Trading System Development

An Interactive Qualifying Project submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By:

Zachary Styer

Andrew Tautkus

Matthew Upham

With help from:

Professor Hossein Hakim

Professor Michael Radzicki

Date: 5/3/2016

Report submitted to:

Professor Hossein Hakim

Worcester Polytechnic Institute

Table of Contents

Chapter 1: Introduction & Overview of the Research

Statement of the Problem

The purpose of this interactive qualifying project is for each of us to create stock trading strategies that make anywhere greater than 8% a year. Ultimately, we want to beat the S&P 500's annual rate of return.

Statement of Problem Importance

Making money through stock trading is important because it leads to making extra earnings on the side to fulfill the lifestyle you want, or you can turn it into a full time job and take control of your life. The general United States population are finding it increasingly important to take control of their financial futures.

As pensions are mostly being phased out completely, and Mutual Funds in 401K's are eating up profits from fees, people are looking to other options to have a comfortable financial retirement.

Another reason for the importance of this is to have the knowledge and skill to create a business to help people take control of their financial futures. Through our success in this IQP, we could bring these tips & tricks to the real world, and manage other professionals' money to hopefully beat returns of other retirement accounts.

Brief Summary of Literature & Statement of Creativity

Ultimately, we have developed a strategies based off a simple but customized stock screener. Our teammate Andrew created a Stock Screener and Neural Net Combination that we have been able to use. The Stock Screener allows us to target stocks with high slopes, and high

R^2 values, along with other parameters. The Neural Net helped us predict the high and low of stock prices in future days.

There are other Stock Screeners in the field such as HotScan or TradeStation , but we wanted a screener which was more customized toward our specific strategies. We were able to pull all the data from Google, and have much more flexibility over the sample of data that we used, as well as creating different charts, graphs and outputs using our stock screener.

Conclusions

We realized immediately that we need to stick to a strategy while trading. While sporadic gains can occur from randomized trading, over time it is much more profitable to follow specific rules and strategies. Also, adding stop losses were extremely important, especially as last minute dips in stock value could diminish returns by a decent amount. Knowing when to exit the market due to the strategies was also very important, especially because if a trader stays in the market too long, their chances of losing money increase.

Chapter 2: Trading & Investing

What it Means to be a Trader or Investor

There are two types of traders or investors: retail and institutional. A retail trader (often called "individual investor" or "small investor") is an individual who trades their own money. An institutional trader trades others money. (Zucchi 2015) As a result, institutional traders often have large amounts of buying power. When an institutional trader makes a trade, it isn't unusual for it to move the price. In order to be successful as a retail trader, one must understand the actions on the institutional traders and how they could affect the retail investor's trades.

Trading and investing have many differences. Traders Trading focuses on buying low and selling high or shorting high and selling low. Traders have a high frequency of buying and selling. Traders are often divided into five groups: Position Traders, Swing Traders, Day Traders, Scalp Traders, and Flash Traders. Position Traders will hold their position (long or short) for a period of months to years. Swing Traders hold their position for days to weeks. Day Traders hold their position for minutes up to just for the day. Scalp Traders hold their positions for seconds to minutes. Flash Traders hold their position for microseconds. (Folger 2012).

Investors tend to favor stocks. Active Investors view themselves as owners of the stock of which they invest. As such, they often believe that shorting a stock is unpatriotic. Active investors focus on the long term. For them, the focus when deciding on a stock is the dividend potential, and the asset appreciation. Active Investors pay attention to their portfolio, and watch the actions of the firms of which they have invested. (Trading System Development Michael J. Radzicki PhD)

In general, the longer the position is held, the smaller the commission and slippage, less need to watch the short term market, more risk, longer capital is tied up, and more attention is required to watch firm, industry, and economic news. The opposite is true for shorter term positions. When dealing with extremely small time frames (scalpers and flash traders), one should pay attention to time for the order to make it to the server. As such, real estate next to the stock exchange is very important for these types of trading. The longer the time frame, this becomes less of a factor.

Behavior Finance or Efficient Market Hypothesis

Traders follow the belief that one can systematically beat the market. They follow 'Behavior Finance' which basically states that information travels irregularly. (Behavior Finance 2003) 'Behavior Finance' directly contradicts the Efficient Market Hypothesis. The Efficient Market Hypothesis states that no one can systematically beat the market. Under the hypothesis, anyone or group that beat the market only did so by random chance.

Persons who believe the efficient market should "park" their money in Exchange Traded Funds or Mutual Funds. Many passive investors fall into this category, and there are many ways to do so efficiently. The key lies in diversification. One should invest in a variety of sectors, and over a variety of markets (London/NYSE/Nasdaq/China). On the other hand, traders believe in 'Behavior Finance', and as such they try to find an edge, or pattern in the market. They believe they can exploit patterns to systematically make money.

Trend and Cycle

In Macroeconomics, there is a trend as well as a cycle. In the United States, the overall long trend is normally positive. The cycle is the sinusoidal pattern in the data caused by periods of booms and busts. (Trend and Cycle 2012). During booms, the economy is climbing more than the trend. During busts, the economy is falling below the trend. A recession is an example of a bust. Investors and traders are often highly interested in how the market is performing. The longer term the trader, the more important the trend and cycle become. During down trends, traders may be focused on shorting and during uptrends, longs.

The Four Asset Classes

There are four asset classes: equities, bonds, commodities, and currencies. Equities are ownership of a company, this is commonly called stock. (Equity Definition 2003) Bonds are loans to the government or corporations. (Bond Definition 2003) Commodities are goods, examples include oil, gold, corn.(Commodity Definition 2003) Commodities are often traded as futures contracts. This means that if you are holding onto the futures contract at the date stated on the contract, you get the good delivered to you.

The four asset classes are connected. In general, the U.S. dollar and commodities trend in opposite directions, commodities and bonds trend in opposite directions, and stock prices and bond prices trend in the same direction. (Inter-Market Analysis & Sector Rotation Michael J. Radzicki PhD) Although, these general rules can change during deflation. Financial institutions pay particular attention to this phenomena and allocate their resources accordingly.

Equities tend to be decently liquid, but it mostly depends on the market capitalization of the stock in question. For example, Apple (AAPL) tends to be very liquid. On the other hand, MannKind Corp (MNKD) is a low liquidity. (Speights 2016) Often, the liquidity of a symbol is connected to the name recognition of the company that it represents.

The margin requirement for equities is 50%. (Spiegelman 2013) That means, when buying on margin, one must have half of the value in either cash or equity. Once the margin debt to total equity ratio hits 0.5, one must either sell off some of the stock (and pay back the margin), add more cash, or add more equity to the account. (Margin Requirements 2016). The advantage of margin accounts for equities comes when dealing with dividends. Many large companies have dividend yields much greater than the interest rates on margin accounts, but one must keep in mind the added risk that comes with the added reward. (Buying Stocks 2016). That is, if the value of the stock decreases, the owner is still responsible for the margin loan.

When it comes to taxes, Traders and investors have many differences in what is allowed to be deducted. Investors can only claim their interest as an expense if they itemize their deductions and it is under the interest limitation. Traders can deduct their full interest expenses. Traders can deduct expenses related to their home office, and seminars. Investors cannot deduct home office, or seminar expenses. Both can deduct the costs associated with books, and subscriptions under miscellaneous expenses, but Investors can only count the amount that exceeds 2% of their adjusted gross income. Traders get the full deduction. Perhaps the greatest advantage for Trader Status is that their taxes depend on their net profit. (Tax Benefits 2016).

The disadvantage of Trader Status is that the trading profits are taxed at the rate of ordinary income. Investors are taxed at the capital gains rate. Traders have to report all profits.

This includes unsold stocks. (Thomas 2016). In addition to finances, Trader Status comes with a higher rate of audits, and fewer tax accountants familiar with the process.

Pattern Day Traders are required to keep a minimum of $25,000 in their account. A Pattern Day Trader is defined as "One who trades the same security four or more times per day (buys and sells) over a five-day period, and for whom same-day trades make up at least 6% of their activity for that period." (Harvey 2013). With this requirement. Pattern day traders are allowed to leverage up to four times the cash value.(Day Trading Margin 2016).

Equity traders must pay attention to stock splits. A stock split is when a stock is split into parts with the separate part prices added together equaling the original price of the stock. The market cap of the stock remains constant. There are a few reasons that companies will split their stock. One of the main reasons is based on psychology. Investors and traders see a lower price and assume it is a better deal. Stock splits also increase the liquidity of the stock. One potential problem with stock splits occurs during back testing. If the back test data is not corrected for the stock split, the report will see the stock split as a huge drop in the value of the stock, registering it as a loss. Commissions can also be affected by stock splits, depending on the commission type of the broker. If the commission is weighted by the number of stocks, then it is most efficient to buy before the split.(Understanding Stock 2016). Dollar value exits can also be affected by stock splits, and percent based exits tend to wear better against stock splits. (Chaos Technical 2012)

Bonds are basically loans. The loan can be to the federal government, a large corporation, or basket of mortgages. Bonds are rated based on their risk level. Triple A (AAA) is the best rating, and it almost guaranteed. Most of the time AAA bonds have low risk and low yield. One

barrier to entry in the bond market is the amount of capital required. Most of the time, the bond market is left to the large banks and financial corporations.

Bonds have the issue of low liquidity, and difficulty in back testing, due to the individualized nature of the bond market. Most of the importance of bonds for retail traders comes from understanding the actions of the large financial institutions. The largest headache when dealing with bonds is the uniqueness of each bond. The most well-known bonds are U.S. government bonds, which are the safest, normally AAA, and lowest yield.

Bonds can be bought on margin, with a margin requirement from 1% to 100% depending on the length of time, the type, and the grade of the bond. One must pay considerable attention to the rate of return when buying bonds using margin. The margin interest rate has the distinct possibility of eating profits. Due to all of these potential problems, retail traders stay out of the bond market, and leave it to the large financial corporations. (Margin 2016).

Commodities tend to be less liquid than stocks and currencies. The most liquid commodities are oil, corn, natural gas, soybeans, and wheat. The rest of the commodities have relatively low liquidity. (Farley 2015) Many retail traders elect to trade ETFs based on the commodity of choice rather than to trade the commodity market.

Margin rates for commodities are based on the commodity of choice. Overall, the margin requirements are relatively high. Account size for commodity trading tends to be relatively large in order to compete for the large order sizes.

The major currency pairs (EUR/USD, USD/JPY, GBP/USD, USD/CHF, USD/CAD, AUD/USD, and NZD/USD) tend to be very liquid markets. They are the most traded currency pairs. On the other hand, exotic pairs like USD/ZAR (U.S.A. / South Africa) tend to be less

liquid. In addition, the spread for exotic pairs tends to be two or three times larger than the spread for major currency pairs. (What is Forex 2011).

Currencies have a margin requirement of 2% for major currency pairs. This enables a trader to trade 50 times the size of their account. All of the other currency pairs have a margin requirement of 5%. This enables a trader to trade an exotic pair at a volume 20 times the size of their account. Like equities, currency trading on margin also has the possibility of a margin call. Most of the time, currency traders do not go above 25 times their account size. (12 CFR 240.9 2016).

Currency trading has the advantage of low transaction costs. The spread is normally less than 0.1%. The market it open 24 hours a day for most of the week. The only period of time that the market is not open is between the close of NYSE on Friday until Australia opens on Monday. Currency traders can have an account size as small as $25. There are many resources for an individual to set up an account at that size. (Why Trade Forex 2016).

Forex traders must pay particular attention to the global economy, and the actions on both side of the currency pair. During periods of time surrounding big announcements, many forex traders elect to stay out of the market. Many of the decisions of forex traders get swayed by the actions of financial authorities.

Sector Rotation in Equity Markets

During different parts of the business cycle (boom and bust cycle), different sectors perform better. This is called “Sector Rotation.” (Fidelity Learning Center 2016). Large financial corporations pay particular attention to this phenomena. During the course of the cycle, they will

adjust their allocation according to where the in the business cycle the market is at. As a retail investor/trader, one should pay particular attention to where the big money is moving. In finance, it is most efficient to go with the flow of money instead of against it.

Breadth of the Market

Stock exchanges are a market in which stocks can be bought and sold. In order for a stock to be on the exchange, they have to have to have an initial public offering (IPO), which is basically the process to get shares put onto the market. The exchange's job is to track the price of the stock. They track both the "bid price" (what someone is willing to pay) and the "ask price" (what someone is willing to sell for). The difference between these two prices is called the bid-ask spread. (Harper 2004).

The largest stock exchange is the New York Stock Exchange (NYSE). It is physically located on Wall St. in New York, New York. The NYSE has existed since 1792, and does $169 billion per day in trading revenue (in 2013). (New York Stock Exchange 2016). The NYSE is open from 9:30 am to 4:00 pm Eastern Time.The NASDAQ is the world's largest online trading stock exchange. It includes stocks such as Apple, Google, Microsoft, and Amazon.(Nasdaq Definition 2016). In addition to the NYSE and NASDAQ, there are many other stock exchanges, but they are the biggest two in the U.S.A.

There are many ways of determining how "The Market" is performing. The most used indices are the S&P 500, the Dow Jones Industrial Average, and the NASDAQ. Each index has its own characteristics. The S&P 500 contains, as the name suggests, 500 of the largest companies listed in the NYSE or NASDAQ. The stocks are weighted according to their market

capitalizations, so the larger the market cap of the stock, the larger the pull in the S&P 500's value. The Dow Jones Industrial Average (DJIA) is comprised of 30 stocks. Historically, The Dow was used to gauge the performance of the industrial sector (the largest sector during its birth in 1896). Today, many of the companies in the DJIA have little or nothing to do with the industrial sector (i.e. American Express, Apple, Goldman Sachs and many more). The NASDAQ (as previously stated) is the world's largest online trading stock exchange, and the second overall behind the New York Stock Exchange (NYSE). Due to its electronic origins, the NASDAQ has become an Index for the electronics industry.

In addition to the overall U.S. economy, "The Market," individual indexes, industries, and stocks all can trend. Typically these trends fall under three categories: Up trending, down trending and side trending. Up trending is what an investor would typically like to see. Higher highs are being made, alongside higher lows this is often associated with the symbol of the bull. Down trending is the opposite of up trending and as such is associated with the symbol of a bear. In a down trending market, lower lows and lower highs are being made. This is a good scenario for shorts, but typically terrible for Investors. Side trending markets happen when neither higher highs nor lower lows are being made.

Indices are not the only way of understanding how "The Market" is performing. Traders and Investors also look to Market breadth analysis. A diary for the stocks listed on the New York Stock Exchange is shown below. The first statistic listed is the "Issues Traded", this is the number of trades that occurred. "Generally, the more active the market in total volume, the more issues traded." (Bajkowski 1998). "Issues Traded" normally isn't the best statistic to monitor due to the relative consistency in its value and lack of indication of direction. On the other hand,

"Advances", "Declines", and "Unchanged" (the next three lines), are important for monitoring the exchange. "When most stocks are participating in a general market price increase, the market is said to have good breadth. However, if blue chip indexes, such as Dow Jones industrials, are advancing while the majority of stocks are declining, the market has poor breadth." (Bajkowski 1998). The overbought/oversold index is found by dividing the advances by the declines and averaging over a few days to smooth out random movements.

Another statistic listed is the new highs and lows. These statistics are the number of stocks hitting a 52 week high or low. In the figure below, a much greater amount of new lows are being created with respect to new highs. Volume is often considered "votes" in favor of a direction. Rising or falling prices are confirmed by large volumes. Volume with direction gives more information to an investor/trader. In the diary for September 1, 1998, significantly more volume was associated with decline than advance.

The traders (Arms) index (TRIN) is the most common combination of these statistics. Basically TRIN is the ratio of advances to declines divided by the ratio of advance volume to decline volume. TRIN values above 1 are bearish, at 1 are neutral, less than 1 are bullish. TRIN values can be misleading due to the nature of the advances to decline ratio being in the numerator. It is not uncommon for a strong bullish ratio to appear proceeding a bearish market.

The final statistic listed is the number of block trades. Block trades are trades in excess of 10,000 shares. This is often an indicator of the actions of large financial institutions, and can be greatly valuable to retail traders/investors. (Bajkowski 1998)

FROM WALL STREET JOURNAL

DIARIES

NYSE	MON	FRI	WK AGO
Issues traded	3,571	3,548	3,559
Advances	414	918	1,420
Declines	2,886	2,213	1,650
Unchanged	271	417	489
New highs	15	18	35
New lows	1,183	813	260
zAdv vol (000)	37,819	249,081	287,923
zDecl vol (000)	874,456	562,642	245,119
zTotal vol (000)	914,696	838,726	557,235
Closing tick[1]	−779	−88	+167
Closing Arms[2] (trin)	3.32	.94	.73
zBlock trades	19,545	17,517	10,935

Source: The Wall Street Journal, September 1, 1998.

Figure 2.1- WSJ Data Example

Derivatives

Derivatives are contracts, and their value is derived from the underlying entity. Be it an equity, index, interest rate, or a commodity. The two most common types of derivatives are futures and options.

Futures Contracts are contracts "between two parties where both parties agree to buy and sell a particular asset of specific quantity and at a predetermined price, at a specified date in future." (Futures Contract Definition 2016) Futures contracts are commonly used in commodities where expenses to obtain the commodity. (Futures Markets 2016).

Options Contracts are contracts "between a buyer and seller that gives the purchaser of the option the right to buy or sell a particular asset at a later date at an agreed upon price." (Options Contract 2016). The purchaser is making a bet that the price of the asset will increase, and the seller is betting that the price with stay the same or decrease. The value of an options

contract comes from the length of time left and the current value of the stock in comparison to the contract price.

Chapter 3: Trading Systems

A trading system is a grouping of different rules, boundaries or criterion that decide entry and exit points for trading equities. Overall, these signals are marked on a set of graphs and charts which are used to make trades usually in real time, or simulated very close to real time. (Kuepper 2004).

Different sets of analysis tools can be used to create the different rules for a trading system. Some of the most common technical tools are Moving Averages and Bollinger Bands. Moving averages are lagging indicators, which are based off of prices from previous time periods. Moving averages help filter out erratic spikes in data, smoothing out charts so they can be analyzed more precisely. (Moving Average 2003).

Bollinger Bands consist of an upper and lower band which are both 2 standard deviation above and below the simple moving average. The standard deviation, which is a measure of volatility means that Bollinger Bands are able to adapt and move with the market during different states. When the market becomes more erratic, the Bollinger Bands distance themselves from the average. When the market becomes less erratic, the Bollinger Bands tighten in toward the average. Technical traders use the bands tightening as an indicator that the market volatility will soon spike. (Bollinger Bands 2003).

Other tools that are commonly used are Oscillators, Stochastic Oscillator, and Relative Strength. These are alternatives to be used with or against Moving Averages and Bollinger Bands.

An Oscillator is a tool used to see which stocks are overbought or oversold within a short time period. The oscillator is banded in between two different values which are considered extreme, and constructed from findings of a trend indicator. When the Oscillator itself flows toward the upper extreme, the asset is considered to be overbought. When the Oscillator approaches the lower point, it is considered to be oversold. Oscillators are mostly useful when a trend cannot be seen from a superficial view. When a corporation's stock is trading horizontally, that is generally when Oscillators are being used. (Oscillator Definition 2004).

The Stochastic Oscillator is a specific type of oscillator which compares the closing price of an equity to the range of its' prices of a period of time. One can change how sensitive the oscillator is by changing the length of the time period, or creating a moving average of all the results.

The specific formula is **%K = 100[(C - L14)/(H14 - L14)]** where:

C = closing price from the most recent period

L14 = 14 day lowest price

H14 = 14 day highest price

This indicator theorizes that in a market that is trending upwards, prices will close near the peak price of the day. During markets that are trending downwards, prices will close near the lowest point of the day. A trader would execute a trade when %K intersects a "three-period moving average", called %D (which takes further analysis). (Stochastic Oscillator 2003).

Lastly, Relative Strength is another technical technique that is commonly used. This uses momentum investing and compares performances of difference equities to the performance of the total market. Relative strength makes a calculation to see which investment has the strongest

performance out of the batch of the trader's choosing. It's basically a "buy high, sell higher" strategy, and makes the assumption that an equity whose price has been increasing will continue to increase. (Relative Strength 2003).

Types of brokerage accounts

There are multiple types of brokerage accounts that are used by the common investor & trader. Having these options is important because the trader can use an account based on their financial needs or goals.

First of all is the cash account. The investor deposits cash into this account, and then can purchase equities using this cash. It is a relatively simple account used by many traders and investors

The margin account adds a line of credit. The cash and other securities are lika collateral for credit. Here you can use credit from the broker in order to purchase equities. These brokers generally charge less interest than credit card companies. The downfall of these types of accounts are that you are required to gain a higher ROI than the cash account in order to make the same profit.

Option accounts allow the trader to use options, which are very risky compared to the alternatives of trading with stocks. An option is a contract which provides the buyer with a right to either buy or sell a security at a strike price or date. These are risky, and generally shouldn't be used by the average investor.

Lastly, the Individual Retirement account (IRA) is almost like a savings account, except the individual receives tax breaks. These are opened by the individual, and there are many

accounts such as Traditional IRAs, Roth IRAs, SEP IRAs and SIMPLE IRAs. Ultimately, the individual has to choose which account they open depending on their financial goals.

There are many advantages and disadvantages for both margin accounts and cash accounts. For margin accounts, these are advantageous for investors who want to leverage their funds. It's a very cost effective way to earn gains if you are experienced, and generally good in the short term. Margin accounts do have their disadvantages though.They come with a line of credit (a debit applied to your account) because you are borrowing money. Margin account brokers charge a daily interest rate, which can add up if your account is not gaining money. This daily interest rate is based on the Current Prime Rate, and an extra percentage added on by the broker. Also, to keep the margin account active, you must stay within a certain margin ratio set by the brokerage.

For the bearish investor, they can take a short if they know the stock will fall. They can also cover the short position by taking a long position if needed. The trader can earn a profit on the difference between the initial short sale and the cost to buy the shares at a lower price, while taking into account the margin interest changes over the course of the time period.

Cash accounts have different advantages and disadvantages, especially for the bearish investor. Advantages includes no credit line, so you're not going negative on you balance account from the start. You get to deposit cash directly in your account, and you do not owe interest to the broker. The disadvantages are that it's harder to leverage your money quickly because you can't run a line of credit. You need alternative strategies to be a good bearish investor because on long positions, you can only use cash. (Cash Account 2003).

For cash and margin accounts, there are generally types of rates depending on the broker you use. Common brokers such as Schwab, TD Ameritrade or Fidelity all charge similar but slightly different rates because they have to stay competitive.

Take Fidelity for example: they charge $7.95 per trade on any number of shares for most stocks, ETFs and Options. Depending on the buy to close order cost, they can either charge more per contract, or no extra depending on what you choose. The difference between a cash and margin account is that the margin rate is applied to the latter. For example, Fidelity charges a different percentage based on your debit balance. For a debit balance between $50,000 and $499,999, there is a Base margin rate + 50%, and an effective rate of 7.075%. (Trading Commissions 2016).

There are many different financial markets where different types of assets are traded. These markets include Capital Markets, Money Markets, Cash or Spot Markets, Forex and the Interbank Market, Primary vs Secondary Markets, OTC Market, and lastly the Third and Fourth Markets. Traders can trade in the different market for many different reasons.

Capital Markets

In a capital market, this is where both individual traders and traders at institutions will trade various securities. Capital markets include the stock market and the bond market.

The stock market allows investors to trade in companies that trade publicly. This allows companies to sell portions of theirs company to investors, and the investors provide these companies with capital. The stock market is divided into the primary market and the secondary market. Basically the primary market is the market where the newest issues are offered before others, and then the secondary market is where subsequent trades happen.

