Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Technical and Machine Learning Trading Strategies for Nasdaq Futures

Author:
Ao Shen
as5017@ic.ac.uk

Supervisors:
Prof. William J Knottenbelt
Alexei Zamyatin

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 10, 2018

Abstract

In finance, futures are standardized forward contracts that obligate parties to buy and sell at a predetermined price and date. Because of this, futures, promptly named, are often considered a leading price indicator to the underlying asset. The increased popularity of ”index investing” whereby investors buy a basket of stocks to diversify risks has driven greater interest in index-tracking securities. This thesis focuses on improving the return profile of trading Nasdaq 100 index futures using an ensemble of machine learning and technical analysis. The Nasdaq 100 Futures or NQ belongs to CME’s E-mini futures series which enjoy immense volume, longer trading hours and cheap margin requirements. This translates into less noise, more data and more flexible strategies, making it the ideal subject for machine learning and technical analysis.

This dissertation examines the entire process of enhancing the Sharpe Ratio or risk-adjusted return of trading Nasdaq Futures. That includes the development of a market simulator which takes in price and volume data, transaction costs, individual or ensemble strategies, processes their signals, and backtests them to produce executable trading instructions in a CSV file. The strategy component seeks to outperform the Sharpe Ratio of simply buying and holding the future over time. Principal Component Analysis was implemented to reduce the number of technical indicators we used from 29 to 7, 4 of which were ultimately chosen to optimize the ensemble strategy. Result validation were performed using Welch’s t-test and confusion matrix statistics analysis.

The study concluded that an ensemble of 4 technical indicators and LSTM produced the most consistent results in out-of-sample testing, achieving a prediction accuracy of 64% as measured by MCC. A Sharpe Ratio improvement of 21% over simply holding the future was also observed. This implies that learnable features exist in the Nasdaq Futures. The ensemble outperformed their counterparts individually, indicating synergies exist. A random forest ensemble strategy also statistically outperformed the benchmark, achieving 61% MCC, and a 16% increase in Sharpe Ratio. Q-learning failed to produce statistically significant results. RF and LSTM Ensemble were applied to Bitcoin and Ethereum, of which only LSTM outperformed the Ethereum benchmark. A meta analysis of the results show that lower frequency strategies perform better than their high frequency counterparts. By acting on rare and and seldom occurring events, signals became more robust to noise, thus increasing the probability of profitable outcomes.

Acknowledgements

I would like to thank the following people, without whom this dissertation would not have been possible:

Prof. William J Knottenbelt for sanctioning and kick-starting this project. His suggestions were also pivotal in improving the quality of this report.

Alexei Zamyatin for offering his time to review my report and getting me in touch with his brilliant colleague.

Family for making this journey through Imperial possible and friends for inspiring ideas and assistance with troubleshooting.

Contents

Chapter 1

Introduction

1.1 Motivation

The introduction of cash-settled e-mini futures by CME[12] in the late 90s brought significantly expanded trading volume to the asset class as smaller investors sought lower transaction costs, flexible trading hours, and leverage. This study focuses on one particular index future – the Nasdaq 100, which tracks the value of the top 100 non-financial companies on the Nasdaq Exchange. The composition of the index is heavily weighted towards large capitalization information technology stocks like Facebook, Apple, Amazon, Netflix and Google, hence the acronym ”FAANG”. This has allowed the index to outperform any other major stock index for the past 20 years.[12]

Indexes basket many securities together to reduce idiosyncratic risk. This results in lower volatility and noise. This is especially beneficial to our machine learning strategies since individual stocks tend to be very noisy, especially during their quarterly reporting period. Indexes allow our models to learn consistent patterns instead of noise, making index futures the ideal instrument for this dissertation.

At the same time, breakthroughs in Deep Learning[2], has significantly expanded the use of Artificial Neural Networks. Long Short Term Memory in particular has produced great results for the world of financial series data analysis.[2] Neural Network's ability to detect non-linear relationships in the underlying data is additionally a great compliment to technical analysis, which is mostly linear.

Being able to use a combination of linear (technical analysis) and non-linear machine learning to study a highly profitable financial instrument is what makes this an exciting proposition to tackle.

1.2 Objectives

In order to make profitable trading decisions, one does not necessarily need to know the exact price of the asset tomorrow, merely whether or not it will increase in price. In this vein, we train our model based on historical price and volume data, to classify the subsequent period's closing price. This classification is sufficient to inform trading decisions, as detailed price forecasts may be unrealistic given market chaos. To this end, we will set up to accomplish the following:

1.2.1 Model Construction

1. Design and implement a model which outperforms a buy-and-hold strategy for Nasdaq Futures using technical analysis, machine learning or both.

2. Verify the assumption that learnable patterns exist in stock indexes

1.2.2 Simulator Construction

1. Build a modular market simulator whose API can be easily reused by others to experiment with their own strategies.

2. Test this simulator with strategies on other instruments to confirm its reusability.

1.2.3 Technical Analysis

1. Design and implement a novel technical indicator that can outperform or assist in the outperformance of the overall model.

2. Investigate which of the 29 most popular indicators perform the best.

1.2.4 Further Investigations

1. Investigate which technical analysis or machine learning algorithms do best in achieving outperformance as measured by Sharpe Ratio.

2. Any other insights that may enhance strategy performance.

1.3 Contributions

1.3.1 Model Construction

1. An ensemble of 4 technical indicators and LSTM was constructed in Jupyter Notebooks using TensorFlow, and trained on the Google Cloud. Hyperparameters were optimized using Random Search. The model achieved 64% MCC and a 21% increase in Sharpe. The ensemble outperformed the indicators and LSTM individually, implying synergies exist when combing technical analysis and machine learning.

2. A Random Forest ensemble model was implemented from raw Python and NumPy to test if greedy algorithms can exploit close-interval data dependencies. A statistically significant 61% MCC and 16% increase in Sharpe Ratio, implies a relationship exists between recent historical and future prices.

3. A Q-learning ensemble model was implemented. Many challenges were overcome due to non-convergence during training. Discretization was implemented using K-means and Gaussian Mixture Model (GMM). Clustering was performed using Silhouette Coefficient to reduce problem space. Dyna-Q was implemented to speed up convergence. However, the model failed to achieve statistically significant results. This disappointment may stem from the fact that there may be longer term dependencies in the market data that Markov chains could not pick up due to lack of memory. Discretization alleviates this problem but may still have reduced the precision of our model due to losses after re-clustering.

1.3.2 Simulator Construction

1. Designed and built a market simulator from scratch using NumPy and Pandas. The API takes in financial data, transaction costs, technical indicator and machine learning model signals. It then sorts them according to a proprietary algorithm. Subsequently, it backtests the signals, and computes metrics such as MCC, t-tests, and confusion matrix statistics. Finally, it produces an executable trading instructions in a Pandas Dataframe or CSV file.

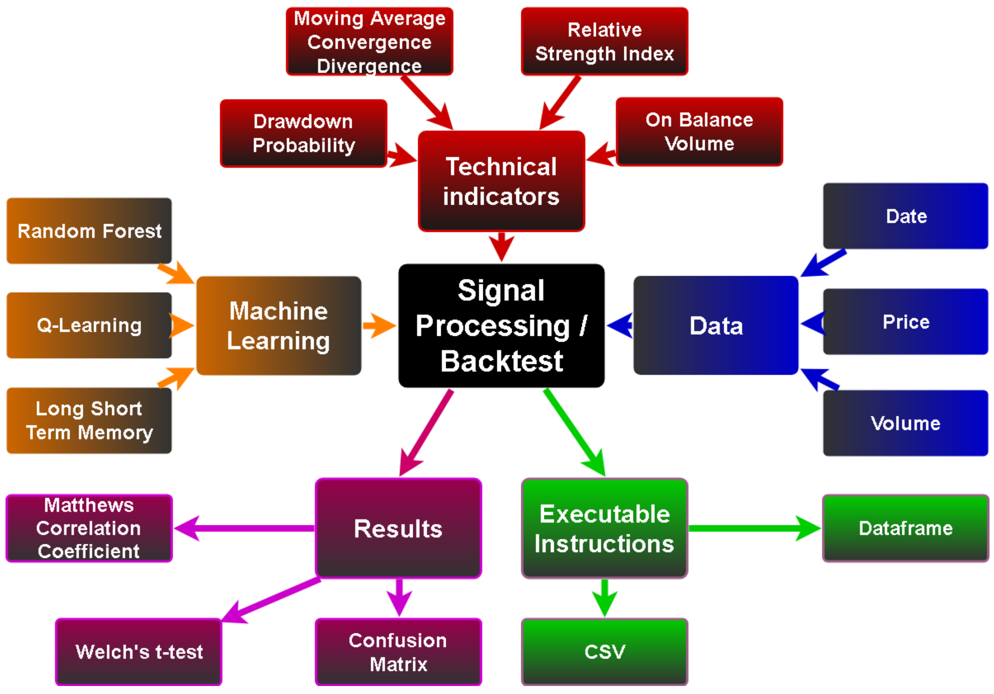


Figure 1.1: The market simulator takes in data and signals from technical indicators, and machine learning algorithms as inputs. It then sorts these signals, and produces performance analytics and trading instructions.

2. Extensive data augmentation and preprocessing, such as Exponential Smoothing, Min-Max-Scaling, splice-adjustments were performed to reduce data noise.

3. The simulator and models were tested with Bitcoin and Ethereum. LSTM outperformed the benchmark on Ethereum, achieving 56% MCC and 13% higher Sharpe Ratio. However, it performed poorly on Bitcoin, which may be due to the high number of unoptimized hyperparameters. Random Forest did not outperform significantly. Note that since the models are optimized for Nasdaq futures, these results cannot definitively conclude if there exists learnable features in cryptocurrencies.

1.3.3 Technical Analysis

1. Concluded that Moving Average Divergence Convergence, Relative Strength Index, and On Balance Volume, in descending order are the strongest individual predictors for technical analysis.

2. Principal Component Analysis and Correlation Matrix were produced using SciPy and Seaborne to find the most important factors among the 29 technical indicators. The indicators could be narrowed down to 6 without compromising the Sharpe Ratio, of which the above 3 were the strongest performers.

3. Designed and implemented a novel technical indicator – Max Drawdown Probability, which allows a strategy to further enhance its performance by not trading on days with potentially high losses.

1.3.4 Further Insights

1. On Balance Volume, and Maximum Drawdown Probability enhances signal accuracy at the expense of signal reduction. In other words, fewer but higher probability success trades are made.
2. Conducted meta analysis by aggregating all 3 ensemble results. The analysis reinforces the above point that signal strength (the extent to which all 4 signals agree with each other) is crucial to profitable strategies.
3. Gained insight into why high classification rates do not translate perfectly into higher Sharpe Ratio through confusion matrix analysis. The biggest factors were the algorithms' blindness to the magnitude of price swings, and its low prediction accuracy during drawdowns, which tend to move faster and more unpredictably. This was the motivation behind our drawdown indicator.

1.4 Thesis Roadmap

- Chapter 2 – Background and Related work, explores previous research in the fields of finance, and machine learning, and result validation methods pertinent to futures, technical, and time series analysis.
- Chapter 3 – Data Preprocessing, explains our data source, collection and processing methods, to ensure our analysis is based on clean data.
- Chapter 4 – Methodology, Performance, and Validation, explains the methods used to process the signals from our various strategies, compute performance metrics, and validate results
- Chapter 5 – Market Simulator Architecture, illustrates the architecture of our core implementation and how it is connected to other aspects of our project.
- Chapter 6 Technical Analysis and Principal Component Analysis, explores 29 different popular technical indicators, and narrows them down through PCA. These are then further narrowed down to 3 by their performance. A fourth proprietary indicator was added to enhance overall performance.
- Chapter 7 – Random Forest, outlines a simple greedy strategy. The if-else localized strategy can help us understand if immediate historical data has any predictive power.

- Chapter 8 – Q-learning outlines a more complex, Markov Decision Process, where we use dynamic programming and a non-label-based reward/penalty system to make non-greedy decisions.
- Chapter 9 – Long Short-term Memory, is the premier method to perform time series analysis. Our results validate its reputation.
- Chapter 10 – Other Applications, explores the use of our models on Bitcoin and Ethereum.
- Chapter 11 – Meta Analysis, aggregates our experiments together to conclude that some of the outperformance from our strategy derives from reducing noisy signals.
- Chapter 12 – Conclusion, summarizes the entire dissertation and outlines the roadmap for future extensions.

Chapter 2

Background and Related Work

2.1 Nasdaq Futures

While US stocks only trade for 6.5 hours a day, most futures exchanges like the CME, are online 22.5 hours a day.[12] This allows for a continuous stream of data where information can be priced immediately, rather than when the US stock exchanges start trading. This also allows for a more global view where information are priced dynamically from other time zones. This ultimately means more data, although at the expense of more noise due to the thinner volumes during non-US hours. All of these factors make futures efficient and indicative of asset prices, thus the reason why every investor should pay attention to the futures market.

Unlike physical commodities, where there is a limit, financially settled futures have none, allowing them to become highly leveraged. To put this into perspective, the US Department of Energy estimates that the dollar value of oil futures traded annually, is on average 20 to 50 times the value of the actual oil where the futures are derived from.[1] Incidentally, this excessive level of ethically questionable leveraged derivatives are what many consider to be the cause of the real estate collapse leading to the 2008 financial crisis.[7]

Yet, the liquidity, and implied leverage are what makes futures the driver of prices in nearly all asset classes, including stocks, bonds, commodities and of course currencies. Since implied leverage means that no interest is actually transacted, it is very cost efficient.

