Bachelor Thesis

Examinations of Stock Trading Algorithms and Applications of Twitter Sentiment for Trading Strategies

Alex Thomas

Date: May 7, 2019

Advisor: Prof. Darren Strash, Prof. Aaron Gember Jacobson

Technical Report: COSC-TR-2019-##

Department of Computer Science
Colgate University
Hamilton, New York

Abstract

Algorithmic Trading (AT) is a financial sector that trades financial instruments, such as stocks, with algorithms and no human interaction. This allows the largest prop-trading firms in the world to conduct thousands of trades per second. In practice, the vast majority of strategies are implemented using mathematical formulas based on a variety of stock metrics, such as closing price or volume. However, the effectiveness of these algorithms isn't publicly available due to the necessity of secrecy of implementation details. Part of the thesis aims to uncover and examine the effectiveness of existing metrics-based strategies. We find both pairs trading and a combined RSI and MACD momentum algorithm to be incredibly effective.

Moving beyond traditional AT strategies, this thesis further aims to investigate using news- or media-content-based strategies. By using Twitter data and Natural Language Processing (NLP), we create a unique trading strategy based on Twitter sentiment of publicly traded companies' tweets using a bevy of machine learning algorithms and a deep learning algorithm. We use models which have simplistic or extended features and apply these features to a stacking model and a deep learning model. We find the simplistic model to be very ineffective while the extended model beats the baseline measure for 87.5% of stocks tested, generating profits of up to 529%. Our stacked model beats the baseline for 62.5% of stocks tested and proves to be very effective for specific stocks. Our deep learning model is far more risk averse than any of the other models explored.

Acknowledgments

I would like to thank both of my thesis advisors Prof. Darren Strash and Prof. Aaron Gember-Jacobson. The former had an incredibly involved role throughout my independent study and development of my thesis. I would not have been able to complete this thesis without his organization, incredible experience, and mentorship. I have had the privilege of being advised by Prof. Gember-Jacobson for both this thesis and as my computer science advisor. I would like to thank him for closely guiding me as I wrote this thesis and the hours spent helping me with my defense. Both have taught me in the classroom and have had profound impact on my experience as a computer science major, as well as giving industry guidance for my pursuits after Colgate. I cannot stress enough how lucky I am to have had both professors acting as my advisors. I also would like to thank Prof. Will Cipoli. He provided valuable consultation related to machine learning algorithms and feedback on my various models implemented for my Twitter sentiment trading contributions. I would additionally like to thank my parents, friends, and family for allowing me to complete this thesis.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . .
(Alex Thomas)

Contents

1 Introduction

Algorithmic Trading (AT) is the generation and submission of orders of a financial asset by an algorithm, or set of instructions [28]. The trading of stocks is one of the most important and prominent profit making utilities in the world. The introduction of 401k's in 1978 into employment benefits has caused the majority of families in the United States to be invested in the stock market [27], allowing millions of people to maximize longterm savings [28]. Each individual's portfolio has a unique distribution of stocks. Because the average person doesn't have enough knowledge to day trade productively, this has created the need for owning stocks that provide safe and constant returns. This is realized in mutual funds, which are collections of funds invested into a variety of different stocks and often compose large percentages of 401k portfolios. These funds often utilize algorithmic trading to produce maximum profit, motivating our study to understand how these strategies work.

To enter into the stock market one must purchase a specified number of shares of a particular security or mutual fund, which consist of companies, individuals, institutions, currencies, and more. However, this is a two-way transaction. To purchase a specific number of shares at a particular price, another party must agree to sell at least as many shares at that same price. This happens via broker or in modern times, via computer. From the trader's standpoint, it is as simple as logging onto a stock trading website and clicking "execute trade".

There are many different types of transactions in financial markets. *Buying* and *selling* of shares are among the most common stock orders. *Limit orders* are buy and sell transactions that are only executed at a given limit price threshold. These orders are generally used to minimize loss and risk as trades will only occur when specific market conditions hold [2]. *Shorting* of stocks works the opposite way of buying. With shorting, the trader is betting against the performance of the stock. The short sellers borrow shares of stock they don't own and sell them at market price. The goal is to re-buy the stock at a later date and return the borrowed shares to the lender by repurchasing the stocks at a lower price than the initial purchase. *Options* trading has gained recent popularity and is comprised of call and put options. This is a derivative type of security, as it is intrinsically linked to the price of something else. *Call options* give the owners the right to buy stock at a certain price, while *put options* gives the holder the right to sell stock at a certain price. Here, we are chiefly concerned with buying and selling transactions, however further study could examine some of the other types of orders mentioned above.

The intended outcome for any trade or transaction is always the same: to maximize profit. However, it is often difficult to pinpoint the exact moment in time that a particular trade maximizes profit. As computers and network connections have improved, trading financial instruments via automation has become more prominent. By using advanced mathematical models and measures, automated financial trading aims to maximize profit, executing up to thousands of trades a day of a particular security. While not often widely publicized, millions of trades each day are made by computer algorithms. In 2011, over 73% of all equity trading volume in the U.S. was performed algorithmically [28]. However, much of this trading is done by large financial institutions or prop-trading firms. Few studies look at the efficacy and inner workings of prominent trading algorithms [10] [13].

The stock market at its inception was entirely analog and trades of stock were carried out in person. Rapid trading traces its roots back to the early 1930s. Specialists and pit traders bought and sold positions and broadcasted trades via new high speed telegram services [28]. Computerization of trades started in the 1980s when the NASDAQ was the first exchange to introduce purely electronic trading. This trend of stock markets moving electronically opened the gate for the use of computers to automatically generate buy and sell signals without human interaction. Today, trading time has changed from a matter of seconds to microseconds [28].

Just like the stock market moving entirely electronic, modern news has also moved entirely online. Social media or news websites, like Twitter or Facebook, are an effective way for a company to share news in the modern day. They allow companies to make global announcements in seconds. Behavioral finance suggests that emotions, mood, and sentiments in response to news play a significant role in investment and affect the price of stocks. Numerous trading strategies have looked to leverage news sentiment from social media websites, most notably Twitter, for AT [15].

However, AT has not always had a positive impact on the market. There have been some negative consequences. The May 6th, 2010 “Flash Crash” brought the public’s attention to the little publicized, but very heavily used algorithmic trading in financial markets [18]. This happened with E-mini, denoted by ES, which is a stock market index futures contract that trades for around 50 times the value of the *Standard’s & Poor’s 500 Index* (S&P). A mutual fund complex sold 75,000 of these contracts valued at approximately $4.1 billion - resulting in the largest net change in daily position of any trader in the E-mini since the beginning of the year. This caused a cascading effect, as other traders reacted to this massive momentary plunge and sold accordingly, with over 20,000 trades across 300 separate securities executing at prices 60 percent away from their initial prices a mere half hour earlier. The *Dow Jones Industrial Average* (DJIA) fell over 1,000 points in moments, causing over $1 trillion to evaporate within 10 minutes.

1.1 Motivation

AT is the generation and submission of orders of a financial asset by an algorithm, or set of instructions, that processes current market data and places orders in stock marketplaces without human interaction. More formally, Chaboud et al. [7] define AT as:

> "In algorithmic trading (AT), computers directly interface with trading platforms, placing orders without immediate human intervention. The computers observe market data and possibly other information at very high frequency, and, based on a built-in algorithm, send back trading instructions, often within milliseconds. A variety of algorithms are used: for example, some look for arbitrage opportunities, including small discrepancies in the exchange rates between three currencies; some seek optimal execution of large orders at the minimum cost; and some seek to implement longer-term trading strategies in search of profits."

High Frequency Trading (HFT), is a subset of algorithmic trading and a far newer phenomenon that has been made possible by the rapid improvement of computerized trading speed [14]. This is the primary form of algorithmic trading found in financial markets today, with billions of dollars constantly traded by machines every second. However, it is difficult for trading of this type to occur at an individual level. Companies have the capital and resources to pay millions of dollars for their own fiber-optic cable connections across the United States and into Wall Street and up to the millisecond stock data whereas the individual doesn't.

This provides an interesting conundrum for the individual trader. Much of AT done in the market is done by large banks or massive prop-trading firms trading billions of dollars of existing capital per day. Is it possible for an individual with much less capital to engage in AT and be significantly profitable? With the current market trend of no-commission brokers such as Robinhood or developer friendly API's such as Alpaca, it has become ever the more possible for individuals to become algorithmic-traders [4]. However, what strategies can be implemented to obtain profits? Is it feasible to use small amounts of capital to have significant returns? While sentiment plays a role in the stock market, is it possible to quantify via Twitter and build a model that predicts stock price? All of these questions provide motivation for this study, but in particular, the last question is of utmost importance. Social networks have provided large companies a platform for mass public communication with the entire planet. This data is publicly available and could provide helpful insight into trading decisions.

1.2 Our Contribution

Because AT is an unsurprisingly secretive industry, we look to examine the effectiveness of both typical and atypical trading strategies. We give an in-depth analysis of a variety of both AT and HFT strategies. Specifically, we examine momentum, arbitrage and mean reversion measures and methods [2]. While many of these methods can be applied to other financial markets, such as Bitcoin markets, we use data exclusively from the US stock market. We develop a test suite that runs a variety of different algorithmic trading strategies on two different periods of data. Some of the strategies tested focus on HFT and use intra-day, minute-by-minute stock data. Others look at a period of over 7 years of closing price data. By using different types of data, we are able to examine strategies in both an AT and a HFT context.

Additionally, we contribute unique trading algorithms using both machine and deep learning using Twitter *sentiment* data. We collect tweets from publicly traded companies twitter pages from the inception of their account until 2017. We then using Natural Language Processing (NLP) to construct average sentiment values for tweets on each day. While we find weak linear correlation with price and average tweet sentiment, we expect more advanced models that use machine learning to be able to more accurately apply this data. We investigate the effectiveness and fit of machine learning classifiers and regressors, such as decision trees, k-nearest neighbors, MLPs, and random forests. We also investigate the effectiveness of LSTM neural networks using deep learning.

1.3 Organization

In Chapter 2, we take a deep dive into existing literature on AT and sentiment analysis applied to stock trading. In Chapters 3 and 4 we examine and test the effectiveness of a variety of different existing AT strategies. In Chapter 5 we investigate applying Twitter sentiment data to trading strategies using different machine learning techniques. In Chapter 6 we give extensive experimental results to understand the effectiveness of our unique Twitter sentiment strategies. We conclude in Chapter 7 and propose future work to improve the robustness of our findings. The appendix in Chapter 8 contains pseudocode implementations of our existing AT strategies.

2 Related Work

Many researchers have studied AT and HFT in the context of specific algorithms, the state of the industry, and greater topics [28]. Because of the secretive nature of AT due to the effects of money making, research on specific algorithms often doesn't include specific implementations or pseudocode. Algorithms lose their advantage when made publicly available, as other influencers acting on the same signals decrease profit margins.

2.1 Algorithmic Stock Trading

Algorithmic stock trading is both an academic discipline and a multi-billion dollar industry. Academically, it is often studied in a general context. Specifically, research highlights how AT has a dysfunctional role in the stock market or may take a general look at how HFT can undercut normal trading. For example, articles review components of AT systems, which generally includes data access, cleaning, pre-trade analysis, trading signal generation, trade execution, and post-trade analysis [28]. Other research takes a deep dive into particular experimental algorithms, like *gated Bayesian networks* (GBN) [5]. GBN's are models that combine several Bayesian networks in a manner that acts like gates – making specific networks inactive or active based on logic. This method is applied to AT systems and performs better than the benchmark while simultaneously substantially decreasing risk of invested capital.

However, the research also has very practical applications. Numerous textbooks exist that take a very general, overarching dive into algorithmic-trading. Aldridge's textbook [2] looks at market structure, HFT data fetching, risk management of HFT, and a variety of naïve strategies, such as market making or statistical arbitrage. Gomber et al. [14] focus on the evolution of AT and take a look at very specific regulations in European and American markets. These practical applications all reinforce the barriers of entry for an individual trader, as acquiring raw and cleaned data is highly expensive due to the time sensitivity of trading algorithms [28].