Bond Markets are where bonds are traded. A bond is where investor loans money to a corporation or government figure, which in return borrows those funds for a specified amount of time at a fixed interest rate. Bonds are utilized by many different entities including governments, companies and municipalities to fund different projects. Bonds are bought and sold by investors around the world, and the market can be also referred as the debt/credit/fixed-income market. The selection of bonds that investors can purchase include corporate bonds, US Treasury bonds, notes, bills and municipal bonds.

Money Market

The money market is another portion of the overall financial market. Generally the money market includes instruments with relatively high liquidity, as well as short maturity times. The money market stands out for people searching for borrowing and lending on a short term time period. This can span from multiple days to about a year. Securities in the money market include certificate of deposits, bills, municipal notes, federal funds, and banker's acceptances to name a few. Investments in the money market can also be referred to as cash investments. The reasoning for this is because of their short maturity time.

Participants in the money market include companies and individual investors, Generally, this market is considered relatively safe because the securities are extremely liquid, and they mature quickly. Risks can include defaulting on specific securities though, especially commercial paper.

Cash or Spot Market

The "Cash" or "Spot" Markets are complex and risky. Goods are sold in the cash market, and then delivered as soon as possible. On the other hand, contracts are sold on the spot market and are effective as soon as possible. All prices for goods or contracts are settled, which is different from most other markets where trades are figure out at forward price points. This market is generally reserved for the more experienced traders.

Derivatives Market

A derivative is a specific value 'derived' from an "underlying asset or assets." It is a specific contract where the price is calculated from the market price of the underlying asset. Derivatives can be used very well and are suitable for a risk management program. This market is not used for inexperienced traders.

Some examples of derivatives are swaps, forwards, options, and futures. All of these are extremely complicated, and the underlying strategies used with them are extremely complicated as well. A lot of these derivatives also have an insignificant part in private investing, and are focused more in the 'non-exchange' market.

Forex and the Interbank Market

The Forex market is based around currency trading. It is the largest liquid market in the entire world of trading. The daily average trading is above $1.9 trillion. The Forex includes all of the currencies of the world, and absolutely anyone (whether it be a person, company or country) can trade in this market. The Forex market is open 5 days a week for 24 hours a day. Although there is no centered marketplace for the exchange, currencies are traded around the world at

financial centers. This includes London, New York, Tokyo, Zurich, Frankfurt, Hong Kong, Singapore, Paris, and Sydney.With relative recent advances of the internet, this has opened the Forex market to even the most average investors.

The Interbank market is the system for trading currency. This happens between banks and institutions, not including smaller trading groups or retail investors. Some trading happens from requests through larger customers, but a majority of it happens between banks' and their own accounts.

Primary Markets & Secondary Markets

Primary markets sell securities on an exchange. Companies & governments can obtain specific financing through these securities. Primary markets are also referred to as 'new issue markets', and investments banks will set the initial price and ranges of prices for each security. In the primary markets, investors have the initial chance to buy into securities of a company.

Investors can buy securities or assets from other investors instead of companies on the secondary markets. This is where a majority of trading takes place throughout the day. Here, prices are set through supply and demand.

The OTC Market

The over-the-counter market is a dealer market. These are stocks not traded on a stock exchange, but rather on an over-the-counter bulletin board or pink sheet. Most stocks or securities that pass through here are extremely small, such as penny stocks or tiny companies.

Third and Fourth Markets

The Third and Fourth Markets generally take extremely high volumes of shares for each trade. These transactions are between brokers/dealers and massive institutions. The third market is built up of over-the-counter transactions between the dealers and institutions. The fourth market is comprised of trades between just large institutions. Third and fourth markets were created to safeguard prices on the main exchanges, because such large trades would have a huge impact and increase variability on the primary and secondary markets. (Types Of Financial 2012).

Time Frames

As stated previously, there are many time frames of which a trader can operate. Since Trading Systems are the rules of which Traders follow, each Trading System has a time frame of which it works optimally. The basis for time frame in Trading systems usually follows the bar sizing of which it operates.

For example, flash trading systems would have a bar sizing of micro seconds. Scalp trading systems would have a bar size of seconds. Day trading systems use a bar size of between 5-15 minutes. Swing trading systems tend to have a bar size between an hour to a day. Position trading systems may look to week bars on up to make a decision. When creating an automated system, one can check to see which time frame works best with their particular system. One set of rules may be more apt to trade on a shorter time span. Knowing the frame of which a system works can be the difference between a “winning” system and a “losing” system. Good systems will have a personality trait that corresponds to their trading frequency, and time frame.

Manual Trading versus Automated/Algorithmic Trading

There are two main types of trading systems that investors/traders use. The first is manual trading. This is when the actual trader/investor is responsible for making the entering and exiting decisions in a given trade. While manual traders often use automated programs to gather and display information as well as different technical indicators, at the end of the day, no trade is made without the traders authorization. The second type of trading system is called an automated trading system. This is when the trader writes computer programs which utilize highly complex mathematical models in order to make market transactions. These systems often have strict rules built into the model in order to attempt to place the buy or sell order at the optimal time. These orders are executed entirely by the trading system itself and do not require the traders permission to execute, thus the system will make trades completely on its own.

Fundamental Trading versus Technical Trading

There are two main approaches a trader/investor can use when deciding whether to buy or sell a stock. The first is call technical analysis. Technical analysis is when the trader/investor looks at the price movement of a stock as well as several technical indicators such as regression lines or moving averages and uses this data to predict its future price movements. This is the kind of analysis that automated trading systems use, however it can just as easily be applied to manual trading as well. The second main analysis type is called fundamental analysis. This approach requires looking at economic factors such as balance sheets, cash flow statements, and price to earnings ratios to predict the movement of a stock, This technique is commonly used in manual

trading and is usually avoided by automated trading systems because it is hard to have an automated process do this kind of analysis.

Personalized Objectives

Trading systems should be created with the usage in mind, and a goal. A system could be the very profitable, but it doesn't mean anything if the trader, or their client is not comfortable with using it. For this reason, systems often have personalized objectives.

One of the most common objectives, is a high winning percentage. Often, traders are not comfortable with losing trades. In this case, the amount of wins is more important than the net profit or any other objective. For this case, entries are very important. Profits often get cut by the delay of entry to confirm trade direction. By the time entries are made, a significant portion of the potential has been lost.

Some trust their systems enough so that the only concern is the highest annual return possible. These systems are higher risk, with the associated volatility. In addition to personality type, traders of a high volatility, high return systems should be prepared to lose big. These systems often aren't recommended for individuals nearing retirement, or those dependent on the income. In the long run, these systems have the best return, and often beat the market by significant amounts.

On the other side, some systems aim for a low account draw-down. This type of system often comes into use with small account sizes. For small accounts, a large draw-down could

potentially wipe out the entire account. To avoid large drawdowns, these systems focus on "safe" trades.

Another form of security comes from diversification. Some systems can transfer across markets easily. These are great systems for remaining with the big money and following the business schedule. The system works uniformly, and doesn't rely as heavily on the characteristics of each market.

In some cases, the amount of time required, or the time of day when the system works is very important. Some traders may have times of day which work best. For example, some prefer to trade during the amateur hour (9:30 am to 10:00). This is one of the times when the market is the most volatile. This type of trading requires a system that is most adept at that market characteristic.

Other systems focus on limiting the amount of time the system has a trade on. This type of system's intent is to limit the risk associated with trade time. The longer money is tied up on the market, the more opportunity of the trade going south. Often, small market time systems have very specific entry conditions and easy exit conditions. This form of system differs from the time management system. A time management system only looks at a window of time. Market time limiting systems may be active a large portion of the day, but the large proportion of the time, no trades are active.

Another psychological difficulty for some is holding trades overnight. Some individuals cannot sleep if a trade is on. Obviously, this is an issue. Some systems may make all of their

profits overnight, and not be easily switched to a day only system. Often, switching to a day only system only requires making sure all trades are off at the end of the trading day.

Some individuals may be looking to have the system work well with a small account size. For those individuals, a longer term outlook (position trading) helps to eliminate the account requirements for day trading. In addition to timeframe, small accounts should pay attention to the type of commission and the asset class. Small accounts tend to do best with currencies and equities.

Special consideration should be paid to maximum adverse excursion. Maximum adverse excursion is the largest that the position is down before exit. Even if a system is automated, the trader of the system may step in to cut their losses. While a system may be extremely profitable, at some point in the trade the position may be indicating a loss. The most important part of maximum adverse excursion is the mental strength of the trader. If the trader does not feel comfortable with large swings, then either the system should be edited, put to the side, or tried with another stock.

Often, one of the most important factor for traders is the profit factor. After all, the point of trading is to grow wealth. Profit factor is the ratio of win amount ($) to loss amount ($). A common target for profit factor is 2. This means twice as much money was made during wins than was lost during losses.

The fundamental law of trading system states that profitability is proportional to the profit factor divided by the profit factor plus the ratio of average winning trade to average losing trade. (Profitability and Systematic Trading: A Quantitative Approach by Michael Harris). This comes with the logic that if one cuts losses short and lets wins run (Effectively increasing the

ratio of average win to average loss) then the number of required winning trades decreases. A trading system could only win 20% of the time (Profitability) but be profitable so long as the ratio of the average win to the average loss is large enough. A trader could have 80% of their trades losing, and still make a profit by having much larger wins. This is essentially what a rocket strategy does. It makes all of its money during big wins and stops losses before they get too large.

$$P = \frac{Pf}{P_f + R_{WL}}$$

Trading Systems

System entry rules often follow the pattern of filter, set-up, and then trigger. The filter finds a stock/currency pair etc. that matches the system and is displaying the desired characteristics. The filter can be a screener (like in our case). It can be a one-time event, i.e. checked to see what stocks a system worked on. A set-up is the system identifying a specific scenario is taking place. This can be as simple as the market is trending in a specific direction, or the set-up can have more complex elements to it. The idea is that the stock/currency pair is operating under a specific condition. The trigger is the event that takes a position. Set-ups and triggers do not have to be solely based on the value of a stock. They can be information gathered from anywhere. Volume is a common setup, as well as market (index) direction. Many experiment with price patterns such as number of red bars before a green bar. One key thing is to

limit the specificity of the entry requirement. Too specific, and barely any trades will be taken, severely limiting profit.

There are two main ways to exit a trade. The trader can either lose money or gain money when they exit. The specific names for these exits are either “take-profit” and “stop-loss”, depending on the circumstances.

Sop losses allow traders to exit at a specified point or price. When this point is reached, the stop-loss will execute and the order will be made to sell. They are set higher than the asking price while buying, or below the order to sell. Good ‘till cancelled stop orders stands until any type of execution happens, or until it is cancelled manually. With day orders, the stop-loss will expire after a full day of trading. The last type is the trailing stop- this stop-loss tracks a specific distance away from the going market price, but will never move in the downward direction.

Take-profits are the second type of orders. These are limit orders that are much similar to stop-losses. They are basically turned into market orders to sell after that specified point is hit. There are not trailing points though, and the exit is always higher than the market price at that point in time.

To develop a good exit strategy, traders must think of multiple points. There are the time-span of the trade, the amount of risk they should be willing to take, and at what point do they want to get out of the market. Overall, exit strategies are extremely important for the success of a trading session. (Kuepper 2004).

Order Types

There are many different Order Types, all having their own unique benefits and drawbacks. When a trade order is place, a broker will obey this order and either enter or exit a position. From a superficial standpoint, these order can seem extremely simple. In reality though, it is a bit more complication. The instant "push a button to buy and sell" method is inefficient, and can expose traders to specific financial risks. They can experience slippage from trades with stop-losses applied, which is the difference between expected and actual price. Having different order types allows traders to choose an exact price for every different trade, which is therefore much for effective than dealing with slippage. By utilizing stop-losses, traders can have extra security in fast-moving markets.

Another concept to note are long and short trades. Trades can go in two directions, all being dependent of the thought if the market is going up or down.

Long trades are more of a traditional method of trading. With a long trade, the trader is expecting or intending to profit from a market that is rising. These can be passed through all brokers, and any losses incurred from a long trade are limited. No matter what price that trade starts at, the lowest the price can go is $0, so losses can be large but still limited no matter what.

Short trades are a less traditional method of trading, and a bit riskier. They are entered with an expectation or intention of making money off a falling market. Basically, the trader puts in an order to borrow a financial instrument from a broker. When the price of the financial instrument hits the target price, the trader proceeds to buy the shares/contracts back to replace the borrowed instrument from the broker. When the price drops, the trade is most likely a profitable one (commission and slippage need to be taken into account). When the price rises, this is

considered a loss. When a trader deals with short trading, this requires a margin account in order for the trader to borrow these financial instruments. Not all of these instruments can go in short, and also different brokers will offer different instruments for short trading.

Having these two types of trades are important because traders can utilize the markets when they are rising and falling. Shorts are especially important because traders can have access to other methods of profitability. Short trades are relatively risky though (especially for the average investor) because of the possibility of unlimited losses. When a short trade decreases in value when the market rises, the market could keep rising (and losses would continue to build up). Utilizing stop losses is important here, and traders can manage their risk relatively well with this tool. (Folger 2012).

Market Order

A market order is the simplest of all the trade orders. It basically tells the broker to buy or sell at the most optimal price that is available at the current moment. Technical trading interfaces generally have buttons that say 'buy' or 'sell', which simplifies these trades to a point. Usually, these orders will be executed as soon as possible.

Using a market order guarantees that the trader will get a trade fully filled at a point. If a trader urgently needs to enter or exit a trade, they will generally use a market order, as it is a proven method for success in entering and exiting trades. The drawbacks of a market order is that it doesn't actually guarantee a price. There is not much in the way of precision when entering an order, which can lead to large losses due to slippage. The best way to curb losses in a market order is to place order in markets that are very liquid.

Overall, when a market order is placed to buy, it is granted at the ask price. When it is placed to be sold, it is filled at a bid price. The previously traded price is not always the price that the market order will be placed at though, especially in extremely fast moving markets, or ones that are rarely traded.

If a trader placed a long market order for 500 shares of Apple at $10.00 for each share, if orders that were placed before the trader were fulfilled, the trader's market order has the possibility of being completed at higher price. In general, a market order never guarantees a specific price, but will always guaranteed that the order is placed.

Stop Order

Stop orders are used when a trader wants to see which direction the market is going before initiating a trade. Stop orders only become active after a 'stop level' has been hit, which is a specific price level. Stop orders are somewhat opposite of limit orders. When a stop order is placed, the buy is placed above the market, while the sell is placed below the market. When the specified limit of the stop level has been hit, the order is then converted to whichever order the trader specified (either a market order or a limit order). Stop orders are basically triggers for the other types of orders.

Stop orders can also be broken down into stop market or stop limit orders. When a stop level has been hit, the stop market order shoots a market order over to the market. Similarly, when the stop limit order will shoot over a limit order to be fulfilled. Buy-stop orders send the message to enter a market or limit order only when the prices reaches the specified stop level. This allows individuals to challenge the price until it hits the price point that they want. If, or

when the price hits the stop level, this gives the trader extra confirmation that the market's direction is going where it was previously predicted.

Stop orders have their common applications too. For the most part, traders generally utilize this for setting stop-losses for trades. Stop-loss orders are set above a trader's risk level where they would not place more money. If it is a long order, the first stop-loss is below the entry trade level, which would allow extra security in case the market goes down. If it is a short order, the first stop-loss is placed higher than the entry point of the trade in the event that the market goes up.

The other common application is a trailing stop. This is a flexible order that tracks a price in order to maximize profit. This stop increases in increments for long trades, and effectively tracks a price as it increases. In short trades, it lessens as the price decreases. Traders themselves specify the weighting of the stop. This weighting takes multiple parameters, including percent or the amount in dollars, as well as the difference between the price that is currently seen and the level of the trailing stop. If a trailing stop is very close together, it will track the price much more closely. If the trailing stop is very wide, it will be farther away from the price. (Folger 2012).

Limit Order

Contrary to Market Orders which guarantee a fill, a Limit Order guarantees a specific price. Limit orders are orders in which you can buy and sell at a price greater than or equal to what you specificy.

There are also specific types of limit orders, which each have their own unique properties. A buy limit order (or also referred to as a limit order to buy) is focused purely on buying. The rule for this order is that a trade can only be initiated at the price specified by the trader or lowers. On the other hand, a sell limit order is only completed at the price chosen by the trader or higher. This is different from a market order where the trader simply lets go of control and the market chooses the prices.

Limit orders are advantageous because they prevent negative slippage, which could really hurt traders trading on Market Orders. The downside is that it never guarantees a fill every time. Overall, limit orders are only able to be filled when the price reaches the traders desired trading prices. Specific trading opportunities might be lost if before being filled, the price moves significantly away from the limit order price. Also, if there are not enough people in the market buying and selling at specified price points, the limit order might not be filled even if it meets all the requirements otherwise.

Limit orders are much more precise than Market Orders, but entering correctly is important. If entered correctly, this is where precision is the highest. If one is utilizing a buy limit order, they need to choose a price equal to or less than the current bid. If one is utilizing a sell limit order, they need to choose a price greater than or equal to the market asking price.

Traders can utilize limit orders in many different ways, and they’re useful for improving price and exploiting pullbacks. Being on the correct side of the market is especially advantageous. Limit orders ultimately will allow the trader to receive a trade at a price they want or better, whatever their scenario is, if they've played their cards right.

Other Types of Orders

There are other types of orders that traders can utilize. The most common other types are conditional orders and duration orders. Conditional orders are orders placed which will either be forwarded or canceled if a set of rules that the traders has applied has been met.

The first type of conditional orders are one-cancels-the-other orders (OCO). OCO orders let traders place multiple orders at once. When a singular order is filled, the rest of the orders placed are then cancelled. An OCO order has the same function as separate stop orders, limit orders, and closing the last remaining order.

Another type of order is the order-sends-order, or OSO. This is another type of automated order which sends orders once one is filled. OSO is made up of one primary order that can pass one or multiple secondary orders after the first order is fulfilled. OSOs can be used with OCOs to optimize the trading process, while making it less work for the trader to make and execute trades. (Folger 2012).

Smoothies

A smoothie is considered a stock that has a high ATR (average true range), a relatively high moving price, and high volume. Average true (daily) range is defined as the daily high minus the daily low price of a stock which is then average over a specified number of days.

Stock prices need to be moving at a decent velocity so that trade profits can be maximized. The velocity can be found in multiple different programs and scanners, such as HotScan. As for high volume, these stocks should have 800,000 + units of volume, and should have a small spread. As a decent rule of thumb for trading with smoothies, for every 1000 share

that you are trading, the stock itself should be trading about 1 million shares every pre trading day, averaged out. This allows for a lot of flexibility with entries and exits.

Chapter 4: Analysing Trading Systems

TradeStation Performance Summary Report

All of the automated trading in this project was done using TradeStation. One of options in TradeStation is a strategy Performance Report. It contains many of the statistics that a trader would like to know about their strategy. The first four lines have correspond to profit and loss. The first line is the total net profit, this is the gross profit (total of all wins) minus gross loss (total of all losses), the subsequent two lines. Profit factor (stated earlier as P_f) is quite simply the profit divided by the loss. It is important to keep in mind time frame and account size while looking at these statistics. In the following case the account size was $100,000 and the time interval was 4 months. This factor can drastically change one's perspective on the stats. In this case, the strategy (although a net profit) was not performing. Profit factor can be a great indicator of the relative strength or weakness of a strategy. Many traders aim for a profit factor of 2 or higher. This strategy had a profit factor barely above 1.

Figure 4.1- TradeStation Performance Summary

TradeStation Performance Summary Expand

	All Trades	Long Trades	Short Trades
Total Net Profit	$2,047.54	$2,047.54	$0.00
Gross Profit	$10,351.00	$10,351.00	$0.00
Gross Loss	($8,303.46)	($8,303.46)	$0.00
Profit Factor	1.25	1.25	n/a

The next 5 lines deal with the number of trades. They state the total number of trades taken, the percent of those trades that made a profit, the number of winning trades, the number of losing trades and the number of even trades. The total number of trades can be quite useful when determining the productivity with an account size. Small account sizes should the number of

round trips (Trades) in order to limit commission costs. Percent profitable (listed earlier as R_{WL} or win-loss rate) helps to determine if the system is psychologically tradable.

Figure 4.2- TradeStation Performance Summary (cont.)

Total Number of Trades	91	91	0
Percent Profitable	45.05%	45.05%	0.00%
Winning Trades	41	41	0
Losing Trades	50	50	0
Even Trades	0	0	0

The following 6 lines show the average and outlying trade profits and losses. In general, average profit should be much higher than average loss. This follows the perspective of letting wins run and cutting losses. Even with sub 50% win rates (like in this case), a system can still be profitable so long as the average win is larger than the average loss. Optimally, one would like to see the ratio of average win to average loss be as high as possible. Although, too high and it might indicate an overfitting or limited data.

Figure 4.3- TradeStation Performance Summary (cont.)

Avg. Trade Net Profit	$22.50	$22.50	$0.00
Avg. Winning Trade	$252.46	$252.46	$0.00
Avg. Losing Trade	($166.07)	($166.07)	$0.00
Ratio Avg. Win:Avg. Loss	1.52	1.52	n/a
Largest Winning Trade	$937.38	$937.38	$0.00
Largest Losing Trade	($669.48)	($669.48)	$0.00

The next grouping shows the patterns of wins and losses. These statistics can give the trader an indication of what to expect out of the system moving forward. It states the maximum winning and losing trades in a row. This is valuable, because often a trader's intuition is to remove a system after too many losses assuming it is broken. Sometimes, the system may just

have a certain characteristic of large groups of losses/wins. This is another question of mental strength. A very profitable system may not be a good suite for a particular individual if there is a trend of many losses in a row. Another statistic shown is the average number of bars in a winning trade or losing trade. This is important for two reasons; the system may be holding onto losing trades for too long (as is the case with this example), and the system could be functioning normally and the trader should know how long to expect to see a losing trade on. Some traders may have the tendency to want to extend the length of a losing trade (get back to black), and others may want to cut a trade too short and effectively make the damage the trade even more.

Figure 4.4- TradeStation Performance Summary (cont.)

Max. Consecutive Winning Trades	7	7	0
Max. Consecutive Losing Trades	9	9	0
Avg. Bars in Winning Trades	18.83	18.83	0.00
Avg. Bars in Losing Trades	19.10	19.10	0.00
Avg. Bars in Even Trades	0.00	0.00	0.00

The block following has three lines. All of the lines have to deal with account sizing. Some institutions charge commissions based on number of shares rather than number of trades. For these cases the total shares held can be particularly important. It can let the system builder know if there needs to be adjustments made. Account size required shows the size of the smallest account that could trade the strategy. This amount is misleading, because it assumes buying on margin and no extra account reserves.

Figure 4.5- TradeStation Performance Summary (cont.)

Max. Shares/Contracts Held	5,780	5,780	0
Total Shares/Contracts Held	347,052	347,052	0
Account Size Required	$1,523.72	$1,523.72	$0.00

The following lines are some of the most important for understanding the risk and reward of a particular system. The first grouping (top two lines) state the return on investment capital (percent gain on account size) and the annual rate of return (return on investment if the system traded one full year). The return retracement ratio is the reward for the risk. The higher the ratio the more return for the risk. RINA index factors in profit and time in the market. The higher the RINA index, the more "efficient" the strategy. Trading period is how long of a window the strategy report has. Percent of time in the market shows the percent of the trading period that a trade is actively on. The lower the percent time in the market, the higher the RINA index. Max equity run-up shows the maximum profit the account had during the window.