2.2 Efficient Market Hypothesis

The Efficient Market Hypothesis states that the market prices in any new public information instantaneously, thereby eliminating profit opportunities. However, this is hardly the case in real life. Event studies have shown[11] that stock prices tend to increase ahead of acquisition announcements as news leak. Speed of information dissemination is exploited by high frequency traders. Arbitrage opportunities can arise across different trading platforms and instruments when investors over or underreact to news. What this means, is that the market becomes efficient because of human exploitation of its inefficiencies not despite it.

2.3 Random Walk

Both technical and machine learning strategies, seek to find a pattern in our underlying dataset that holds into a future period. However, what if stock movements are random? The application of Random Walk to the financial markets assume that price changes are independent of each other. Cuthbertson defines random walk as an individual stochastic series with a drift.[3]

$$X_t = \alpha + X_{x-1} + \epsilon_{t+1}$$

$$\epsilon_{t+1} \approx iid(0, \sigma_\epsilon^2)$$

where X is a stochastic variable. α is the noise. and ϵ is the drift.

The drift is simply a weighted average of probabilities of each price the stock could possibly move in the next period. Note that this definition assumes independence between successive price movements. In fact, successive conditional variances have been proven to be auto-correlated in semi-efficient and efficient markets given information Ω and martingale X.[3] Bidirectional momentum can build in the price chart of a stock as psychology and big investors try to buy or sell slowly to achieve better average prices. Seasonal trends also occurs as commodity-weather cycles, political and holiday seasons boosts or dampens sentiment. The liquidity and dividend yield of companies were also somewhat correlated to their long-term returns. Only in inefficient markets where past prices do not reflect historical information, can we have a random walk.

2.4 A Bitcoin and Ethereum Example

The distributions of both cryptos are quite ”normal”, although Ethereum has a fatter right tail.

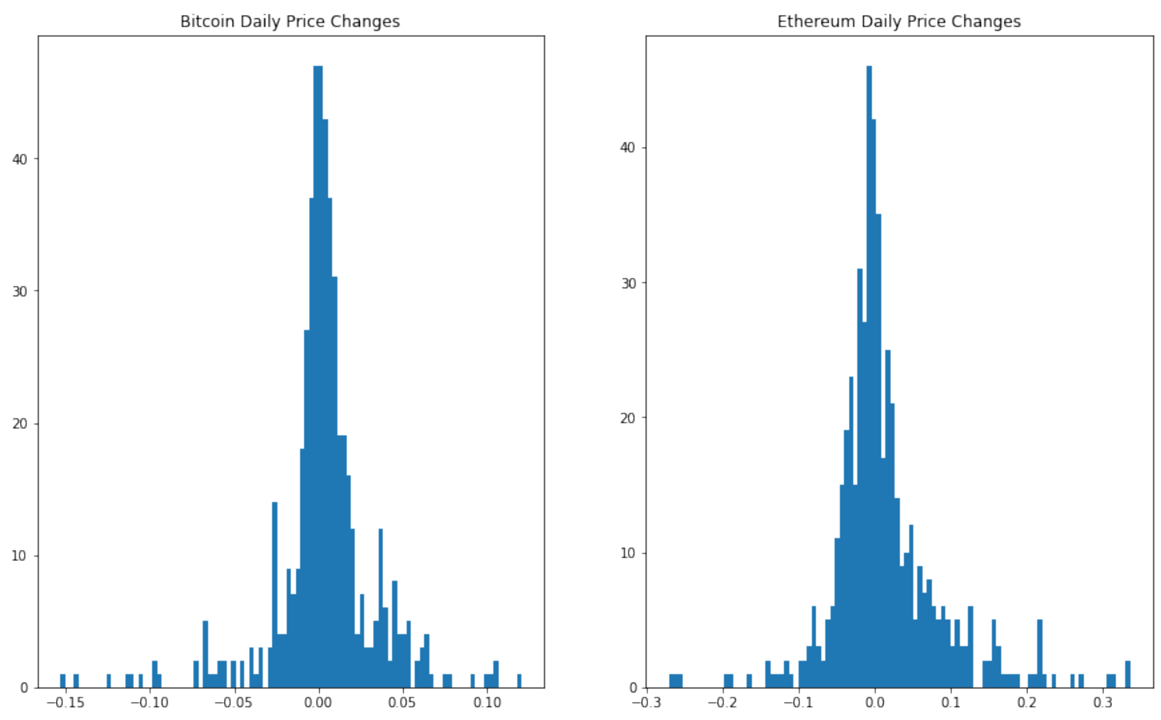


Figure 2.1: Bitcoin and Ethereum daily percentage price change (% change vs frequency), both of which appear normal. Normal distributions are important because they allows us to perform many statistical tests.

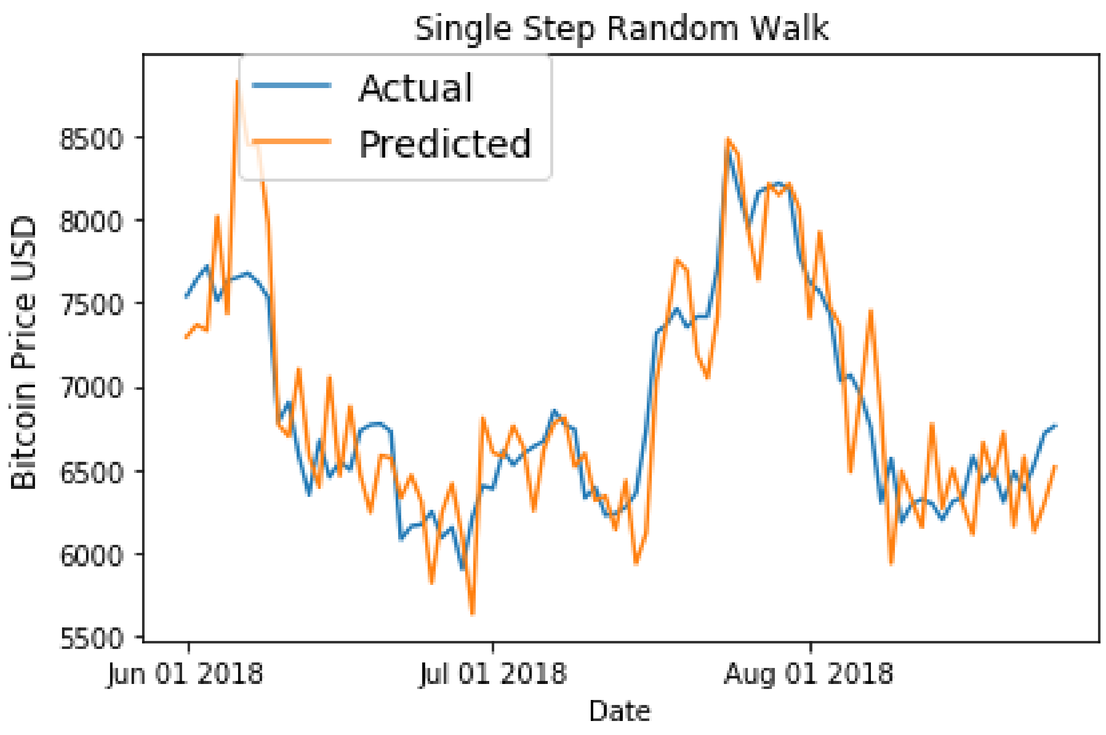


Figure 2.2: Single future period prediction of Bitcoin price. Accurate, but not very useful.

Using Single Step predictions seem to predict future prices pretty well. However, this is very deceiving as the model always seems to lag the actual price by 1 period, and often over/undershoots in terms of price. Further, adjusting every single period, is a simple way of overfitting the dataset.

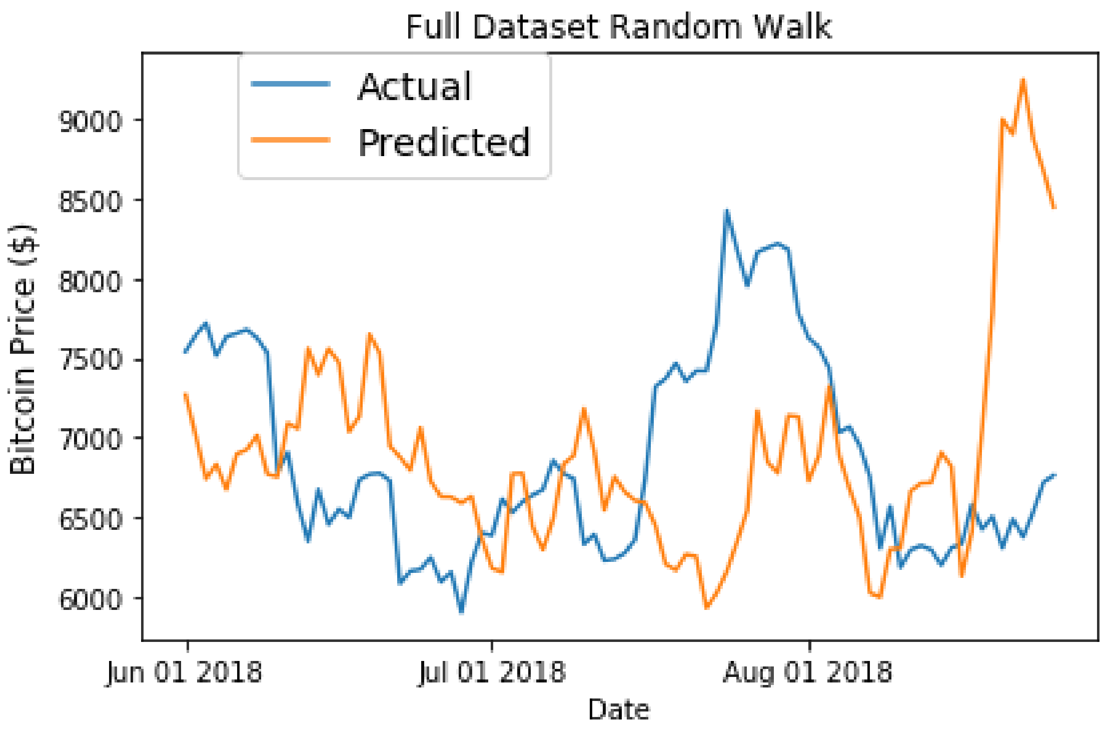


Figure 2.3: Full Dataset Random Walk. Inaccurate, and not useful. A random seed was picked to make the lines fit better.

Our Random Walk trained and tested on the entire dataset (instead of just one period) is much less predictive. While it forecasts the trend of Bitcoin fairly well, there are long periods (in August and September) where our model undershoots and overshoots significantly. The results are also affected by the random seed generator. The model is hardly predictive, but the normal distribution of the data does give some credence to the fact that Bitcoin pricing may contain random factors.

2.5 Sortino ratio

One criticism of the Sharpe ratio is that it punishes volatility in both directions, even though volatility that results in positive returns is welcomed by investors. In order to account for this, the Sortino ratio was developed. The Sortino ratio is similar to the Sharpe ratio, but uses only downside deviation in the denominator.[21]

2.6 Technical Analysis

Technical analysis seeks to predict asset prices using purely trading data. This is in contrast to fundamental analysis which seeks to predict prices based on economic fundamentals. The efficacy of technical indicator is heavily reliant on the assumption that market prices reflect the underlying fundamentals of financial assets. Additionally, it assumes that historical data (even short term ones) can predict future trends. In reality, technical analysis is often used to follow money-movement of big money managers. This has significantly contributed to the decline of actively managed funds in recent years compared to passive investments.[30]

Linear rule-based technical indicators like moving averages and oscillators seek to learn simple statistical patterns from historical price and volume data. These have been used to predict prices of financial assets, including stocks, derivatives and cryptocurrencies since their inception. Most of the technical indicators except Simple Moving Average, Moving Average Convergence-Divergence, Relative Strength Index, Bollinger Bands, On Balance Volume, Pivot Points, Maximum Drawdown Probability is sourced from TA-Lib.[16]

It is also argued that technical analysis becomes a self-fulfilling prophecy, where stocks begin following major technical indicators rather than the other way around due to critical mass of traders believing in that indicator. However, if this is the case, every participant in the market, would try to front-run each other and make the market efficient disappear, only yielding profits to very few players. Additionally, technical analysis is also theoretically impossible to apply to inefficient markets (where information is not priced timely), as it is essentially a random walk, which we explored earlier on.

Note that this study uses logarithmic instead of linear price charts for technical analysis. The reasoning is that we are more concerned with percentage swings rather than changes in absolute prices.

2.7 Dividend Noise

Dividends are permanent asset (cash) outflows from the company to shareholders. Since we are concerned with total return of our trading, having periods of value spikes (before dividend payment) and drops (immediately after), distorts our analysis. Note that since dividends are paid quarterly but we use daily data, this noise cannot be eliminated completely. The discount is usually applied near the dividend declaration date, and not distributed throughout the quarter evenly. Nevertheless, since the Nasdaq-100 companies pay out less than 0.7% in dividends, this noise should not be material.

2.8 Tracking Error or Slippage

Indexes are theoretical in nature and are thus frictionless. However, this is difficult to achieve without slippage. One culprit of tracking error are transaction costs. Futures are settled daily, meaning that rebalancing and mark-to-market occur at the end of the trading day, everyday. This daily re-settlement has impact and transaction costs, which is detailed in Chapter 4.

Another problem is dividend reinvestment. Our dividend-adjusted data assumes dividend reinvestment on the day of issuance every quarter. This is unrealistic without algorithmic execution, and even then it incurs additional costs. In general, while indexes are good benchmarks, real-life returns still see degradation overtime due to the compounding effect of these tracking errors.