2.2 Sentiment Analysis, Twitter, and Stock Trading

The stock market often behaves as a function of news sentiment. Behavioral finance suggests that emotions, mood, and sentiments in response to news play a significant role in investment [15]. Particularly, Ho and Wang identify that sentiment has the most significant impact on investors in the market [15]. Take the GE downswing on April 8, 2019 as an applicable case study. GE's stock plunged over 6% in pre-market trading as an influential J.P. Morgan analyst slashed his rating on the stock [26]. Significant actors can have major effects on the entire market purely based on sentiment of the actor's remarks.

2.2.1 Sentiment Analysis Application to Machine Learning and Classification

Previous literature often uses machine learning for sentiment classification problems. Pang et al. [23] employ different machine learning methods on movie review data and find that while it does vastly outperform human-produced baseline, it is not as nearly as effective on traditional topic-based categorization. Their work stems from the issue of automatically classifying the vast amount of online text documents' sentiment, which would be incredibly useful for business intelligence applications and review sites. Using naïve Bayes, a maximum entropy model, and support vector machines (SVM), they find that they aren't able to produce very accurate topic-based categorization results. They estimate that a large problem with their sentiment analysis is deliberate contrast in the data, which humans can easily discern unlike machines. Their paper shows how even though machine learning is often relied on for classification and does outperform the baseline, it isn't incredibly effective. These discrepancies could potentially lead to challenges for our own methods.

2.2.2 Past Examinations of Sentiment and Stock Trading

Using sentiment as an indicator for stock trading has been heavily studied in academia. Li et al. study news impact on stock price using sentiment analysis [19]. Because financial news articles are believed to have an impact on stock price return, they use news sentiment to implement a stock price prediction framework using the Harvard psychological dictionary and the Loughran-McDonald financial sentiment dictionary. This sentiment analysis framework outperforms previous *bag-of-words* stock price prediction frameworks [19]. This literature demonstrates how news impacts the stock market and can be used to predict stock price movement, which is at the crux of our study.

Financial markets outside of the major United States heavyweights like the DJIA have also been studied using sentiment analysis. Garcia et al. use social signals to create a Bitcoin trading algorithm that is highly profitable [12]. They integrate various datasources that provide social signal data on information search, word of mouth volume, emotional valence and opinion polarization in the analysis of Bitcoin. Cryptocurrency is intriguing to study due to its extreme volatility and sustained price fluctuations. Their analysis reveals that increases in opinion polarization and exchange volume precede rising Bitcoin prices and are able to create a Bayesian model that is highly profitable. Shah et al. [25] discuss the efficacy of Bayesian regressions on predicting Bitcoin price variation. They use a latent-source model for the purpose of binary classification, which has been previously established as effective for this specific scenario. They are able to predict price change every 10 seconds accurately enough to double the initial investment in less than 60 days. While these studies consider use of cryptocurrency markets, it still proves that financial instruments are linked with sentiment.

2.2.3 Past Examinations of Twitter as a Valid Means of Stock Trading

Past literature has extensively examined using Twitter as a means of data for trading algorithms. Mao et al. [22] use Twitter volume spikes to identify wether or not they are useful for stock trading. They find that Twitter volume spikes are correlated with stock price movement, acting as a surprise to market participants based on implied volatility. The authors use a Bayesian classifier to develop a trading strategy that has significant returns in a short period of time. Zhang et al. [29] look to predict stock market indicators through Twitter. After collecting randomized twitter posts for six months, the researchers measure collective hope and fear measures each day and analyze the correlation between those and stock market indicators. They find that emotional tweet percentage is significantly negatively correlated with all of the major United States stock markets. Bollen et al. [6] use Twitter mood to predict the movement of the stock market by classifying specific tweets into six different categories or moods. The authors use a Granger causality analysis and a self-organizing fuzzy neural network to investigate how indicative public mood states are of the DJIA closing prices. They use large-scale Twitter feeds and determine the mood of tweets over 6 dimensions and find that Twitter mood can predict the movement of the DJIA with incredible accuracy. All of these works suggest that Twitter activity can be applied to stock prices and thus used in trading algorithms, which is precisely what we aim to exploit in our study.

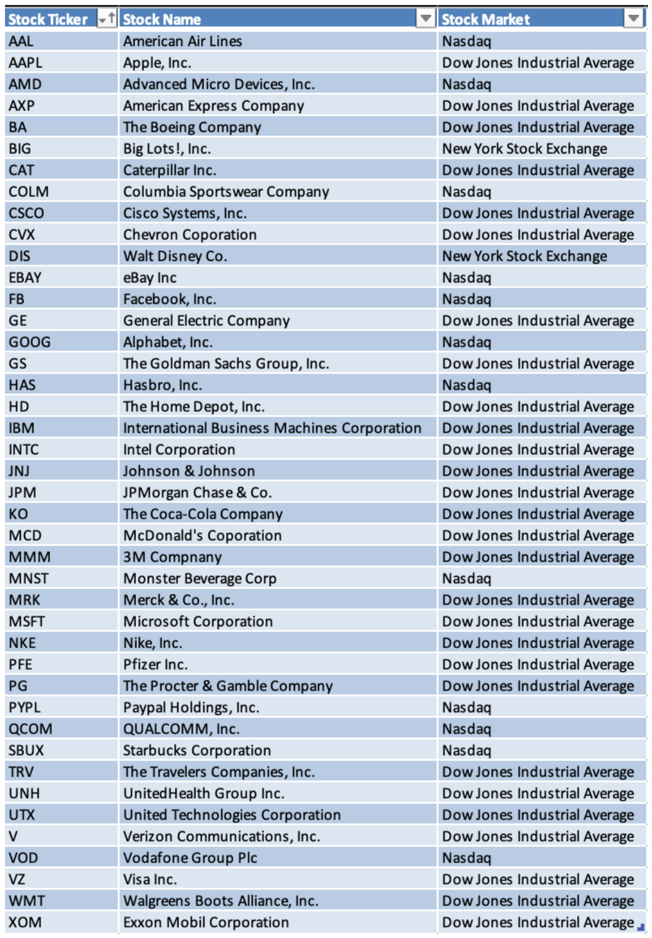
3 Examination of Trading Algorithms

In this chapter we examine numerous algorithmic trading methods. We discuss stock trading terms and data before moving into different momentum based strategies followed by a discussion on pairs trading. Chapter 4 examines the effectiveness of these strategies discussed below.

3.1 Stock Trading and Data

We use Quandl, a financial data platform, to obtain the different stock ticker data. Specifically, we used the *WIKI* and *AS500* datasets to acquire stock data. The former contains end-of-day stock pricing data while the latter contains intraday data, with updated pricing for each minute in a day of trading. While these datasets include numerous stock statistics, such as volume and adjusted price metrics, we only consider closing price for both of our datasets. The stocks used in all strategies are found in Figure 3.1. These stocks were chosen given the availability of NASDAQ and S&P500 stocks in the Quandl dataset and cover a variety of industries. This is important because we need to account for different performances of stocks across multiple industries. Certain large tech stocks like `FB` or `GOOG` have had incredible trajectories and performances whereas other stocks outside of the tech industry like `HAS`, have performed more moderately. The range of stocks chosen encompass nearly all of the major industries throughout the world.

Buy and *sell signals* are used throughout the study. Algorithms can generate a variety of buy and sell signals at a specific point in time, indicating that a denomination of stocks should be purchased or sold at that specific point in time. The strategies use closing price data from October 1st, 2006 to January 1st, 2017. Some strategies use intra-day stock data, which is minute-by-minute data of stock prices from October 26, 2018. Other additional terms need to be defined. *Securities* and *stocks* are used interchangeably throughout the paper and effectively have the same meaning. Specifically, securities have more of a broad definition, as securities include stocks, bonds, mortgages, and others. When discussing how stocks will move in the future, we use *bearish signals* to signal trends of downturn and *bullish signals* for positive, upward trends [2].



Stock Ticker	Stock Name	Stock Market
AAL	American Air Lines	Nasdaq
AAPL	Apple, Inc.	Dow Jones Industrial Average
AMD	Advanced Micro Devices, Inc.	Nasdaq
AXP	American Express Company	Dow Jones Industrial Average
BA	The Boeing Company	Dow Jones Industrial Average
BIG	Big Lots!, Inc.	New York Stock Exchange
CAT	Caterpillar Inc.	Dow Jones Industrial Average
COLM	Columbia Sportswear Company	Nasdaq
CSCO	Cisco Systems, Inc.	Dow Jones Industrial Average
CVX	Chevron Coporation	Dow Jones Industrial Average
DIS	Walt Disney Co.	New York Stock Exchange
EBAY	eBay Inc	Nasdaq
FB	Facebook, Inc.	Nasdaq
GE	General Electric Company	Dow Jones Industrial Average
GOOG	Alphabet, Inc.	Nasdaq
GS	The Goldman Sachs Group, Inc.	Dow Jones Industrial Average
HAS	Hasbro, Inc.	Nasdaq
HD	The Home Depot, Inc.	Dow Jones Industrial Average
IBM	International Business Machines Corporation	Dow Jones Industrial Average
INTC	Intel Corporation	Dow Jones Industrial Average
JNJ	Johnson & Johnson	Dow Jones Industrial Average
JPM	JPMorgan Chase & Co.	Dow Jones Industrial Average
KO	The Coca-Cola Company	Dow Jones Industrial Average
MCD	McDonald's Coporation	Dow Jones Industrial Average
MMM	3M Compnany	Dow Jones Industrial Average
MNST	Monster Beverage Corp	Nasdaq
MRK	Merck & Co., Inc.	Dow Jones Industrial Average
MSFT	Microsoft Corporation	Dow Jones Industrial Average
NKE	Nike, Inc.	Dow Jones Industrial Average
PFE	Pfizer Inc.	Dow Jones Industrial Average
PG	The Procter & Gamble Company	Dow Jones Industrial Average
PYPL	Paypal Holdings, Inc.	Nasdaq
QCOM	QUALCOMM, Inc.	Nasdaq
SBUX	Starbucks Corporation	Nasdaq
TRV	The Travelers Companies, Inc.	Dow Jones Industrial Average
UNH	UnitedHealth Group Inc.	Dow Jones Industrial Average
UTX	United Technologies Corporation	Dow Jones Industrial Average
V	Verizon Communications, Inc.	Dow Jones Industrial Average
VOD	Vodafone Group Plc	Nasdaq
VZ	Visa Inc.	Dow Jones Industrial Average
WMT	Walgreens Boots Alliance, Inc.	Dow Jones Industrial Average
XOM	Exxon Mobil Corporation	Dow Jones Industrial Average

Figure 3.1: Table of stocks used in our study

Graphs can be interpreted as follows. Figure 3.2 shows the "Golden Cross" strategy applied to AAPL. Purple triangles, which are angled upwards, are buy signals while black triangles angled downwards are sell signals. After about 400 days through the trading period, the strategy generated a buy signal, as denoted by the purple triangle. At about 500 days through the trading period, the following sell signal was generated. This generated a profit of almost $20 per share for this specific instance. The orange line is the long moving average, the blue line is the short moving average, and the red line gives the price of AAPL.

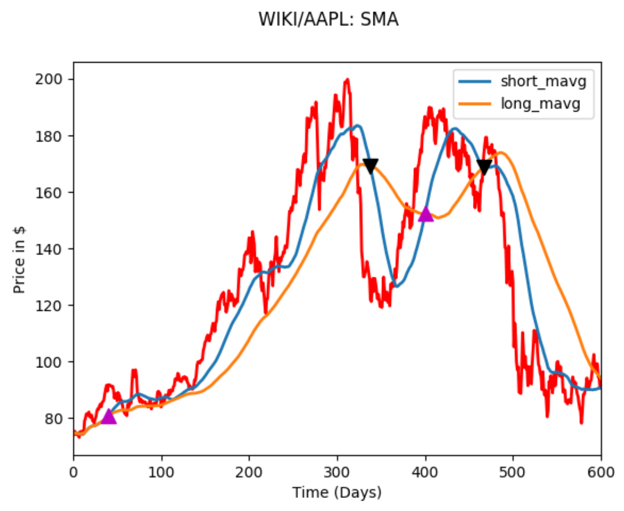


Figure 3.2: Simple Moving Average - "Golden Cross" Strategy applied to AAPL

3.2 Momentum Trading Strategies

Momentum indicators are AT tools used to measure the recent performance, or momentum, of a stock [8]. In this section, we examine trading strategies that leverage different momentum indicators to generate optimal buy and sell signals.