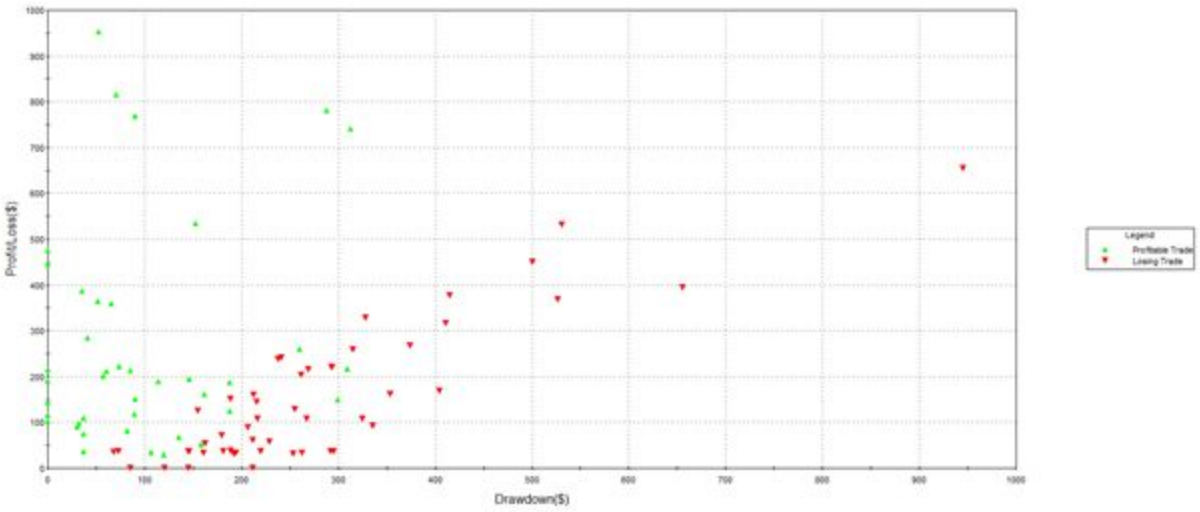
Figure 4.6- TradeStation Performance Summary (cont.)

Return on Initial Capital	2.05%
Annual Rate of Return	6.85%
Return Retracement Ratio	38.61
RINA Index	39.31
Trading Period	3 Months, 16 days, 2 hours, 30 Minutes
Percent of Time in the Market	28.62%
Max. Equity Run-up	$4,371.92

Another valuable resource in the performance report is the graph of maximum adverse excursion. It is a visual representation of the worst case scenario during the life of a trade. A trade could have ended up with a significant profit, but it could have had to travel through a loss in order to get there. It is another question of mental strength. It is up to the trader to decide whether they could let the trade run its path or if they would intervene. Another possibility one can find in the maximum adverse excursion plot is whether the system is just randomly traveling through losses. One can find issues with stop losses in the plot. For example, in the plot below,

the profits show a characteristic of not having drawdowns more than $300, but the losses exceed that mark significantly. One could simply add a stop loss at $300 to cut those losses before they got any worse.

Figure 4.6- Drawdown vs. Profit Plot



Optimization

Optimizing serves many purposes. As the name suggests, optimization finds the best scenario for a particular statistic. The most common statistic to be optimized is net profit. Although, other statistics can be optimized as well. Some personalized objectives can be optimized for. For example, one can optimize the percentage win rate. There are two basic types of optimization: exhaustive and genetic. Exhaustive tests go through all of the options given by the user. In effect, it is exhausting that list. The output given by the function is the optimal out of all of the options. Genetic algorithms have multiple passes and "breed" the best solutions from one pass to create the selection for the next pass. The advantage of an exhaustive test is the fact that it will test all of the inputs, and all combinations of those inputs. The advantage to genetic

algorithms is the speed. The larger the number of combinations gets, the more of an advantage that the genetic algorithm has.

While optimizing, one should keep separate periods of time to optimize over (train) and another to test out the new values. This helps to remove over fitting the system. Over fitting happens when the inputs are set too closely to the data range and may display good statistics for that section, but when tested on another period, the system may perform poorly.

Expectancy, Expectunity, and System Quality

Systems are often analyzed using expectancy, expectunity, and system quality. Expectancy is the "casino" nature of the system. Basically, it is the win rate multiplied by the average win. Expectunity is a combination of expectancy and opportunity. One can have a system that wins, and wins big, but if the trade opportunity is rare, then ultimately, the system may not be the best. Quite simply, expectunity is expectancy multiplied by the number of opportunities to trade. (Diary of a Perth 2012). System quality is similar to expectunity. The system quality number (often written as SQN) is the square root of the number of trade opportunities multiplied by the expectancy and divided by the standard deviation of the profit/loss R multiples.

SQN = root(n) * expectancy / stdev(R)

root(n) – the square root of the number of all trades

expectancy – as shown above and measured in R multiples

stdev(R) – the standard deviation of your profit/loss R multiples

(Scott 2014).

The basic principle behind these statistics is whether or not the system as an "edge." Most of the time, this means that it is doing something that not every other trader is doing, or is taking advantage of the decision processes of everyone else's relative homogeneity. Understanding how a system is different, and where the advantage comes from is very important. It can help the trader decide whether the system should be scrapped, put aside, left in, or made the star. A system could be underperforming due to the nature of the market, or due to a fundamental change to the mindsets of traders.

Chapter 5: Literature Review

Precision Trading Systems

For most of our systems, we followed the logic of Mel Raiman's Precision Trading System. Basically, Precision Trading Systems (PTS) follow the guide of the general long term trend. Their general logic follows: scan to find a few stocks with the "right" character, check the long term regression for which direction the trade will be going (long or short), use medium and short bars to determine when to enter a trade. A typically day trading light setup is shown below.

The real finesse of the system is the exact timing of the trade. Typical setups and triggers involve moving averages, regression lines, and standard errors based on those regression lines. Often, PTS systems treat the regression lines as a form of support and resistance. When the trend is upward and the price hits the bottom regression line, then it is time to buy, and when the price hit the top it is time to sell. The opposite is true when shorting.

Neural Nets

Neural networks are a machine learning algorithm that have been becoming increasingly popular lately with the increase in big data. This increase in popularity has caught the eye of many traders who have then tried to apply them to trading with mixed success.Software that will generate a neural network for you has also begun to pop up. This software has allowed the average investor with no coding experience to access the power of neural networks. Even some of the large hedge funds have begun experimenting with neural networks as well. Since most of the neural networks used to trade are either generated by software or manually written by

someone who is very knowledgeable in the computer science field it is hard to find out exactly what is going into a lot of the neural networks that people are using to trade since all the software's code is kept secret and no person who wrote their own successful neural network is going to share what they put into it. However it's easy to assume that investors have tried all kinds of different neural networks with varying node configurations and inputs. Now with all this neural network technology already out there it's important to realize why our group decided to create our own neural network instead of using software to create one for us. Basically,, creating our own neural network allowed us to give it the customizability we wanted that we would have never been able to achieve through the use of any kind of existing software.

Scanners

While our team used created a stock screener, we are not the first to traders to create or use a stock screener in our trading strategy. Currently, there are numerous free stock screeners available online as well as ones that you need to purchase.These screeners usually have scanning criteria that falls into three categories.The first category is descriptive. This category contains filters that generally describe the stock in general such as such as exchange, market cap, volume, sector, etc. The next category that these screeners usually have is fundamental. This category contains filters that have to do with the stocks overall performance such as P/E ratio, gross margin, EPS growth, etc. Then the final category is technical. This category lets you filter using different basic technical indicators such as moving averages or beta values. Typically these screeners will allow you to combine filters from all three of these categories in your screen. Now because screeners allow you to sort through thousands of stocks to find ones that meet a certain

criteria, they are often used in trading strategies to find stocks that met the traders stock criteria. There are many examples of strategies that use stock screeners such as the popular PTS strategy which uses a stock screener to find stocks know as smoothies to trade for the day. While, our strategy did not invent the idea of a stock screener, what we did do was create one that allows us to screen for things that no other free screener is capable of doing. This includes thing such as correlation with market, regression line slope and r squared value, opening percent, etc. Building on the already existing ideas of stock screeners, we were able to design our own screener that we could then use to for the criteria we wanted which if we had just used existing ones we would not have been able to do.

Chapter 6: Neural Net, Screener, and Regression Analyzer Build

Neural Net Creation

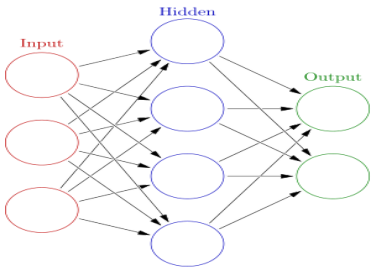
An artificial neural network is a machine learning model that can be used to used to estimate functions that depend on a large number of inputs that are generally unknown. The idea of an artificial neural network was based on the biological neural networks that are used in our brains to pass information between neurons. Artificial neural networks are represented as a series of interconnected neurons which exchange information with each other. Each of these connections carries a weight which determines how strong the connection is. These weights can be adjusted which allows the neural network to learn based on the information it receives.

Artificial neural networks have a large variety of applications, but in general they are mainly used for pattern recognition. For example common applications for neural networks are facial or handwriting recognition due to the network's ability to find patterns that are not noticeable to the human eye. This being the case, neural networks can be a very useful tool in the world of investing as well. When looking at a stock chart the human eye alone can't determine any particular pattern, but that doesn't mean a pattern isn't there. Through the use of an artificial neural network it is possible find at least a general pattern in a given stock which can then be used to extrapolate future prices of the stock.

While there are many different types of neural networks, I decided to use a simple multilayer perceptron network as my model. A multilayer perceptron network consists of three layers of neurons or nodes as they are also called. These layers are called the input, hidden and

output layer. As seen in the image below, each layer consists of one or more neurons that is connected to the neurons in the layer ahead of it through a series of weights.

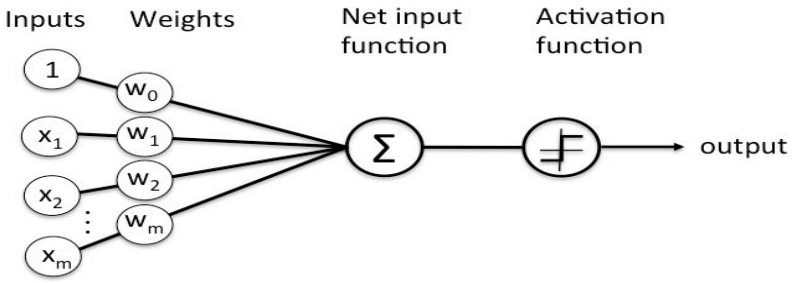
Figure 6.1- Neural Net Example



In this kind of network the information flows from input to output or from left to right. The input nodes are considered to be passive nodes. This is because they don't modify the data. They simply receive a single piece of data and duplicate that value to its multiple outputs. The other two layers are made up of active nodes. These nodes actively modify the data using some function. Each value from the input layer is sent to all the hidden nodes. The values entering the hidden nodes are multiplied by weights. These weights are simply predetermined numbers that are stored in the network. Since a single hidden node will receive multiple weighted values, these values are summed to produce a single value which then becomes the value of that hidden node. Before the values of the hidden nodes are passed to the output layer they are first sent through what's known as the activation function. This function is responsible for taking the linear net input data and applying a nonlinear function to it which essentially squashes the data between two values. Common activation functions are a sigmoid function or hyperbolic tangent function which can be combined with a stepper function which produces an output of the activation function if it the activation functions output value exceeds a given threshold value and if it does

not, it produces zero. However, combining the two /functions is optional. Optionally, the activation function can be applied with or without the stepper function as the sole input. The values of the hidden nodes are then duplicated and sent through a second series of weights to the output nodes. Similar to the hidden nodes, each output node will receive multiple weighted values so these values are are again summed to produce a single value and run through an activation function whose result then becomes the value of that output node. The below diagram illustrates this process for a single node.

Figure 6.2- Single Node Process Example



In order for a neural network to produce an accurate output it is necessary to train the network before using it. To start the training process the weights are first initialized to random values usually between -.5 and .5. From there a set of data known as the training data is run through the network. The network is given the correct output for the training data so it is able to evaluate how close the output it produced is to the desired output and adjust the weights accordingly to decrease the produced valued to bring it closer to the desired value. This process is repeated thousands of times until the produced network values are similar to the desired values of the training data. This method of training is called backpropagation. However in order

for this training strategy to work, there must be a single value that can represent how well the neural network is currently doing so the network is able to tell if it is improving. This value is called the error sum. The error sum will start off high but gradually decrease as the network iterates through the training process. Once the error sum reaches a certain set value you know the network is trained properly.

Understanding the backpropagation process is key to understanding how neural networks learn. The first part of backpropagation is to feed training data forward through the network. From there we can use what is called gradient descent in which we can change the weights in proportion to the negative of an error derivative with respect to each weight. To do this we first need to get the total error which we can calculate by using

$$E_{total} = \sum_{i} E_i$$

Where E_i is the error of the i^{th} output node.

To get the error for an individual output node we can do

$$E_i = \frac{1}{2}(target\ output - actual\ output)^2$$

The reason there is more than one error output is because the training data will have more than one set of data in it and each of those sets of data will produce an output. Once the total

error is calculated, we are then able to begin determining how much each weight in the network contributed to that error. To do this for the output layer we can use the formula

$$\frac{\partial E}{\partial w_{ji}} = -(t_j - y_j)\, g'(h_j) x_i$$

Where:

w_{ji} = an individual weight

t_j is the desired value

y_j is the actual value

g(x) is the activation function

h_j is the weighted sum of the neuron's input

x_i is the ith input

We can then multiply $\frac{\partial E_{total}}{\partial w_{ji}}$ by a small constant known as the learning rate or α in order to see how much to adjust w_{ji}. Thus

$$\Delta w_{ji} = \alpha\left(\frac{\partial E_{total}}{\partial w_{ji}}\right).$$

We then pass the above info to the hidden layer which allows us to calculate how much to adjust the weights connecting the input nodes to the hidden nodes. For the hidden layer we still need to calculate $\alpha\left(\frac{\partial E_{total}}{\partial w_{ji}}\right)$ for each weight but since there is no target value to compare

against the actual value of the hidden node we can't find Δw_{ji} the same way. Instead we must use the formula

$$\frac{\partial E_{total}}{\partial w_{ji}} = \frac{\partial E_{total}}{\partial out_i} * \frac{\partial out_i}{\partial net_i} * \frac{\partial net_i}{\partial w_{ji}}$$

where:

out_i = the output of the i^{th} hidden node

Net_i = the the sum of the input weights connecting to hidden node h_i

$$\frac{\partial E_{total}}{\partial out_i} = \sum_j \frac{\partial E_j}{\partial out_i}$$

where:

E_j is the error for the j^{th} output node

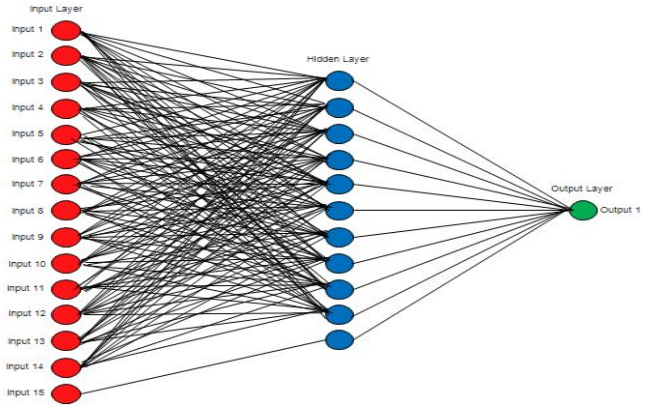
This will produce the amount to adjust the weights connecting the input and hidden nodes. We then repeat this process for every training iteration until the network's total error has decreased to our desired level.

Neural Network

The purpose of the neural network I created is to predict the high or low of the next trading day for a given stock. For my neural network I used 15 input nodes, 11 hidden nodes, and 1 output node. The first 14 input node take an input of the highest price of the day for the previous 14 trading days. The 15th input node takes the most recent closing price. The first 14

input nodes are connected to each of the first 10 hidden nodes while the 15th input node connects to the 11th hidden node.From there all 11 hidden nodes connect to the one output node. This design can be see below

Figure 6.3- Input, Hidden and Output Nodes



All data fed into the input nodes is first normalized between 0 and 1 and for my activation function I used a sigmoid function which is defined as

$$S(t) = \frac{1}{1+e^{-t}}$$

While this was my final design that I decided worked the best, there was a lot of testing of different designs to get to this final one.

Once I knew I wanted to make a neural network to predict the highest price of the next day, I had several key questions I needed to answer. The first being, what data to feed into the input nodes and on top of that how many input nodes did I want to use. After some thought, I decided I wanted to use past highest prices as my input. Since neural networks are so good at

pattern recognition I figured it would definitely be able to find a pattern in daily highest prices. Once I decided what data I was going to feed to the inputs I still had to figure out how many days back I wanted to go. I decided to solve this problem by testing different input amounts and analysing the results. For my first design I decided to go with 14 input neurons which means I would be using the previous 14 highs of the day to predict the highest price of the next day. I chose 14 as my initial input amount because it seemed like a good number of nodes to start with. Once I knew the amount of input nodes I wanted to start with I could then decide the amount of hidden nodes I wanted to use. In neural networks, it is recommended to use around 3/4ths the amount of input nodes as hidden nodes so I figured I would start out using 10 hidden nodes. Then since I only needed one output value which would be the predicted highest price of the next day, I knew I only needed one output node. From there the thing I needed to figure out before I could begin testing was what activation function I wanted to use. I decided to start off using a hyperbolic tangent function with no stepper function attached to it since hyperbolic tangent functions are common activation functions in neural networks. The hyperbolic tangent function is defined as

$$tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$

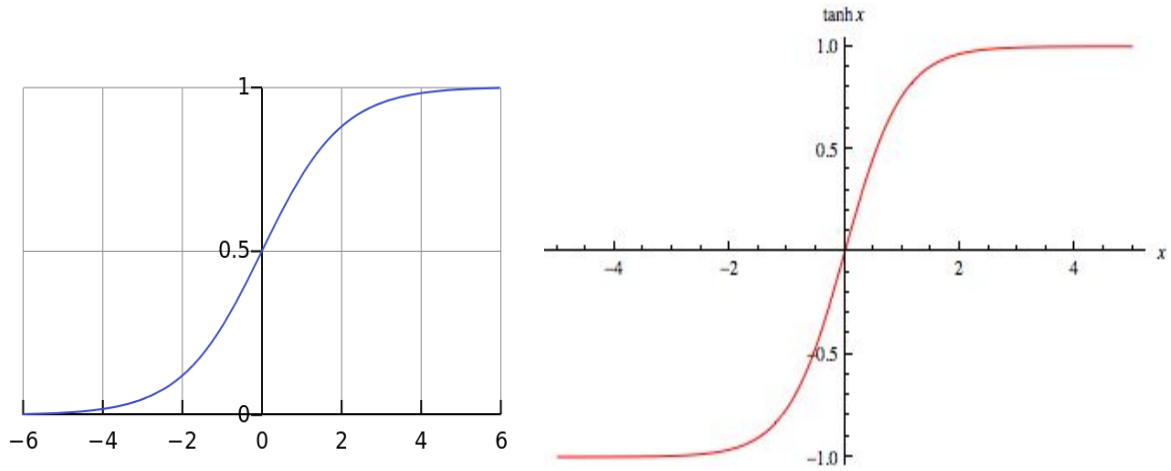
Now that I decided the basic design of my initial network all I needed to do was decide how I wanted to train it and then I could begin the testing process. For training data I decided to start with using the previous 200 days as training data knowing I could always change it easily later if it was too much or too little training data. For the actual training process I decided to have my network start at the 200th day back and use the 200th-186th day as the 14 inputs and then use the 185th day as the desired output. This would be the first line of training data. The second line

of training data would then be 199th-185th as input data and the 184th as the desired output. The process of making lines of training data would continue until all 200 days had been used. The network would iterate through the training data 8000 time before moving on to the actual prediction.

Now that I established a training process as well as the actual design of my network I was ready to begin testing. To test the network I wrote a script that would test the network on each trading day going a year back, retraining after each new day, and compare the actual output against the expected output of each day. Doing this I would able to determine how accurate the network was as well as what changes I might need to make. I decided to use the S&P 500 as the test stock since it is a major benchmark for the market as a whole. While the initial results weren't great, the network was accurate enough to show that with some minor improvements there was some real potential to perform very well.The network had an average error of 1.52% with a standard deviation of .53% after removing major outliers. In order to try and improve these results I decided to modify the network and try it with different amounts of training days. After trying training day amounts ranging from 50 to 800 I found that using 300 days produced the optimal results. From there I decided to modify the amount of inputs to see if I could get better results that way. I tried testing the network with 5 inputs and then again with 30. What I found was that the 30 input version didn't move the predicted high enough from the previous days high and that 5 day version moved it way too much thus both were not as accurate as the 14 day neural network so I eventually decided to stick with the 14 day input version since it performed the best. The next change I tried was switching the activation function from a

hyperbolic tangent to a sigmoid function in hopes of better results. The main difference between these two functions is that a sigmoid function scales values between 0 and 1 while a hyperbolic tangent function scales them between -1 and 1. However it is important to remember with both of these functions that the transformation is not linear. The graphs of these functions can be seen below where the left image is a sigmoid function and the right is a hyperbolic tangent function.

Figure 6.4- Sigmoid Function (Left), Hyperbolic Tangent Function (Right)



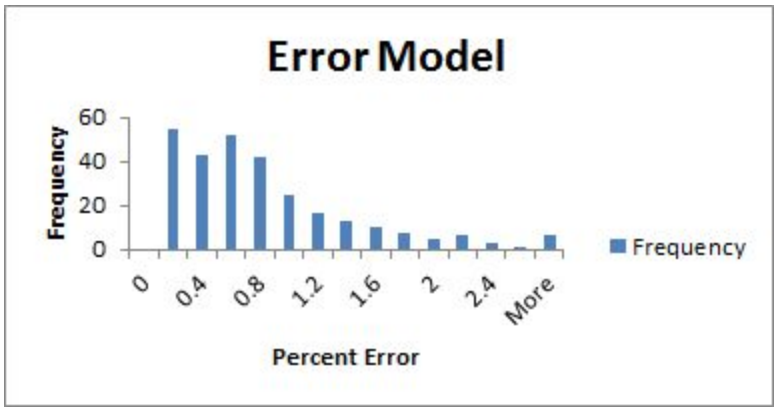
After testing with the sigmoid function instead, the results were significantly better so I decided to use that instead for my activation function. The final change I decided to add to the network was an 11th input node which receives the close of the previous day as an input and then connects to a newly added 11th hidden node which then connect to the output node. I added this because I was noticing that the network tended to have issues predicting days in which the S&P 500 hit its high of the day and then dropped significantly before closing. Without adding in the closing price the network has no idea this significant drop occurred and in most situations overpredicted the high of the next day because of this. I decided to keep the 15th input node from connecting to the initial 10 hidden nodes was because I wanted the network to factor in the

closing price separately. I decided to test the network with the closing price connected to all the other hidden nodes as well just incase, but as expected the network when the closing price was kept separate performed better. This network also performed much better than the network without the closing price factored in so I decided to keep the change.