2.9 Short Selling

The liquidity and cash-settled nature of futures allow them to be borrowed from a broker, and sold at a lower price. This is known as ”shorting” and can double our problem space, allowing for machine learning to perhaps perform better. Nevertheless, we limit our study to a long-only portfolio since downturns are usually sharper and shorter, thus harder to predict. Further, a short position’s theoretical downside is infinite, as stock prices can climb infinitely. However, the upside is only 100% in the case of bankruptcy. This is worsened by the fact that markets do go up overtime both due to improved earnings and inflation. This makes short-selling a losing long-term strategy.

2.10 Idiosyncratic and Systematic Risk

During moments of market stress like the recent 2009 Financial Crisis, asset prices becomes very correlated, where all stocks drop simultaneously. Thus, it is impossible to eliminate systematic risk even if we do use an index. However, these events are still worth learning, so we can avoid pitfalls in investing. A transition into shorting strategies might perform well in these environments, although it is be difficult for retail investors to borrow stocks at a reasonable price.

2.11 Biases

2.11.1 Data dredging, fishing, or snooping

Data dredging, fishing or snooping is the process in which multiple hypothesis are tested on a single dataset to forcefully find a correlation when none may exist. Conventional statistical tests like Welch's t-test assumes probabilities that a particular result would be tested against the null hypothesis. By testing multiple variables, there is always a chance that the results fall out of the confidence interval, resulting in erroneous results. In our research, we test our results against a benchmark one at a time. In fact, we only perform the t-test half of a dozen times throughout the entire paper.

2.11.2 Look-ahead Bias or Data Leakage

Look-ahead bias occurs when our training data can see into the test set. This occurs when data sets are not separated properly. In our case, we keep out-of-sample completely separate. Cross validation is performed only on the training set.

2.11.3 Survivorship Bias

The constituents of indexes change over time due to privatization, and bankruptcies. Stocks today, may not exist in 10 years. The churn of the Nasdaq 100 Index, is anywhere between 1-3% depending on economic conditions.[29] Fortunately, index futures track indexes which do rebalance annually, eliminating the need for us to do so.

2.11.4 Overfitting

Overfitting is also another way of saying that "past performance does not guarantee future results." For LSTM we wish to tune our dropout. Dropout regularization randomly selects a percentage of neurons to ignore from the specified layer during training, so contributions to the activation of downstream neurons is temporarily interrupted on the feed forward step, and weight updates are not applied to the neurons during the backwards pass. This increases co-dependence between neurons such that the network as a whole is less sensitive to specific weights of individual neurons, resulting in multiple independent internal representations. This in turn improves the ability of the network to generalize and mitigates the probability of overfitting. Similar concepts apply to Random Forest, where we trim decision trees short, or increase leaf size. Discount Rate will be varied for Q-learning.

2.12 Machine Learning and Factor-based Models

State of the art financial models combine multiple ordinary least squares factors into a single multi-factor model for price prediction. However, recent advances in machine learning has improved, although only slightly the performance of these models using Recurrent Neural Networks. The following chart explains how linear multifactors models can be improved with machine learning techniques.[24]

Method	IC
Linear Multifactor	6.2
Boosted Decision Trees	6.3
Kernel Method	6.9
ML Multifactor with 3 Extra Features	7.2

Table 2.1: Machine Learning Improvements using Technical Indicators[24]

2.13 Random Forest

Random Forest is a greedy supervised learning algorithm seeking to infer a function from the input (a labeled training dataset) to hopefully produce correctly labeled outputs (as verified by the test set). It is robust to noise and often produces surprisingly strong results.

It grows multiple trees based on random feature selection (node splitting). This makes the algorithm inherently greedy, similar to how day traders may think in if-else statements. To put it simply, Random Forest builds multiple decision trees and merges them together to produce more consistent and accurate predictions. This is done through bagging.

It overcomes the problem of decision trees' tendency to overfit. (low bias, but very high variance) Random Forest averages multiple trees, reducing variance, but introducing only a slight bias. Since correlation and noise is very strong in stock data, bagging leads to greatly improved performance of the final model.[17]

The classification simplicity of the algorithm means it does not require any data scaling or normalization. Its allows feeding in of numerical and categorical data. In our case, we want to allow the model to learn both from continuous stock data and ternary technical indicator signals.

SVM maximizes the ”margin” thus allowing for the concept of ”distance.” This is important to gauge the magnitude of a move. However, since we already have technical indicators which measures these, it is largely redundant. Further, the goal of this project is not to find the magnitude of price swings, but its direction.[35]

2.14 Q-Learning

In reinforcement learning, the model trains from reward and punishment, rather than labeled data. This allows our agent to learn with the goal of risk-adjusted return in mind instead of telling it what is the correct action to start.

Rather than having greedy if-else statements from decision trees which works better in the immediate short-term, Q-learning seeks to maximize reward in a manner similar to depth first search. It can be understood as a model-free approximation of dynamic programming. The drawback is that the maximum reward path may be random and not be applicable to future market environments.[34]

Q-learner works in two phases - exploration and exploitation. Initially when the Q table is empty, Q-learner should 'explore' all the actions randomly. As the table gets filled up, Q-learner can choose to 'exploit' the best action from the Q table. As a Q-function converges, it returns more consistent Q-values and the amount of exploration decreases.

A simple and effective fix for the above problem is random greedy exploration – where either a random action, or a greedy action with the highest Q-value is chosen. DeepMind actually decreases randomness over time from 1 to 0.1 – in the beginning the system makes completely random moves to explore the state space maximally, and then it settles down to a fixed exploration rate.[20]

In a Markov chain, for any given state the process has a certain probability of transitioning to another state with every step. A famous application of Markov chains is Google's PageRank algorithm, which models link traversal as a Markov process, with each link on a webpage having a certain probability of being utilized.

2.15 Mean-Variance Optimization

Rather than optimizing the Sharpe Ratio directly, the Mean-Variance Optimization (MVO) can pick a return and try to optimize for the lowest standard deviation, therefore improving the Sharpe Ratio afterwards. The uniqueness of the MVO algorithm is based on the strategic transformation used for mutating the offspring based on mean-variance of the n-best dynamic population.[10]. However, this may be a subject for future work, as it would expand the scope this project too much.

2.16 LSTM

Traditional neural networks such as feed-forward neural nets have a drawback in that they assume all inputs are independent of each other. In many problems like price predictions, inputs (and outputs) are most definitely interdependent. Recurrent Neural Nets (RNNs) fix this problem by sharing information between steps. They add the output of a step to the input of the next one. This allows the RNNs to have a “memory”, enabling them to distinguish patterns over time.

In an LSTM network, the neurons in an RNN are instead replaced with memory “cells”, which consist of an internal memory state and several “gates”. These gates are neural network layers in their own right, learning a set of weights that allows them to determine what to forget (the “forget gate”), what to update the internal state with (the “input gate”) and what to output (the “output gate”). LSTM’s, by design, have no difficulty discerning connections over long sequences of data. Each cell’s internal memory state allows it to keep any relative weightings intact over time. This makes LSTMs the perfect candidate for price predictions, since in the market cause and effect can be quite far apart.

The cell always remember the information (hence memory) while the gates can sample from closer or farther data points in a time series. This optionality allows the network to gain additional insight into the network, compared to the greedy method demonstrated in Random Forest. LSTM units were devised to partially tackle the vanishing gradient problem in backward propagation, whereby neurons can reach near 0 gradient, and thus not able to relay useful information to the output node.

The following is a visual representation of the LSTM process. Credits go to Colah.[9]

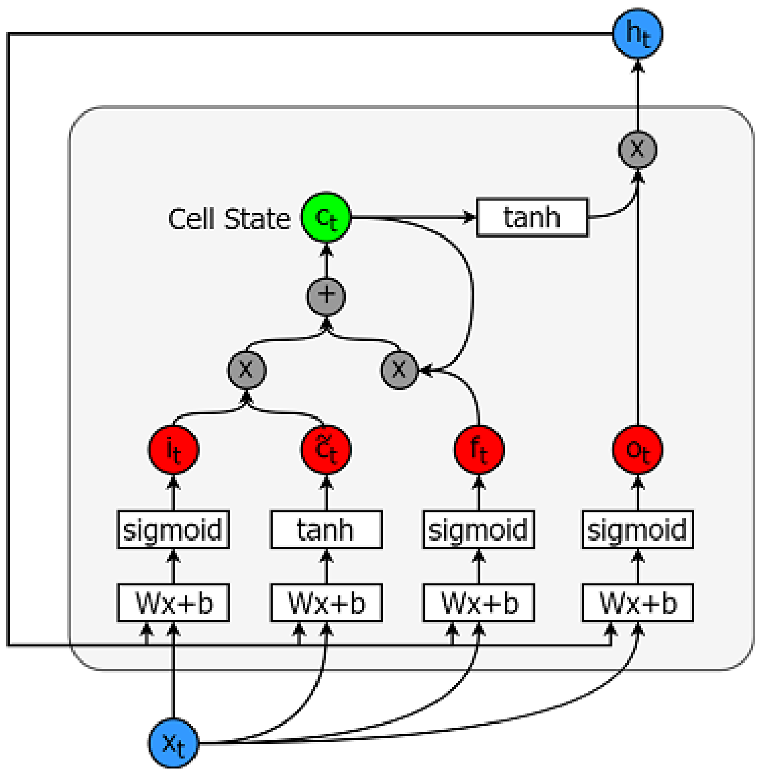


Figure 2.4: An unrolled recurrent neural network[9]

As we explored in the Random Strategy section, errors compound very quickly in long-term forecasts. Just like how we might be able to use previous and current frames to inform us about what might happen in future frames in video processing, LSTM might be able to predict future asset prices by understanding short or long term dependencies.

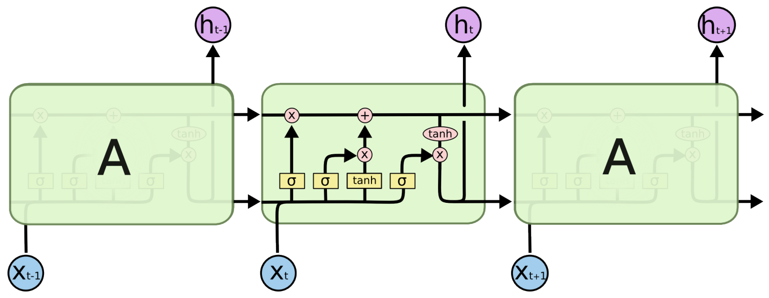
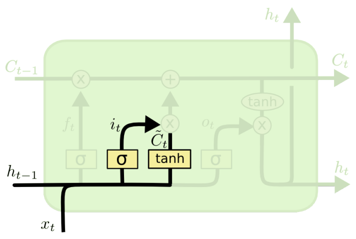


Figure 2.5: Repeating module in a chain[9]

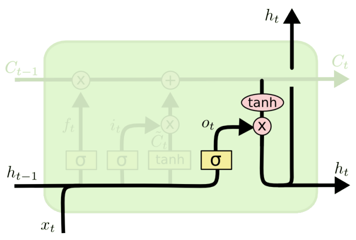
It achieves this by having the sigmoid layer gate selectively letting information through to control the cell states. A weight of 0 "forgets" and 1 "remembers." To update the values, we first decide which values we want to update through the "input gate layer", then subsequently create a vector values update the values.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 2.6: Rour interacting layers[9]

Then we finally filter our cell state with a sigmoid gate (which parts of the cell state to output) and then push it through a tanh (-1 to 1, whether we want to output it).



$$o_t = \sigma\left(W_o \ [\, h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.7: Four interacting layers[9]

Chapter 3

Data Preprocessing

3.1 Data Source

Our stock price data derives from Alpha Vantage [33], a free, real-time data streaming solution for our analysis. Having streamed online data is very critical as local, downloaded databases can become outdated quickly. The downside is that for unpaid users, the download speeds can be slow during peak trading hours. However, this can be avoided, since we use adjusted daily close data.

Bitcoin and Ethereum price come from CoinMarketCap.[8] It's free and fast. We chose late 2013 to be the commencement of our tracking for Bitcoin because that was when trading volume became reasonably high, and more exchanges started forming, increasing the interest and USD liquidity in the product significantly. Periods prior to 2013, were very volatile, and volumes stayed exclusively in Asia. Low volume data is exceptionally susceptible to market manipulation. The choice to use Bitcoin and Ethereum is also because they have the highest volume, allowing our algorithms to learn patterns instead of noise.

3.2 Data Structure

Our data is indexed by Date and contains numerous columns.

- Open: is the price in which the first transaction occurred on the day. It is heavily influenced by events during non-trading hours and is often different than previous day's closing price.
- High: is the highest price reached during the day. It is a measure of sentiment and emotional overshooting in the market.

- Low: is the lowest price reached during the day. It is a measure of sentiment and emotional undershooting in the market.
- Last: is the most recent price of the security. Depending on the frequency of the data, it could be the previous second, hour, or month. This is not used.
- Change: is the difference to the previous day. Percentage change is computed by ourselves, thus this is not used.
- Settle: is the most important data point, as it marks the daily-resettled price of futures.
- Volume: is also an important data point, as it can confirm the strength of changes in price.
- Open Interest: marks the amount of participants with an open order in the market. When neither parties wishes to haggle, situations can occur where buyer and seller expectations don't meet. This can lead to high open interest but low volume. We have no use for this.

The earliest available data for E-mini Nasdaq Futures are from 1999-06-22. Given that we use 5 columns of data, there is about 25,000 data points.

Date	Open	High	Low	Settle	Volume
2018-08-31	7641.75	7682.25	7633.25	7661.25	342731
1999-06-22	2240.00	2310.00	2238.00	2242.50	9250

Table 3.1: Nasdaq Futures Data from AlphaVantage[33]

3.3 Splice Noise

Front month CME E-mini index futures mature every 3 months. This means that buyers and sellers have to re-arrange the new futures price for the next quarter. This usually leads to big gaps in price, since there is a cost of carry (margin cost or time value of money) associated with holding any financial instrument. Further, speculation on future prices or liquidity on the rolling day may also play a factor in the gap. Best efforts in smoothing this gap using the carry cost is made.