3.2.1 Simple Moving Average

Simple Moving Average (SMA) is an elementary AT measure. It is mostly used to measure average stock price over a period of time, however certain strategies solely rely on SMA. It looks at a rolling average of a specified window. Mathematically this can be defined as [3]:

$$SMA(t) = 1/n \sum_{i=t-n}^{t} x(i)$$

In words, this gets the average price over a specified window of time for a specified function. In the context of the stock market, SMA can be applied for both short term and long term averages, with the former under an hour while the latter can be hundreds of days or more. In application to the stock market and closing prices, this creates an average closing price for a specified amount of time, which gives an indicator of price swings in that period of time.

SMA is commonly leveraged into a strategy that uses a dual moving average, also known as the "Golden Cross" [3]. It works by taking two different SMA's - a short window and a long window. In our implementation, we chose a 40-day window and 200-day window, giving insight into a strategy that uses longer averages. The short window crossing below the long window gives a buy signal reinforced by high trading volumes. The long window crossing below the short window is considered bearish and gives a sell signal, as this signals that the stock is currently overvalued and will devalue. As demonstrated in Figure 3.2, the strategy generates about 30 buy or sell signals over the period tested.

3.2.2 Expected Moving Average

Expected Moving Average (EMA) is closely linked to SMA. Like SMA, EMA functions over a rolling window; however, it is calculated differently. Mathematically, EMA is calculated with F_i = the value associated with the moving average at period zero, α = smoothing constant, and X_i = closing price of the security at period i [17]:

$$F_i = F_{i-1} + \alpha(X_i - F_{i-1})$$

In words, the EMA is found by multiplying the closing price subtracted by the EMA from the previous day times a multiplier plus that prior day's average. This makes this measure far more weighted towards recent prices than SMA.

Like our implementation of SMA, EMA is commonly implemented with dual high and low value average. It takes two different EMA's, one with a low window and the second with a high window. However, because EMA reacts more closely to recent stock prices shorter windows are more commonly chosen. For our implementation, 12-day and 26-day windows were used. The short window crossing below the long window gives a buy signal reinforced by high trading volumes while the opposite is considered bearish and gives a sell signal. Figure 3.3 demonstrates the EMA trading strategy applied to `AAPL`. Because smaller windows are used than our SMA implementation, we see far more frequent buy and sell signals. Additionally, we see the short and long expected moving average lines follow the price line far closer than the SMA implementation.

3.2.3 Bollinger Bands

Bollinger Bands were introduced by John Bollinger in the 1980s [20]. This strategy uses three bands, with the two outer bands derived from a standard deviation of the moving average. They provide a relative definition for high and low stock prices. Just like the previously

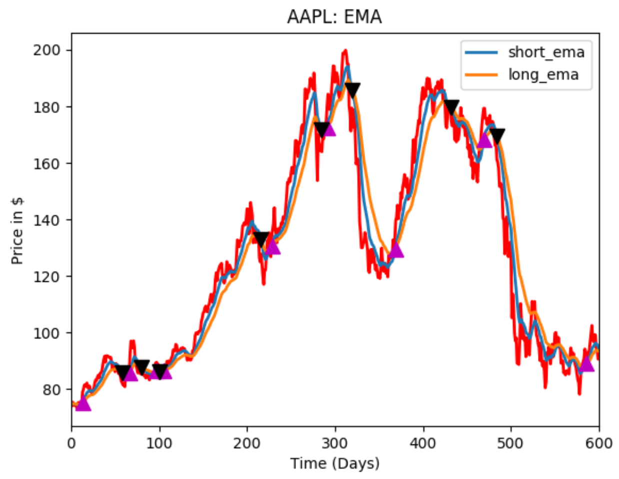


Figure 3.3: Dual Moving Average Strategy with Exponential effects applied to AAPL

examined techniques, the method uses a moving average as its basis, but instead incorporates 3 different bands separated by standard deviation. Because of this use of standard deviation, which is determined by market volatility, Bollinger Bands adjust themselves to market conditions. Precisely, the bands are calculated as follows with M - Middle Band, U - Upper Band, L - Lower Band, STD(x) - standard deviation of period x:

$$M = SMA(12), U = M + 2 * STD(12), L = M - 2 * STD(12)$$

When the market is more volatile the bands widen and when the market becomes less erratic, the bands move closer together [20]. Unlike the previous techniques, this strategy uses market conditions to evaluate trading orders.

This strategy is implemented with these 3 bands as mentioned above. For our implementation, a 12 day window was chosen to test out the algorithm, as this has been proven to be the most effective [20]. When the closing price drops below the lower band, this gives a buy signal, as once a lower band has been broken due to heavy selling, the stock price will revert back and head towards the middle band. The opposite is true for when the closing price breaks the upper band, as this is indicative of heavy buying. The closing price approaching the upper band gives a bearish signal while approaching the lower band gives a bullish signal. This strategy is showing in Figure 3.4. This measure can generate multiple buy or sell signals in a row. Looking at the range of days between 150 and 250, six straight sell signals are generated. In the range of days between 400 and 500, many buy signals are generated. Both of these trends are indicative of the market giving only bearish or bullish indicators stemming from the use of standard deviation measuring the volatility of price conditions.

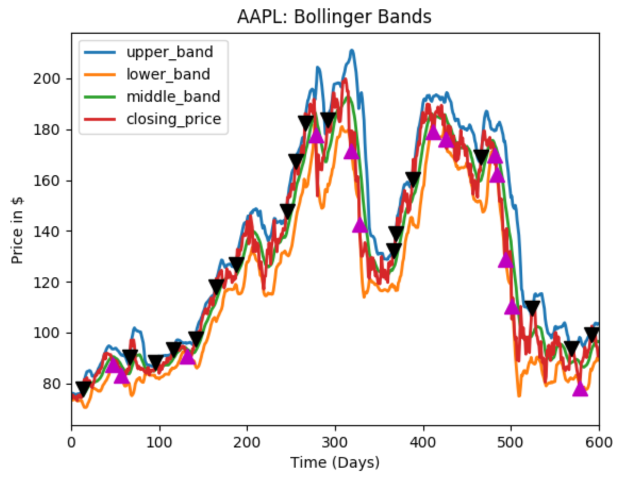


Figure 3.4: Bollinger Bands strategy applied to AAPL

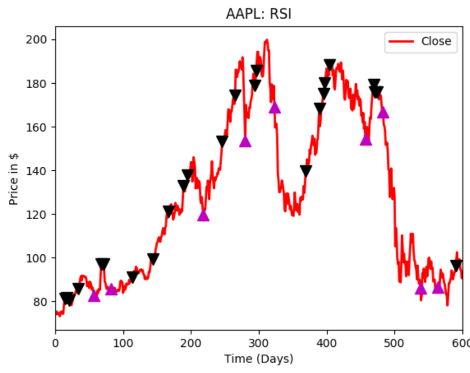
3.2.4 RSI - Relative Strength Index

Relative Strength Index (RSI) is a momentum indicator that measures the magnitudes of price changes to analyze overbought or oversold stocks. It demonstrates a particular security's recent performance over a relatively short window compared to the mean. This indicator is widely used today in algorithmic trading. The measure is a value between 0 and 100 at a specific date. The equation for RSI is as follows with RS defined as average gain of up periods divided by the average gain of down periods over a specified window x [8]:

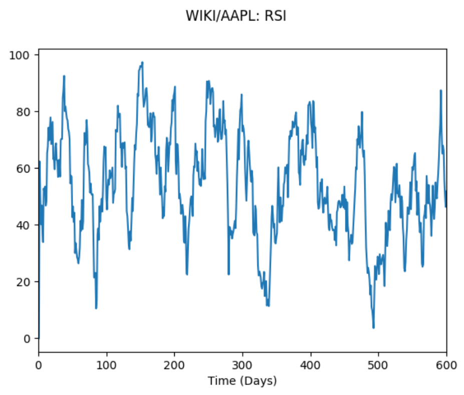
$$RSI = 100.0 - (100.0/(1.0 + RS(x))$$

Therefore, a large RSI value is indicative of stocks that have had recent larger gains compared to losses while a low RSI value is indicative of stocks with poor recent performance compared to the mean.

This strategy takes advantage of mean reversion. Our implementation uses a very simple method. Sell signals are generated when the RSI is over 70 and buy signals are generated when the RSI is under 30 [8]. This is very logical, as expected gains are the largest when a stock has performed poorly recently and is expected to revert back to the mean. This strategy also uses a 14 day window, which is standard across the industry. Figure 3.5 shows this strategy when run on AAPL. The second subfigure generates the buy and sell signals, as it shows the RSI of the stock throughout the period tested. These signals generated from the second subfigure are then plotted on the first subfigure over the stock price. Clearly, this strategy generates a large quantity of trading signals over the period.



(a) RSI Strategy



(b) RSI of AAPL

Figure 3.5: RSI strategy applied to AAPL

3.2.5 Combining Momentum Indicators - RSI and Moving Average Convergence-Divergence (MACD)

MACD uses two moving averages to identify trend changes while RSI performs exactly as stated in the above section. The MACD is constructed by subtracting two different sized exponential moving averages from each other. The equation for MACD is as follows and uses $EMA(x)$, with x representing the period in minutes [8]:

$$MACD = EMA(12) - EMA(16)$$

The MACD is then plotted against a signal line, which is defined as S and is the EMA of the 9 minute MACD:

$$S = EMA(MACD(9))$$

Buy and sell signals are then generated by the “Golden Cross" method. Specifically, this is when the MACD crosses the signal line from below - signaling a momentum swing and a bullish buy signal. A sell signal is the reverse - when the MACD crosses the signal live from above.

Unlike the previously mentioned strategies, this one combines indicators to generate more robust trading signals. Because there are two different measures working at the same time, buy signals are generated by both the RSI and MACD measures giving bullish signals within 3 minutes of each other. Sell signals are generated by either one of the measures producing a sell. Under this strategy, buy signals are very rarely generated while either strategy generating a sell denotes all shares to be sold. Because of the selectivity of buy signals, we can expect this technique to be very profitable. This strategy is most effective when run over intra-day data because of the relative infrequency of buy signals, as demonstrated in Figure 3.6. Throughout an entire day, only a handful of buy signals were generated.

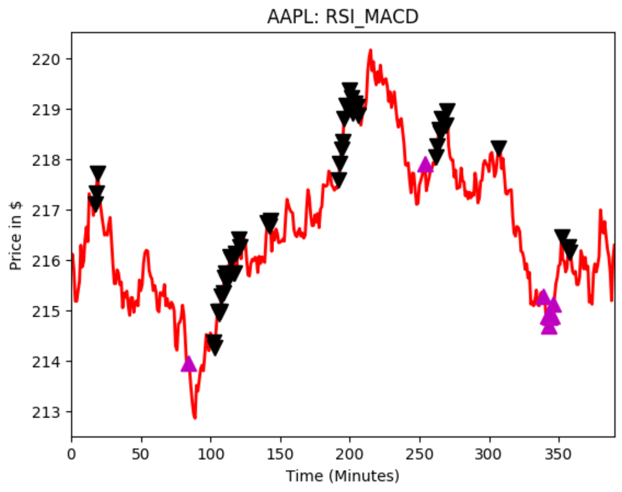


Figure 3.6: RSI and MACD strategy buy and sell signals

3.3 Pairs Trading - Arbitrage and Mean Reversion

Pairs trading is an AT strategy that chooses two economically linked stocks and profits off the divergence in spread of prices. Pairs trading takes advantage of statistical arbitrage, which is attempted profit from pricing inefficiencies identified through mathematical models. The most basic assumption is that prices will move towards their historical average, known as mean reversion. However, unlike other instances of mean-reversion, this strategy has a distinct advantage of always being hedged against market movements [11].

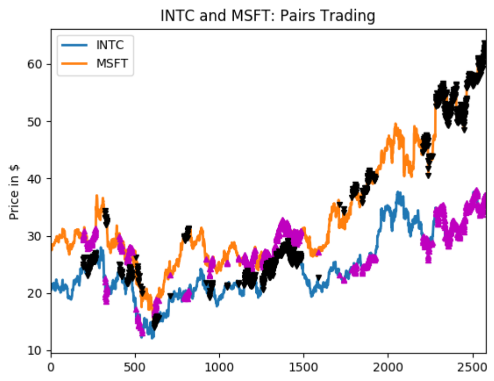
A challenge with pairs trading is choosing pairs of stocks that have related performances. Often when given two stocks that are linked economically (i.e. Pepsi and Coca-Cola), we expect the spread to remain relatively constant over time. However, there might be divergence in the spread between these two pairs cause by factors such as supply/demand changes or changes in volume in a particular stock. Another problem with this is finding stocks that are closely related. Because we need to have stocks that behave similarly, we use *cointegration* to identify pairs of similarly behaving stocks.