After making the above changes the network preformed much better than initially. The next step was to create an error model. Knowing the error of the network is key to trading its results. The first step to doing this was stripping any major outliers from the data. I did this because I felt that any day in which it was drastically off was most likely caused by major news dramatically affecting the markets movement on that day. Since that day would not have normal market conditions I wouldn't expect the market to follow the pattern my network was predicting so I don't want those days to count against its error rate. After the outliers were removed I plotted the error of each test day going a year back and got the following results. The formula to get the error for a given day was

$$e(i)\ 100 * \left| \frac{actual - expected}{expected} \right|$$

Figure 6.5- Error Model



Mean 0.736777927%

Standard Error 0.037942656%

Median 0.575188582%

Mode #N/A

Standard Deviation 0.638294273%

Sample Variance 0.407419579%

Kurtosis 3.022192117

Skewness 1.598270844

Range 3.59910148%

Minimum 0.002374053%

σ Maximum 3.601475532%

Count 284

Looking at the error model above, it is easy to see that the results are normally distributed, While the error graph above is only one half of a normal curve, we can still apply normal distribution to it as long as we adjust the probability density function to account for the fact that it is a folded normal curve. A variable has a folded normal distribution if the variable is the absolute value of a variable with normal distribution which is exactly the case here. Therefor the new probability density function or PDF is

$$p(x) = \sqrt{\frac{2}{\pi\sigma^2}} e^{\frac{(x^2+\mu^2)^2}{2\sigma^2}} * cosh(\frac{\mu x}{\sigma^2})$$

σ=standard deviation(.638)

μ=mean(.737)

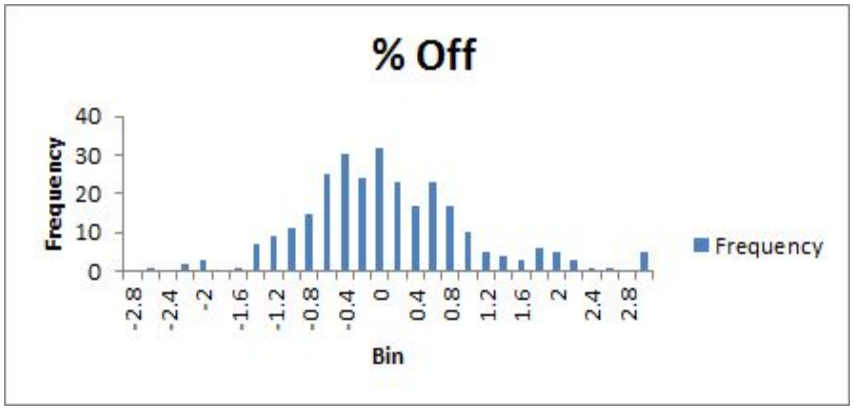
This PDF function will only give us the percent chance of the neural networks error being a certain amount. While this is useful it would be much better if we could instead get the probability that the network's error will be within a certain range. To do this we can add the integral form a to b to our PDF equation. This will allow us to get a specified error range or area under the probability curve. The new PDF equation looks like the following

$$p(x) = \int_a^b \sqrt{\frac{2}{\pi\sigma^2}} e^{\frac{(x^2+\mu^2)^2}{2\sigma^2}} * cosh(\frac{\mu x}{\sigma^2})$$

Using this probability density function we can estimate that around 34% of the time the neural network will be less than .45% off, 68% of the time it will be less than .92% off and 95% of the time the prediction will less than 1.61% off. Knowing the neural networks error rate can be very useful when trading based on its prediction, While this error may be useful, knowing just the

percent chance of the neural networks error being within a certain range is hard to apply to trading. For example if the neural network says the high of the day is going to be $100 and the current price is $98 and I want to know if I should buy the stock. All this current error rate will tell me is that there is let's say a 68% chance the prediction will be less than $0.64 off. While this allows you to see that you have a good chance of making money if you buy at $98, there is no way to take this current error and say that there is an x% chance of making money off of this trade. However if we were to say that there is an x% chance the high of the day will be above a certain price. Now we can gage exactly how much risk we are taking on the trade. For example if we are able to say that there is a 70% chance the high of the day is going to be greater than $98. If we then buy at $98 we know we have a 70% chance of making money off the trade. Lucky measuring error in this way is in fact possible. If we remove the absolute value from our error formula it becomes $e(i)\ 100 * \frac{actual-expected}{expected}$. Now when we graph model our error instead of a folded normal curve we get a regular normal curve as seen below.

Figure 6.6- % Off (Regular Normal Curve)



Mean -0.016577608

Standard Error 0.057996907

Median -0.112700136

Mode #N/A

Standard Deviation 0.975658993

Sample Variance 0.951910471

Kurtosis 1.322713639

Skewness 0.683916466

Range 6.382783052

Minimum -2.78130752

Maximum 3.601475532

Sum -4.691462923

Count 284

Now instead of measuring percent error we are measuring the the percent error along with whether it overestimated or underestimated. Using the the PDF formula for standard normal distribution along and adding an intergral to it like we get the following formula

$$P(x) = \int_{a}^{b} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Using this new error formula we are able to generate the below probabilities.

p=predicted high of day

a=actual high of day

σ=standard deviation (.98) note: initially in % form so need to convert to dollars before using

15.5% chance $a<p-\sigma$

30.8% chance $a<p-\frac{1}{2}\sigma$

50.7% chance $a<p$

70.3% chance $a<p+\frac{1}{2}\sigma$

85.3% chance $a<p+\sigma$

Or

14.8% chance $a>p+\sigma$

29.7% chance $a>p+\frac{1}{2}\sigma$

49.3% chance $a>p$

69.1% chance $a>p-\frac{1}{2}\sigma$

84.5% chance $a > p - \sigma$

Now going back to our example. If a stock is at $98 and the neural network says the high if the day is going to be $100. Using this new error equation we know that there is a 69.1% chance that the the actual high of the day is going to be greater than the predicted value minus .49% or in this case $0.49. This means that there is a 69.1% chance that the high of the day will be above $99.51. Using the error formula we are also able to tell that there is a 98.3% chance the high of the day will be above $98. This means we effectively have a 98.3% chance of making money off the trade if we buy at $98. As you can see being able to determine this error is very useful and by using it we are able to shift the percent chance of making a winning trade in our favor by only buying stocks in which we are comfortable in the percent chance of making money.

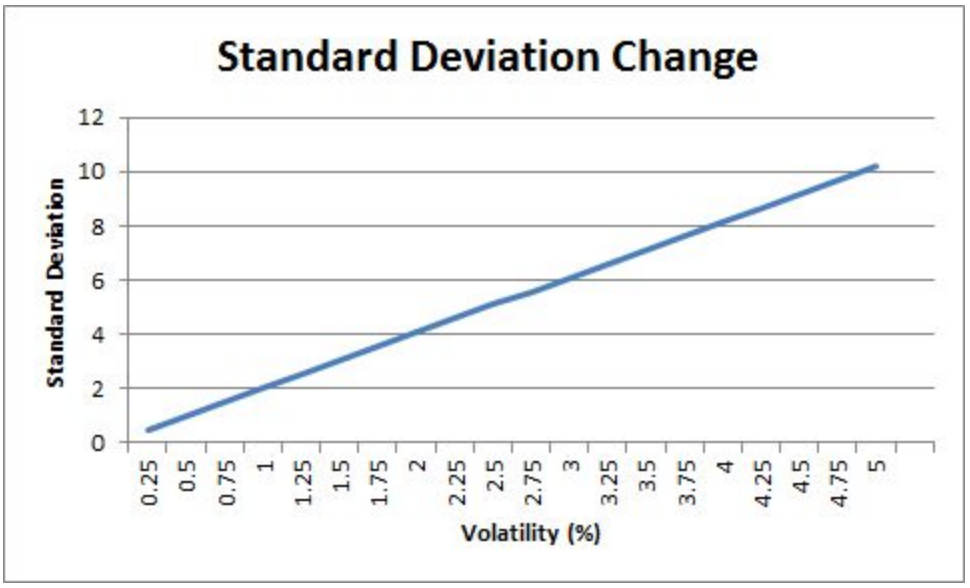
While being able to predict the S&P 500 is very useful what if we wanted to be able to predict the high of any stock. In order to do this we could easily just switch the network's input to be the past 14 highs of that stock along with the most recent close of that stock and it would run fine. While the network would run fine, its error rate and PDF would be completely different. One way to fix this would be to back test the network on the given stock going back a year and then make new error models that correspond to that stock. This would infact work, but back testing the neural network a year back takes around a half an hour. If we had to do this each time we wanted to run it on a new stock it would be very inconvenient and make running a quick

prediction on a stock before you buy it nearly impossible. Luckily I was able to come up with a way around this issue.

After backtesting the network on several stocks all of different volatility, what I found was that the mean error stayed around 0% just like it was with the S&P 500. However, the greater the stock's volatility, the greater the standard deviation on the error for that stock. Infact, the standard deviation actually increases linearly with the stock's volatility thus I was able to create a linear model to estimate the standard deviation of the network's error for any stock using the stock's volatility.This model can be found below with the equation

$$y = 2.035x - 0.21625$$

Figure 6.7- Standard Deviation Change



As for the mean, what I found for the mean after backtesting stocks can be seen below

Mean error %

mean 1	-0.17594
mean 2	-0.1645
mean 3	-0.18572
mean 4	-0.19345
mean 5	-0.18475
mean 6	-0.19456
mean 7	-0.16482
mean 8	-0.21034
mean 9	-0.16732
mean 10	-0.14596
average	-0.17874

Thus for the mean in my PDF formula I decided to use the average of the means for the 10 stocks as a constant for the mean error. Now using this new model we are now able to run the neural network on any stock and accurately predict the error. Thus the new probability density function for any stock is as follows. Notice how we are able to replace the standard deviation with the

slope of our error model times the volatility so that the standard deviation will change based on the stock's volatility multiplied by the slope constant.

$$P(x) = \int_{a}^{b} \frac{1}{2.035v\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2(2.035v)^2}}$$

Where:

v = percent volatility of stock

μ = mean (-0.17874)

Once the neural network was able to predict the high for any stock as well as generate an error model for it, it was time to expand it even more. My next goal was to get the network to predict the low for the day. In order to do this I simply switched the data being fed to the network from the previous high of the past 14 days and the most recent closing price to the previous low of the past 14 days and the most recent closing price. After backtesting the network's ability to predict the low on several stocks what I found was that the network was still able to predict the lows as well, but the errors standard deviation was slightly higher than it was for predicting the high of the day. However the backtest results were still normally distributed so and the increase in the errors standard deviation still increased linearly with volatility so just like with the high I was able to generate an error model for the neural networks low of the day predictions as well which can be seen below

$$P(x) = \int_{a}^{b} \frac{1}{2.129v\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2(2.129v)^2}}$$

Where:

v = percent volatility of stock

μ = mean (-0.17874)

Once this was complete I decided to see if the network could predict the high or low of any given time interval. In order to do this I fed the previous high or low of the past 14 intervals of time I wanted to predict for. For example, if I wanted to predict the high of the next hour I would feed the network the previous highs of the past 14 hours and then the current price of the stock as the closing price. After backtesting different time intervals what I found was that their was a linear relation between the amount of time I wanted to predict the high or low for and the difference in the error's standard deviation.Thus I was able to make a new model to predict the error's standard deviation for any stock for any time interval as seen below.

$$\sigma = 2.035v\frac{t}{390} \text{ or for low } \sigma = 2.129v\frac{t}{390}$$

Where:

v=stock's volatility in %

t=time interval you want to predict (in minutes)

Since their is a linear relation between the change in the error's standard deviation and the time, we are able to scale the standard deviation based on the time interval we want to predict. The reason I use $\frac{t}{390}$ is because the old equation calculated the standard deviation for the day, but since there is a linear relation we are able to scale the standard deviation based on how much time we want to predict in relation to what the error would be for a full day or 390 minutes. Now if we apply this new formula to our probability density function we get for predicting the high or low for any time interval.

$$P(x) = \int_a^b \frac{1}{(2.035 v \sqrt{\frac{t}{390}})\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2(2.035 v \sqrt{\frac{t}{390}})^2}}$$

Or for predicting the low

$$P(x) = \int_a^b \frac{1}{(2.129 v \sqrt{\frac{t}{390}})\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2(2.129 v \sqrt{\frac{t}{390}})^2}}$$

Where:

μ=mean (-0.17874)

v=stock's volatility in percent

t=time interval you want to predict in minutes

Now this is a very complicated formula so here is an example of it in use so that it will a little more sense. Let's say we have a stock that has a volatility of .75% and is currently at $98 a share. We want to see what the high of the next 200 minutes will be so we feed the necessary data to the neural net and it says the high of the next 200 minutes will be $100. If we want to know the percent chance that the high of the next 200 minutes would be at least $99 we could use the above probability density equation to figure this out.

$$P(x) = \int_{-1}^{\infty} \frac{1}{(2.035 * .75\frac{200}{390})\sqrt{2\pi}} e^{\frac{-(x+.17874)^2}{2(2.035 * .75\frac{200}{390})^2}}$$

The reason we make the interval -1 to infinity is because $99 is roughly 1% less than the predicted high so we want to find the probability that the actual high of that 200 minute interval will lie between 1% less than the predicted high and infinity percent greater than the predicted high. When we evaluate this expression we get .8536 or 85.36% chance that the the actual high of the next 200 minutes will be greater than $99 which means we have an 85.36% chance of making money off the trade. Now as you can imagine this kind of probability estimation can be very useful when deciding whether to make a trade.

Regression Analysis

Regression analysis is a statistical technique that can be used to estimate the relationship between two or more variables. In regression analysis one or more independent variables are compared against a dependant variable in order to determine how the dependant variable changes

as a given independent variable changes. One of the more common techniques used to do regression analysis is called linear regression. In a linear regression model, a straight line also known as the line of best fit is drawn through the data in such a way that the sum of the squared residual is as small as possible. A residual is a the vertical distance between the actual point and the line of best fit. The equation for the line of best fit can be seen as

$$y = \alpha + \beta x$$

Where:

α = the y-intercept

β =the slope of the line

We can calculate α and β by using the following equations.

$$\beta = \frac{\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n}(x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta\bar{x}$$

Where:

$\bar{x}$ = the mean of the x variable

$\bar{y}$ = the mean of the y variable

n = the amount of data points

There is also a value known as the r squared value. This number ranges between zero and one and is responsible for telling how well the line of best fit fits the data. A value of one means it fits the data perfectly while a value of zero means it doesn't fit the data at all. To calculate this value we can do the following.

$$r = \frac{\frac{1}{n}(\sum_{i=0}^{n} x_i y_i) - \bar{x}\,\bar{y}}{\sqrt{(\overline{x^2} - \bar{x}^2) - (\overline{y^2} - \bar{y}^2)}}$$

$\bar{x}$ = the mean of the x variable

$\bar{y}$ = the mean of the y variable

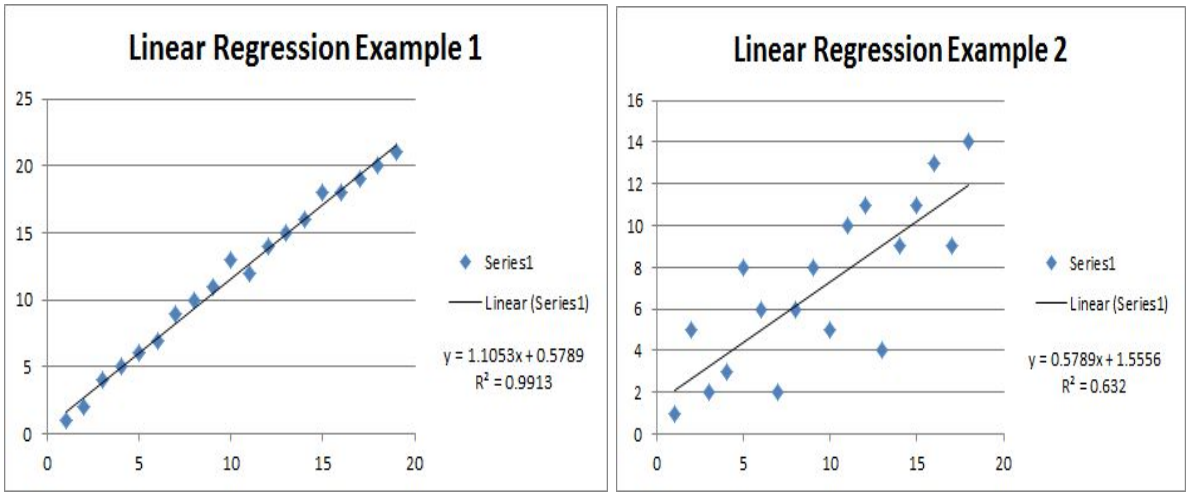
$\overline{x^2}$ = mean of all x values squared

$\overline{y^2}$ =mean of all y values squared

n = the amount of data points

If we then square the value of r we can get the r squared value. Below are few linear regression models and their r squared values.

Figure 6.8- Linear Regression Graphs



As you can see the regression line from example one fits the data much better thus the r squared value is higher. Now what if we wanted to predict what the 20th value would be for example one. To do this we can actually extend the regression line up to x=20 and see what the y value would be at that point. This with an r squared value of .9913 this predicted value would most likely be pretty close to the actual value. Now if we wanted to predict the next value for example to we could do the same thing we did for example one but this prediction would most likely be as accurate because as you can see by looking at the data itself or the r squared value for the second example there is much more spread in the data and the line of best fit does not fit it as tightly as in example one. To add to these linear regression models we can use the standard deviation of the regression lines error to create channels on either side of the data. These channels usually lie two standard deviations away on either side of the regression line. This means that around 95% of the time the actual value of the predicted data point will lie within these channels. An example of this can be seen below.

Figure 6.9- Example of Chart with Channels

Green line= data

Blue line=regression line

Red lines=channels

As this graph shows, the regression line shows the overall trend of the data which is why it is also called a trendline and it can be said that the r squared value represents how strong that trend is. It can also be said that if the trend holds and the data is currently near the lower channel that it will need to go back up to the top channel. Then again the trend might and the data could just keep going down. In this case the current model would no longer hold and it would be necessary to create another regression model to predict the data.

This same concept can be applied to a stock chart. In fact the above graph is stock graph. In order to see if a stock is trending as well as what direction it is trending in a linear regression model can be generated for the stock. Now depending on the amount of time you go back in the

stock data, it is possible to capture different trends in the data. For example if you were to generate a regression model going 100 days back their might be strong upward trend in the data. However if you were to generate a graph going 25 days back the stock might have a strong downward trend. This is key because now we can say that the stock is trending up long term but trending down shorter term. Now if that short term downward trend continues for another 25 days or so it will begin to influence the long term trend going 100 days back so eventually both the long and short term trends could be downward, Being able to evaluate the different long and short term trends in a stock is key in deciding if a certain regression model will hold. For example, if we look at the above graph which goes back roughly 4 days. If we want to see if this trend is going to hold which would mean it would continue to go up as well as head towards the top of the channel, we could evaluate shorter as well as longer term trends to get a better estimate of whether the current 4 day trend we are looking at is going to hold. For example if we look at both a 2 day model and an 8 day model and they both have a very strong negative trend and the 4 day trend while trending up is currently very close to the lower channel, it is safe to say that the 4 day regression model that we are currently using is probably not going to hold and the stock is going to continue to go down. While using just a few different time length regression models to predict a stock's movement can be effective, what if their was a way to factor in a regression model for every time interval to see what a stock is going to do.

To do this we first pick a time length to create an initial regression model. Typically I will do 500 five minute ticks back which is around 4 days back because this is a generally a good length to look if you are planning on holding the stock for a day or two. This current model will

be our base model which means we will use other regression models to predict if the base model will hold or break. From here, using an algorithm created in python, a regression model for every time interval between 5 five minute ticks back to 1000 five minute ticks back is created. The slope of each of those regression lines is then multiplied by its corresponding r squared value then added to the rest and then divided by the total amount of models. The value produced will represent the slope of the combined trends.

$$s_c = \left(\frac{1}{2n-5} * \overline{r^2}\right) \sum_{i=0}^{2n-5} r_i^2 s_i$$

Where:

s= the slope of the ith trendline

r^2 = the ith r squared value

$\overline{r^2}$ =average of the r squared values

n= amount of ticks used in base model

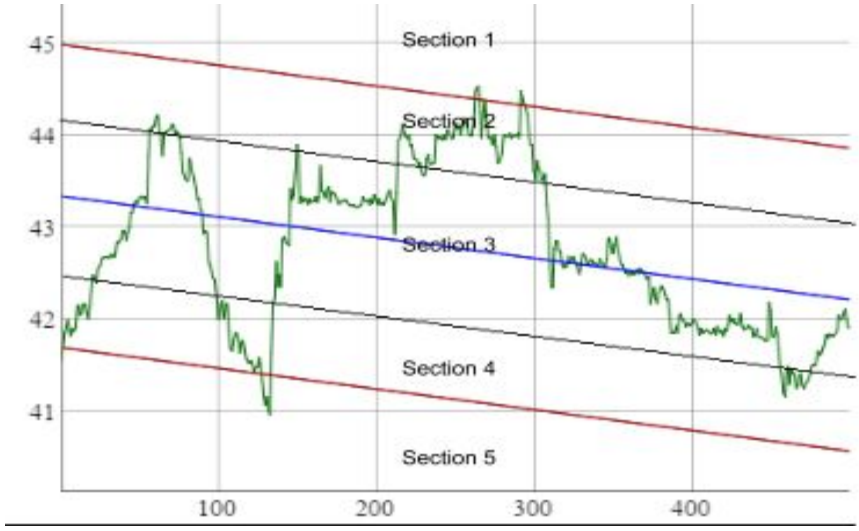
Notice how this formula not only factors in each regression line, but also how strong that trend is by using the r squared value. This value is key in predicting the price of a stock and hints in which direction it will move in and by how much. I will be referred to as s_c from now on and this value will be used more later in other prediction models.

The Rating System

Using this the idea of linear regression analysis above I was able to produce a system that is able to assign a rating for a given stock that essentially gives the expected percent return of that stock given the model you feed into the rating system. The rating system takes as input a stock, the amount of ticks to model back, and the size of a tick. For example an example input would be, the stock ticker SPY, and then go back 500 five minute ticks. From here the rating system would calculate the expected percent return for the next 100 ticks which is a fifth of the input ticks. Below is a detailed overview of how the rating system works.

First, the input is taken and a regression model is made with it. From there the regression model is broken into 5 sections. Section one is the area above the top channel. Section two is the area from the top channel to the halfway point between the regression line and the top channel. Section three is the area from the halfway point between the regression line and the top channel and the halfway point between the regression line and the bottom channel. Section four is the area between the halfway point between the regression line and the bottom channel and the bottom channel itself. Section five is the area below the bottom channel. Below is an image showing the different sections.

Figure 6.10- Channels with Different Sections



After the regression model is broken up into the different sections, the algorithm determines which section the stock is currently in. In the case above, the stock is currently in section three since the point that's the farthest to the right is in between the two black lines. Once the section the stock is currently in is determined the algorithm moves onto the next step which varies depending on which section the stock is in. Therefore I will go into detail on each of the five sections.