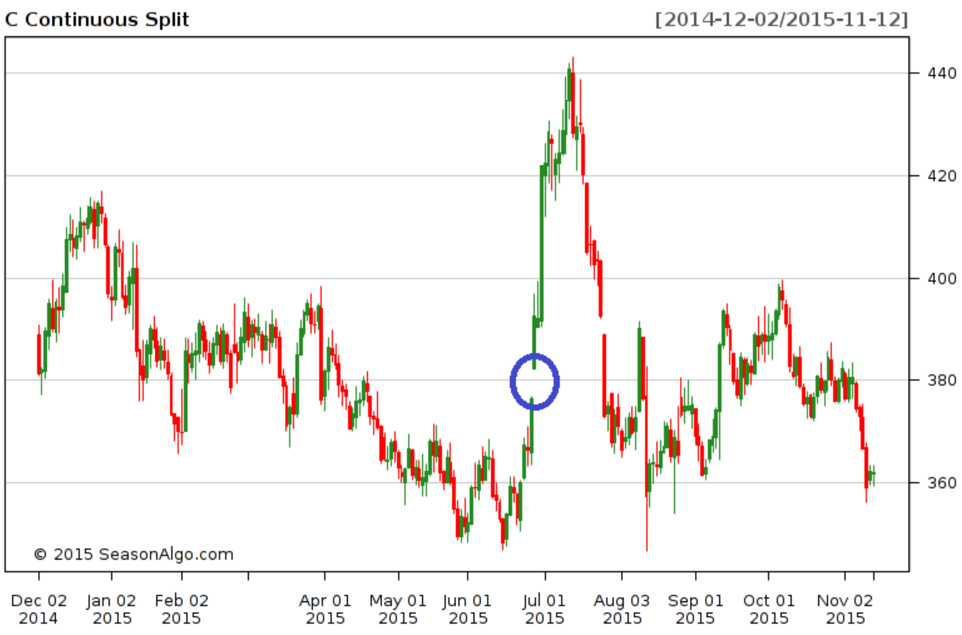


Figure 3.1: Futures data series adjusted for gaps or splice during roll-forward periods[28]

3.4 Sampling Frequency

We have a choice of using data in secondly, minutely, hourly, monthly, or even yearly intervals, though daily data has the biggest advantages.

Futures are settled at the end of each day. Lower interval does not take into account the action of this settlement which constitutes 70% of the market makers, even if some of them are automated. Automated statistical arbitrage algorithms enter the markets near the end of the day to take advantage of any mispricing. This usually ensures that daily data is free of any major distortions in regular market conditions.

Higher frequency data is exceptionally noisy. Human sentiment and reaction during major market events are likely to overshoot, only to rebound later, creating noisy volatility. Lastly, to fully make use of high frequency data points, we require additional resources, such as proximity to an exchange, fast internet and efficient algorithms, all of which are beyond the scope of this project.

Chapter 4

Methodology, Performance, and Validation

4.1 Ensemble of Machine Learning Models and Technical Indicators

One of the most important contributions from this study is the usage of technical analysis to augment machine learning. This is achieved by sending technical indicator signals to the market simulator alongside the machine learning signals. A lower weight is put on probable technical signals while a higher weight is placed on less probable ones. Next, a decision on the technical indicator is made by signal strength, of which the machine learning algorithm has the final say in. This has the effect of reducing the number of overall trading signals, but produce results better than the machine learning or technical indicators can do individually.

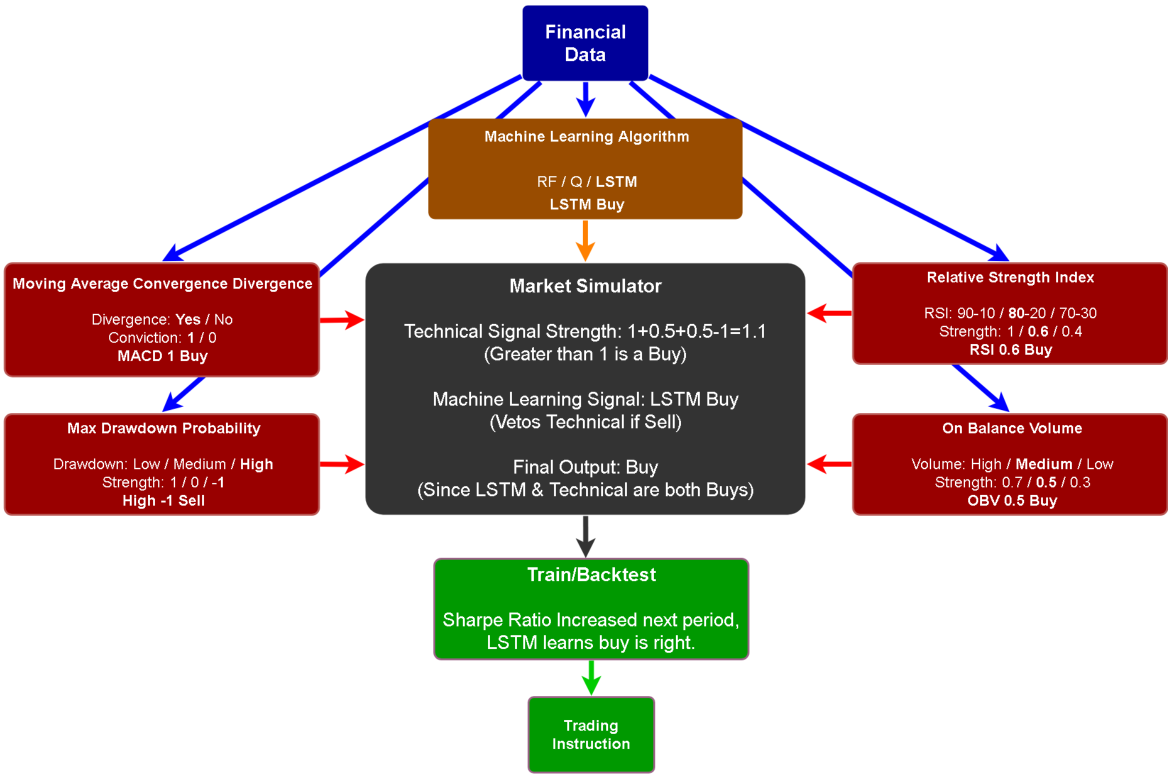

Financial Data
Machine Learning Algorithm
RF / Q / LSTM
LSTM Buy
Moving Average Convergence Divergence
Divergence: Yes / No
Conviction: 1 / 0
MACD 1 Buy
Max Drawdown Probability
Drawdown: Low / Medium / High
Strength: 1 / 0 / -1
High -1 Sell
Market Simulator
Technical Signal Strength: 1+0.5+0.5-1=1.1
(Greater than 1 is a Buy)
Machine Learning Signal: LSTM Buy
(Vetos Technical if Sell)
Final Output: Buy
(Since LSTM & Technical are both Buys)
Relative Strength Index
RSI: 90-10 / 80-20 / 70-30
Strength: 1 / 0.6 / 0.4
RSI 0.6 Buy
On Balance Volume
Volume: High / Medium / Low
Strength: 0.7 / 0.5 / 0.3
OBV 0.5 Buy
Train/Backtest
Sharpe Ratio Increased next period, LSTM learns buy is right.
Trading Instruction

Figure 4.1: Visual Representation of Signal Strength Decision Process

The net effect is that when technical indicators wants to sell, but machine learning algorithm wants to buy, a sell occurs. The goal of this method, is so that the machine learning decision is never overturned by technical indicators, so learning can occur. The drawback of this method is that the machine learning model might conform to the technical indicator signals over time. Measures against this bias would be to add a random factor, or allow the algorithm to learn which indicators to use beyond the Signal Strength method.

4.2 Financial Performance Metrics

Financial metrics such as daily average return, volatility, Sharpe Ratio, will be discussed. Optimizing for the highest Sharpe Ratio is the top priority for both technical and machine learning strategy. Note that this is possible, because Sharpe Ratio is convex function.[14]

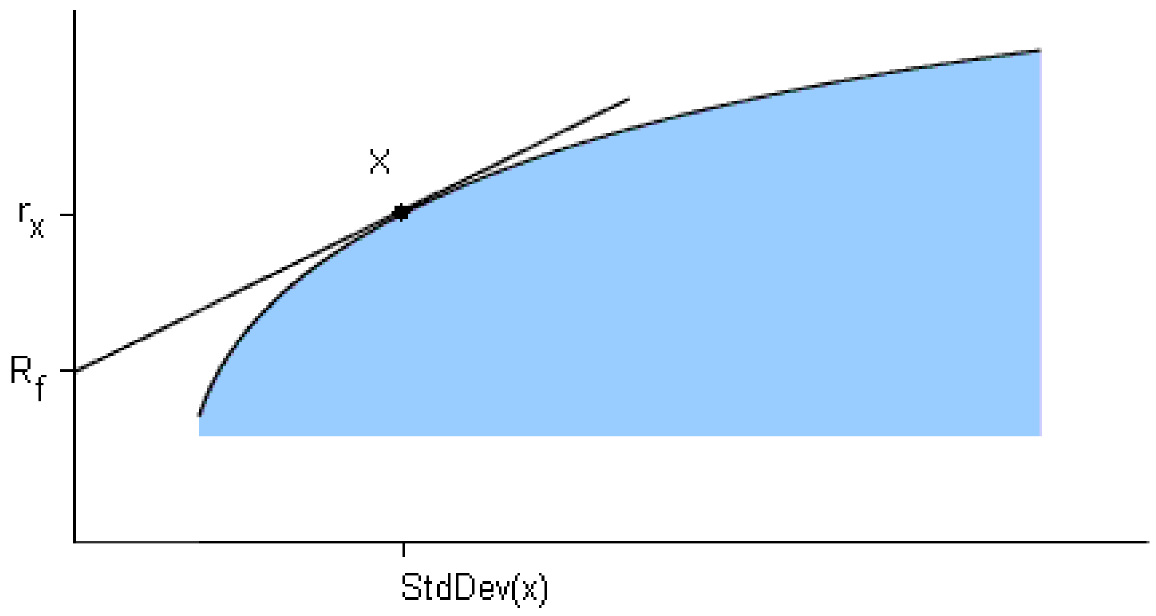


Figure 4.2: Sharpe is the Ratio of Return over Risk. Given its convexity, there should exist only one theoretical maximum.[23]

4.2.1 Daily Return

Daily return is important as a reward/penalty mechanism to our trading, because it is used along with standard deviation to compute the Sharpe Ratio. It is also important for higher frequency trading strategies. In the extreme case, a daily return can be the cumulative return if our window is 1. It is computed by taking today's closing price and divide it by the previous day's close. Higher is better

$$Daily\ Return = \frac{Price_1}{Price_0} - 1$$

4.2.2 Volatility

Volatility or risk is a critical measure, especially if levered. With a high volatility strategy, we may lose all of our money before the expected value is achieved. "The market can stay irrational longer than you can stay solvent." by Keynes, perfectly explains the risks of volatile strategies. We explore this further with maximum drawdown to prevent negative volatility. It is computed using the standard deviation of daily returns. Lower is better.

$$Volatility_n = \sigma\ Daily\ Return_n$$

where n is the time window

4.2.3 Sharpe Ratio – The Goal

Sharpe Ratio optimization is our goal, because a strategy with high cumulative return but higher volatility, can be easily outperformed by a strategy with higher Sharpe Ratio given the allowance of leverage. In fact, most volatile derivatives like futures have very leveraged underlying assets. When Investors want to exit these risky positions quickly, more cascading volatility is created. Higher is better.

$$Daily\ Sharpe\ Ratio = \sqrt{Days}\,\frac{Mean\ Return - Risk\ Free\ Rate}{Volatility}$$

$$Daily\ Risk\ Free\ Rate = (1 + Proxy)^{(1/Days)} - 1$$

where days is the number of trading days or compounding periods per year, usually 252. Proxy is usually the 10-year US Treasury Yield, currently near 3%. Note that any return below 3%, makes Sharpe Ratio negative, thus discarded, since we would rather put our money into risk-free treasury bonds.

4.3 Classification Performance and Validation Metrics

Correct classification only measures direction not magnitude. For example, a strategy that classifies correctly 5 times on 1% up days, performs poorly if an incorrect classification occurred on a single 10% down day. While classification are important metrics to help us tune our strategies, Sharpe Ratio is our ultimate goal.

4.3.1 Confusion Matrix

The confusion matrix is a visual representation of classification correctness. In our case, since we only have 2 decisions: buy or hold, a two by two error matrix results.

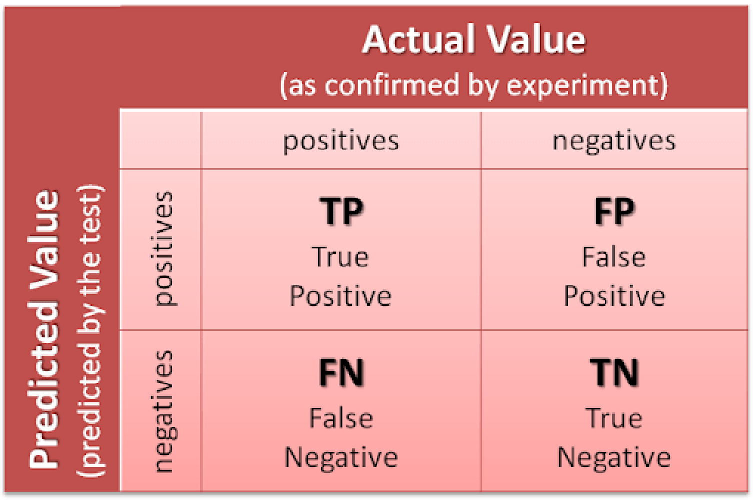


Figure 4.3: Confusion Matrix Visualization[25]

Accuracy is the ratio between correct and total classifications.