Cointegration is a stationary measure that highlights horizontal trends [13]. Unlike correlation, which only tracks similarly moving magnitudes over time, it instead tells how the difference between two regression lines changes over time. Our implementation uses the Python extension `statsmodels` to generate the p-value for cointegration. It is absolutely key to have related stocks, as this strategy takes advantage of mean reversion. In other words, we expect stocks to revert back to their original mean in relation to other similarly behaving ones. Now this is quite difficult, as it is hard to find stocks that behave very similarly. After running comparisons of cointegration, certain tech stocks such as `INTC` and `MFST` were found to behave similarly. We tested cointegration for all combinations of stocks tested and only found five satisfactory combinations: `QCOM/SBUX, INTC/MSFT, AMD/SBUX, AMD/EBAY, AMD/VOD`.

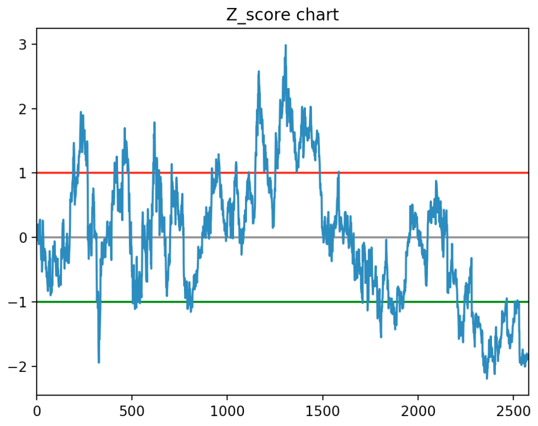
To generate trading signals, a Zscore is generated from the ratio of prices R(i) = ClosingPriceStockX/ClosingPriceStockY on a particular day *i*, the overall mean of ratio of prices M = AVG(R), and the standard deviation of the price ratio STD = STD(R):

$$Z = (R(i) - M)/STD$$

The Zscore is a measure of how far away the current ratio of prices is away from its mean. Figure 3.7b shows the Zscore of both `INTC` and `MSFT`. Because of mean reversion, we can use this as a good indicator for buy and sell signals. A buy signal is generated when the zscore drops below negative one, as we expect the zscore to return to its mean of zero. A sell signal is generated when the zscore goes above one, as the stock is currently overvalued because we expect the stock to return to its mean of zero [11]. These thresholds are shown in Figure 3.7b. In context of the pair of stocks, on buy signals the first stock in the pair is bought while the second is sold or shorted. The reverse is true on sell signals. Figure 3.7a shows this strategy applied to `INTC` and `MSFT`. Each time a buy signal is generated at a particular day the other stock has a matching sell signal and vice versa.



(a) Pairs Trading Strategy



(b) Zscore of INTC and MSFT

Figure 3.7: Pairs Trading strategy applied to INTC and MSFT

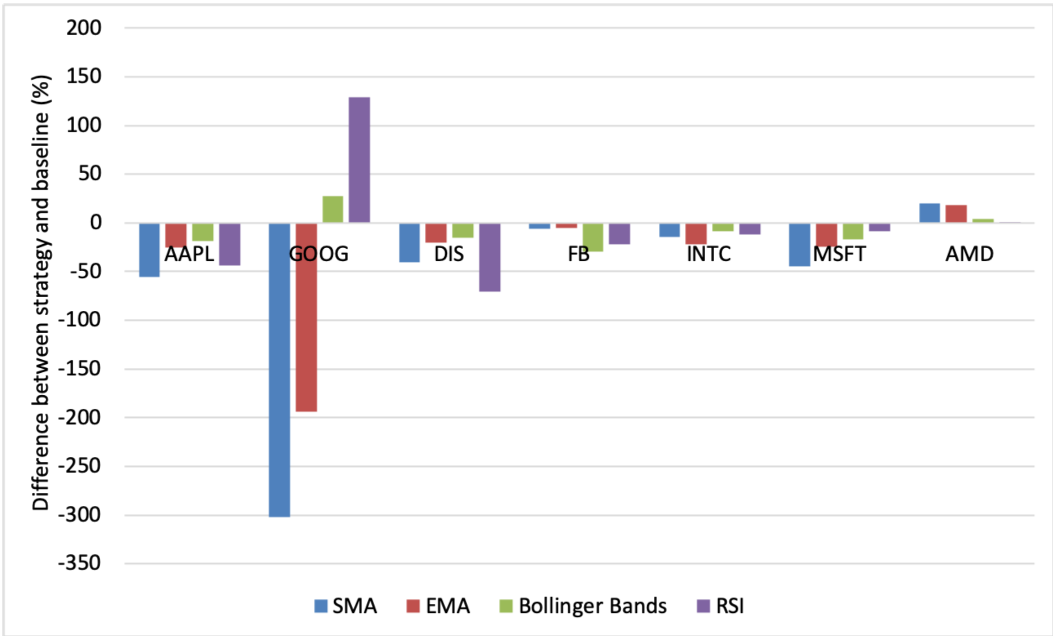
4 Experimental Results of Standard Algorithms

Our Python test suite runs all of the algorithms discussed above on a subset of stocks and generates profit figures. The implementation leverages `Pandas` Dataframes to store and manipulate the stock data. `Pandas` is a powerful open source data analysis library. See Chapter 8 for pseudocode of each strategy's implementation. For every buy or sell signal generated in our implementation, the strategies purchased or sold 100 stocks at a time.

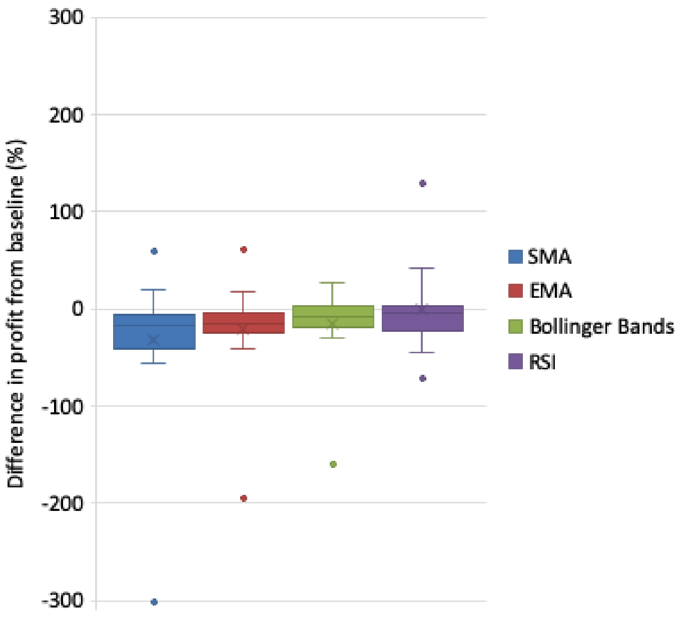
To measure the effectiveness of a particular strategy, we use a quality metric to analyze productivity. We compare against a *buy-and-hold* baseline. The baseline measure is created by holding a long position of a specific denomination throughout the entire period of trading. Simply put, shares are bought at the beginning of the period and sold at the end. The profit level is then compared against the performance of the AT strategy. Much of the literature studies the best performance of these strategies over different periods and uses the same baseline comparison [20] [2] [12]. We therefore include the same baseline in this study. However, there are other baseline measures that are implemented in the industry. Some baseline measures compare against mutual fund performances over the trading period. For this baseline method, strategies are often tested against the performance of `SPY`, a historically high performing fund [2].

4.1 Momentum Strategies

Standalone momentum strategies as a whole very rarely beat out the baseline measure. We tested against a selection of 15 stocks across different industries. Figure 4.1a displays the results of a selection of stocks using SMA, EMA, Bollinger Bands, and RSI: `AAPL, GOOG, HAS, QCOM,` and `AMD`. Only a few stocks had performances that beat the baseline. One was `GOOG`, with both Bollinger Bands and RSI outperforming the baseline profit significantly. `AMD` for both SMA and EMA proved to be effective as well. However, for most it was more profitable to simply just buy and hold over the period. This performance is counterintuitive, as we should be generating profitable buying and selling trading signals with all of the computation, but that is clearly not the case.



(a) Momentum Strategies Individual Results



(b) Momentum Strategies Results Across All Stocks

Figure 4.1: Momentum Strategies Results

Figure 4.1b shows the performance of each strategy across all 15 stocks tested. We observe that SMA and EMA are the least effective of the strategies, with both of their upper quartiles not even beating the baseline. Bollinger Bands and RSI both have upper quartiles above the baseline, but the median score still doesn't beat the baseline. We observe improved performance as algorithms become more involved and incorporate different statistical measures, but with no median scores outperforming the baseline, we don't observe any success with these strategies. Clearly, these strategies aren't viable as standalone trading algorithms. Now we will discuss why we observed overall poor performance for each algorithm.

The "Golden Cross" strategy when applied to both SMA and EMA isn't particularly effective. Due to the nature of SMA, which generally looks at long periods of time to highlight trends, this can induce a lagged effect to the buy and sell orders. A lagged effect in the context of moving averages is that the current moving average doesn't react to the current trend because of this longer observation window, often making this method ineffective. This lagged effect can be fixed by looking at an EMA analysis [10]. However, unlike the literature suggests, in practice this doesn't translate to profit. `AAPL` nearly doubled its profit from using the exponential window that removes the lagged effect yet still had 25.3% less returns than the baseline. While as a whole the strategy was more effective than SMA and did come closer to baseline performance, it rarely ended up beating it. This suggests that these indicators should be looked at in a different, combined context where multiple indicators could provide more robust trading signals.

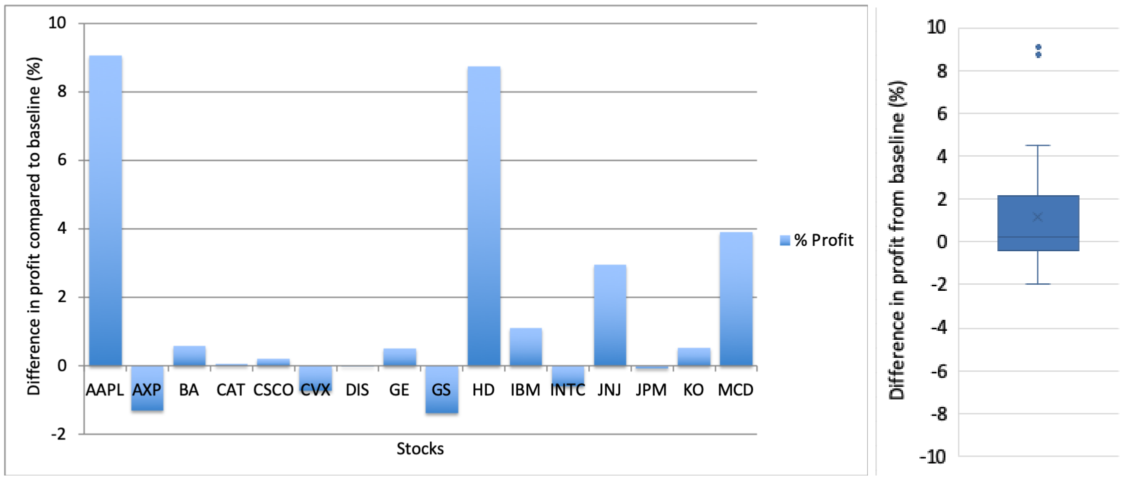
With Bollinger Bands, buy and sell signals are still reliant on changes in rolling average prices. However, because of the incorporation of standard deviation, this strategy uses volatility in the market to further base buy and sell decisions. While a significant portion of the chosen stocks didn't end up actually out performing the baseline, a majority of them were within 1-2 percent performance of the baseline. 33% of stocks tested on the strategy out performed each of its own baseline measure of buying and holding a specific denomination of shares throughout the entire period. Most notably, `GOOG` outperformed the baseline by 27.7% while its SMA and EMA performances were an abysmal -302.5% and -194.1%, respectively. While still not a passable standalone algorithm, the improvement over SMA and EMA is encouraging.