If the stock is in section one the probability of the stock staying above the upper channel is first calculated. To get this, we use the the regression summing technique I talked about earlier in which I calculated a value I called s_c which represents the overall slope of the combined regression lines, Except this time we are only using the most recent 10% ticks in the calculation . For example if the model was 500 ticks back I would use the most recent 50 ticks in the calculation. However, we need to slightly modify the s_c calculation because we want a

probability not a slope. So instead we first check if the slope of the base model is greater than zero. If it is we calculate the amount of regression models that had a slope greater than that of the base regression model we are using and then divide that value by the total amount of regression models used and if the base model slope is less than zero we count the amount of models with a slope greater than zero and divide that by the amount of models. This value is the estimated probability of the stock staying above the upper channel.

$$p = \frac{s_g}{.10n-5}$$

Where:

s_g = amount of regression models with slope greater than base model's or the amount of models greater than 0 if base models slope is negative.

n=total amount of ticks used in base model

Once we have the probability of the stock staying above the upper channel, we then go on to calculate the percent return of the stock if it stays above the top channel. To get this value we simply multiply the slope of the base regression model times 20% of the total amount of ticks used in the base model and then add a standard deviation to it. We do this because if it is going to stay above the upper channel, the stock will need to atleast go up on average the slope of the regression line per tick which is why we multiply the slope of the regression line by the amount of ticks we are predicting. We then add a standard deviation because we assume if the stock is staying above the regression line it will go up a decent amount at one point which is why we add a standard deviation to it. The following formula can be seen below.

$$return = \frac{1}{5}sn + \sigma$$

Where:

s = the slope of base model

n = amount of ticks used in base model

σ = standard deviation of model

Then since we are calculating the expected value of the stock if it says above the upper channel, we need to multiply the probability of it staying above the upper channel; with the estimated return if it stays above the channel. The combined formula is as follows

$$E_{above} = (\frac{s_g}{.10n-5})(\frac{1}{5}sn + \sigma)$$

The expected value of the stock staying above upper the upper channel is only half the calculation however. We still need to find the expected value of the stock going below the upper channel. To do this we first need the probability of this happening which is simply one minus the probability of the stock staying above the upper channel since it is the only other outcome. Then for the percent return if it drops below the upper channel is 4% because that will be our stop loss and we will sell if we are down 4% so that is the most we can lose. Thus the formula for the expected value of the stock going below the upper channel is.

$$E_{below} = 4(1 - p_{above})$$

Now that we have the expected value of the stock staying above the upper channel and the expected value of it going below we can add them together to get the expected value or rating of the stock if it is above the upper channel

$$E_{section\ 1} = E_{above} + E_{below}$$

For section two the expected value calculation is almost the same as section one. The stock can do one of two things. The first is that the regression model will hold in which the stock will tap the top channel and then begin to drop down to the lower channel or just drop initially, The other option is that the stock will go up and break the upper channel and stay above it. This would mean that the regression model would no longer be accurate and we would need to make a new one to predict the stock. To calculate the expected value of the stock if it breaks the top channel and stays above it we can calculate the probability of the stock breaking the top channel. To do this we can use the same probability formula as we used for section one. Then to get the value of the stock if it break the upper channel we can first to calculate how much it will go up to get to the upper channel. We can then add a standard deviation to this value because we are assuming that it is breaking the upper channel and continuing to go up and a standard deviation is a good estimate of how much it will move up in the predicted time frame.

$$E_{break} = \left(\frac{s_g}{.10n-5}\right) * 100\left(\frac{v_t - v_c}{v_c}\right) + \frac{1}{5}ns + \sigma$$

Where:

s_g = amount of regression models with slope greater than base model's or the amount of models greater than 0 if base models slope is negative.

v_t = the price at top channel

v_c = current price

σ = the standard deviation

n=amount of ticks used in base model

We then need to calculate the expected value of the stock if it drops. We can do this the same way we did for section one.

$$E_{drop} = 4(1 - (\frac{s_g}{.10n-5}))$$

Then if we add the two expected values together we get the following formula for the expected value of stock if it is in section 2

$$E_{section\ 2} = E_{break} + E_{drop}$$

Now calculating the expected value if the stock is in section three is a little more difficult.This is because since the stock is in the middle of the regression model it could very easily move in either direction and in each case it would still fit the overall regression model. While this may be the case, we can calculate the expected value for the stock when it's in the middle. To do this, we can calculate the amount of regression models that have a slope greater than that of the base model plus a half a standard deviation and divide this by the total amount of models. This will be the probability of it going towards the top channel. We then count the

amount that are less than the base slope by at least a half a standard deviation and divide by the total amount of models. This will be the probability of it going down to lower channel. Then do one minus these two probabilities added together to get the probability of it staying middle. Now that we have a way to calculate the values of each outcome as well as the probabilities we can now calculate the expected value for each outcome,

$$E_{top} = (\frac{s_g}{.10n-5}) * 100(\frac{v_t-v_c}{v_c}) + \frac{1}{5}ns$$

Where:

s_g = amount of models greater by at least a half standard deviation

v_t = the price at top channel

v_c = current price

n = amount of ticks in base model

s = slope of base regression line in percent

$$E_{bottom} = -4(\frac{s_L}{.10n-5})$$

Where:

s_L = amount of models less by at least a half standard deviation

n = amount of ticks in base model

$$E_{middle} = (1 - s_g + s_L)\frac{1}{5}ns$$

Where:

s_g = amount of models greater by at least a half standard deviation

s_L = amount of models less by at least a half standard deviation

n = amount of ticks in base model

s = slope of base regression line in percent

Now that we have the expected value of all three possible outcomes we can add them together to get the total expected value of the stock if it is in section three as seen below.

$$E_{section\ 3} = E_{top} + E_{middle} + E_{bottom}$$

For a stock in section four, the way we calculate the expected value is very similar to a the way we did it for section two. For a stock in this section, there are two possible options. The first is that it will tap the bottom channel and shoot up or just shoot up initially. The second option is that it will break the bottom channel and continue to fall. To calculate the expected value of the first option, we can assume that the current regression model is holding since it is heading towards the top channel. Therefor we can use the slope of the base model in our calculations. To get the value of the stock if its going towards top channel we take the distance between the stock's current location and the top channel and convert it to a percent. We then add this value to the slope times the amount of ticks we are predicting. To finish the calculation we multiply that value by the percent chance of this outcome which is simply the amount of models with a slope greater than that of the base model divided by the total amount of models. Then to

get the expected value of the stock breaking the bottom channel we simply do negative four times 1 minus the probability of it going to top channel. Once again we use negative four because our stop loss is at four percent so that is the most we will lose because we will sell at that point.

$$E_{top} = (\frac{s_g}{.10n-5}) * 100(\frac{v_t-v_c}{v_c}) + \frac{1}{5}ns$$

s_g = amount of regression models with slope greater than base model's or the amount of models greater than 0 if base models slope is negative.

v_t = the price at top channel

v_c = current price

n = amount of ticks in base model

s = slope of base regression line in percent

$$E_{break} = -4(1 - (\frac{s_g}{.10n-5}))$$

We can then add the expected value of the two possible outcomes together to get the expected value for a stock that lies in section 4 as seen below

$$E_{section\ 4} = E_{top} + E_{break}$$

Finally, for a stock in section 5 we can calculate its expected value by analyzing each of the two outcomes. The first is that the stock will come back through the lower channel and head towards the top channel and the other option is that it will stay below the lower channel and continue to fall. To calculate the value of the first outcome we can use the same slope since just like before it will be following the current model by heading up towards the top channel. Thus we can calculate the expected value by doing by finding the distance between the top channel and the current position of the stock and then subtracting a standard deviation from it since it is unlikely that the stock will move up all the way to the top channel in the interval we are predicting. We can then add this to the slope of the model times the amount of ticks we are predicting. Then we multiply this by the probability which as before is just the amount of models with a slope greater than that of the base model divided by the total amount of models as seen below. We can also calculate the expected value of the other outcome the same way as last time and just multiply negative four by one minus the probability of outcome one.

$$E_{top} = (\frac{s_g}{.10n-5}) * 100(\frac{v_t-v_c}{v_c}) - \sigma + \frac{1}{5}ns$$

s_g = amount of regression models with slope greater than base model's or the amount of models greater than 0 if base models slope is negative.

v_t = the price at top channel

v_c = current price

n = amount of ticks in base model

s = slope of base regression line in percent

σ = standard deviation of base model in percent

$$E_{stay\ below} = -4(1 - \frac{s_g}{.10n-5})$$

Now that we have the expected value of both possible outcomes we can add them together to get the expected value for a stock in section five.

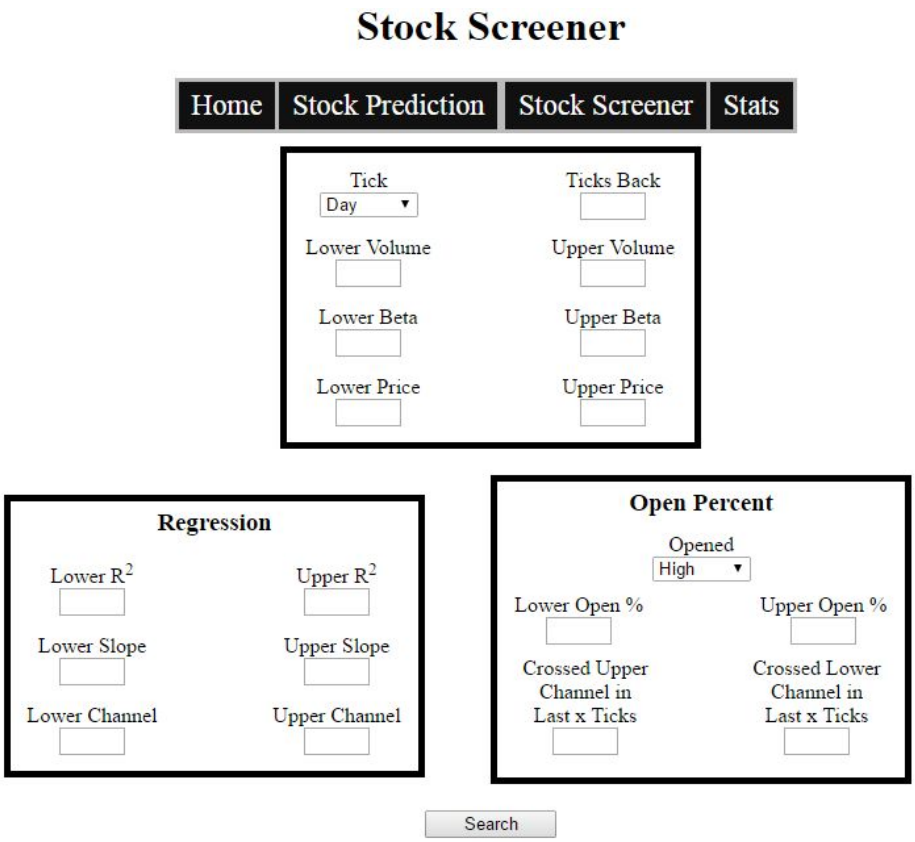
$$E_{section\ 5} = E_{top} + E_{stay\ below}$$

Now that we have a rating system, we can now rate any stock to determine how good it will do in the future. This means that theoretically rating wise, there is a best and worst stock to trade on any given day. The question now is how to find the stock that will have the best rating out of all the stocks in the market. While technically we could just go through by hand and rate every single stock in the market, this is very time consuming and impractical. Instead, we determined a much more efficient way to achieve this.

The Stock Screener

We developed a stock screener that is able to screen every stock on the major exchanges for certain criteria and return the stocks that fit it. The screeners input looks like the following.

Figure 6.11- Stock Screener



To use the screener, you start off by choosing the tick size as well as how many ticks you want to screen back. From there you have the option to screen for stocks that fall within a certain volume,beta, or price range. You then have the option to also screen for stocks who have a regression model with a r squared value, slope or channel width within a certain range. Finally, you have the option to screen for stocks that opened high or low by a certain percent or have recently crossed the upper or lower channel of the regression model. It's important to note however that each of these fields are optional and if a field is left blank the screener will not screen for it. When run, the screener will pull live tick data from google and use it to calculate the necessary values that the screener is screening for and then see if those values fall within in

the screeners criteria. If not the screener will discard the stock and repeat the process on the next stock, but if the stock matches all the screening criteria, the screener will calculate the rating for that stock and then add it to a list of stocks that fit the criteria which will be returned to the user along with the rating and other stats for each of those stocks when the screener is finished. The screener will do this process to every stock on the major exchanges and return the ones that match sorted from greatest to least by their rating. Below is a sample output of the screener which screened 45 hour ticks back with a slope greater than .33 per tick and a r squared of over .75.

Ticker	Volume	Price	Beta	R Squared	Slope	Channel Width	Rating	Correlation
MUX	238973	1.991	0.563	0.758	0.419	12.584	22.021	-0.052
SA	151931	12.234	-0.688	0.906	0.577	9.864	14.664	-0.111
CRBP	13496	1.921	-0.252	0.815	0.352	8.902	11.748	0.23
MOMO	619631	13.663	3.02	0.822	1.226	30.288	7.668	-0.072
SCTY	562568	26.665	2.157	0.754	0.415	12.6	4.428	-0.017
XIN	59184	4.72	0.575	0.949	0.337	4.159	4.156	0.222
KGC	207850	3.582	-0.333	0.763	0.379	11.246	4.055	-0.133

	5							
PTLA	150894	22.746	1.715	0.821	0.356	8.833	3.539	0.169
XBIT	12169	10.27	-0.548	0.809	0.392	10.126	2.873	0.006

Now as you can imagine this is a very useful tool to have since it gives you the ability to find stocks that have the highest rating, but also have other criteria such as high slopes or beta values. However if you simply want to find which stock has the highest rating and not filter any stocks out based on other criteria, you can simply just screen with just the ticks back and tick size fields filled in. Also, because the screener has so many different criteria you can screen for it is able to support many different trading strategies since you are able to vary what you want to screen for depending on what your strategy. .The screener also has the ability to find short and long term trends since it will calculate the criteria based on the time interval you input. For example if you only put in 50 hours, it will calculate the beta or slope for just the past 50 hours. On the other hand if you put in 50 day ticks, it will calculate values such as slope or beta for the past 50 days thus capturing more of a long term trend thus allowing the screener to be an effective tool for both short and long term traders. As far as run time, the screener takes around 10-15 minutes to run. This is because the screener needs to pull and analyze data for over 6000 stocks. The screener however utilizes several optimization techniques to make sure it runs as fast as possible with its available resources. To start, as soon as a stock fails to meet one of the criterias, the stock is discarded and the screener moves onto the next one. More importantly though, the screener utilizes all four cores my computer's processor to screen four stocks at once

which greatly increases its speed. Overall the stock screener is an extremely effective tool which allows you to find the perfect stocks to trade for any given strategy.

Rating Two

While the initial rating system was very effective, what I found was that it didn't factor in how the market is going to preform into its calculation. This caused the ratings to be inaccurate occasionally due to the fact that the current rating didn't factor in the markets pull on a given stock. In order to fix this problem with the rating system, we first need to determine how to measure how strong the markets pull on a given stock is. To do this we can calculate the stocks correlation with the S&P 500 which generally represents the market as a whole. A correlation of 1 means that the given stock follows the market perfectly, a correlation of 0 means the stock doesn't follow the market at all and a correlation of -1 means the stock does the exact opposite of what the market does or inversely follows the market. To calculate the correlation of a given tock with the market we can do the following.

$$r = \frac{n(\sum_{i}^{n} x_i y_i) - (\sum_{i}^{n} x_i)(\sum_{i}^{n} y_i)}{\sqrt{[(n\sum_{i}^{n} x_i^2) - (\sum_{i}^{n} x_i)^2)]\,[(n\sum_{i}^{n} y_i^2) - (\sum_{i}^{n} y_i)^2]}}$$

Where:

x = the ith tick value of the S&P 500

y = the ith tick value of the given stock

n = the amount of tick back

There is however two major issues with this standard correlation formula. The first is that if we use the price at each tick for the values of x and y then the correlation will not be accurate. This is because we measure a stock's movement relative to another stock in percent not price, Therefor we need to convert the given prices into percents before we can calculate the correlation. In order to do this we can modify the formula slightly to get the following.

$$r = \frac{n(\sum_{i}^{n-1}(\frac{(x_{i+1}-x_i)}{x_i}\frac{(y_{i+1}-y_i)}{y_i})-(\sum_{i}^{n-1}\frac{(x_{i+1}-x_i)}{x_i})(\sum_{i}^{n-1}\frac{(y_{i+1}-y_i)}{y_i})}{\sqrt{[(n(\sum_{i}^{n-1}(\frac{(x_{i+1}-x_i)}{x_i})^2)-(\sum_{i}^{n-1}\frac{(x_{i+1}-x_i)}{x_i})^2][(n(\sum_{i}^{n-1}(\frac{(y_{i+1}-y_i)}{y_i})^2)-(\sum_{i}^{n-1}\frac{(y_{i+1}-y_i)}{y_i})^2]}}$$

Where:

x = the ith tick value of the S&P 500

y = the ith tick value of the given stock

n = the amount of tick back

While the above formula will calculate the actual correlation of the given stock with the market, this correlation value in my opinion fails to capture the real underlying correlation with the market. For example, if a stock always moves three times as much as the market so for every 1% the market moves this will move 3%. The current correlation formula would give this stock a

.33 correlation with the market because it moves three times as much as the other stock. While according to a real correlation measurement this would be the case, for my system, I want a stock that follows the market's direction exactly but three times a much to receive a correlation of 1. This is because if the market is going to go up than this given stock will go up as well and I want the correlation value to reflect that. To accomplish this, I multiply the correlation by the beta value of the stock. The beta value of a stock is essentially the ratio between how much the given stock moves and the market. In the case above the stock that moves three times as much would have a beta value of 3 and a correlation of ,33 so when we multiply the two together we get an adjusted correlation of 1 which is what we want. With our adjusted correlation value we can measure if the stock moves the same amount as the market relative to its volatility.

$$r_{adjusted} = \beta r$$

Where:

β = the beta value of the given stock

r = the stock's actual correlation with the market

We can use this adjusted correlation value to see how much the given stock moves with the market. This also means we know how much of a given stock's rating to base on how we think the market is going to do in the interval we are predicting. The question now is how do we determine how well we think the market is going to do in the interval we are predicting. In order to do this we can utilize the neural network from earlier to predict the high and low of S&P 500 for the interval and then use these predicted values to determine how well the market is going to do and assign a rating to the market as a whole. In fact we can actually create two separate

ratings for the market. The first we will use for stocks that move with the market and the second we can use for stocks that move inversely with the market. To get the rating we will use for stocks that move with the market we can simply find how much the percent the S&P 500 will increase from its most recent price to the predicted high. We then subtract .3% from that value to build in some error to make sure the market is going to go up more than just a little before we start to positively factor it into our rating. We then use this value as the expected value or rating of the market. To get for stocks that move inversely with the market we can get the percent movement from the most recent price to the predicted low and take the absolute value this and then subtract .30 like before. We then use this value as the expected value or rating. These formulas can be seen below.

$$Rating_1 = 100(\frac{p_h - p_c}{p_c}) - .30$$

$$Rating_2 = 100|\frac{p_L - p_c}{p_c}| - .30$$

Where:

p_h = the predicted high of interval

p_l = predicted low of interval

p_c =the current price of S&P 500

Now that we have a rating system for the market itself we need to factor this rating into our rating system for a given stock. To do this we turn the stock's adjusted correlation with the

market into a percent. For example a correlation of .35 would turn into 35%. If the correlation is positive we will using the first market rating and if the correlation is negative we will use the second rating market rating. We then multiply the percent correlation by the market rating and multiply that value by the stock's beta value. Multiplying by the beta value will effectively scale the rating so it factors in the given stock's volatility. We then add it to one minus the correlation percent times the old rating. This will give us the stocks new expected value or rating.

$$E = (1 - r_a)E_{old} + |\beta| r_a E_{market}$$

Where:

r_a = the adjusted correlation converted to a percent

E_{old} = the old expected value/rating of the stock

E_{market} = the expected value/rating of market

β = the beta value of the given stock

Essentially what we have done is factor in the markets pull on a given stock into its expected value or rating. Now a stock that gets a high initial rating but correlates heavily with the market could get its rating dropped significantly if the market is predicted to do bad. This also means that a stock which is initially predicted to do bad could get its rating bumped if the market is going to have a bad day and it correlates heavily inversely with the market. Factoring the markets predicted performance is key in improving the rating system. Its also important to note

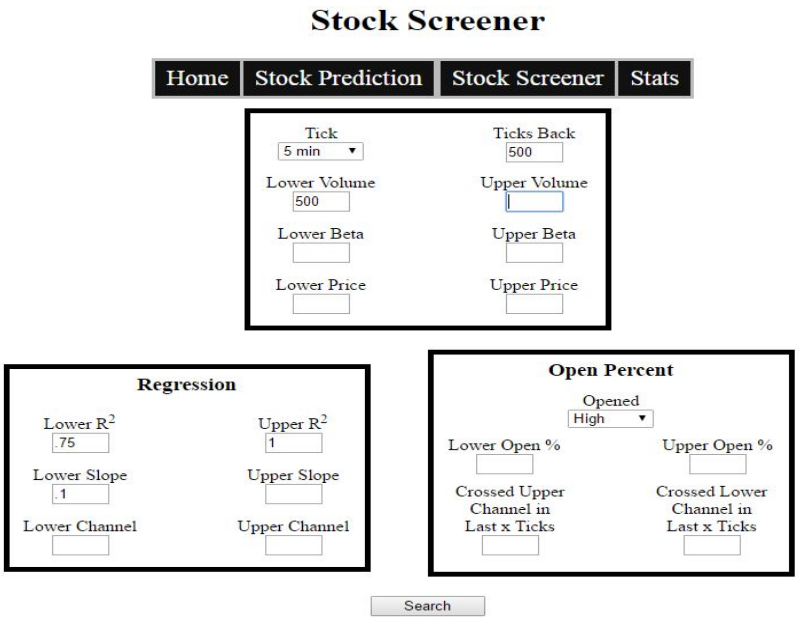
that the more a stock correlates with the market the more the markets rating will factor into the stock's rating.

Chapter 7: Trading System 1

Using the tools I created above, I was able to design a trading system that can be very profitable as well as easy to use. While I only used one trading strategy in particular, the tools available are designed to accommodate a large variety of different trading strategies. The strategy I used is a day trading system in which stock are held anywhere from 5 minutes to a day or two. The basic idea is to capitalize on uptrending stocks and buy two or three of them right before the market closes and then sell them the next day to capitalize on an opening price jump along with the overall uptrend. The question is though, how to determine what stocks to buy as well as when to sell them the next day.

We can utilize the stock screener to solve this first issue, While one can simply run the stock screener with just a tick size and ticks back fields filled and it will rate every stock in the market, for this given strategy I prefer to utilize some of the screeners other fields as well. Since my strategy involves buying up trending stocks, I screen for stocks with a regression model that has a high slope and r squared value. This is because if a stock has a high r squared value and slope it is uptrending and will most likely open high and continue going up for the rest of day. I will also screen for volume as well because I don't want to trade stocks that don’t have much trading volume. Thus a picture of the usual screen can be seen below.