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

Precision just measures true predictions. This is not very useful for us, since we want to know if we are selling at the right time.

$$Precision = \frac{TP}{TP + FP}$$

Recall is the same as Precision but takes into account FN instead of FP.

$$Recall = \frac{TP}{TP + FN}$$

F1 score is a harmonization between Precision and Recall. It is a better metric. However, given the important of True Negatives in our computation, F1 is not adequate.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

As expected, there are no perfect statistic that can summarize the confusion matrix. Nonetheless, MCC does this the better than most.[26]

4.3.2 Matthews Correlation Coefficient

Matthews Correlation Coefficient or MCC is our choice of prediction accuracy. It is essentially a correlation coefficient between the observed and predicted binary classifications. It ranges from -1 to 1, with -1 having inverse agreement between classification and prediction, 0 having weak agreement, and 1 being perfect agreement. We choose this metric because it ameliorates the problem of imbalance classification results, which is a definite problem for low signal strategies.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

4.3.3 Cross Validation

All strategies have their in-sample data sets go through a 10-fold cross-validation process to reduce variance or overfitting to certain segments of our dataset. We slice off or reserve the most recent period for out-of-sample testing. The split is 90-10, because our real-life testing period tend to be short also.

4.3.4 Random Walk vs Welch's unequal variances t-test

While we can achieve good classification, the good classification may simply come from the skewed classification distribution. To test for this fallacy, we run a one-sided Welch's unequal variances t-test where we need to reject the null hypothesis that our strategy does not outperform the random walk with a 95% confidence interval.

To past the test, we need to ensure that the t-stat is not negative and the critical p value is < 0.05.

$$t = \frac{\bar{X_1} - \bar{X_2}}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}}$$

where $\bar{X}, \bar{X_2},\ and\ S_1^2, S_2^2,\ and\ N_1, N_2$ are the sample means, sample variances and sample sizes, respectively.

Chapter 5

Market Simulator Architecture

The market simulator is an important implementation component of this project. The API takes in input parameters such as the data, date, and transaction cost. It also takes in the output signals from our technical and machine learning algorithms. It then sorts all these signals, deciding on which ones to ultimately use. It finally backtests the outputs against the test set to create performance and validation metrics. Finally, it writes an executable buy/sell trading instructions in a dataframe or a CSV file.

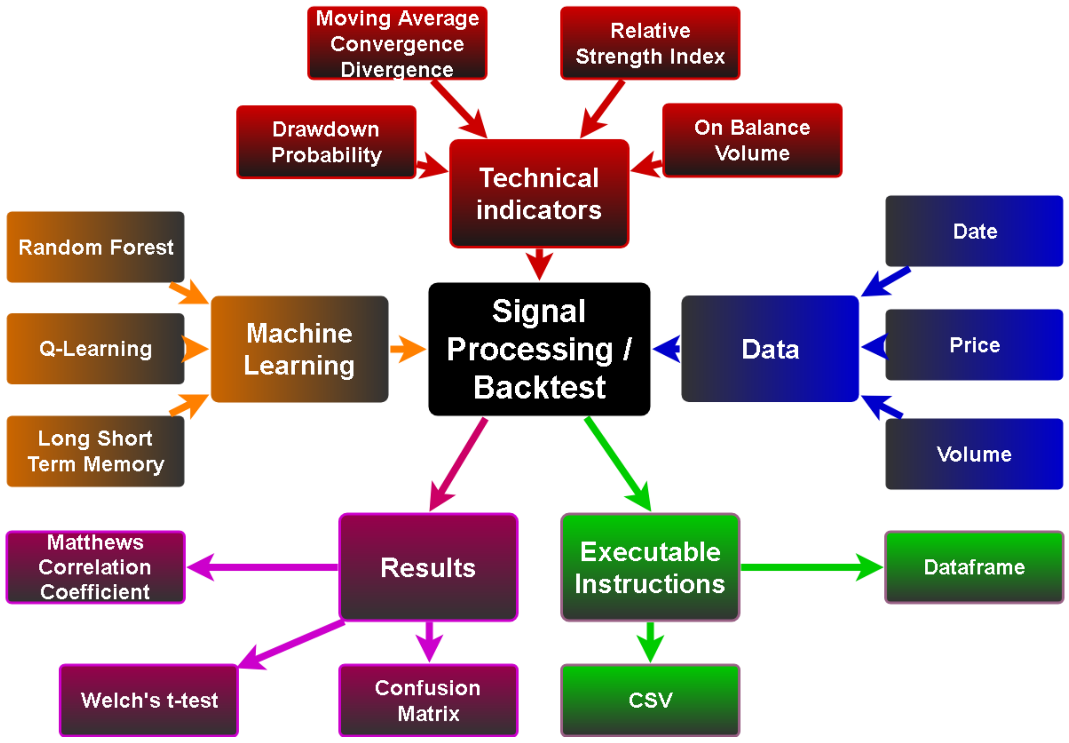


Figure 5.1: The market simulator takes in data, and signals from technical indicators and machine learning algorithms as inputs. It then sorts these signals, and produces performance analytics and trading instructions.

5.1 Input Parameters

Our market simulator takes several parameters, including the start/end date of our data, initial investment, commission, and impact.

5.1.1 Start and End Date (Time-Period Bias)

The start and ending period of our analysis is important, as it affects our perceived return. This is a problem specific to time-series data. For example, if we started our evaluation of Nasdaq after the Dotcom Bubble in 2000, our return would be less than 80%. In contrast, if we set our training/testing period after the Bubble in 2002, the return would be over 700%. We must choose our training and testing periods carefully, particularly with cross-validation, to ensure our strategy is robust at all times, but still able to detect large price swings.

In our case, we chose to start in 2005, because the underlying composition of the Nasdaq Index then is very similar to the one today. While it won't be able to experience the Dotcom Bubble, it will be able to experience the 2008 Financial Crisis.

5.1.2 Transaction Cost (Commission and Impact)

Transaction costs are one of the primary contributors to the failure of high-frequency trading strategies and retail traders. Even if a trader (human or robot) is right 51% of the time, he or she would still lose money. Commission refers to the money paid to stock brokers which are licensed institutions that holds custody of our shares and cash.

Impact refers to the percentage of money lost to not being able to bid at the desired price. This is very well explained in Flash Boys.[15] In the book Michael Lewis describes how a RBC trader bidding on the price of a stock he saw on his screen often did not materialize, even though it was higher than the ask price. This phenomenon was attributed to high-frequency firms like Citadel which had their servers moved near exchanges and invested in private optical fibers which traveled faster and through shorter distances. This allowed the HFT firms to "front-run" large orders from RBC by reading it, buy it at the current price, and then sell it back to RBC at a higher price, making a risk-less profit.

As one can see from the below chart, a retail trader, trading from a computer at a home, often a continent away, at 0.02 impact, is at a significant disadvantage to a High Frequency Trading firm with fast and close cables at 0.0005 impact. The difference can be up to 3.5% of the entire portfolio value! This has become a serious ethical question, as many debate whether this type of exploitation actually contributes to the efficiency of the markets, and whether it amounts to market manipulation. Thus far, regulators have not taken any action, even though throttling technologies that can "even the playing field," already exists.[18] To simulate the effects of impact, our models will assume a midpoint transaction slippage of 0.005.

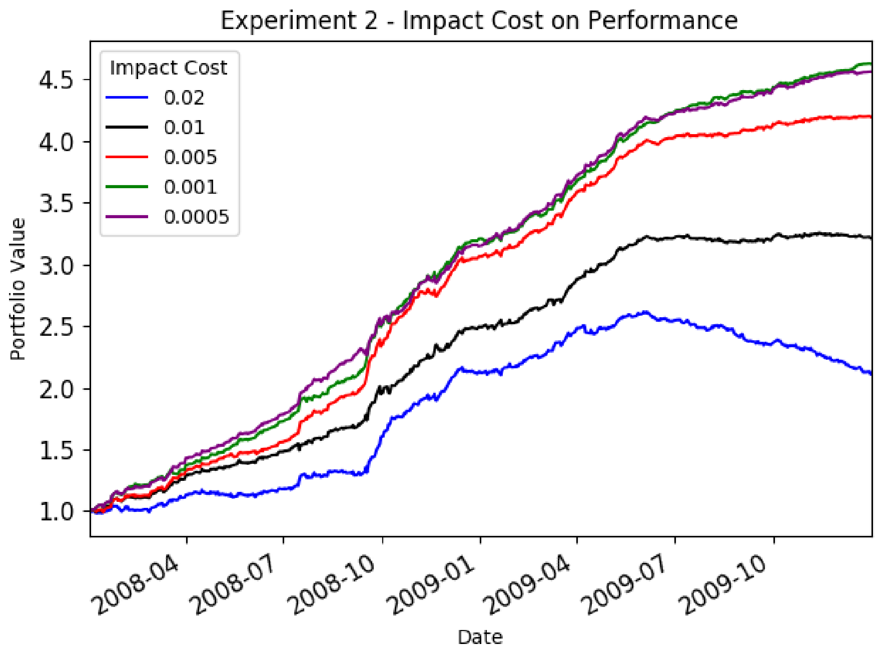


Figure 5.2: Trading Impact Cost in %

5.2 Signals, Position, Shares, and Cash

Our Simulator continuously rebalances our portfolio of cash and shares depending on the signals we receive from our trading strategies.

5.2.1 Buy/Sell Signals

Our simulator takes 1, or -1 as signals to buy, sell or hold. These are outputs from our technical indicators and machine learning algorithms. Note that in the absence of a buy or sell signal, we hold the investments by doing nothing.

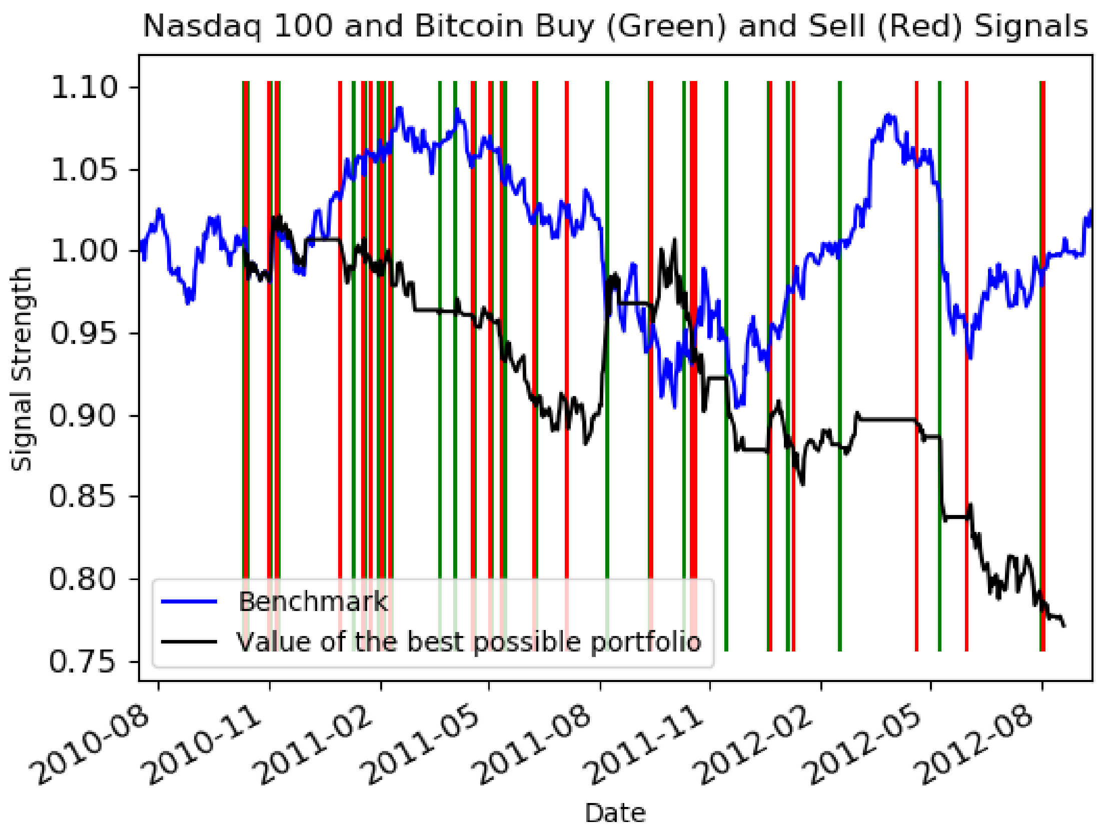


Figure 5.3: Buy(Green) Sell(Red) Hold(Nothing)

5.2.2 Long/Short Position and Leverage

By taking these signals, we increase the share count (buy), and thus reducing our cash. Negative cash can occur in cases where we own more shares than our entire equity. This is called "buying on margin" or using leverage/borrowed money from our broker. A daily commission penalty is applied at 3.07% (as currently charged by Interactive Brokers, the most popular retail broker[5]) annualized rate each day. This acts as a natural penalty towards levered strategy. Margin call is a situation in which the broker forces an investor to sell securities until a safe leverage ratio is reached to ensure the remaining equity absorbs any potential losses from current holdings.

In the case where we sell, we can be in a position where we have more total cash than equity, along with negative numbers of shares. This is known as "short-selling." In this case, we are borrowing shares, and given cash instead. These are usually lent out at similar rates as margin, but only for selected stocks.