RSI takes advantage of over-purchasing or overselling of securities to predict convergence back to the mean, taking advantage of mean reversion. Unfortunately, most RSI implementations didn't outperform the baseline. The performance was comparable to Bollinger Bands but slightly improved, as our median performance for this strategy is -3.88%. While negative, it beats Bollinger Bands median value by over five percentage points. `GOOG` had by far and away the most impressive performance of any momentum strategy and beat the baseline by 128.8% while on the other hand `AAPL` underperformed and posted a -44%

performance compared to the baseline. Overall, our lack of profitable results motivate the need for looking at a more advanced strategy over more fine-grained price data.

4.2 Combination Momentum Strategies

The RSI and MACD combined implementation performed incredibly well. Unlike the above section, we use intraday stock data. Our results are from one day of trading on October 26, 2018. Our results were corroborated by choosing a random selection of other days of trading and finding similar results. Of all 29 stocks tested, only 4 stocks ended up with negative profit at the end of the day. Figure 4.2 shows this strong performance. The most notable performances were from `AAPL` and `HD`. `AAPL` out-gained the baseline by 9.07% while `HD` had 4.99% of profit with the strategy while the baseline lost 3.75%. We only observed 27% of stocks in total that were out-gained by the baseline. `AXP` is a good example of a poor performing stock - generating a loss of 1.3% more than the baseline. Across all stocks, we observe the mean performance of this strategy to be 1.18% and a median performance of 0.23%. This is a drastic improvement from Section 4.1, where all median performances were negative. With enough capital, this strategy could be used to generate incredible returns when using this strategy over multiple days.



(a) RSI/MACD Individual Results **(b)** Total Results

Figure 4.2: RSI/MACD Strategy Results

This algorithm generates incredibly robust buy signals and frequent sell signals which can nearly guarantee gains after buy signals. Therefore, having millions of dollars of existing capital and allowing us to purchase higher denominations of shares of a given stock can generate even more profits with less risk of massive losses. However, rather than focusing on individual stocks over time, this combination strategy is best implemented over a pool of multiple stocks because it isn't uncommon for days to generate absolutely zero buy signals for a given stock. Unfortunately, the barrier to entry for individuals to generate these massive profits is being able to access up to the second stock data, which requires requires tens of thousands of dollars per year [28]. All data found via the internet or Quandl is delayed by upwards of 20 minutes, which can make the difference between thousands of dollars and profits and thousands of dollars of losses.

4.3 Pairs Trading

Just like our combined momentum strategy, our pairs trading implementation performed incredibly well. Because we are trading two different stocks in this method, we *buy-and-hold* both stocks over the trading period. However, we only found 5 combinations of stocks that were cointegrated enough to conisder trading. When the cointegration p-values of a pair of stocks is less than .05, this signals cointegration at a 95% confidence level [13]. Further study would identify more pairs of cointegrated stocks from a larger pool of stocks. When trading `AMD` with `SBUX`, the strategy profited 205.89% over the baseline. `QCOM` and `SBUX` also found success - culminating in 122.37% returns over the baseline. However, even the lowest performing combination, `AMD` and `VOD`, beat out the baseline by over 32.4%. Figure 4.3 shows these truly astounding results.

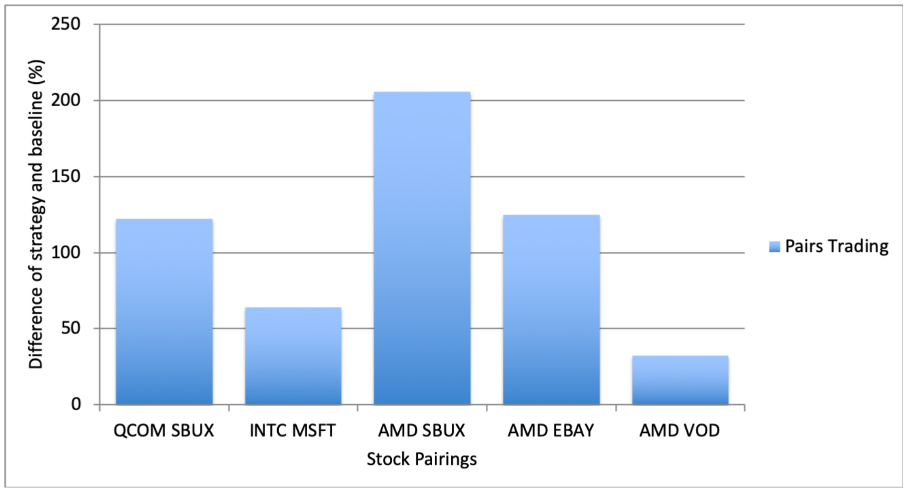


Figure 4.3: Pairs Trading Results

This highlights a common trend. More involved strategies that use multiple measures or inputs always perform better than single measures and nearly always beat the baseline. Both the combined RSI/MACD and pairs trading strategies were very successful. Using these strategies, someone could make incredible profits. However it is important to acknowledge limitations as well. For all of our algorithms studied in this section, we only test a very small subset of the larger DJIA. Future studies would look to include many more stocks to better understand the effectiveness of these strategies.

5 Twitter Sentiment and Stock Trading

We create unique twitter sentiment stock trading algorithms using a variety of different models. As mentioned in Chapter 2, sentiment can be used to effectively trade in the stock market. While a variety of different papers have looked at more general tweet sentiment directed at particular stocks, such as specific stock ticker mentions [22], we choose a different method. Unlike previous literature, we leverage large companies' twitter activity. The motivation is that companies often use social media as a primary medium for announcements, product launches, and generally important events. While mentions of a particular stock can gauge the public's reaction or sentiment to a particular event, we attempt to circumvent this reactionary sentiment measure and look at the sentiment of the content the company is producing on Twitter. By analyzing a company's Twitter output for a day, we expect that we can use that information to more effectively predict the following days stock price movement and thus generate more profit. This method has the added benefit of not only gauging the sentiment of a company's tweets, but also identifying the public's reaction to particular tweets measured by retweets and favorites. We consider five different learning algorithms and four different models that use Twitter sentiment. Below we discuss why we choose such algorithms and how they work. Chapter 6 examines the effectiveness of our Twitter sentiment stock trading strategies.

5.1 Twitter Data and Aggregation

To acquire tweets from the companies, we use a slightly modified version of Github user *bpb27*'s repository *twitter_scraping*[1]. We scrape tweets from within our trading period of 2006 until the start of 2017. We found that most major companies didn't start tweeting until after the start of our initial trading period so our models trade over the time from the companies first tweet until the start of 2017. We use 16 major companies tweets and stock closing price data for our experiments, which cover a variety of different industries as shown in Figure 3.1.

[1]https://github.com/bpb27/twitter_scraping

To acquire Twitter sentiment values, we use Natural Language Processing (NLP). NLP is a branch of artificial intelligence that allows machines to understand and decipher human language. For our implementation, we used the Python library `textblob` to perform NLP on the Twitter data [21]. We perform sentiment analysis on the text from the tweets and use the *polarity* score, which is a float from negative one to one, to understand the sentiment of individual tweets. The `textblob` library generates polarity from a naïve Bayesian classifier that has been pre-trained on a movie review data corpus by identifying positive and negative words from the input text. While it is possible there are differences between the movie review corpus and the language used in tweets, it is impossible to quantify. While building our own NLP Bayesian classifier would have potentially made Twitter sentiment more accurate, future work could look to implement a tweet-based NLP classifier.

We then aggregate the Twitter data over each day of stock trading by merging both the Twitter data stock data. Replies, which are tweets directed in response to particular users, are dropped from the models, as this would skew the intent of the data as we are concerned with tweets directed at the general public. We create inputs for our model over each day of average tweet sentiment, amount of tweets, total favorites, and total retweets. After, we then generate indicators for supervised learning: closing price change and closing price signed change. For the regressors we use the float value of change in closing price while the classifiers only accept integer values for the y-value input. Therefore, a one signals a positive change in price while a negative one signals a negative change in price from day to day.

5.2 Statistical Approach

We initially look to understand if there is any statistical correlation between closing price and average tweet sentiment. By using the Python library `Numpy`'s function `corrcoef()`, we are able to obtain correlation of price and sentiment. This correlation measure ranges from one to negative one. We find very minimal correlation, which is shown in Figure 5.1. We see that `BIG` has the most correlation of any stock tested with a measly 0.30. The vast majority of the stocks are under +/- 0.15 correlation. This lack of correlation could have negative implications for a very basic trading strategy that uses sentiment values statistically.

Because there is little correlation between closing price and average sentiment we should expect a naïve strategy to perform poorly. We create a simple strategy that generates a buy signal when the average tweet sentiment is above 0.25 and generates a sell signal when it is negative. We rarely find companies that generate overtly negative Twitter content. Most tweets from major companies are often positive, which is logical given the way social media is used for promotion. Therefore, we choose a buy and sell signal threshold that reflects the

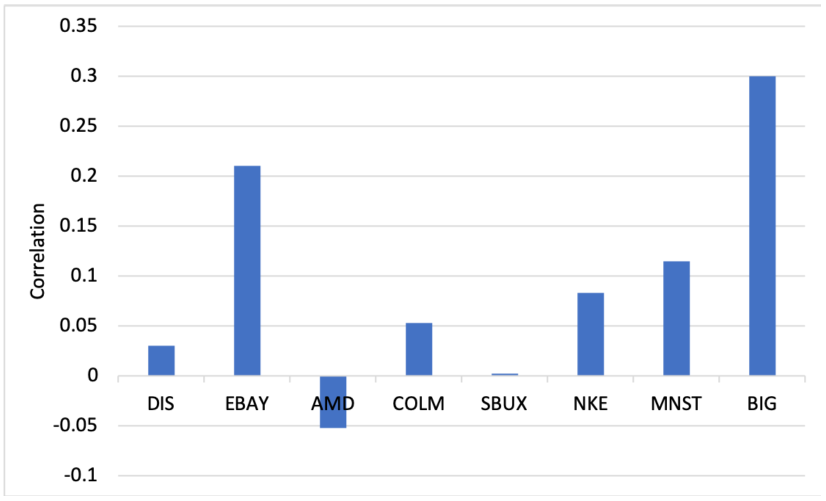


Figure 5.1: Correlation Results

nature of our dataset. We should expect that if the companies tweets for the day were on average negative, the stock could perform poorly the next day.

Figure 5.2 shows the abysmal performance of this naïve strategy. Only 25% of the stocks end up beating the baseline, with the underperforming stocks performing terribly. EBAY characterizes the overall performance of this strategy well – with the strategy returning 47.2% while the baseline nets a 237.6% profit. If sentiment was more correlated with closing price, we would expect this strategy to be more profitable. This lack of correlation motivates the need for a more sophisticated approach, such as using machine learning.

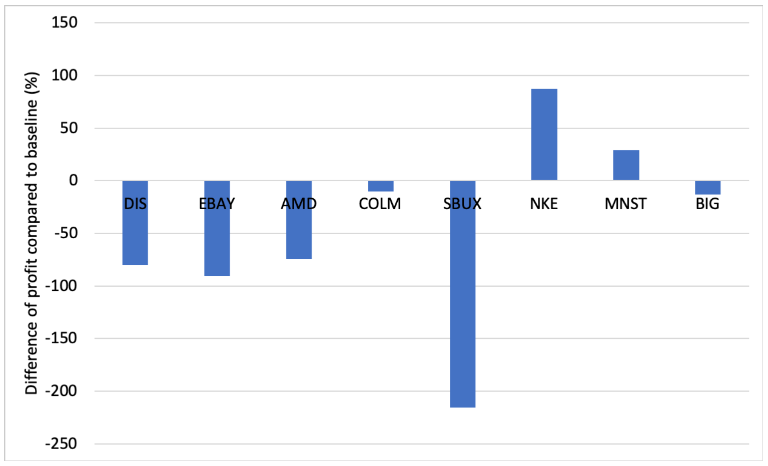


Figure 5.2: Naive Strategy Results

5.3 Stock Trading and Machine Learning Algorithms

Our implementation uses the both the `Scikit-learn` and `TensorFlow` Python libraries [24] [1]. We use NLP to conduct a sentiment analysis of tweets for our models. We then aggregate the twitter data across days of stock trading and then feed the combined stock and twitter data into different types of machine learning algorithms including trees, k-nearest neighbors, and neural networks. We use this variety of machine learning algorithms to attempt to understand if one in particular works well with Twitter sentiment data or if the effectiveness of each algorithm varies from stock to stock. A majority of these models have been used for trading algorithms relying on sentiment in previous literature and have been proven to be effective, further motivating their inclusion in our study [6] [22] [25]. We discuss the different types of features inputted to these algorithms in section 5.4.