Figure 7.1- Stock Screener in Use

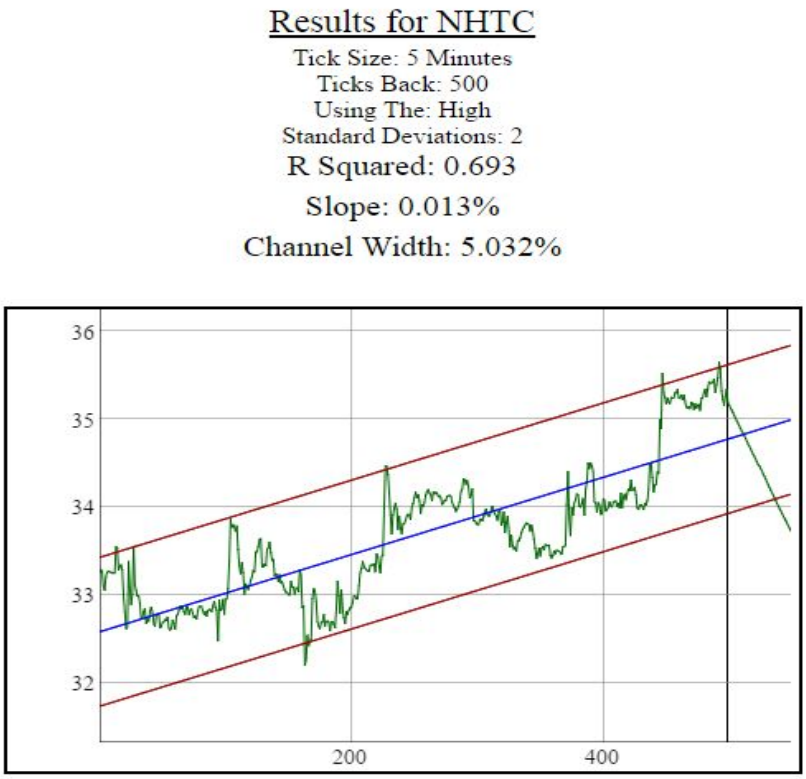


Looking at the image above, we have set the screener to look for stocks that in the past 500 five minute ticks have had an average trading volume of at least 500 per five minute tick and a 500 tick regression model that has an r squared value of at least .75 and a slope of at least .1% per tick.The screener will now go through every stock in the market and return the ones that match this criteria as well as rate them and put the highest rated stocks first in the results. Now that we have a list of stocks that match our criteria, we can go through and pick which ones we want to buy.

In order to do this we can use the generate different regression models for each of the top rated stocks to see which ones we think are going to do the best. Luckily, the website not only has a screener but also allows you to generate regression models for a stock using any tick size and interval. Thus we are able to examine short, medium and long term trends in the stock to see how good we think it is going to do the next day, The regression model generator will also

generate a prediction line which will show the direction it is predicted to go to help out in the analysis process as well. Below is an example of how this process would work on one of the stocks that the screener could return. First, we would generate a regression model for the initial tick interval we screened for. Note that the part of the model after the vertical black line is the predicted direction and is not part of the actual regression model.

Figure 7.2- Regression Model Example

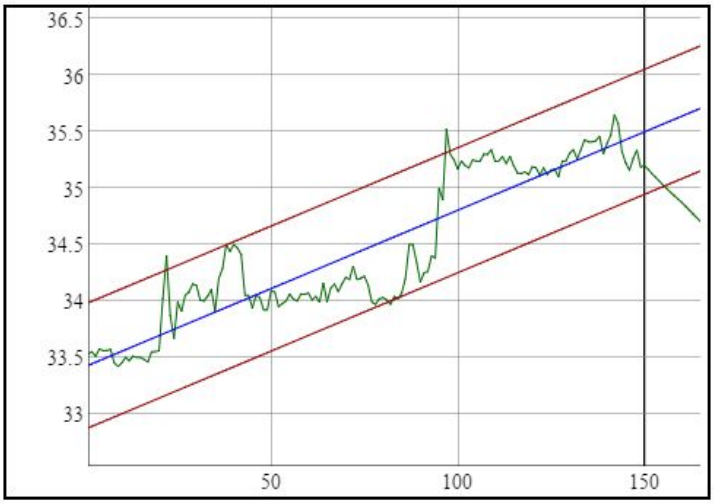


From looking at the above model along with the predicted direction, it is easy to see that the stock is most likely going to fall towards the bottom channel. However to get a sense of what it is going to do shorter term we can generate a 150 tick model.

Figure 7.3- Regression Model Example

Results for NHTC

Tick Size: 5 Minutes
Ticks Back: 150
Using The: High
Standard Deviations: 2
R Squared: 0.827
Slope: 0.04%
Channel Width: 3.216%



From looking at the 150 tick model, we can see that the shorter term direction isn't looking good either. From the above model, it looks like it's going to break the bottom channel and continue to keep going down. This would definitely support the 500 tick model which showed it was going to head towards the bottom channel. It is important to remember though that since the 150 tick model is a shorter time frame, the distance between the channels is a lots smaller so when the stock breaks the bottom channel on the 150 tick model it does not mean it has also broken the 500 tick channel as well. It just means that its dropped a fair amount in the 500 tick model After looking at the 500 and 150 tick model, normally I would look at a 50 tick model too, however, it is pretty clear from these two models that the stock is going to drop the

next day so it is not necessary. While NHTC is an example of a stock I would not buy, let's look at an example of one that I would buy.

Below is a 500 five minute tick model for FAZ. From the 500 tick model we can see that it has just recently bounced off the bottom channel and is heading towards the top channel. While the prediction line is pointing down, in this case I think the stock will infact go up to the top channel. To know for sure though we must examine the shorter term models

Figure 7.4- Regression Model Example

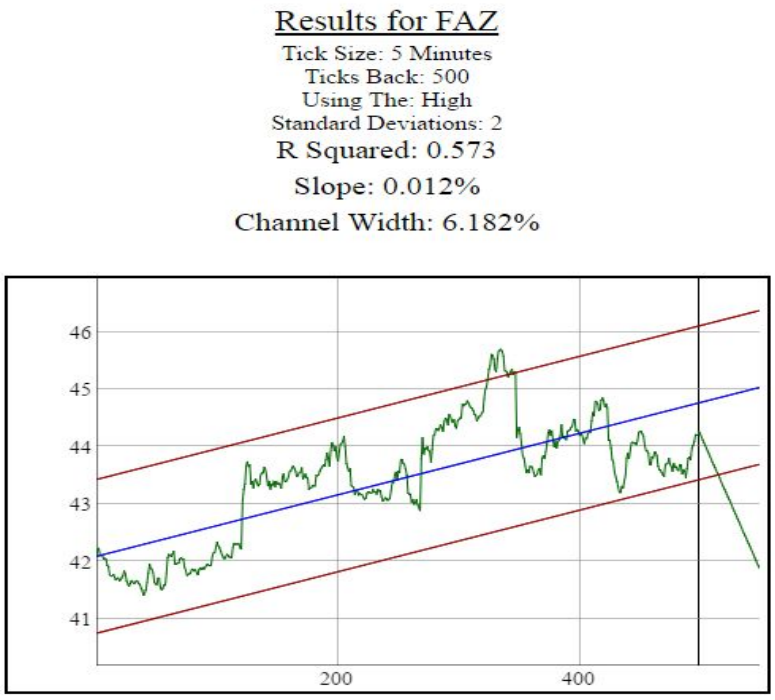
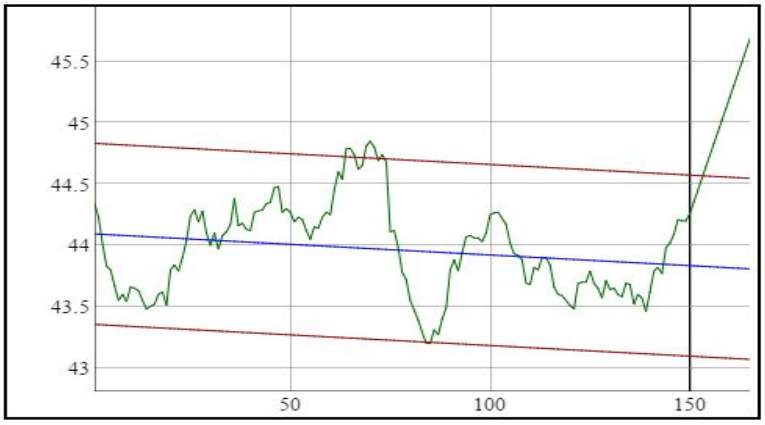


Figure 7.5- Regression Model Example

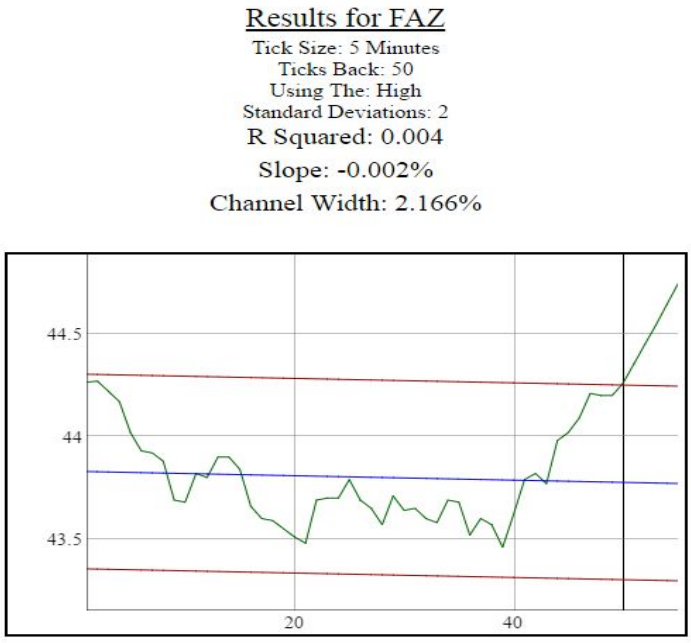
Results for FAZ

Tick Size: 5 Minutes
Ticks Back: 150
Using The: High
Standard Deviations: 2
R Squared: 0.04
Slope: -0.004%
Channel Width: 3.361%



From the above 150 five minute tick model, we can see the the stock has lots of short term momentum and it looks like it's going to shoot up and break the top chanel. The prediction line for this model supports this as well. If the stock does in fact shoot up and break this top channel then it will definitely continue to shoot up to the top channel of the 500 tick model. However to make sure it is in fact going to break the top channel in the 150 tick model, we need to look at an even shorter trend model.

Figure 7.6- Regression Model Example



From the above 50 five minute tick model, we can see that it's about to break the top channel and the prediction line supports this too. From this model we can see that the stock will most likely have enough short term momentum to break the top channel in the 150 tick model which intern means it will in fact be heading up to the top channel in the 500 tick model which exactly what we want. After doing the above analysis on this stock, I would definitely buy it and it would most likely shoot up for the day as the models suggest.

The above examples show how we can use regression models to decide if a given stock that the screener returns is going to perform well the next day. While doing this regression analysis is key to picking the stocks that will most likely do the best, it is not always necessary in order to make money. Since all the stocks we are deciding between will have high ratings, most

of them will do well the next day, so it's more picking which one is going to do the best rather than which one is going to make money. This being the case if one is lazy or just doesn't have the time to look through the top 10 stocks or so they can simply just buy the two or three top rated stocks and they will most likely perform well the next day.

Using the above trading strategy along with the software I designed I did the following trades and overall did fairly well.

Figure 7.7- Trading Examples

Date	Name	Order Type	Price	Shares	Total Amount	% Return
02/17/16, 09:32 AM	GLUU	Market Buy	3.43	975	3344.25	n/a
02/17/16, 09:32 AM	NHTC	Market Buy	29.6	113	3344.8	n/a
02/17/16, 09:32 AM	S	Market Buy	2.85	351	1000.35	n/a
02/17/16, 09:33 AM	JPEP	Market Buy	4.25	410	1742.5	n/a
2/17/16,11:18 AM	NHTC	Market Sell	31.56	113	3566.28	6.60%

2/17/16, 11:22 AM	S	Market Sell	2.89	351	1014.39	1.40%
2/17/16, 01:49 PM	NHTC	Market Short	32.29	100	3229	n/a
2/17/16, 02:54 PM	GLUU	Market Sell	3.55	975	3461.25	3.50%
2/17/16, 3:59 PM	NHTC	Market Cover	31.55	100	3303	2.30%
2/17/16, 03:59 PM	JPEP	Market Sell	4.15	410	1701.5	-2.30%
2/18/16, 09:31 AM	CSCO	Market Buy	26.46	90	2381.4	n/a
2/18/16, 09:31 AM	DGAZ	Market Buy	19.54	122	2383.88	n/a
2/18/16, 09:31 AM	CRTO	Market Buy	38.53	58	2234.74	n/a
2/18/16, 09:32 AM	NAII	Market Buy	11	300	3300	n/a

2/18/16, 01:33 PM	DGAZ	Market Sell	20.02	122	2442.44	2.50%
2/18/16, 02:32 PM	UPRO	Market Buy	51.57	45	2320.65	n/a
2/18/16, 03:06 PM	CSCO	Market Sell	26.38	90	2374.2	-0.30%
2/18/16, 03:22 PM	CRTO	Market Sell	38.19	58	2215.02	-0.90%
2/18/16, 03:55 PM	NAII	Market Sell	10.99	300	3297	-0.10%
2/18/16, 03:58 PM	W	Market Short	37.97	82	3113.54	na
2/18/16, 03:58 PM	SRNE	Market Short	6.03	550	3316.5	na
2/18/16, 03:59 PM	WYNN	Market Short	78.47	18	1412.46	na
2/19/16, 9:37 AM	WYNN	Market Cover	76	18	1456.92	3.20%

2/19/16, 03:55 PM	W	Market Cover	39.62	82	2978.24	-4.30%
2/19/16, 03:56 PM	IAG	Market Short	2.46	1747	4297.62	n/a
2/22/16, 9:38 AM	IAG	Market Cover	2.31	1747	4559.64	6%
2/22/2016, 01:37 PM	UPRO	Market Sell	53.29	45	2398.05	3.30%
2/22/2016:, 3:44 PM	SRNE	Market Cover	6.17	550	3239.5	-2.50%
2/25/16, 10:07 AM	FDC	Market Buy	12.51	260	3252.6	n/a
2/25/16, 10:10 AM	LSCC	Market Buy	6.24	515	3213.6	n/a
2/25/16, 10:11 AM	MNOV	Market Buy	6.23	650	4049.5	n/a
2/25/16, 12:27 PM	MNOV	Market Sell	6.74	650	4381	8.10%

2/26/16, 10:03 AM	UPRO	Market Buy	54.77	60	3286.2	n/a
2/26/16, 3:43 PM	FDC	Market Sell	12.89	260	3351.4	3%
2/26/16, 3:57 PM	LSCC	Market Sell	6.11	515	3146.65	-2.10%
2/26/16, 3:57 PM	UPRO	Market Sell	53.66	60	3219.6	-2%
2/26/16, 3:58 PM	DGAZ	Market Buy	25.11	150	3774.5	n/a
2/26/16, 3:58 PM	MEET	Market Buy	3.67	1000	3673	n/a
2/29/16, 10:19 AM	S	Market Buy	3.31	1000	3318	n/a
2/29/16, 11:29 AM	MEET	Market Sell	3.25	1000	3122	-12.9%%
2/29/16, 11:30 AM	DGAZ	Market Sell	27.38	150	4099.5	9%

2/29/16, 2:25 PM	S	Market Sell	3.46	1000	3447.01	4.50%
2/29/16, 3:58 PM	FEYE	Market Buy	16.82	200	3372	n/a
3/1/16, 9:40 AM	MDR	Market Buy	3.41	1150	3918	n/a
3/1/16, 2:48 PM	FEYE	Market Sell	17.58	200	3507.01	4.50%
3/1/16, 3:48 PM	MDR	Market Sell	3.35	1150	3844.51	-1.80%
3/1/16, 3:55 PM	ACW	Market Buy	1.51	2500	3758	n/a
3/1/16, 3:55 PM	DYN	Market Buy	10.62	350	3725	n/a
3/2/16, 9:57 AM	DYN	Market Sell	11.83	350	4132.51	11%
3/2/16, 9:35 AM	ACW	Market Sell	1.51	2500	3769.51	0%

3/4/16, 3:55 PM	ENPH	Market Buy	3.03	1000	3029.91	n/a
3/4/16, 3:55 PM	SDLP	Market Buy	3.54	850	3009	n/a
3/4/16, 3:56 PM	EYEG	Market Buy	3.85	285	1097.25	n/a
3/7/16, 9:35 AM	SDLP	Market Sell	3.91	850	3317.04	10.40%
3/7/16, 9:36 AM	ENPH	Market Sell	3.24	1000	3240.11	6.80%
3/7/16, 11:39 AM	EYEG	Market Sell	3.85	285	1097.28	0%
3/7/16, 3:53 PM	MCEP	Market Buy	1.87	1650	3085.51	n/a
3/7/16, 3:55 PM	TWI	Market Buy	6.11	367	2236.74	n/a
3/7/16, 3:55 PM	S	Market Buy	3.96	588	2328.42	n/a

3/8/16, 9:35 AM	MCEP	Market Sell	1.62	1650	2672.51	-13.30%
3/8/16, 9:45 AM	TWI	Market Sell	5.82	367	2135.94	-4.50%
3/9/16, 3:53 PM	GLF	Market Buy	6.78	500	3390	n/a
3/9/16, 3:57 PM	S	Market Sell	3.96	588	2328.42	0%
3/10/16, 3:53 PM	EXK	Market Buy	2.74	839	2298.86	n/a
3/9/16, 9:45 AM	GLF	Market Sell	6.93	500	3465	2.20%
3/14/16, 9:35 AM	GLUU	Market Buy	3.21	650	2086.51	n/a

3/14/16, 9:36 AM	TCK	Market Buy	7.49	296	2217.04	n/a
3/17/16, 9:55 AM	TCK	Market Sell	7.91	296	2338.41	5.60%
3/17/16, 10:31 AM	EXK	Market Sell	2.83	839	2374.37	3.30%
3/17/16, 3:31 PM	GLUU	Market Sell	3.14	650	2042.04	-2.10%
3/21/16, 3:51 PM	W	Market Buy	37.55	130	4881.5	n/a
3/21/16, 3:55 PM	DGAZ	Market Buy	22.3	150	3345	n/a
3/21/16, 3:56 PM	WYN	Market Short	75.56	47	3551.32	n/a
3/22/16, 11:18 AM	WYN	Market Cover	75.46	47	3546.62	0.07%

3/22/16, 3:55 PM	NHTC	Market Short	37.24	94	3500.56	n/a
3/22/16, 3:58 PM	DGAZ	Market Sell	21.67	150	3250.5	-2.80%
3/23/16, 2:09 PM	NHTC	Market Cover	32.72	94	3075.68	12.13%
3/23/16, 3:47 PM	CARA	Market Buy	5.85	650	3802.5	n/a
3/28/16, 3:20 PM	W	Market Sell	40.31	130	5240.3	7.35%
3/28/16, 3:58 PM	DGAZ	Market Buy	22.28	150	3342	n/a
3/29/16, 3:55 PM	NHTC	Market Short	35.02	94	3291.88	n/a
3/31/16, 1:50 PM	CARA	Market Sell	6.31	650	4101.5	7.86%

3/31/16, 3:49 PM	DGAZ	Market Sell	21.11	150	3166.5	-5.25%
3/31/16, 3:52 PM	TCK	Market Short	7.65	422	3228.3	n/a
3/31/16, 3:52 PM	UVXY	Market Buy	20.52	128	2626.56	n/a
4/1/16, 1:53 PM	NHTC	Market Cover	32.84	94	3086.96	6.23%
4/7/16, 12:18 PM	TCK	Market Cover	7.02	422	2962.44	8.23%
4/7/16, 3:58 PM	UVXY	Market Sell	21.99	128	2814.72	7.10%

Chapter 8: System 2

The fully automated system was built using a fairly simple algorithm based on the ideas of precision. During many trials and errors, the idea came along on having the regression trend for just one day. To create this, the number of bars since the start of the trading day has to be considered. The strategy called for a long entry if both the 15 bar regression and the day's regression are above set amounts (optimized). In addition to the regressions being positive, the number of entries per day were limited to 1, and trades were not allowed if there was already a trade on or if the time was before 10:30 am (amateur hour). The exit was controlled by either the time being past 3:30 pm (end of day), or less than the intercept minus a designated value (optimized) times the standard error. After optimization the values came out at follows: the number of standard deviance away was found to be 1, the slopes had to be higher than 0.003.

The easy language code was as follows:

```
Inputs:
Length1(3),
Dist(0.78),
Slo1(0.07),
Slo2(0.07);
Variables:
Length2(0),
Reg1(0),Reg2(0),
Error1(0),Error2(0),
oLRSlope(0),
oLRAngle(0),
oLRIntercept1(0),
oLRIntercept2(0),
oLRValueRaw(0);
```

```
If Time>=1030 and Time<1530 then Begin
If Time>=1000 and Time<1100 then
Length2=Round((((Time-1000))/5)-1,0);
If Time>=1100 and Time<1200 then
Length2=Round((((Time-1100)+60)/5)-1,0);
If Time>=1200 and Time<1300 then
Length2=Round((((Time-1200)+120)/5)-1,0);
If Time>=1300 and Time<1400 then
Length2=Round((((Time-1300)+180)/5)-1,0);
If Time>=1400 and Time<1500 then
Length2=Round((((Time-1400)+240)/5)-1,0);
If Time>=1500 and Time<1600 then
Length2=Round((((Time-1500)+300)/5)-1,0);
Value1=LinearReg(close,Length1,0,Reg1,oLRAngle,oLRIntercept
1,oLRValueRaw);
Error1=Stderror(close,Length1);
Value2=LinearReg(close,Length2,0,Reg2,oLRAngle,oLRIntercept
2,oLRValueRaw);
Error2=Stderror(close,Length2);
If marketposition=0 and EntriesToday(Date)=0 and Reg1>Slo1
and Reg2>Slo2 then buy next bar market;
End;
If marketposition=1 and Time>1530 then sell next bar at
market;
If marketposition=1 and close<(oLRIntercept2-Dist*Error2)
then sell next bar at market;
```

Overall, the system works decently well. The real advantage of the system comes when using the screener to find the stock. During back testing, the strategy performed decently well on its own. The system overall made a net profit which is the ultimate goal of any system. The profit factor (2.12) is above the cutoff of 2.

Figure 8.1- TradeStation Results

	All Trades	Long Trades	Short Trades
Total Net Profit	$3,772.69	$3,772.69	$0.00
Gross Profit	$7,141.70	$7,141.70	$0.00
Gross Loss	($3,369.01)	($3,369.01)	$0.00
Profit Factor	2.12	2.12	n/a

The profitability ratio of this system (33%) was extremely low. This is not in itself a problem, but it does indicate that some added precision to the system, but the intention is to have the system working in tangent with the screener.

Figure 8.2- TradeStation Results (cont.)

Total Number of Trades	18	18	0
Percent Profitable	33.33%	33.33%	0.00%
Winning Trades	6	6	0
Losing Trades	12	12	0
Even Trades	0	0	0

The ratio of average win to average loss (4.24) greatly makes up for the low win rate, but the sheer number of losses reduces the average net profit to $209.

Figure 8.3- TradeStation Results (cont.)

Avg. Trade Net Profit	$209.59	$209.59	$0.00
Avg. Winning Trade	$1,190.28	$1,190.28	$0.00
Avg. Losing Trade	($280.75)	($280.75)	$0.00
Ratio Avg. Win:Avg. Loss	4.24	4.24	n/a
Largest Winning Trade	$2,217.00	$2,217.00	$0.00
Largest Losing Trade	($605.40)	($605.40)	$0.00

The max number of winning trades is about as expected at 3, but the max number of losing trades is excessive. When used with the screener, the number of consecutive losses would be significantly reduced. Winning trades are held about twice as long as losing trades at 43 bars (about three and a half hours), and losing trades lasting 21 bars (about an hour and 45 minutes).

Although losing trades are about half as long as winning trades, optimally, the amount of time should be closer to 8 bars (40 minutes).

Figure 8.4- TradeStation Results (cont.)

Max. Consecutive Winning Trades	3	3	0
Max. Consecutive Losing Trades	9	9	0
Avg. Bars in Winning Trades	42.83	42.83	0.00
Avg. Bars in Losing Trades	20.75	20.75	0.00
Avg. Bars in Even Trades	0.00	0.00	0.00

The maximum position size was set by a constant position sizing of $20,000 on a $100,000 account. The commission costs were estimated at $7.50 per trade, so the contract sizing did not play into commission costs.

Figure 8.5- TradeStation Results (cont.)