Date	Symbol	Order
2018-09-10	NQ	HOLD
2018-09-11	NQ	Buy
2018-09-12	NQ	HOLD
2018-09-13	NQ	HOLD
2018-09-14	NQ	HOLD

Table 5.1: Table representation of trade instructions for 5 days.

Chapter 6

Technical Analysis and Principal Component Analysis

6.1 Overview

In this strategy, we first extract 6 out of the 29 most popular technical indicators using Principal Component Analysis. 3 of the 6 is then chosen for their individual predictive power. This process allows us to eliminate many other correlated but less accurate indicators. A self-developed maximum drawdown indicator is developed to prevent likely high loss trades.

6.2 Feature Optimization using Principal Component Analysis

There are many technical indicators. To make our research practical, we must narrow down our choices. Thus, we use PCA to find the the common factors in each indicator. This narrowed our choices down to 6, of which 3 stood out to be the strongest performers.

6.2.1 Problem Definition

Each vector is defined by the name of its technical indicator. It returns a dataFrame of daily Sharpe Ratios. Our goal is to find the correlation between the daily returns produced by these signals, to see how much of an overlap exists between each indicator. We then choose the one with the highest return amongst the group as our default choice.

Note that we can also use signals (-1/1) in our PCA vector. However, signals do not capture the magnitude. If a certain indicator is better at classifying large gains, than another, but had same direction of signals, both indicators would be equally desirable, which should not be the case. The classifier which sees large magnitude should win.

6.2.2 Standardization

The dataFrame is first standardized to produce comparable results.

$$X_m = \frac{Return_m - Average\ Return_i}{Average\ Standard\ Deviation_i}$$

where i is the feature before transformation and m is the feature state after transformation.

6.2.3 Covariance Matrix

While we can find the covariance matrix, it does not have a diagonal, requiring us to perform linear transformations.

$$Covariance\ Matrix_{ij} = \frac{1}{N-1}\sum_{N=1}^{N} X_{ij}X_{nj} = \frac{1}{N-1}(X^T X)$$

where ij are the Sharpe Ratios of the two indicator, and n is the total number of indicators

6.2.4 Linear Transformation

In order to rotate our data in order to give it new axis, we can apply linear transformation.

$$Z = XV$$

where Z is the data we want to transform. X is the decoded signal. V is an orthogonal matrix.

6.2.5 Eigenvalue Decomposition and Correlation Matrix

$$C = UAU^T$$

where C is correlation matrix. A is the diagonal matrix of ordered eigenvalue. U is orthogonal matrix that stores eigenvectors

We then compute the Correlation Matrix

$$Cov[Z] = \frac{1}{N-1}Z^TZ = \frac{1}{N-1}V^TX^TXV = V^TCV = V^TUA(V^TU)^T \xrightarrow{V=U} A$$

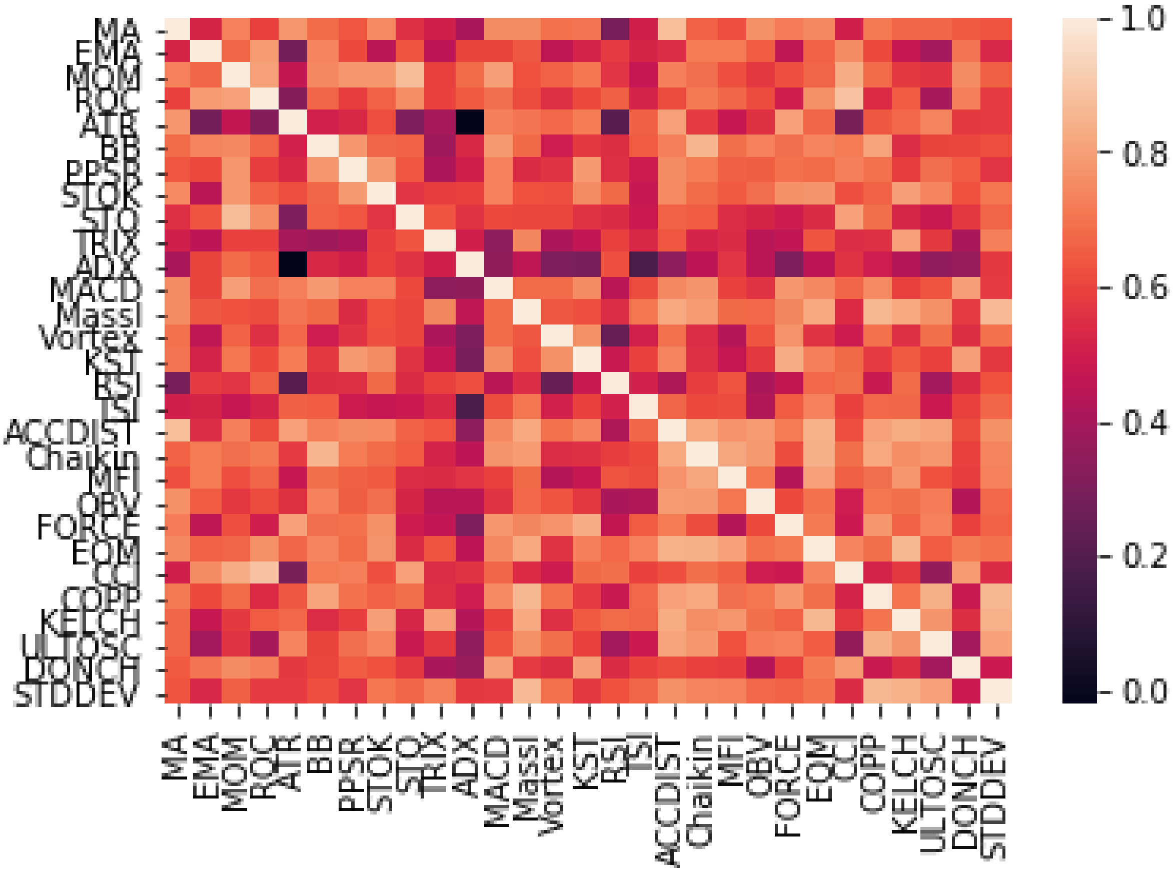


Figure 6.1: Correlation Matrix of 29 Technical Indicators

Note the light shade of the Correlation Matrix. This implies a significant amount of correlation between the indicators. This is reasonable given that many technical indicators tend to give correlated signals since they are generally linear and based on just price and volume data.

6.2.6 Principal Component Analysis Results

Features	MCC	DR	SD	SR
29	0.5904	0.001011	0.0148	0.5510
12	0.6215	0.001066	0.0138	0.5610
6	0.6205	0.001082	0.0168	0.5620
5	0.4315	0.000584	0.0208	0.4888

Table 6.1: Reduction below feature 6, causes significant drop in MCC and SR, while 12 to 6 seems stable.

Note that at 29 features, the MCC (signal prediction-test correlation) was quite low, and that perhaps hurt the final Sharpe Ratio. Also notice that between features 12 and 6, 6 had higher SR but also higher MCC. This happened perhaps because while the signal was more frequently correct for 12, they weren't right on days when the gains are large, or when the losses are small. While improving prediction is important, it is ultimately the Sharpe Ratio, or our risk-adjusted return that is important. Regardless, differences tend to be minor and they generally trend in the same direction.

6.3 Simple Moving Average

Moving average is the pillar of technical analysis. More than half of all technical indicators adopt some form of rolling mean (moving average). It is defined as

$$SMA(X_t, n) = \frac{1}{n}\sum_{t=1}^{n} X_{t-n}$$

where n is the time period (ex. 20 days). t is the date. X is value at time t.

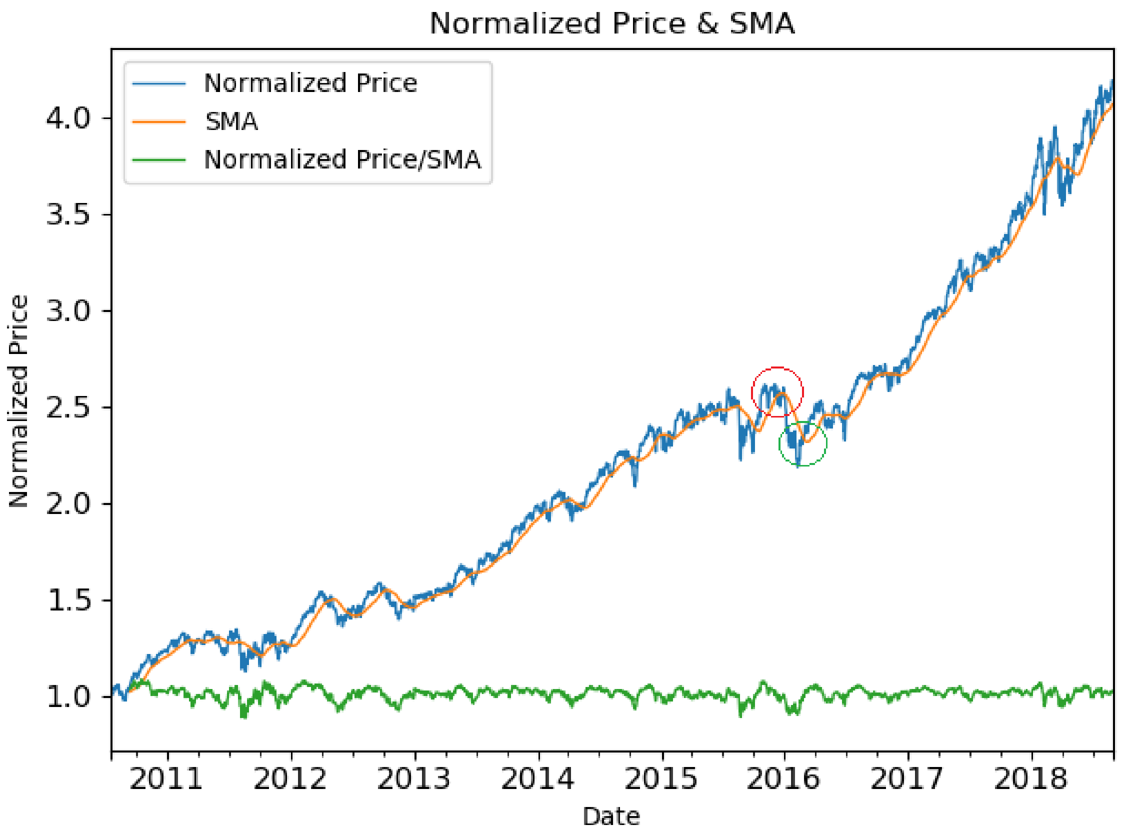


Figure 6.2: Simple Moving Average of Nasdaq. Red circle is bearish. Green is bullish %

As can be seen from the graph, the orange line is delayed by about 50 days (window) compared to the actual price chart. When a potential cross occurs, a trade signal is produced. If the price ascends into the SMA, it is a buy. If it descends into it, sell.

To make more use of this tool, SMAs of different windows can be used to gauge whether long-term trends are being broken. For example, when a 50-day SMA descends across a 200-day, it is known as a “Death Cross” or a breakdown of long-term trend, signaling a sell. When a shorter window SMA ascends across a longer window one, it is called “Golden Cross,” signaling a long-term buying opportunity.

Window	Signals	MCC	SR
10	1056	0.4885	0.5481
50	781	0.4954	0.5582
200	123	0.5315	0.5778
BM	N/A	N/A	0.5591

Table 6.2: Simple Moving Average Crossover Results for NQ

Note that compared to the buy-and-hold benchmark, the 200-day moving average crossover indicator significantly out-performs in risk-adjusted return. Nonetheless, with only 123 signals, this is a good indicator of how rare divergence can be robust to noise.

6.4 Moving Average Convergence-Divergence

The Moving Average Convergence-Divergence indicator (MACD) is among the strongest performer in backtests. Instead of a simple moving average, it uses exponential moving average, giving later or more recent values higher weight. This is generally consistent with the reality that if noise is constant, the older the data, the less likely it will affects future events.

$$EMA_t = \frac{Y_1, t = 1}{\alpha \cdot Y_t + (1 - \alpha) \cdot -1, t > 1}$$

where Y is the data series. t is the time window. α is the smoothing factor between 0 and 1 where higher weight discounts earlier data faster.