5.3.1 Trees

Decision Tree Classifier

Decision trees predict the value of a target variable by learning simple decision rules inferred from the data features [24]. It breaks down a dataset into smaller and smaller subsets, with the leaf nodes representing the classification prediction based on information gain. For features into the classifier, we use the default features specified in the `Scikit-learn` library. Specifically these features contain: a Gini impurity for criterion, utilizing best split, and using two as the minimum number of samples to split a node. Because we are using a classifier, our implementation predicts whether or not the following days closing price will be higher or lower, which is represented by a 1 or -1. Another benefit of decision trees is that they can be visualized easily. Figure 5.3 shows the decision tree fitted to `DIS`. However, due to the complexity of our model it is difficult to interpret the visualization.

RandomForest Regressor

Random forests build off of decision trees. It uses a randomized decision tree making a diverse set of classifiers by introducing randomness in the classifier construction [24]. This prediction of the ensemble is then given as an averaged prediction of the individual classifiers. Each tree in the ensemble when it is constructed is built from a random sample drawn from the training set. More randomness is added during construction by choosing the best split of nodes from a random subset of the features. This generally induces more bias from the randomness, but due to averaging often decreases variance and hence yields a

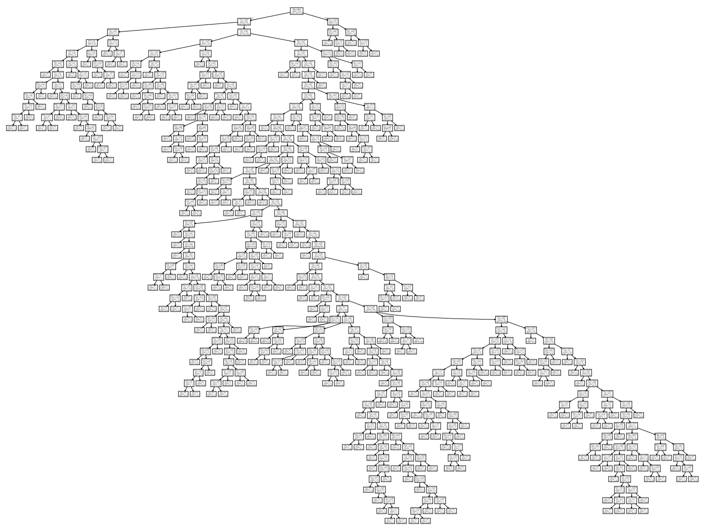


Figure 5.3: Decision Tree Visualization of `DIS`

better overall model [24]. Just like the features used in the decision tree, we use the default values in the `Scikit-learn` library. We use 10 trees in the forest and measure quality of a split using mean squared error. Unlike a classifier, a regressor is able to predict and model float values, giving us a different insight, as our implementation predicts the magnitude of change in closing price rather than just the direction of price movement.

5.3.2 K-nearest Neighbors Classifier

This algorithm is one of the simplest machine learning algorithms that exists. After choosing a k, or amount of nearest neighbors, it calculates the distance between the query sample and all of the training samples via Euclidian distance [24]. The algorithm then sorts the ordered collection of distances and chooses the mode of the k selected labels for classification. In a regression, it would choose the mean of the k selected labels. For our algorithm, we choose the default `Scikit-learn` amount of neighbors: $k = 5$. This is motivated by the variety of different features that are used to test the model, generating different groups of neighbors. Future study could take a closer look at this algorithm and identify the ideal amount of neighbors for each particular stock by comparing trading profit across different k values. Because we are doing classification, our algorithm predicts binary price change, as with the other classification algorithms discussed.

5.3.3 Neural Networks

MLP Classifier

An MLP is a neural network that perform binary classification. It has an input layer, a certain number of hidden layers, and an output layer which makes a decision about the given input and are generally used in a supervised learning context. The network *learns* by using a back-propagation algorithm consisting of two steps [16]. In the forward pass, predicted outputs from a given input are evaluated mathematically. In the backwards pass, partial derivatives of the cost function are propagated back through the network. Figure 5.4 shows how a signal flows through a simple MLP with one hidden layer.

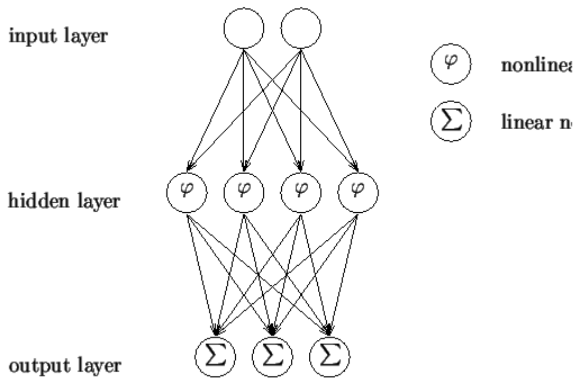


Figure 5.4: Signal-flow graph of an MLP

We choose a classifier with 3 hidden layers, each with 100 neurons, and use a stochastic gradient descent for the solver. While MLP classifiers can handle a variety of different layers and solvers, this combination proved to be most effective during testing of the algorithm for our combined Twitter and stock data. The MLP classifier predicts a one or negative one, signaling a price rise or alternatively a price drop the following day.

Recurrent Neural Networks - Long Short Term Memory Model

We implement a deep learning model as well. We utilize a Long Short Term Memory model (LSTM), which is an example of a recurrent neural network (RNN). Unlike traditional neural networks, RNNs use previous data to help classify or predict the current value [9]. LSTM networks are a special type of RNN that is capable of learning long-term dependencies unliked traditional neural networks which struggle with this problem. They are trained via back-propogation over time and instead of neurons have memory blocks that are connected through layers. Each block contains 3 different types of gates which are triggered by sigmoid activation units: forget, input, and output gates. The first decides wether or not to throw away information from the block. The second decides which values from the input should be allocated to update the memory state. The third decides what to output based on input and memory state. Figure 5.5 demonstrates the architecture of an LSTM model. Because the structure of an RNN allows the model to use past events to aid future states unlike a typical neural network, this has an ideal application for long term time-series based data [9]. This LSTM model is ideal for our use case, as we utilize time-series stock data.

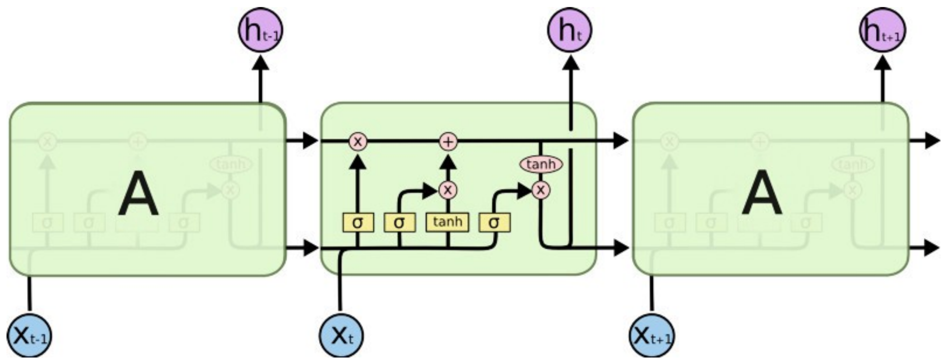


Figure 5.5: Architecture of an LSTM model

The model for our implementation uses a look back of 60 days to give the model enough input to predict the following closing price. At its core, our LSTM model processes 60 days prior of inputs including closing price, tweet sentiment, and a variety of Twitter data, and then predicts the following days closing price. Furthermore, we add a second layer into our model. This stacked LSTM makes the output of the first layer become the input to the second layer, giving our model more depth and accuracy. Unlike the above methods which predict signaled change, this model predicts the stock price.

5.3.4 Stacking

We use stacking to attempt to find more consistent performance with our machine learning algorithms. Stacking works by aggregating the buy and sell signals from each machine learning technique on each day of stock trading. This method is analogous to taking a popular vote, as we trade based on what the majority of the models dictate. We employ two different strategies – one that behaves more conservatively and another that generates buy signals far more leniently. The former sells all positions if any machine learning model generates a sell signal and only buys when all four models generate a buy signal. The latter gives a buy signal if more than half of the models generate a buy signal and sells if under half give buy signals. By employing these two different strategies, we expect to see more regular performance.

5.4 Feature Sets and Trading Strategy

We implement a simple trading strategy used across all machine learning techniques. We want to understand the efficacy of particular machine learning techniques, or if a particular model is more effective than the others. For each model, we scale our Twitter data using `Scikit-learn's` `minmaxscaler()` preprocessing tool to remove skew, as we find that the majority of companies produce tweets that have far more positive than negative sentiment.

5.4.1 Trading Strategy

For all of our different machine learning algorithms and models, we use the same simple trading strategy. We buy and sell based on the predicted change in price. Starting with $1000 of capital, we will buy as many shares as possible if our algorithm predicts that the price will rise the following day. If our algorithm of interest predicts that the price will fall on the following day, we sell all of our positions. We compare against a baseline *buy-and-hold* measure. The baseline is obtained by buying up to $1000 worth of shares at the beginning of the trading period and selling on the last day. We use two different feature sets for our models, simplistic and complex, and test both using this strategy.

5.4.2 Simplistic Features

The most simplistic model employed in our research only uses two inputs: closing price and average tweet sentiment.

5.4.3 Complex Features

Building on the simplistic features, we add other twitter features to the input. We use amount of tweets, favorites, and retweets to provide more detailed input to the model, which should translate to more accurate results. We also considered adding more features such as stock volume or moving averages, but we determine that it had no impact on the model.

6 Twitter Sentiment Results

We find that each of our different Twitter sentiment strategies using extended features have overall strong results. Different models have differing effects on various stocks. To understand the effectiveness of our strategies, we utilize the same baseline *buy-and-hold* measure as described in the opening of Chapter 4. For all of our models, 100% beat the baseline for all stocks with baseline *buy-and-hold* losses. This unanimous success for poor performing stocks gives great promise towards a live-trading implementation during poor overall stock market performances. Each specific model utilizing the extended feature set finds at least two stocks with performances that beat the baseline by over 50%. Stocks that have strong baseline performances are rarely beaten by any of our strategies, but still mostly generate positive profits within roughly 10% of the strong baseline measure. On an individual stock level, our results have somewhat random performances on each model for different stocks. This lack of consistency is due to the difference in data and sentiment for each stock. Because we fit the model for each stock on each machine learning algorithm, it would be highly improbable to see one model give incredibly persistent and consistent results due to the random nature of market movement.

Of the 16 stocks we tested, we examine eight in-depth that have the most remarkable results across different algorithms. However, we do discuss the overall results using all of the stocks that we tested in Section 6.5. We exclude the lowest performing in our in-depth discussion of the models to emphasize the highest performing stocks for our strategies. Common practice in the field is to focus on and examine the highest performing stocks across different strategies [2]. Therefore, we choose `DIS, EBAY, SBUX, BIG, NKE, AMD, COLM, and MNST`. Most of these stocks share a common feature: the baseline strategy posts sizable losses, whereas our trading algorithms make profit when the stock does poorly. Therefore, stocks that had strong baseline performances during the trading period such as `HAS` or `FB`, were excluded from individual analysis of each of the models and are instead examined in Figure 6.5.

As a whole, it would be difficult to trade using this entire group of tested stocks, but our testing reveals some encouraging results. Trading on this Twitter sentiment strategy certainly comes with lots of risk and is at an individuals discretion. However, individual results that beat the baseline by nearly 80 or more percentage points offer some starting points for making a live trading implementation. Specifically, using our deep learning strategy on `SBUX` nets a 54.5% profit over the trading period, while the baseline *buy-and-hold* nets a loss of 23.7%. The stacking lenient strategy for `MNST` nets a 50% profit while the baseline nets a loss of 31.2%. Another encouraging result worth pursuing in live trading is the extended model decision tree algorithm for `AMD`. For this particular stock, our algorithm nets a 529.1% profit, more than doubling the baseline's already impressive 207.3% profit. Our research shows that starting with these three stocks and the associated strategies would be a great start to a sample portfolio.