Max. Shares/Contracts Held	11,560	11,560	0
Total Shares/Contracts Held	154,137	154,137	0
Account Size Required	$1,818.13	$1,818.13	$0.00

Based on the test period, the system was predicted to have an annual rate of return of roughly 16% which is twice the average for the market (S&P 500). Another item to consider is that the market was dropping during the testing period. Another positive characteristic of the system was the low percentage of time in the market (about 2.5%). This factor added to the relatively favorable RINA index of 558.

Figure 8.5- TradeStation Results (cont.)

Return on Initial Capital	3.77%
Annual Rate of Return	16.91%
Return Retracement Ratio	n/a
RINA Index	558.05
Trading Period	2 Months, 19 days
Percent of Time in the Market	2.37%
Max. Equity Run-up	$7,052.46

The following is an example of a trade made by this system:

Figure 8.6- example of a trade

Chapter 9: Trading System 3

In this trading system, I focused on using the stock scanner that we created to choose stocks in the moment that had high R^2 (0.8-1.0) while having relatively high slopes (1-1.5% increase per day). The maximum amount of stocks I chose to trade with per day was 2.

This strategy was based off buying when the predicted low of the day was hit, and selling when the predicted high of the day was reached. Overall, if no strategy was executed 30 minutes before the market closed, then the stock was ultimately sold, whether it was a loss or gain.

The nights before a trade was placed, I would use the scanner to predict which stocks to use for the next day using our criteria for R^2 and slope. Then, I would run our system so it would output predicted highs and lows for the next day.

Ultimately, there were some downfalls that I ran into in this trading system. Some of the most notable issues were when the low estimated price or the high estimated price were overestimated or underestimated.

If the low was underestimated, then sometimes the trade would never execute because the buying price was never hit. When the high was overestimated, the trades would be sold 30 minutes before the end of the day, whether at a profit or a loss. Sometimes, there weren't even trades executed. Overall, the best run I had was in the chart below.

Overall, this strategy was relatively manaul because I had to enter I the high and low for each day on TradeStation. For that reason, along with the variability of the stock tracker, I would not use this system for the future. It was very variable, and I have decided to stick more toward longer term investing than day trading.

Figure 9.1- Sample Trading Period (Best Run)

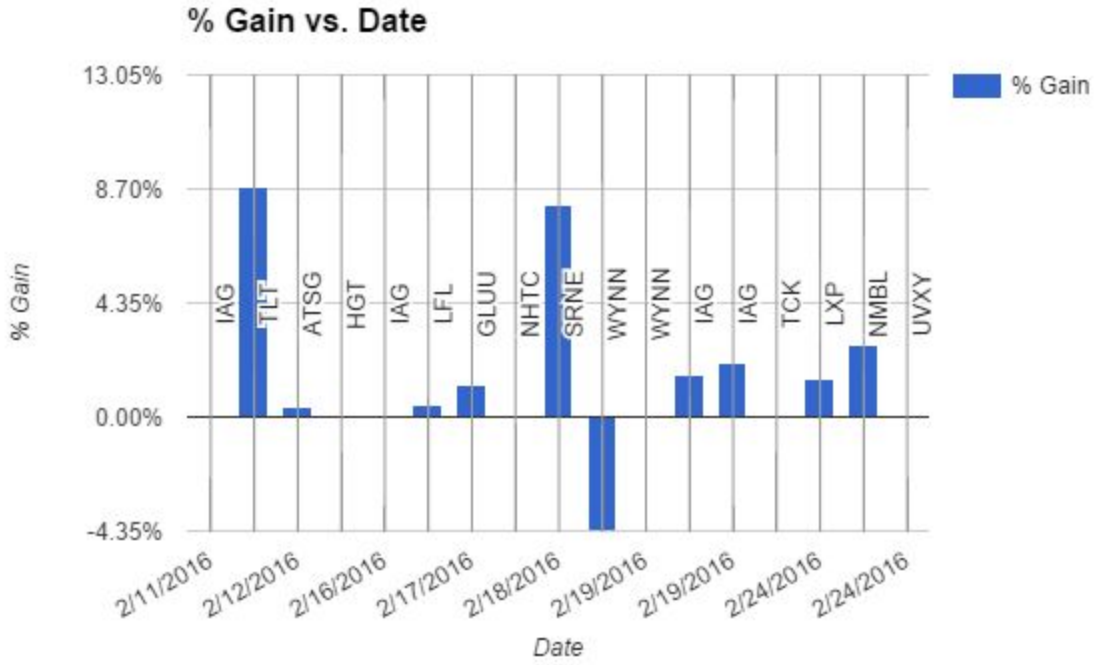


Figure 9.2- Overestimation of High (Sold 30 minutes before the day end)

Figure 9.3- Underestimated of Low (Sold 30 minutes before the day end, never executed otherwise)

Figure 9.4- Sample of Trades

Date	Ticker	(P) Low	(P) High	(P) Bought	(P) Sold	% Gain
2/11/2016	IAG	$2.35	$2.17	$2.17	$2.36	8.76%
2/11/2016	TLT	$133.20	$136.00	$133.10	$133.59	0.37%

2/12/2016	ATSG	$10.65	$11.17	$10.66	$10.66	0%
2/12/2016	HGT	$1.58	$1.63	$0.00	$0.00	0%
2/16/2016	IAG	$2.18	$2.34	$2.18	$2.19	0.46%
2/16/2016	LFL	$5.74	$5.83	$5.76	$5.83	1.22%
2/17/2016	GLUU	$3.39	$3.57	$0.00	$0.00	0%
2/17/2016	NHTC	$29.70	$31.50	$29.61	$32.02	8.10%
2/18/2016	SRNE	$6.25	$6.90	$6.10	$6.08	-4.35%
2/18/2016	WYNN	$74.00	$78.20	$0.00	$0.00	0%
2/19/2016	WYNN	$76.10	$77.30	$76.01	$77.26	1.64%

2/19/2016	IAG	$2.43	$2.52	$2.42	$2.47	2.07%
2/22/2016	TCK	$5.64	$5.68	0	0	0%
2/24/2016	LXP	7.644	7.67	7.46	7.57	1.47%
2/24/2016	NMBL	$7.50	$7.54	$6.99	$7.18	2.72%
2/24/2016	UVXY	$38.67	$39.21	$0.00	$0.00	0%

Chapter 10: Summary and Conclusions

Based on your testing, does each of your systems look like it can make money?

For our manual trading system based in Chapter 7, we believe that this system would be the most profitable out of the 3. Using the stock screener followed up with the regression analysis, this system was usually able to predict a profitable stock to trade for the day.

For system two, while profitable, could utilize the advantage of the screener to find a lower risk equity to trade daily. Major problems in the system lie in the low win ratio. With the added help of a screener, this ratio would greatly increase.

For system three, this was not very profitable overall, and was too manual for not being profitable. Therefore, reverting back to system one would be the best option if we want to make the most profit.

What Problems did you encounter during your project?

We encountered problems with non profitable trading systems, difficulties with backtesting a semi automatic system due to the nature of screeners, and the availability of past data through our chosen source.

What modifications would you do in the future?

For modifications, we would continuously track the long term potential for System 1, and learn how to optimize it further for the best results. Modifying our neural net over time to better fit the market, and continuously checking the accuracy of the predicted prices would be key to its continued profitability.

Works Cited

Bajkowski, John. "Measures of Market Breadth: A Look at Trading Statistics." *Aaii.com*. Aaii, Oct. 1998. Web. 3 May 2016.

"Behavioral Finance Definition | Investopedia." *Investopedia*. N.p., 19 Nov. 2003. Web. 03 May 2016.

"Bollinger Band® Definition | Investopedia." *Investopedia*. N.p., 23 Nov. 2003. Web. 03 May 2016.

"Bond Definition | Investopedia." *Investopedia*. N.p., 23 Nov. 2003. Web. 03 May 2016.

"Buying Stock on Margin." - *For Dummies*. N.p., n.d. Web. 03 May 2016.

"Cash Account vs. Margin Account: What's the Difference?" *Cash Account vs. Margin Account: What's the Difference?* N.p., n.d. Web. 03 May 2016.

"Chaos in Technical Analysis and Backtesting – Part III: Split-adjusted Data." *Price Action Lab Blog*. N.p., 27 Nov. 2012. Web. 03 May 2016.

"Commodity Definition | Investopedia." *Investopedia*. N.p., 19 Nov. 2003. Web. 03 May 2016.

"Day-Trading Margin Requirements: Know the Rules." *FINRA.org*. N.p., n.d. Web. 03 May 2016.

"Equity Definition | Investopedia." *Investopedia*. N.p., 18 Nov. 2003. Web. 03 May 2016.

Farley, Alan. "Analyzing The 5 Most Liquid Commodity Futures (WTI, ZC) | Investopedia." *Investopedia*. Investopedia, 02 Sept. 2015. Web. 03 May 2016.

"Fidelity Learning Center: Sector Rotation: An Introduction." *Fidelity Learning Center: Sector Rotation: An Introduction*. N.p., n.d. Web. 03 May 2016.

Folger, Jean. "Introduction To Order Types: Introduction | Investopedia." *Investopedia*. N.p., 04 Sept. 2012. Web. 03 May 2016.

Folger, Jean. "What Is the Difference between Investing and Trading? | Investopedia." *Investopedia*. Investopedia, 30 Aug. 2012. Web. 03 May 2016.

"Forexiation - Diary of a Perth Forex Trader." *: Possibly the Most Important Metric for Any Trading System*. N.p., 4 June 2012. Web. 03 May 2016.

"Futures Contract Definition | Futures Contract Meaning - The Economic Times." *The Economic Times*. N.p., n.d. Web. 03 May 2016.

"Futures Markets - Part 4: What Is a Futures Contract?" *What Is a Futures Contract?* N.p., n.d. Web. 03 May 2016.

Harper, David. "Getting To Know The Stock Exchanges | Investopedia." *Investopedia*. Investopedia, 19 Sept. 2004. Web. 03 May 2016.

Harvey, Ian. "Pattern Day Traders | Investopedia." *Investopedia*. Investopedia, 08 Oct. 2013. Web. 03 May 2016.

Knott, Kevin G. "Convection Baking." Franz Haas Machinery of America, Inc., 20 May 2013. Web. 03 May 2016.

Kuepper, Justin. "A Look At Exit Strategies." *Investopedia*. N.p., 26 Sept. 2004. Web. 03 May 2016.

Kuepper, Justin. "Trading Systems: What Is A Trading System? | Investopedia." *Investopedia*. N.p., 07 Oct. 2004. Web. 03 May 2016.

"Margin." *Interactive Brokers*. N.p., n.d. Web. 03 May 2016.

"Margin Requirements." - *Firstrade Securities Inc.* N.p., n.d. Web. 03 May 2016.

"Moving Average (MA) Definition | Investopedia." *Investopedia*. N.p., 24 Nov. 2003. Web. 03 May 2016.

"Nasdaq Definition | Investopedia." *Investopedia*. N.p., 24 Nov. 2003. Web. 03 May 2016.

"New York Stock Exchange (NYSE) - FXCM." *FXCM Insights RSS*. N.p., 11 Dec. 2014. Web. 03 May 2016.

"Options Contract." *Definition & Example*. N.p., n.d. Web. 03 May 2016.

"Oscillator Definition." *Investopedia*. N.p., 04 Jan. 2004. Web. 03 May 2016.

"Relative Strength Definition | Investopedia." *Investopedia*. N.p., 25 Nov. 2003. Web. 03 May 2016.

Scott. "BUILDING YOUR TRADING SYSTEM." *Http://evilspeculator.com/*. Evil Speculator, 11 Mar. 2014. Web. 3 May 2016.

Speights, Keith. "3 Biotech Stocks with Low Liquidity Ratios -- Are They in Trouble? -- The Motley Fool." *The Motley Fool*. N.p., n.d. Web. 03 May 2016.

Spiegelman, Rande. "Margin: How Does It Work?" *Margin: How Does It Work?* Shwab, 9 May 2013. Web. 03 May 2016.

"Stochastic Oscillator Definition | Investopedia." *Investopedia*. N.p., 26 Nov. 2003. Web. 03 May 2016.

"Tax Benefits of Professional Trader Status." *Tax Benefits of Professional Trader Status*. N.p., n.d. Web. 03 May 2016.

Thomas, Kaye A. "Fairmark.com." *Benefits of Trader Status*. Fairmark, n.d. Web. 03 May 2016.

"Trading Commissions and Margin Rates - Fidelity." *Trading Commissions and Margin Rates - Fidelity*. N.p., n.d. Web. 03 May 2016.

"Trend and Cycle Together." *Open Mind*. N.p., 07 Jan. 2012. Web. 03 May 2016.

"12 CFR 240.9 - Margin Requirements." *LII / Legal Information Institute*. Cornell, n.d. Web. 03 May 2016.

"Types Of Financial Markets And Their Roles - Complete Guide To Corporate Finance | Investopedia." *Investopedia*. N.p., 28 Mar. 2012. Web. 03 May 2016.

"Understanding Stock Splits | Investopedia." *Investopedia*. N.p., 30 Dec. 2003. Web. 03 May 2016.

"What Is Forex: Buying And Selling In Currency Pairs." *BabyPipscom*. N.p., 17 Apr. 2011. Web. 03 May 2016.

"Why Trade Forex: Advantages Of Forex Trading." *BabyPipscom*. N.p., 14 Apr. 2011. Web. 03 May 2016.

Zucchi, Kristina. "What Is the Difference Between Institutional Traders and Retail Traders? | Investopedia." *Investopedia*. Investopedia, 05 Mar. 2015. Web. 03 May 2016.

Appendix

Neural Network Code

```
import numpy as np
import openpyxl
import sys
import csv


#print 'Neural Netwrok gathering and formating data...'
inputNodeCount=14   #this is the amount of input nodes you want
hiddenNodeCount=10  #this is the amount of hidden nodes you want
outputNodes=1       #this is the amount of output nodes you want NOTE: only works for one output
trainingIterations=80000 # this is the amount of training iterations you want. less iterations means faster but less acurate training
learningRate=.02 #this is how fast you want the neural net to train. you can play around with this value but i wouldn't go above .5
momentum=.7 #this term allows the neural net to avoid local mins. NOTE: to much momentum will cause neural net to overshoot goal


# the rule is if predicted high is <13 from the closing price than market is opening down



class Layer():
    def __init__ (self, neuron_number, inputs_per_neuron):
        self.neuron_number=neuron_number
        self.inputs_per_neuron=inputs_per_neuron
        self.weights=np.random.uniform(-.5,.5,(inputs_per_neuron,neuron_number))

class Data():

    def __init__  (self, inputNodes, outputNodes,mode,setting):
        self.inputNodes=inputNodes
```

```
    self.outputNodes=outputNodes
    self.mode=mode
    self.setting=setting
    self.openPrice=0 #opening price of the day
    self.closePrice=0 #closing price of the day before
    self.currentPrice=0 #current price of the stock
    self.currentHigh=0 #current high of the day
    self.currentLow=0 # current low of the day
    self.time=0 # current time in ticks
    self.trainInput2=0
    self.trainOutput=0
    self.testData=0
    self.testData2=0
    self.list=[]
    self.closeList=[]
    self.readFile()

def readFile(self):
    valueList=[]
    closeList=[]
    setting=2
    File = open('data.csv')
    Reader = csv.reader(File)
    Data = list(Reader)
    if str(self.setting) =='low':
        setting=3
    if int(self.mode)==1:
        self.readFile2()
        self.readFile3()
    elif int(self.mode)==2:
        for x in xrange(300):
            try:
                float(Data[x+1][setting])
```

```
            valueList.append(Data[x+1][setting])
            self.list.append(Data[x+1][setting])
            self.closeList.append(Data[x+1][4])
            closeList.append(Data[x+1][4])
        except:
            pass
    valueList.reverse()
    self.list.reverse()
    closeList.reverse()
    self.closeList.reverse()
    valueList=self.checkForSplit(valueList)
    self.list=self.checkForSplit(self.list)
    closeList=self.checkForSplit(closeList)
    self.closeList=self.checkForSplit(self.closeList)
    self.closePrice=closeList[-1]

elif int(self.mode)==3600:
    for x in xrange(len(Data)-7):
        valueList.append(Data[x+7][setting])
        self.list.append(Data[x+7][setting])
        self.closeList.append(Data[x+7][1])
        closeList.append(Data[x+7][1])
else:
    for x in xrange(len(Data)-7):
        valueList.append(Data[x+7][setting])
        self.list.append(Data[x+7][setting])
        self.closeList.append(Data[x+7][1])
        closeList.append(Data[x+7][1])
    size=len(valueList)-300
    valueList=valueList[size:]
    self.list=self.list[size:]
    self.closeList=self.closeList[size:]
    closeList=closeList[size:]
```

```
        if int(self.mode==2):
            self.readFile2()

        if int(self.time)>=100 and int(self.mode)==2:
            if self.setting=='high':
                if self.currentHigh > (float(valueList[-1])+.05) or
self.currentHigh<(float(valueList[-1])-.05):
                    valueList.append(self.currentHigh)
                    closeList.append(self.currentPrice)
            if self.setting=='low':
                if self.currentLow > (float(valueList[-1])+.05) or
self.currentLow<(float(valueList[-1])-.05):
                    valueList.append(self.currentLow)
                    closeList.append(self.currentLow)

        if int(self.mode)!=1:
            closeList=self.normalize(closeList)
            valueList=self.normalize(valueList)
            self.populateMatrix(valueList,closeList)
            self.getTestData(valueList,closeList)



    def readFile2(self):
        File = open('data2.csv')
        Reader = csv.reader(File)
        Data = list(Reader)
        self.openPrice=Data[7][4]
        self.currentPrice=Data[-1][1]
        self.currentLow=Data[7][3]
        self.time=len(Data)
        for x in xrange(len(Data)-7):
            if float(Data[x+7][2])>self.currentHigh:
                self.currentHigh=float(Data[x+7][2])
```

```
        if float(Data[x+7][3])<self.currentLow:
            self.currentLow=float(Data[x+7][3])

def readFile3(self):
    valueList=[]
    closeList=[]
    #79 ticks is full day
    if self.time>382:
        exit(3)
    if self.time<20:
        exit(4)
    interval=79-int(self.time/5)
    #print interval
    File = open('data3.csv')
    Reader = csv.reader(File)
    Data = list(Reader)
    setting=2
    if str(self.setting) =='low':
        setting=3
    Data=Data[7:]
    makeEven=len(Data)%interval
    Data=Data[makeEven:]
    count=0
    current=0
    Data2=[]
    if setting==2:
        for x in xrange(len(Data)):
            if count<interval:
                if float(Data[x][setting])>current:
                    current=Data[x][setting]
            if count==interval-1:
                valueList.append(current)
                self.list.append(current)
```

```
                closeList.append(Data[x][1])
                self.closeList.append(Data[x][1])
                Data2.append(current)
                count=0
                current=0
            else:
                count=count+1
    current=10000
    count=0
    if setting==3:
        for x in xrange(len(Data)):
            if count<interval:
                if float(Data[x][setting])<current:
                    current=Data[x][setting]
            if count==interval-1:
                valueList.append(current)
                self.list.append(current)
                closeList.append(Data[x][1])
                self.closeList.append(Data[x][1])
                Data2.append(current)
                count=0
                current=100000
            else:
                count=count+1
    closeList=self.normalize(closeList)
    valueList=self.normalize(valueList)
    self.populateMatrix(valueList,closeList)
    self.getTestData(valueList,closeList)



def getTestData(self,lst,lst2):#still need to add second one to it
    listIndex=len(lst)-self.inputNodes
```

```
    self.testData=np.zeros([1,self.inputNodes])
    self.testData2=np.zeros([1,1])
    for x in xrange(self.inputNodes):
        self.testData.itemset((0,x),lst[listIndex])
        listIndex+=1
    self.testData2.itemset((0,0),lst2[-1])#fix this value
    #print self.denormalize(self.testData)
    #print '-----------------------------------'
    #print self.denormalize(self.testData2)

def checkForSplit(self,valueList):
    count=1
    for x in xrange(len(valueList)-1):
        prevValue=float(valueList[x])
        if prevValue > (float(valueList[x+1])*1.8):
            if count<100:
                sys.exit(2)
            return valueList[count:]
        count=count+1

    return valueList

def populateMatrix(self,lst,lst2):
    listIndex=0
    outputCount=0
    np.set_printoptions(threshold=np.nan)
    self.trainInput=np.zeros([len(lst)-self.inputNodes,self.inputNodes])
    self.trainInput2=np.zeros([len(lst)-self.inputNodes,1])
    self.trainOutput=np.zeros([1,len(lst)-self.inputNodes])
    for y in xrange((len(lst)-self.inputNodes)):
        for x in xrange(self.inputNodes+1):
          if x==self.inputNodes:
              self.trainOutput.itemset((0,outputCount),lst[listIndex])
```

```
            listIndex-=(self.inputNodes-1)
            self.trainInput2.itemset((y,0),lst2[outputCount+(inputNodeCount-1)])
            outputCount+=1
        else:
            self.trainInput.itemset((y,x),lst[listIndex])
            listIndex+=1

    #print self.denormalize(self.trainInput)
    #print "-------------------------------------"
    #print self.denormalize(self.trainInput2)

def getPrice(self):
    size=len(self.list)-100
    theList=self.list[size:]
    theList=[float(i) for i in theList]
    theAverage=sum(theList)/len(theList)
    return theAverage

def getVol(self):
    size=len(self.list)-100
    volList=[]
    theList=self.list[size:]
    theList=[float(i) for i in theList]
    for i in xrange(len(theList)-1):
        volList.append(theList[i]-theList[i+1])
    volList=np.absolute(volList)
    theAverage=sum(volList)/len(volList)
    return theAverage
```

```
    def normalize(self,lst):
        listMin=float(min(self.list))
        listMax=float(max(self.list))
        y=0
        for x in lst:
            lst[y]=(float(x)-listMin)/(listMax-listMin)
            y+=1
        return lst

    def denormalize(self,x):
        listMin=float(min(self.list))
        listMax=float(max(self.list))
        return x*(listMax-listMin)+listMin




class NeuralNetwork():
    def __init__ (self,layer1,layer12,layer2):
        self.layer1=layer1
        self.layer12=layer12
        self.layer2=layer2

    def sigmoid(self, x):
        return 1/(1+np.exp(-x))

    def run(self,inputs,inputs2):
        output_from_layer1=self.sigmoid(np.dot(inputs, self.layer1.weights))
        output_from_layer12=self.sigmoid(np.dot(inputs2, self.layer12.weights))
        combined=np.append(output_from_layer1,output_from_layer12,axis=1)
        output_from_layer2=self.sigmoid(np.dot(combined,self.layer2.weights))
        return output_from_layer1, output_from_layer12, combined, output_from_layer2
```

```
    def sigmoid_derivative(self,x):
        return x*(1-x)

    def trainNetwork(self, iterations, trainingSet, trainingSet2, desiredOutput):
        m11=np.zeros([inputNodeCount,hiddenNodeCount])
        m12=np.zeros([1,1])
        m2=np.zeros([hiddenNodeCount+1,1])
        for iteration in xrange(iterations):
            output_from_layer1,
output_from_layer12,combined,output_from_layer2=self.run(trainingSet,trainingSet2)
            layer2_error=learningRate*(desiredOutput-output_from_layer2)
            layer2_delta=layer2_error*self.sigmoid_derivative(output_from_layer2)
            layer2_adjust=combined.T.dot(layer2_delta)+(m2*momentum)
            self.layer2.weights+=layer2_adjust
            m2=layer2_adjust

            layer1combined_error=learningRate*(layer2_delta.dot(self.layer2.weights.T))
            layer1_error,layer12_error=np.array_split(layer1combined_error,[10],axis=1)

            layer1_delta=layer1_error*self.sigmoid_derivative(output_from_layer1)
            layer1_adjust=trainingSet.T.dot(layer1_delta)+(m11*momentum)
            self.layer1.weights+=layer1_adjust
            m11=layer1_adjust

            layer12_delta=layer12_error*self.sigmoid_derivative(output_from_layer12)
            layer12_adjust=trainingSet2.T.dot(layer12_delta)+(m12*momentum)
            self.layer12.weights+=layer12_adjust
            m12=layer12_adjust

def main():
    mode=int(sys.argv[1])
    setting=sys.argv[2]
    layer1=Layer(hiddenNodeCount,inputNodeCount)
```

```
    layer12=Layer(1,1)
    layer2=Layer(outputNodes,hiddenNodeCount+1)
    network=NeuralNetwork(layer1,layer12,layer2)
    data=Data(inputNodeCount,outputNodes,mode,setting)
    #print data.denormalize(data.testData2)
    network.trainNetwork(trainingIterations,data.trainInput,data.trainInput2,data.trainOutput.T)
    t1,t2,t3,output=network.run(data.testData,data.testData2)
    predictedPrice=float(data.denormalize(output))

    percentVol=100*(data.getVol()/data.getPrice())
    slope=(10*percentVol)*.016
    intercept=(-60*slope)+(percentVol*10)
    theOutput = data.denormalize(output)
    theOutput=theOutput.item((0,0))
    print np.round(predictedPrice,3)
    print data.getVol()
    print data.getPrice()
    print percentVol
    print slope
    print intercept
    print np.round(float(data.denormalize(data.testData2[-1])),3)
    sys.stdout.flush()
if __name__ == "__main__": main()
```

