springer, 2005.

[245] Lean Yu, Shouyang Wang, and Kin Keung Lai. *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks*. International Series in Operations Research & Management Science. Springer, 2007.

[246] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Machine Learning International Workshop then Conference*, volume 20, page 856, 2003.

[247] Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

[248] Xue Zhang, Hauke Fuehres, and Peter A Gloor. Predicting stock market indicators through twitter 'i hope it is not as bad as i fear'. *Procedia-Social and Behavioral Sciences*, 26:55–62, 2011.

[249] Xue Zhang, Hauke Fuehres, and Peter A Gloor. Predicting asset value through twitter buzz. In *Advances in Collective Intelligence 2011*, pages 23–34. Springer, 2012.

[250] Inde Žliobaite. Learning under concept drift: An overview. Technical report, Vilnius University, 2009.