6.1 Simplistic Model

Our simplistic model has the worst performance of any of the tested strategies. Only a few of the stocks tested had truly notable performances. Figure 6.1 shows the performance of a selection of stocks that were tested. A majority of stocks tested don't end up beating the baseline, as only 29.4% post performances better than the baseline measure. The vast majority of stocks tested performed like `COLM`, with all four of our machine learning algorithms generating profits slightly below or right at the baseline. However, some did actually find success. `MNST`'s MLP beat the baseline by 31.2% whereas `SBUX`'s decision tree classifier beat the baseline by 50.7%. Only a minority of stocks show profitable trading margins. We need to utilize a more advanced feature set to more effectively predict following day stock price movement. We find improvement across the board by adding more inputs into the model as seen in the following section.

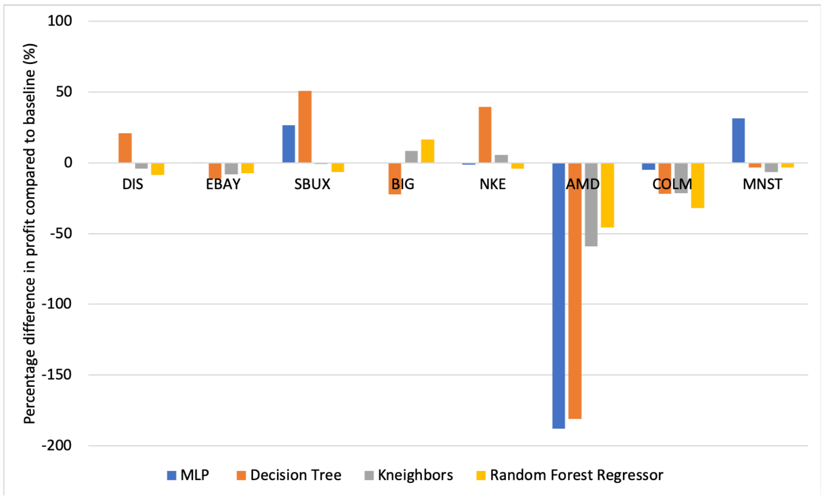


Figure 6.1: Simplistic Model Results

6.2 Extended Model

By adding total tweets, retweets, and favorites as additional features, we see an overall improvement over the simplistic model that only uses closing price and twitter sentiment. Figure 6.2 shows the performance of a selection of stocks that were tested. We find that 87.5% of the stocks tested have a particular algorithm that beats the baseline. This is an improvement of 53% moving to a model with more features. This means that including features such as amount of tweets and retweets are useful for stock trading. Specifically, `AMD` has the most remarkable performance, with the decision tree classifier generating 529.1% profit and more than doubling the baseline profit. We find `MNST` to also have impressive returns with this model. The decision tree classifier generates a 79% profit while the baseline nets a loss of 31.2%. The random forest regressor of `BIG` generates 88.7% profit, beating the baseline by 55.7%. These returns for specific models on specific stocks are significant enough to implement for a realtime for-profit trading algorithm, showing that Twitter sentiment can be used for AT.

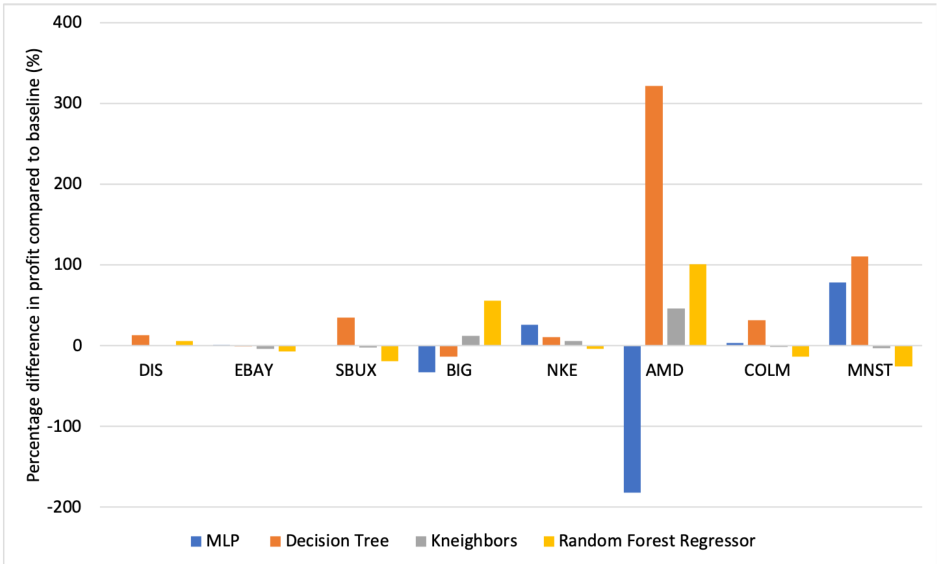


Figure 6.2: Extended Model Results

6.3 Stacking Model

Our stacking model that aggregates all of our various machine learning models into one buy or sell signal doesn't perform as highly as the extended model, but handily beats the simplistic model. Figure 6.2 shows the performance of a selection of stocks that were tested. We find that 58.8% of the stocks using the conservative strategy beat the baseline, while only 29.4% of the lenient strategy beat the baseline. These results show how the conservative strategy performs far more consistently than the lenient strategy. `EBAY's` conservative strategy has a 6.9% return while the baseline has a 39.5% loss. `MNST` nets a 50% profit while the baseline records a loss of 31.2% while using the lenient strategy. This performance sets a high individual performance threshold for this model, as the baseline measure is significantly negative while our strategy has large positive returns. While a few stocks do end up beating a high performing baseline, this strategy, like others using Twitter sentiment, performs much better when the stock performs poorly over the trading period.

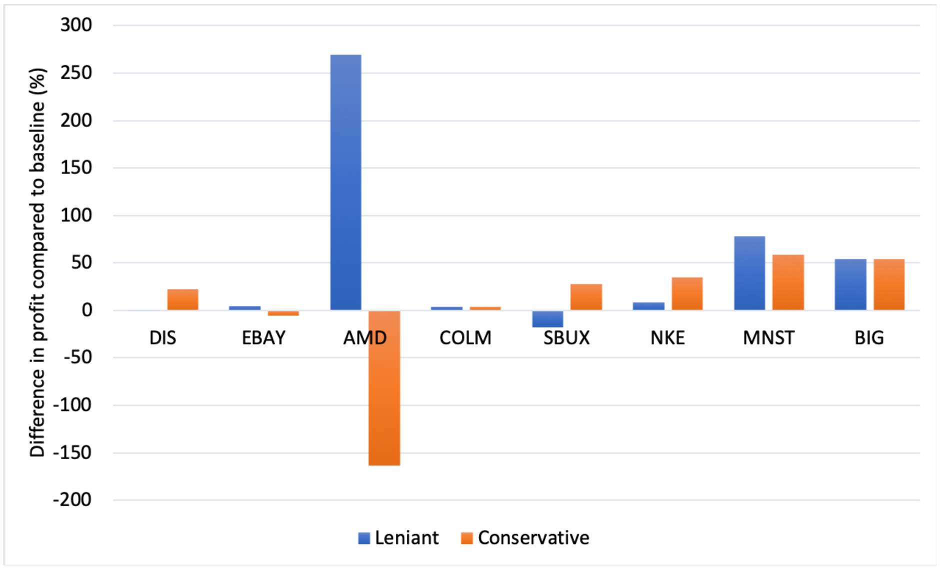


Figure 6.3: Stacking Model Results

6.4 Deep Learning Model

Our deep learning model acts differently than any of the above models, as it predicts the price of the following days close. This model has more average performance, but is very consistent. Rarely does a particular stock trading on the strategy fall significantly below the baseline. Figure 6.4 shows our results for some specific stocks within the data. `DIS, NKE,` and `COLM` categorize the results of this model well – showing how the strategy hovers close to the baseline and gives consistent performance. Interestingly, `EBAY` and `SBUX`, perform far better than the baseline. There is a respective improvement of 55.9% and 78.6% for our strategy over the baseline.

For this model we don't find as many instances of beating the baseline as some of our other models. This is most likely attributed to the accuracy of the predicted stock prices. To verify that our model is accurately predicting the following days stock prices well, we check the efficacy of the predictions by looking at root mean squared error (RMSE). This is done in *Scikit-learn* metrics function call to `mean_squared_error()`. The RMSE of all of our stocks range between 1.5 and 2.5, signifying that our model is able to predict the closing price of stocks extremely well. While the predicted price is close, it isn't accurate enough to be consistently profitable. This performance is most likely due the model predicting stock prices for multiple days in a row either above or below the current price. This prediction differs from the other models that all predict price signal changes. We find that this change in prediction hampers performance for our end-of-day trading strategy that is entirely reliant on price signal changes. Future work could predict multiple days of stock prices in the future and then act on sustained performance rather than specific day-by-day changes to help with this issue in our strategy.

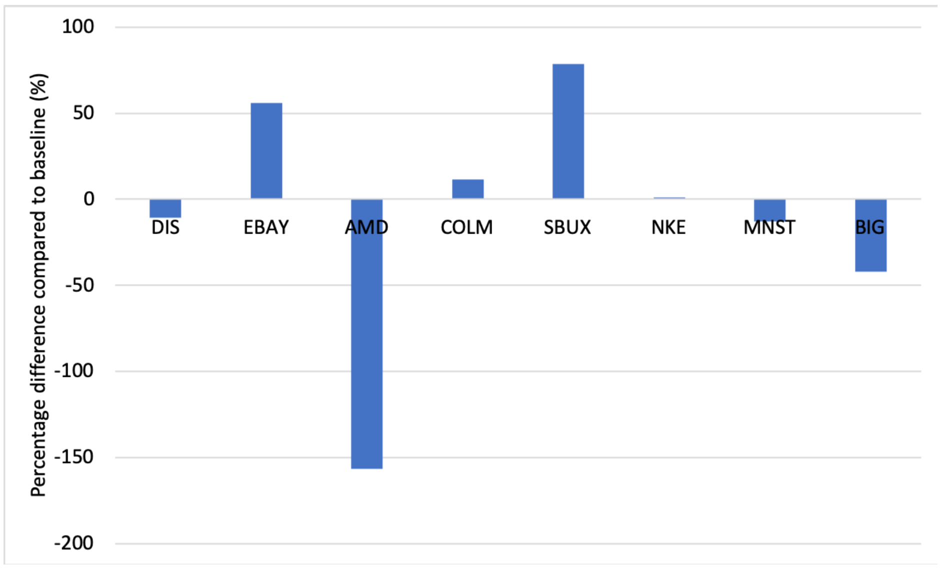


Figure 6.4: Deep Learning Model Results

6.5 Discussion and Comparison

Now that we have examined some particular notable performances of specific models, we must identify trends between all of our models. Figure 6.5 shows our results for all 16 stocks tested. We don't find that one model has particularly astounding performance across all of the stocks tested. However, the decision tree classifier is the most successful of the models examined. With a median performance of 6.4% and an upper quartile performance of 28.7%, this classifier has the most positive results, as it has the largest concentration of the models with performances that beat the baseline. Both of our strategies that use the stacking model have positive medians, signaling its effectiveness. We additionally find that the upper quartile performances for each model are all greater than the baseline. We generally see a consistent spread between the upper and lower quartiles of roughly 25 percentage points. As discussed in the above sections, each of our strategies has a few stocks that generate incredible performance. This performance is reflected in Figure 6.5 as we see a collection of positive outlier datapoints above each different model.