Screener Code

```
import numpy as np
import sys
import csv
import math

volAverage=0
priceAverage=0
```

```
betaValue=0
R2=0
twoSD=0
slope=0
b=0
RR=0
intercept=0
channelWidth=0
varience=0
opened=0
theValue=0
tickAmount=0
Data=[]
correlation=0
ratingTwo=0


def
checkstock(f1,f2,lowerVol,upperVol,lowerBeta,upperBeta,lowerPrice,upperPrice,SPYReturn,tic
k,daysBack,

lowerR2,upperR2,lowerSlope,upperSlope,lowerChannel,upperChannel,Open,lowerOpen,upperO
pen,percentAbove,percentBelow):
        skipDays=1
        global tickAmount
        tickAmount=int(daysBack)
        closePos=4
        fileLength=0
        theFile=f1
        if tick !='Day':
                skipDays=7
                theFile=f2
                closePos=1
        File = open('data'+theFile+'.csv')
        Reader = csv.reader(File)
```

```
Data = list(Reader)
if len(Data)==0:
        exit(2)
Data=Data[skipDays:]
if skipDays==7:
        Data.reverse()
#Data=Data[10:]
global volAverage
global priceAverage
global betaValue
global R2
global slope
global channelWidth
global opened
global correlation
for x in xrange(int(daysBack)):
        try:
                volAverage+=float(Data[x][5])
        except:
                pass
        try:
                priceAverage+=float(Data[x][2])
        except:
                pass
volAverage=volAverage/daysBack
priceAverage=priceAverage/daysBack
betaValue=getBeta(Data,SPYReturn,daysBack,closePos)
opened=getOpen(f1,f2,tick)
if str(lowerVol) != '':
        if volAverage <=lowerVol or volAverage>=upperVol:
                return 0
if str(lowerPrice)!='':
        if priceAverage <=lowerPrice or priceAverage>=upperPrice:
```

```
                return 0
if str(lowerBeta)!='':
        if betaValue <=lowerBeta or betaValue>=upperBeta:
                return 0
if daysBack != '':
        R2,slope,channelWidth=getRegData(Data,tick,daysBack)
if lowerR2!='':
        if R2 <=float(lowerR2) or R2>= float(upperR2):
                return 0
if lowerSlope!='':
        if slope <=float(lowerSlope) or slope>=float(upperSlope):
                return 0
if lowerChannel!='':
        if channelWidth <=float(lowerChannel) or channelWidth>=float(upperChannel):
                return 0
if lowerOpen!='':
        if opened <=float(lowerOpen) or opened>=float(upperOpen):
                return 0
passedOne=0
if percentBelow!='':
        if getPercentChannel(0,Data,daysBack,percentBelow)==0:
                passedOne=-1
        else:
                passedOne=1
if percentAbove!='':
        if getPercentChannel(1,Data,daysBack,percentAbove)==0:
                passedOne=-1
        else:
                passedOne=1
if passedOne<0:
        return 0
correlation=getCorrelation(Data,daysBack)
return 1
```

```
def getPercentChannel(mode, Data,daysBack,percent):
        daysBack=int(daysBack)
        ticks=int(percent)
        Data=Data[:ticks]
        Data.reverse()
        upperChannelStart=intercept+twoSD+(b*(daysBack-ticks))
        lowerChannelStart=intercept-twoSD+(b*(daysBack-ticks))
        #print(Data)
        #print str(upperChannelStart)+'    '+str(b)
        if mode==0:
                for x in xrange(ticks):
                        skip=0
                        try:
                                float(Data[x][2])
                        except:
                                skip=1
                        if(skip==0):
                                if float(Data[x][2])<=lowerChannelStart:
                                        return 1
                        lowerChannelStart=lowerChannelStart+b
        else:
                for x in xrange(ticks):
                        #print upperChannelStart
                        skip=0
                        try:
                                float(Data[x][2])
                        except:
                                skip=1
                        if(skip==0):
                                if float(Data[x][2])>=upperChannelStart:
                                        return 1
```

```
                upperChannelStart=upperChannelStart+b
        return 0

def getOpen(f1,f2,tick):
        closePrice=0
        openPrice=0
        '''
        if tick!='Day':
                File = open('data'+f2+'.csv')
                Reader = csv.reader(File)
                Data = list(Reader)
                Data=Data[7:]
                Data.reverse()
                i=0
                while(1):
                        skip=0
                        try:
                                float(Data[i+1][2])
                        except:
                                skip=1
                                pass
                        if(skip==0):
                                if Data[i+1][0].isdigit()==0:
                                        closePrice=float(Data[i+2][2])
                                        openPrice=float(Data[i+1][4])
                                        break
                                value1=int(Data[i][0])
                                value2=int(Data[i+1][0])
                                if value1!=value2+1:
                                        closePrice=float(Data[i+1][2])
                                        openPrice=float(Data[i][4])
                                        break
                                i=i+1
```

```
        else:
                File = open('data'+f1+'.csv')
                Reader = csv.reader(File)
                Data = list(Reader)
                closePrice=float(Data[2][4])
                openPrice=float(Data[1][1])
        return 100*((openPrice-closePrice)/closePrice)
        '''
        return 5



def getBeta(Data,SPYReturn,daysBack,closePos):
        averageReturn=0
        skip=1
        Data=Data[:int(daysBack)]
        Data.reverse()
        #print Data
        SPYReturn=SPYReturn[1:]

        SPYReturn=SPYReturn.split(' ')
        #SPYReturn.pop(len(SPYReturn)-1)
        #SPYReturn.pop(0)
        stockReturn=[]
        for x in xrange(int(daysBack)-1):

stockReturn.append(100*((float(Data[x+1][closePos])-float(Data[x][closePos]))/float(Data[x+1]
[closePos])))
                SPYReturn[x]=float(SPYReturn[x])
        SPYmean=np.mean(SPYReturn)
        stockMean=np.mean(stockReturn)
        theSum=0
        for x in xrange(int(daysBack)-1):
```

```
            theSum=theSum+(((SPYReturn[x] - SPYmean) * (stockReturn[x] - stockMean)))
        covarience=theSum/(daysBack-1)
        return covarience/np.var(SPYReturn)

def getRegData(Data,tick,ticksBack):
        xSum=0
        ySum=0
        xySum=0
        global intercept
        global b
        n=int(ticksBack)
        xsquredSum=0
        Data=Data[:n]
        valueList=[]
        for x in xrange(len(Data)):
            valueList.append(float(Data[x][2]))
        valueList.reverse()

        for y in xrange(len(valueList)):
            ySum+=float(valueList[y])
            xSum+=(y)
            xySum+=(y)*float(valueList[y])
            xsquredSum+=(y)*(y)
        b=((n*xySum)-(xSum*ySum))/((n*xsquredSum)-(xSum*xSum))
        a=(ySum-(b*xSum))/n

        varience=0
        SStot=0
        mean=np.mean(valueList)
        for y in xrange(len(valueList)):
            predictedVal=a+(b*y)
            actualVal=float(valueList[y])
            val=actualVal-predictedVal
```

```
        varience+=(val*val)
        val2=actualVal-mean
        SStot+=(val2*val2)

global twoSD
SSres=varience
varience=varience/(n-2)
SD=math.sqrt(varience)
twoSD=SD*2

channelWidth=twoSD*2
channelWidthPercent=100*(channelWidth/np.mean(valueList))
RR=1-(SSres/SStot)
slopePercent=100*(b/np.mean(valueList))
intercept=a
'''
print 'RR: '+str(RR)
print 'Sum of Squares: '+str(SStot)
print 'varience: '+str(varience)
print 'slope: $'+str(b)
print 'slope %: '+str(100*(b/np.mean(valueList)))+'%'
print 'mean: $'+str(np.mean(valueList))
print 'SD: $'+ str(SD)
print '2SD: $'+str(twoSD)
print 'channel width: $'+str(channelWidth)
print 'channel width %: '+str(channelWidthPercent)+'%'

#y=a + bx
#y=201.37+.01523x
print 'a: '+str(a)
print 'b: '+str(b)
'''
return RR,slopePercent,channelWidthPercent
```

```
def checkArg(arg):
	if arg!='':
		return float(arg)
	return arg


def getCorrelation(Data,ticksBack):
	File = open('SPY.csv')
	Reader = csv.reader(File)
	SPYdata = list(Reader)
	ticksBack=int(ticksBack)
	SPYdata=SPYdata[7:]
	SPYdata=SPYdata[-ticksBack:]
	Data.reverse()
	Data=Data[-ticksBack:]
	iterations=ticksBack
	stockList=[]
	spyList=[]
	for x in xrange(iterations-1):
			valx=(float(Data[x+1][2])-float(Data[x][2]))/float(Data[x][2])
			valy=(float(SPYdata[x+1][2])-float(SPYdata[x][2]))/float(SPYdata[x][2])
			valy=valy*abs(betaValue)
			stockList.append(valx)
			spyList.append(valy)
	meanX=np.mean(stockList)
	meanY=np.mean(spyList)
	xySum=0
	sX=0
	sY=0
	for n in xrange(ticksBack-1):
		xVal=stockList[n]-meanX
		yVal=spyList[n]-meanY
```

```
                xySum+=xVal*yVal
                sX=sX+(xVal*xVal)
                sY+=(yVal*yVal)
        sX=math.sqrt(sX/(ticksBack-2))
        sY=math.sqrt(sY/(ticksBack-2))
        rSquared=(1/(float(ticksBack)-2)*(xySum/(sX*sY)))
        return rSquared
        #return str(xySum)+' '+str(sY)+' '+str(sX)+ ' '+ str(1/(float(ticksBack)-2))

class Node():
    def __init__ (self, parent,children,nodeType,value):
        self.parent=parent
        self.children=children
        self.nodeType=nodeType
        self.value=value

def runTree(Start):
    global theValue
    if Start.nodeType=='choice':
        nextNode=choosePath(Start,0)
        nextNode.value+=Start.value
        runTree(nextNode)
        return 0
    if Start.nodeType=='chance':
        for child in Start.children:
            try:
                Start.value+=runTree(child)
            except TypeError:
                error='catch'
    if Start.nodeType=='end':
        return Start.value
    theValue+=Start.value
```

```
def choosePath(node,path):
    if path==0:#seeing which choice we are at
        position=getPosition()
        if position==0:
            stayAboveUpper.value,probability=stayAboveUpperVal()
            goBelowUpper.value=goBelowUpperVal(probability)
            #print 'above channel'
            return aboveChannel
        if position==1:
            goAboveLower.value,probability=goAboveLowerVal()
            stayBelowLower.value=stayBelowLowerVal(probability)
            #print 'below channel'
            return belowChannel
        if position==2:
            breakUpper.value,probability=breakUpperVal()
            tapUpper.value=tapUpperVal(probability)
            #print 'upperMiddle'
            return upperMiddle
        if position==3:
            tapLower.value,probability=tapLowerVal()
            breakLower.value=breakLowerVal(probability)
            #print 'lowerMiddle'
            return lowerMiddle
        if position==4:
            stayMiddle.value=stayMiddleVal()
            #print 'middle'
            return Middle


    return aboveChannel

def getData(f1,f2):
    File = open('data'+f2+'.csv')
    Reader = csv.reader(File)
```

```
    Data = list(Reader)
    Data=Data[-tickAmount:]
    valList=[]
    for x in xrange(len(Data)):
            try:
            valList.append(float(Data[x][2]))
        except:
            pass
    return valList

def makeRegressionModel(Data,ticksBack):
    xSum=0
    ySum=0
    xySum=0
    n=ticksBack
    xsquredSum=0
    valueList=Data[-n:]
    for y in xrange(n):
            skip=0
            try:
            ySum+=float(valueList[y])
        except:
            skip=1
            pass
        if(skip==0):
            xSum+=(y)
            xySum+=(y)*float(valueList[y])
            xsquredSum+=(y)*(y)
    b=((n*xySum)-(xSum*ySum))/((n*xsquredSum)-(xSum*xSum))
    a=(ySum-(b*xSum))/n
    varience=0
    SStot=0
    mean=np.mean(valueList)
```

```
    for y in xrange(len(valueList)):
        predictedVal=a+(b*y)
        actualVal=float(valueList[y])
        val=actualVal-predictedVal
        varience+=(val*val)
        val2=actualVal-mean
        SStot+=(val2*val2)

    SSres=varience
    varience=varience/(n-2)
    SD=math.sqrt(varience)
    requestedSD=SD*2
    channelWidth=requestedSD*2
    channelWidthPercent=100*(channelWidth/np.mean(valueList))
    RR=1-(SSres/SStot)
    slopePercent=100*(b/np.mean(valueList))
    return a,b,requestedSD,RR

def getPosition():
    global intercept
    global slope
    global twoSD
    upperChannelStart=intercept+twoSD+(slope*tickAmount)
    halfUpperChannelStart=intercept+(twoSD/2)+(slope*tickAmount)
    lowerChannelStart=intercept-twoSD+(slope*tickAmount)
    halfLowerChannelStart=intercept-(twoSD/2)+(slope*tickAmount)
    #print upperChannelStart
    if Data[-1]>=upperChannelStart:
        return 0
    if Data[-1]<=lowerChannelStart:
        return 1
    if Data[-1]>=halfUpperChannelStart:
        return 2
```

```
    if Data[-1]<=halfLowerChannelStart:
        return 3
    return 4

def stayAboveUpperVal():
    global slope
    global intercept
    global twoSD
    global RR
    slopePercent=100*(slope/np.mean(Data))
    oneSDPercent=100*(twoSD/intercept)/2
    upTo=(tickAmount*.20)
    value=(upTo*slopePercent)+(oneSDPercent)
    theSlope,probability=sumRegressionLines()
    #print 'val1: '+str(value)
    #print 'probability: '+str(probability)
    #print 'BreakVal: '+str(value*probability)
    return value*probability,probability

def goBelowUpperVal(probability):
    p=1-probability
    #print 'tapValue: '+str(-4*p)
    return -4*p

def tapUpperVal(probability):
    p=1-probability
    #print 'tapValue: '+str(-4*p)
    return -4*p

def breakUpperVal():
    global slope
    global intercept
    global twoSD
```

```
    global RR
    slopePercent=100*(slope/np.mean(Data))
    oneSDPercent=100*(twoSD/intercept)/2
    currentSpot=Data[-1]
    upTo=tickAmount+(tickAmount*.20)
    theSlope,probability=sumRegressionLines()
    theSlope=100*theSlope/np.mean(Data)
    topChannel=intercept+(tickAmount*slope)+twoSD
    currentSpot=Data[-1]

value=(100*(topChannel-currentSpot)/currentSpot)+(slopePercent*(.20*tickAmount))+oneSDPe
rcent
    return value*probability,probability


def tapLowerVal():
    global slope
    global intercept
    global twoSD
    global RR
    slopePercent=100*(slope/np.mean(Data))
    oneSDPercent=100*(twoSD/intercept)/2
    topChannel=intercept+(tickAmount*slope)+twoSD
    currentSpot=Data[-1]
    value=(100*(topChannel-currentSpot)/currentSpot)+(slopePercent*(.20*tickAmount))
    theSlope,probability=sumRegressionLines()
    #print 'SD: '+str(oneSDPercent)
    #print 'RR: '+str(RR)
    #print 'value: '+str(value)
    #print 'probability: '+str(probability)
    #print 'tapVal: '+str(value*probability)
    return value*probability,probability


def breakLowerVal(probability):
```

```
    p=1-probability
    #print 'breakValue: '+str(-4*p)
    return -4*p

def stayMiddleVal():
    global slope
    global intercept
    global twoSD
    global RR
    slopePercent=100*(slope/np.mean(Data))
    oneSDPercent=100*(twoSD/intercept)/2
    theSlope,probability=sumRegressionLines()
    upTo=tickAmount+(tickAmount*.20)
    topChannel=intercept+(tickAmount*slope)+twoSD
    currentSpot=Data[-1]
    value1=(100*(topChannel-currentSpot)/currentSpot)+(slopePercent*(.20*tickAmount))
    value2=slopePercent*(.20*tickAmount)
    value3=-4
    p1=probability
    p3=getLowerProb(twoSD)
    p2=1-p1-p3
    value=(value1*p1)+(value2*p2)+(value3*p3)
    #print 'value: '+str(value)
    #probability=abs(momentum+.5)
    #print 'probability: '+str(probability)
    return value

def stayBelowLowerVal(probability):
    p=1-probability
    #print 'breakValue: '+str(-4*p)
    return -4*p

def goAboveLowerVal():
```

```
    global slope
    global intercept
    global twoSD
    global RR
    slopePercent=100*(slope/np.mean(Data))
    upTo=tickAmount+(tickAmount*.20)
    oneSDPercent=100*(twoSD/intercept)/2
    theSlope,probability=sumRegressionLines()
    topChannel=intercept+(tickAmount*slope)+twoSD
    currentSpot=Data[-1]

value=(100*(topChannel-currentSpot)/currentSpot)+(slopePercent*(.20*tickAmount))-oneSDPe
rcent
    return value*probability,probability



def sumRegressionLines():
    slopeAverage=0
    global slope
    count=0
    for x in xrange(int(tickAmount*.15)):
        intercept,theSlope,twoSD,RR=makeRegressionModel(Data,x+3)
        slopePercent=100*(theSlope/np.mean(Data))
        slopeAverage+=slopePercent
        if slope<0:
            if theSlope>0:
                    count+=1
        else:
            if theSlope>slope:
                    count+=1
    return slopeAverage/(tickAmount*.15),count/(tickAmount*.15)

def getLowerProb(SD):
```

```
        slopeAverage=0
        global slope
        count=0
        SD=SD/4
        for x in xrange(int(tickAmount*.15)):
                intercept,theSlope,twoSD,RR=makeRegressionModel(Data,x+3)
                slopePercent=100*(theSlope/np.mean(Data))
                slopeAverage+=slopePercent
                if slope<0:
                        if theSlope<(0-SD):
                                count+=1
                else:
                        if theSlope<(slope-SD):
                                count+=1
        return count/(tickAmount*.15)

Start=Node('NULL',[],'choice',0)
aboveChannel=Node(Start,[],'chance',0)
belowChannel=Node(Start,[],'chance',0)
upperMiddle=Node(Start,[],'chance',0)
Middle=Node(Start,[],'chance',0)
lowerMiddle=Node(Start,[],'chance',0)
Start.children=[aboveChannel,belowChannel,upperMiddle,Middle,lowerMiddle]

stayAboveUpper=Node(aboveChannel,[],'end',0)
goBelowUpper=Node(aboveChannel,'NULL','end',0)
aboveChannel.children=[stayAboveUpper,goBelowUpper]

stayBelowLower=Node(belowChannel,'NULL','end',0)
goAboveLower=Node(belowChannel,'NULL','end',0)
belowChannel.children=[stayBelowLower,goAboveLower]

tapUpper=Node(upperMiddle,'NULL','end',0)
```

```
breakUpper=Node(upperMiddle,'NULL','end',0)
upperMiddle.children=[tapUpper,breakUpper]

stayMiddle=Node(Middle,'NULL','end',0)
Middle.children=[stayMiddle]

tapLower=Node(lowerMiddle,'NULL','end',0)
breakLower=Node(lowerMiddle,'NULL','end',0)
lowerMiddle.children=[tapLower,breakLower]

def main():
        f1=sys.argv[1]
        f2=sys.argv[2]
        lowerVol=sys.argv[3]
        upperVol=sys.argv[4]
        lowerBeta=sys.argv[5]
        upperBeta=sys.argv[6]
        lowerPrice=sys.argv[7]
        upperPrice=sys.argv[8]
        SPYReturn=sys.argv[9]
        tick=sys.argv[10]
        daysBack=sys.argv[11]
        lowerR2=sys.argv[12]
        upperR2=sys.argv[13]
        lowerSlope=sys.argv[14]
        upperSlope=sys.argv[15]
        lowerChannel=sys.argv[16]
        upperChannel=sys.argv[17]
        Open=sys.argv[18]
        lowerOpen=sys.argv[19]
        upperOpen=sys.argv[20]
        percentAbove=sys.argv[21]
        percentBelow=sys.argv[22]
```

```
        SpyHigh=float(sys.argv[23])
        SpyClose=float(sys.argv[24])
        SpyLow=float(sys.argv[25])

        #print 'lower open: '+str(lowerOpen)
        #exit(0)

        if daysBack!='Day':
                daysBack=checkArg(daysBack)
        lowerVol=checkArg(lowerVol)
        upperVol=checkArg(upperVol)
        lowerBeta=checkArg(lowerBeta)
        upperBeta=checkArg(upperBeta)
        lowerPrice=checkArg(lowerPrice)
        upperPrice=checkArg(upperPrice)


passed=checkstock(f1,f2,lowerVol,upperVol,lowerBeta,upperBeta,lowerPrice,upperPrice,SPYR
eturn,tick,daysBack,

lowerR2,upperR2,lowerSlope,upperSlope,lowerChannel,upperChannel,Open,lowerOpen,upperO
pen,percentAbove,percentBelow)
        if passed==1:
                print passed
                print str(int(volAverage))
                print np.round(priceAverage,3)
                print np.round(betaValue,3)
                print np.round(R2,3)
                print np.round(slope,3)
                print np.round(channelWidth,3)
                print np.round(opened,3)
                global Data
                global intercept
                global slope
```

```
            global twoSD
            global RR
            Data=getData(f1,f2)
            intercept,slope,twoSD,RR=makeRegressionModel(Data,tickAmount)
            runTree(Start)
            print np.round(theValue,3)
            print np.round(correlation,3)
            partOne=theValue*(1-abs(correlation))
            if betaValue>=0:
                  partTwo=betaValue*(SpyHigh-SpyClose)*abs(correlation)
            else:
                  partTwo=betaValue*(SpyClose-SpyLow)*abs(correlation)
            ratingTwo=partOne+partTwo
            print np.round(ratingTwo,3)
      else:
            print passed

if __name__ == "__main__": main()
```