Subsequently, a third EMA usually of 9 days, is used to smooth the MACD, creating a crossover point (signal) for the other 2 EMAs (creating an oscillator). When the 9-day EMA intersects the MACD, in an ascending formation, is when one should buy. When the lower EMA is descending towards the moving average, is when one should sell.

$$MACD\ Line = EMA(12) - EMA(26)$$

$$Signal\ Line = EMA(MACD\ Line, 9)$$

$$MACD = MACD\ Line - EMA(9)$$

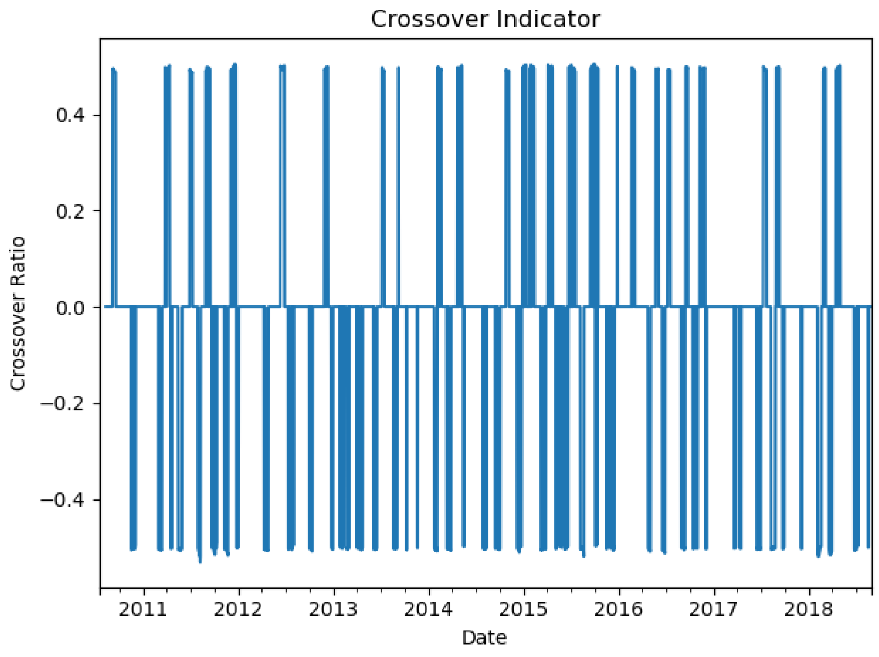


Figure 6.3: MACD Crossover Indicator of Nasdaq 100 Futures

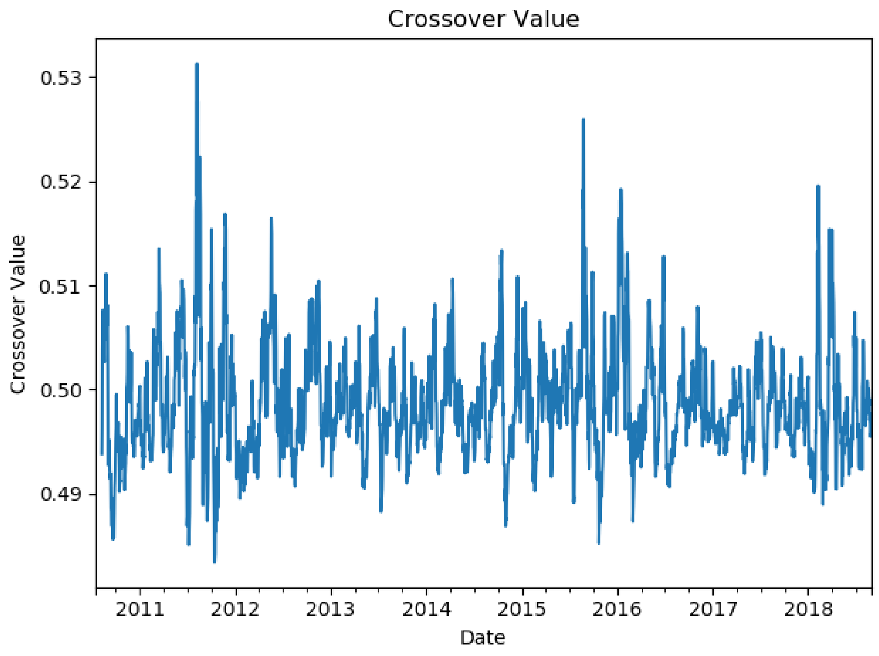


Figure 6.4: Volatility of Nasdaq 100 Futures During MACD Crossover

Not that when MACD indicates a buy or sell as dictated by a crossover, volatility increases. This is indicative of increased price activity (up or down). This demonstrates that our indicator is actually working (when market moves).

Window	Signals	MCC	SR
M12-26-9	162	0.6158	0.6088
BM	N/A	N/A	0.5591

Table 6.3: MACD Divergence Results for Nasdaq

6.5 Relative Strength Index

Relative Strength Index (RSI) is a momentum oscillator. The standard usage of the indicator dictates that when RSI surpasses 70, it is overbought, implying an impending drop in price. When the oscillator drops below 30, a rally should ensue. Its predictive power however, lies when the oscillator reaches extremes (¿90% or ¡10%).

Relative Strength Factor is then converted into an Index between 0 and 100:

$$RSI = 100 - \frac{100}{1 + Relative\ Strength}$$

$$Relative\ Strength = \frac{Simple\ Moving\ Average(Up, n)}{Simple\ Moving\ Average(Down, n)}$$

where Upward Momentum is

$$Up = Close_{now} - Close_{previous}$$

$$Down = 0$$

where Downward Momentum is

$$Up = 0$$

$$Down = Close_{previous} - Close_{now}$$

Usually The RSI would oscillate between 70 and 30. Traders take anything beyond that range as a trading signal. However, sometimes increased volatility in the market can create many false positives. Nonetheless, when the oscillator reaches extremes (above 90 or below 10), it usually is very predictive.

Another very high probable phenomenon is called a “bearish divergence.” This occurs when the RSI trends lower, but the price heads higher. It tells the investor, that the recent rise in price disobeys the long-term trend, and is thus a selling opportunity.

Extremity	Signals	MCC	SR
10/90	26	0.6072	0.6122
20/80	584	0.5654	0.5551
30/70	802	0.5113	0.5488
BM	N/A	N/A	0.5591

Table 6.4: RSI Results for Nasdaq

6.6 Bollinger Band

The Bollinger Band is similar to RSI in that it looks for overbought or oversold signals. However, instead of price momentum, it uses standard deviation to determine the entry points. Every time the price exceeds above the upper band, or drops below the lower band, we have a buy or sell signal. Computation of Bollinger Bands is fairly simple:

$$upper\ band = SMA + 2SMA$$

$$lower\ band = SMA - 2SMA$$

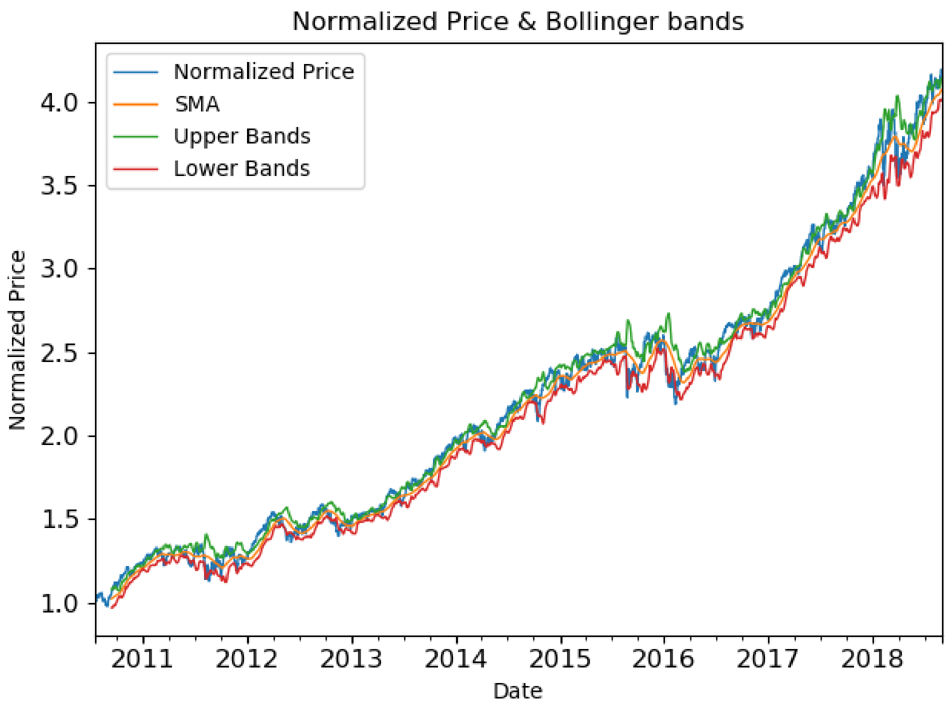


Figure 6.5: Bollinger Band for Nasdaq 100

Window	Signals	MCC	SR
10	56	0.3333	0.5715
50	221	0.4654	0.5524
200	22	0.5415	0.5431
BM	N/A	N/A	0.5591

Table 6.5: Bollinger Band Results for Nasdaq 100

6.7 Support and Resistance

Support levels are recent or long term low in price that a stock has reached that technical analysts and many market participants believe to be difficult to break. In other words, a short-term trend reversal is expected at a support level thus, signaling a buy.

Resistance levels are the opposite where recent or long term highs of a stock is perceived to be difficult to breakout from a second time, indicating a downward trend reversal, thus a sell.

The empirical evidence is fairly strong for support and resistance levels.[31] When a stock approaches a previous high, investors who bought near the previous high, often wants to "break-even," and sell at their original purchase price. Thus, selling pressure increases near a previous high. However, once the buying power overwhelms the "weak-hand" sellers, the stock breaks very strongly over resistance level, as selling pressure dissipates.

Support levels functions in the opposite way. We utilize a higher probability form of support and resistance known as a double "double or triple bottom." This means that a support level is tested twice or even three times, indicating huge buying power near these levels. By repeatedly exhausting selling pressure at these levels, it is very likely that a double or triple bottom supports a long-term upward bounce.

Window	Signals	MCC	SR
Support	89	0.5724	0.5778
Resistance	145	0.5836	0.5658
BM	N/A	N/A	0.5591

Table 6.6: Support and Resistance Results for Nasdaq

It can be seen that both support resistance perform well in classification (MCC), but its gains in Sharpe Ratio are modest. This agrees with the study.[31]

6.8 On Balance Volume

Volume is an important supplementary indicator to price, as price-based indicators are subject to low-volume noise. A breakout or down is only significant if sufficient volume was used to push it through. High volume is often (but not always) the consensus of a large number of market participants. Low volume environments are also subject to market manipulation.

Volume is also an indicator of movements of “large or smart money.” When large money moves into a stock, it tends stay a bit longer than traders. This makes rallies more sustainable rather than technical (fickle). While volume itself is not a signal, it is a great validator to other indicators, confirming high volume trends, and eliminating low volume noise, increasing technical indicators’ probability of success. Purchasing during high volume also reduces impact, as the bid-ask spread shrinks.

$$OBV = OBV_{prev} \begin{cases} volume & if close > close_{prev} \\ -volume & if close < close_{prev} \end{cases}$$

Total volume is assigned a positive or negative value each day depending on if the price went up or down. If OBV does not reach new highs, while the price does, it indicates a weak rally. If OBV moves up without a corresponding increase in price, it might mean that there is a pent up price increase coming.

In Conjunction With	Signals	MCC	SR
SMA 200	30	0.5715	0.5798
BB 200	22	0.4812	0.5325
MACD 12-26-9	162	0.6219	0.6090
RSI 10/90	26	0.6171	0.6291
Support+Resistance	126	0.5823	0.5722
BM	N/A	N/A	0.5591

Table 6.7: OBV as a Force Multiplier to Other Indicators

As witnessed in the above table, volume confirmation improved MCC at the expense of killing off significant amount of signals, This has improved the Sharpe Ratio for almost all indicators except for Bollinger Bands, which actually decreased slightly. This may be possible given that high volume tends to drive standard deviation higher, giving Bollinger Bands false signals.

6.9 Max Drawdown Probability

As noted in the Market Simulator section, leverage is a core component of our outperformance. Max Drawdown Probability (MDP) measures peak to trough declines in a security during major macroeconomic events. The magnitude of the gap and its frequency can inform us of large potential losses. Note that this is different than value-at-risk which seeks to compute the probability of a worst-case scenario or tail-risk occurring.

MDP can be implemented by dynamic programming.

Algorithm 1 Max Drawdown Probability

```
1: procedure MAX DRAWDOWN PROBABILITY($DR$)
2:     $cumsum = DR.cumsum()$
3:     $start = np.argmax(np.maximum.accumulate(cumsum) - cumsum)$
4:     $end = np.argmax(cumsum[: start])$
5:     $MDP = price[end]/price[start] - 1$
6:     if $MDP \geq drawdown$ then
7:         $count += 1$
8:         $prob = 1/len(cumsum)$
9:     end if
10:    Return $drawdown, count, prob$
11: end procedure
```