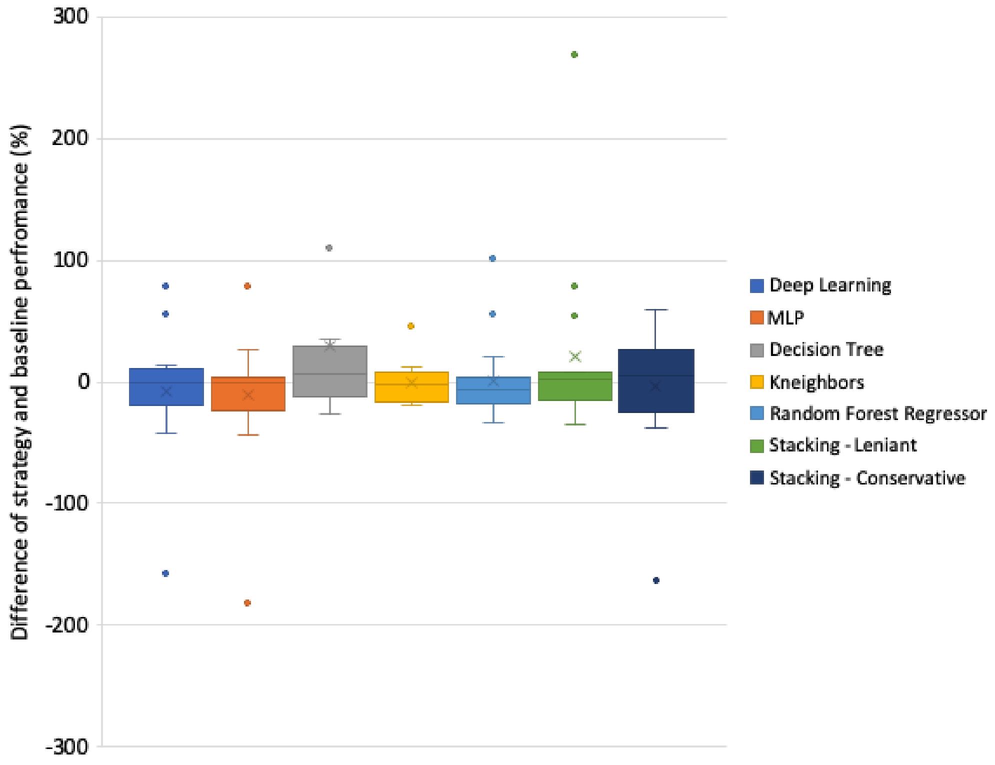


Figure 6.5: Twitter Sentiment Results Compared

However, we also find some limitations with our models and data. Even though the 16 stocks we examined for our sentiment strategies were spread across different industries, it is still possible that skew exists within our data. Incorporating many more stocks to test our models would give more basis to our findings. We are looking at a very small subset of the entire market. But, this small subset is a result of data availability and expense. For acquiring Twitter data, Twitter makes data available at a fee that is prohibitively expensive. Our method for scraping twitter data is incredibly time intensive and restricts our study to 16 stocks. A more thorough study would examine a greater collection of stocks. Additionally, examining more classifiers and regressors could lead to discovering more machine learning algorithms that are highly productive.

7 Conclusion

7.1 Contributions

We make 2 different contributions to AT. Our first contribution examines existing AT strategies to understand how they work mathematically to generate buy and sell signals. We then run these different strategies using our test suite to understand the effectiveness of different strategies. We find that combining momentum indicators, specifically RSI and MACD, and pairs trading are by far and away the most effective existing algorithms that we examined.

Our second contribution looks at using Twitter sentiment as a means of stock trading. We find a lack of statistical correlation between average sentiment and price, motivating our use of different machine learning algorithms. We test two different feature sets for our models. We find that an extended feature set using additional Twitter features, such as total retweets, is far more effective than a simplistic feature set. Using this complex feature set, we test the effectiveness of each machine learning algorithm, combine multiple signals into a stacking model, and additionally use a deep learning model. We find that our strategies are always able to beat a baseline measure that posts losses. Even when the stock performs well and the baseline posts profits, our strategies are generally able to match the performance.

7.2 Future Work

In the future, our study could be extended greatly. First, the NLP component of the research could be tailored exactly to Twitter. Currently we use the *textblob* implementation trained on movie review data. Training on tweets would be more applicable to our research and potentially give more accurate sentiment measures. Some of the existing literature looks at applying sentiment classification to cryptocurrencies. Extending this model to other financial markets, such as cryptocurrency, could give differing results. Also, we could extend our models to predict overall market movement. Rather than our models using individual stocks as input, our models could instead use a collection of stocks as input to try to predict

DJIA price movement. This mixed effects model for market prediction could be used as an overall portfolio management tool.

8 Appendix

8.1 Implementation

Included in this section is the pseudocode for the algorithms described in Chapter 3

8.1.1 SMA

```
def execute(stock, start_date, end_date):
    stock = stockDataRetriever(stock, start_date, end_date).getStock()

    # Initialize the short and long windows and buy sell in df
    short_window = 40
    long_window = 100

    # set Pandas df with stock data
    df = pd.DataFrame(index=stock.index)

    # Create short and long simple moving average
    df['short_moving_average'] = df.movingWindow(short_window)
    df['long_moving_average'] = df.movingWindow(long_window)

    # mark signal based on comparison of the two averages
    df['signal'] = df.compare(short_moving_average, long_moving_average,1,0)

    # when signal changes from 1 to 0 or 0 to 1 - is a buy or sell
    df['positions'] = df['signal'].diff()
```

8.1.2 EMA

```
def execute(stock, start_date, end_date):
    stock = stockDataRetriever(stock, start_date, end_date).getStock()

    # Initialize the short and long windows and buy sell in df
    short_window = 12
    long_window = 26

    # set Pandas df with stock data
    df = pd.DataFrame(index=stock.index)

    # Create short and long simple moving average
    df[short_ema] = df.ema(short_window)
    df[long_ema] = df.ema(long_window)

    # mark signal based on comparison of the two averages
    df[signal] = df.compare(short_ema, long_ema,1,0)

    # when signal changes from 1 to 0 or 0 to 1 - is a buy or sell
    df[positions] = df[signal].diff()
```

8.1.3 Bollinger Bands

```
def execute(stock, start_date, end_date):
    stock = stockDataRetriever(stock, start_date, end_date).getStock()

    # Initialize the window
    window = 12

    # set Pandas df with stock data
    df = pd.DataFrame(index=stock.index)

    # Create the bollinger bands
    df['middle_band] = df.sma(window)
    df['moving_deviation'] = df.std(window)
    df['upper_band'] = df['middle_band'] + df['moving_deviation'] * 2
    df['lower_band'] = df['middle_band'] - df['moving_deviation'] * 2

    # mark signal based on upper or lower prices
    df[signal] = df.compare((df_closing_price, df_lower band),
          (df_closing_price, df_upper_band),1,0)

    # when signal changes from 1 to 0 or 0 to 1 - is a buy or sell
    df[positions] = df[signal].diff()
```

8.1.4 RSI

```
def execute(stock1, stock2, start_date, end_date):
    stock = stockDataRetriever(stock1, start_date, end_date).getStock()

    # set Pandas df with both stocks
    df = pd.DataFrame(index=stock.index)

    # create measures of up and down performance
    delta = stock['Close'].diff()
    roll_up = delta[delta > 0].rolling.mean()
    roll_down = delta[delta > 0].rolling.mean()

    # create RSI
    RS = roll_up/roll_down
    RSI = 100.0 - (100.0 / (1.0 + RS))

    # create the buy and sell commands
    df['buy'] = np.where(RSI < 30, 1, 0)
    df['sell'] = np.where(RSI > 70, -1, 0)
    df['signal'] = df['buy'] + df['sell']

    # when signal changes from 1 to 0 or 0 to 1 - is a buy or sell
    df[positions] = df[signal].diff()
```

8.1.5 Combination Momentum Strategies

```
def execute(stock, start_date, end_date):
    stock = stockDataRetrieverIntraday(stock, start_date, end_date).getStock()
    # create df for both RSI and MACD and merge
    rsi_df = RSI(name, stock)
    macd_df = MACD(name, stock)
    df = pd.merge(rsi_df, macd_df)

    # buy signals when both rsi and macd indicate buy
    # sell signals when either one of rsi or macd indicate sell
    df['buy'] = (df['signal_rsi'] == 1) & (df['signal_macd'] == 1)
    df['sell'] = (df['signal_rsi'] == -1) | (df['signal_macd'] == -1)
    df['signal] = df['buy'] + df['sell']
```

8.1.6 Pairs Trading

```
def execute(stock1, stock2, start_date, end_date):
    stock_1 = stockDataRetriever(stock1, start_date, end_date).getStock()
    stock_2 = stockDataRetriever(stock2, start_date, end_date).getStock()

    # set Pandas df with both stocks
    df = pd.DataFrame(index=stock_1.index, stock_2.index)

    # create zscore from price ratios
    ratios = df['Close_stock1']/df['Close_stock2']
    zscore = (ratios - ratios.mean())/ np.std(ratios)

    # Create the buy and sell commands
    df['buy'] = np.where(zscore < -1, 1, 0)
    df['sell'] = np.where(zscore > 1, -1, 0)
    df['signal'] = df['buy'] + df['sell']

    # when signal changes from 1 to 0 or 0 to 1 - is a buy or sell
    df[positions] = df[signal].diff()
```

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, and G. Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Technical report.

[2] I. Aldridge. *High-Frequency Trading*. John Wiley and Sons, Hoboken, New Jersey, 1st edition, 2010.

[3] L. Almeida Teixeira, A. Lorena, and I. De Oliveira. A method for automatic stock trading combining technical analysis and nearest neighbor classification.

[4] Alpaca. Who is Alpaca? - Documentation | Alpaca.

[5] M. Bendtsen, J. M. Peña, and G. Bayesian. Gated Bayesian Networks for Algorithmic Trading. *International Journal of Approximate Reasoning*, (69):58–80, 2016.

[6] J. Bollen, H. Mao, and X.-J. Zeng. Twitter mood predicts the stock market. Technical report.

[7] A. Chaboud, B. Chiquoine, E. Hjalmarsson, C. Vega, G. Bodnar, C. Jones, L. Marques, D. Rime, A. Schmidt, J. Schoen, and N. Sto¤man. Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market. Technical report, 2009.

[8] T. Chong, W.-K. Ng, V. Liew, T. T.-L. Chong, W.-K. Ng, and V. K.-S. Liew. Revisiting the Performance of MACD and RSI Oscillators. *Journal of Risk and Financial Management*, 7(1):1–12, feb 2014.

[9] Colah. Understanding LSTM Networks – colah's blog, 2015.

[10] J. Ehlers. SIGNAL ANALYSIS CONCEPTS. Technical report.

[11] X. Fu and A. Patra. Machine Learning in Statistical Arbitrage. Technical report, 2009.

[12] D. Garcia and F. Schweitzer. Social signals and algorithmic trading of Bitcoin. 2015.

[13] E. Gatev, W. Goetzmann, and G. Rouwenhorst. Pairs Trading: Performance of a Relative-Value Arbitrage Rule. *The Review of Financial Studies*, 19(3):797–827, 2006.

[14] P. Gomber, B. Arndt, M. Lutat, and T. Uhle. POLICY PLATFORM White Paper High-Frequency Trading. Technical report, 2011.

[15] K.-y. Ho and W. W. Wang. Artificial Neural Network Modelling. 628(July), 2016.

[16] A. Honkela. Multilayer perceptrons, 2001.

[17] F. E. James. Monthly Moving Averages–An Effective Investment Tool? *The Journal of Financial and Quantitative Analysis*, 3(3):315–326, 1968.

[18] A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun. The Flash Crash: High-Frequency Trading in an Electronic Market. *Journal of Finance*, 2017.

[19] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14–23, 2014.

[20] W. Liu, X. Huang, and W. Zheng. Black–Scholes' model and Bollinger bands. *ScienceDirect*, 2006.

[21] S. Loria. Textblob Documentation. Technical report, 2018.

[22] Y. Mao, W. Wei, and B. Wang. Twitter volume spikes. (August 2013):1–9, 2013.

[23] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. Technical report.

[24] F. Pedregosa FABIANPEDREGOSA, V. Michel, O. Grisel OLIVIERGRISEL, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot andÉdouardand, andÉdouard Duchesnay, and F. Duchesnay EDOUARDDUCHESNAY. Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. Technical report, 2011.

[25] D. Shah and Z. Kang. Bayesian Regression and Bitcoin. *Institute of Electrical and Electronics Engineers*, 2014.

[26] B. Sozzi. GE stock may be poised to crash more than 50%, 2019.

[27] T. Stobierski. 401(k) Basics: When It Was Invented and How It Works | LearnVest, 2018.

[28] P. Treleaven, M. Galas, and V. Lalchand. Algorithmic Trading Review. 56(11), 2013.

[29] X. Zhang, H. Fuehres, and P. A. Gloor. Predicting Stock Market Indicators Through Twitter "I hope it is not as bad as I fear". *Procedia - Social and Behavioral Sciences*, 26(2007):55–62, 2011.