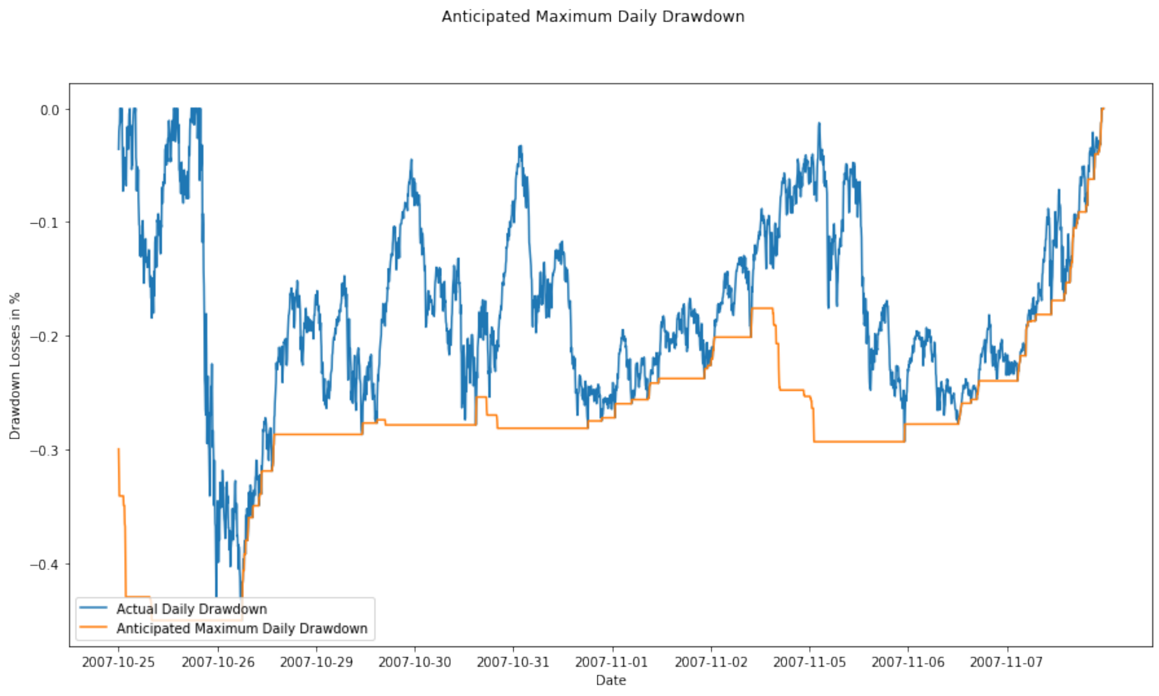


Figure 6.6: Predicted vs Actual Daily Drawdown for NQ

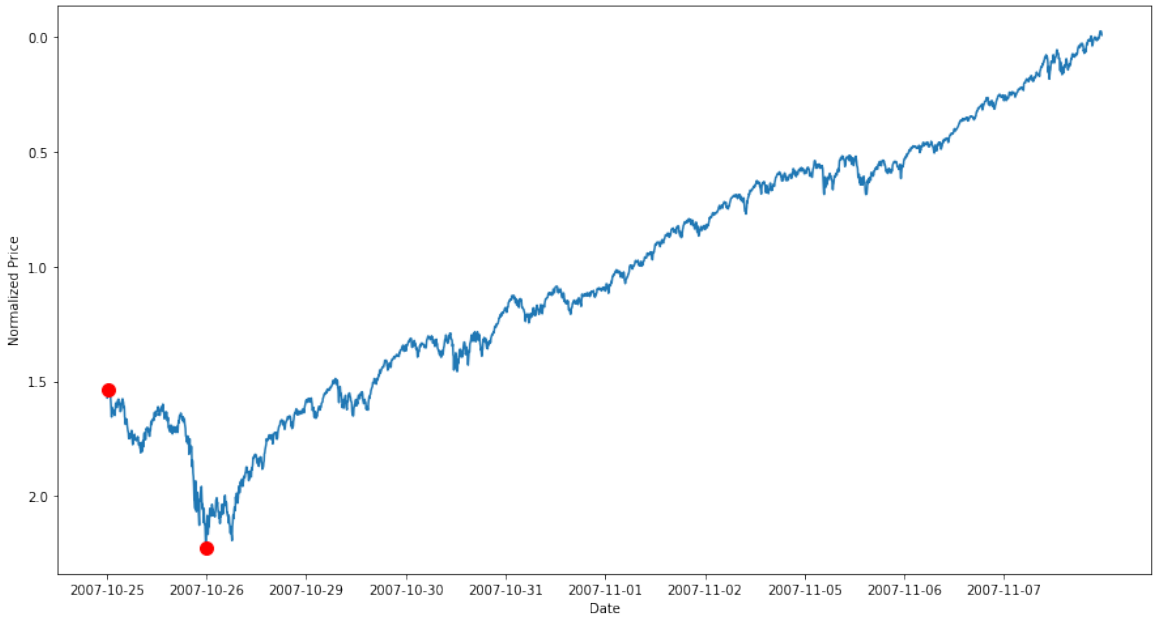


Figure 6.7: 50.45% Maximum Drawdown between 2007-11-01 and 2008-11-21

We can enhance our portfolio risk-adjusted return by stopping trades on days with high potential drawdown.

6.10 Best Performing Indicators

MACD, RSI and OBV were clear winners in terms of classification accuracy and risk-adjusted returns. We use these 3 indicators along with our very own Maximum Drawdown Probability to enhance the strategy going forward.

Chapter 7

Random Forest

7.1 Overview

In this strategy, we seek to define classification trees based on feature X. This can come from the pricing data, or any of the other technical indicators. Feature selection will be based on Information Gain. Random split will be used when the features are tied. We allow the tree to split until all data are classified, hence the minimum leaf size is 1. We reduce over-fitting by pruning after 300 decisions.

7.2 Feature Selection

Our features include daily average returns, standard deviation, high price, low price, and volume. A positive Sharpe Ratio in the next period means a buy. A negative return implies a sell. Otherwise it's a hold. We also take input signals from the 4 technical indicators we mentioned in the last chapter.

Information Gain seeks to reduce entropy in a data. Entropy measures the split in the information contained in a particular node of a tree. Entropy is the highest when all the classes are contained in equal proportion in the node. It is the lowest when there is only one class present in a node. It is computed as follows.

$$H(N) = -\sum_{i=d}^{i=1} P(W_i) log2(P(W_i))$$

where d is number of classes and $P(W_i)$ is the proportion of the population labeled as i.

Thus the best split is characterized by the greatest reduction in entropy:

$$\Delta I(N) = I(N) - P_L * I(N_L) - P_R * I(N_R)$$

where $I(N)$ is Information Gain of the node. P_L is the population in node N that splits into the left child. P_R is the population in node N that splits into the right child. $N_L N_R$ is the left and right child of N respectively.

7.3 Bagging

Bootstrap aggregating or Bagging is a way to randomly sample the training set with replacement to decrease the resultant tree variance without increasing much bias. It was first introduced by Breiman in 2001.[4]

$$B\hat{a}gs = \frac{1}{B}\sum_{b=1}^{B} Bags_b(\grave{x})$$

After training, predictions for unseen samples can be made by taking the majority vote of all the individual classification trees. While the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated.

Simply training many trees on a single training set would give strongly correlated trees or even the same tree many times, since our data is fixed. Bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Algorithm 2 Train Random Forest

```
1: procedure TRAINRF($RandomForest(Data)$)
2:     $RF = newArray$
3:     for i=0 to B do
4:         $Bag = Bagging(Data)$
5:         $DT = newDecisionTree(Data)$
6:         $Features = RandomSelection(Data)$
7:         $DT.train(Data, Features)$
8:         $RF.add(DT)$
9:     end for
10:    Return $RF$
11: end procedure
```

7.4 Result Discussion and Validation

7.4.1 Results

Parameter	Signals	MCC	SR
RF	413	0.5536	0.5791
RF Ensemble	87	0.6089	0.6421
BM	N/A	N/A	0.5591

Table 7.1: RF Results for next 2 years

As can be seen from the above table, Random Forest with 4 Technical Indicators (MACD, RSI, OBV, MDP) performed better than just Random Forest. An achievement of 0.61% prediction accuracy and a 16% increase in Sharpe Ratio in out-of-sample testing shows great promise.

7.4.2 Confusion Matrix

	Actual BUY	Actual Sell
Predicted Buy	31	23
Predicted Sell	12	21

Table 7.2: Confusion Matrix for Random Forest Ensemble

Accuracy	Precision	Recall	F1
0.60	0.57	0.72	0.64

Table 7.3: RF Matrix Statistics

As one can glean from the matrix, recall is much higher than precision. This shows that our classifer is much better at predicting buys than sells. This makes sense, because greedy algorithms tend not to be able to see much farther than immediate rewards. Decision trees will not be able to forecast rapid changes in the trend, given only immediate drawdowns.

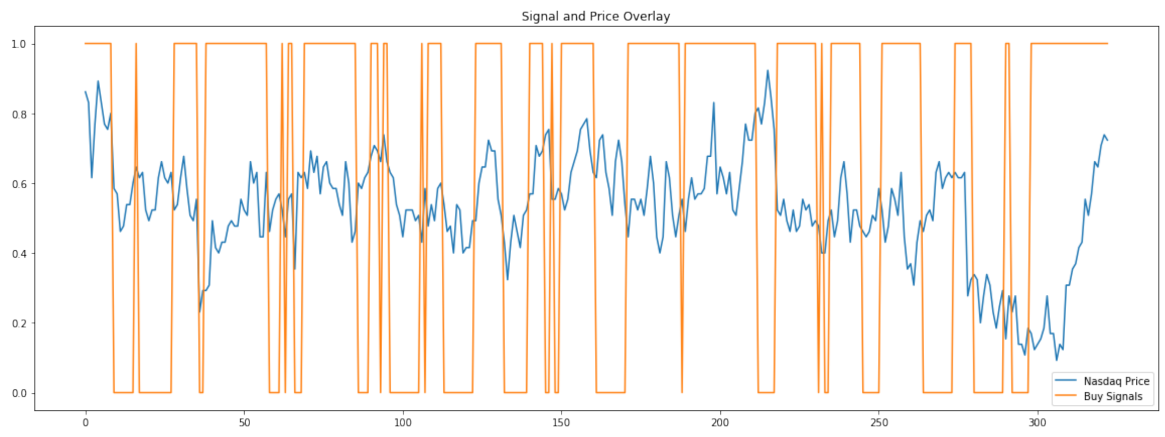


Figure 7.1: Stock price (blue) overlaid with trade signals (orange). We want the orange lines to remain on top when stocks go up, and on bottom when stocks go down.

Note that, what we want is for the orange line (buy signal) to be on top during upward trend, and form a bottom cup during downtrend, which is generally true.

7.4.3 Welch's Unequal Variances t-test

T-value of 5.97 and single tail of $P - value < 0.05$ means that we can reject the hypothesis that our results are not statistically significant with 95% confidence. Our results are statistically significant.

Chapter 8

Q-learning

8.1 Overview

In this strategy, we follow the Markov Decision Process to define our agent and its environment. Increase or decrease in Sharpe Ratio will be the reward or penalty. Discretization is performed to reduce our problem space. Dyna-Q is implemented using Memory Replay to help the model converge faster.

8.2 Markov Decision Process

According to Chen[6], MDP is the most effective model for implementing reinforcement learning. The MDP environment contains discrete set of States S, and a discrete set of Actions A. At each step t it is allowed to choose an action a_i from a subset of action space A. In our case, the actions is buy, sell, or nothing.

At each discrete time step t, the agent reads the current state s_t and chooses to take action a_t. The simulator responds by providing the agent a reward:

$$r_t = r(s_t, a_t)$$

The succeeding state $s_{t+1} = \delta(s_t, a_t)$ is then reached. The functions r and δ only depend on the current state and action. It is memoryless meaning that one should be able to predict the future of a given process based on its present state just as well as if given the entire history of the process.

In an MDP environment, the agent must be able to visit every possible state-action pair infinitely often. Simple split and combination results in a huge number of states to explore. To reduce the state space, we grouped those variables using K-Means and Gaussian Mixture Model (GMM). Then we quantified the performance of the clustering by calculating each data point's Silhouette Coefficient.

8.3 Discretization and Silhouette Coefficient

Discretization is necessary in Q-learning in order to reduce the state space, create time steps for the algorithm to assess its own performance, and adjust rewards and penalties. We use the mean Silhouette Coefficient[27] of all samples to justify the clustering method. It is composed by the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The score for a single cluster is given by

$$s = \frac{b - a}{\max a,\ b}$$

We compute the average of these scores from all samples. The silhouette coefficient for a data point measures how similar it is to its assigned cluster from -1 (dissimilar) to 1 (similar). The figure below computes the mean silhouette coefficient to K-Means and Gaussian Mixture Model using a different number of clusters. Since we cannot assume distribution parameters, different covariance structures to GMM are also tested.

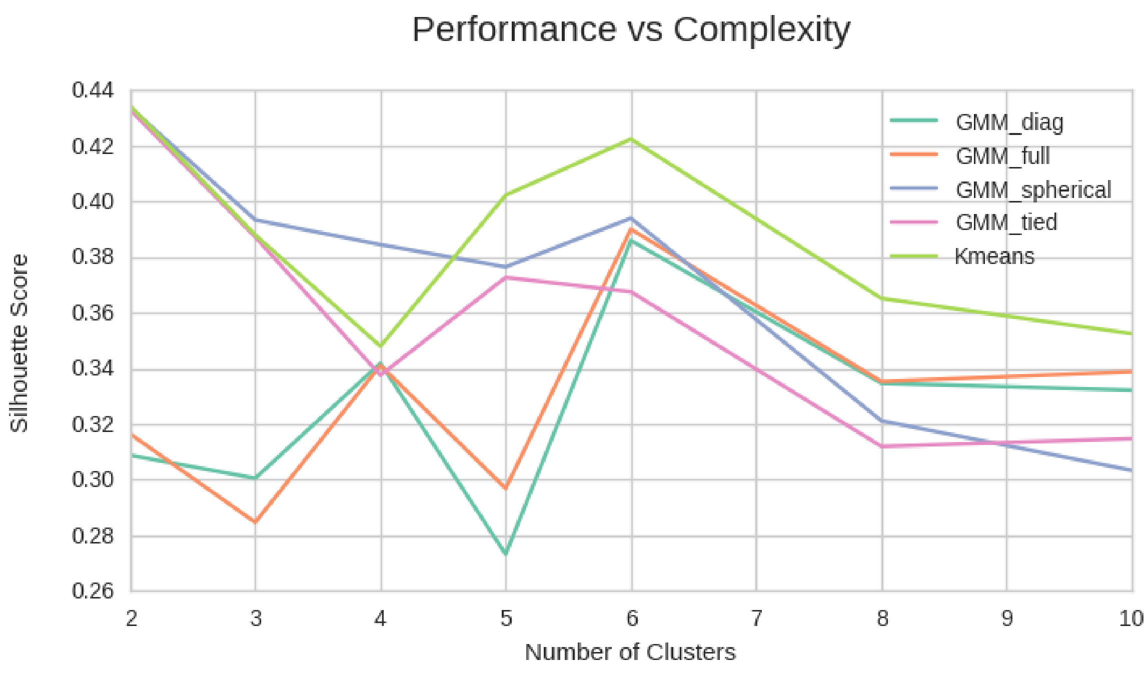


Figure 8.1: Performance vs Complexity of Gaussian Mixture Model and K-Means

It's clear that K-Means outperforms all types of GMM. Additionally, performance peaks at 6 clusters. Thus, we use K-Means to group our variables into 6 buckets after discretization.

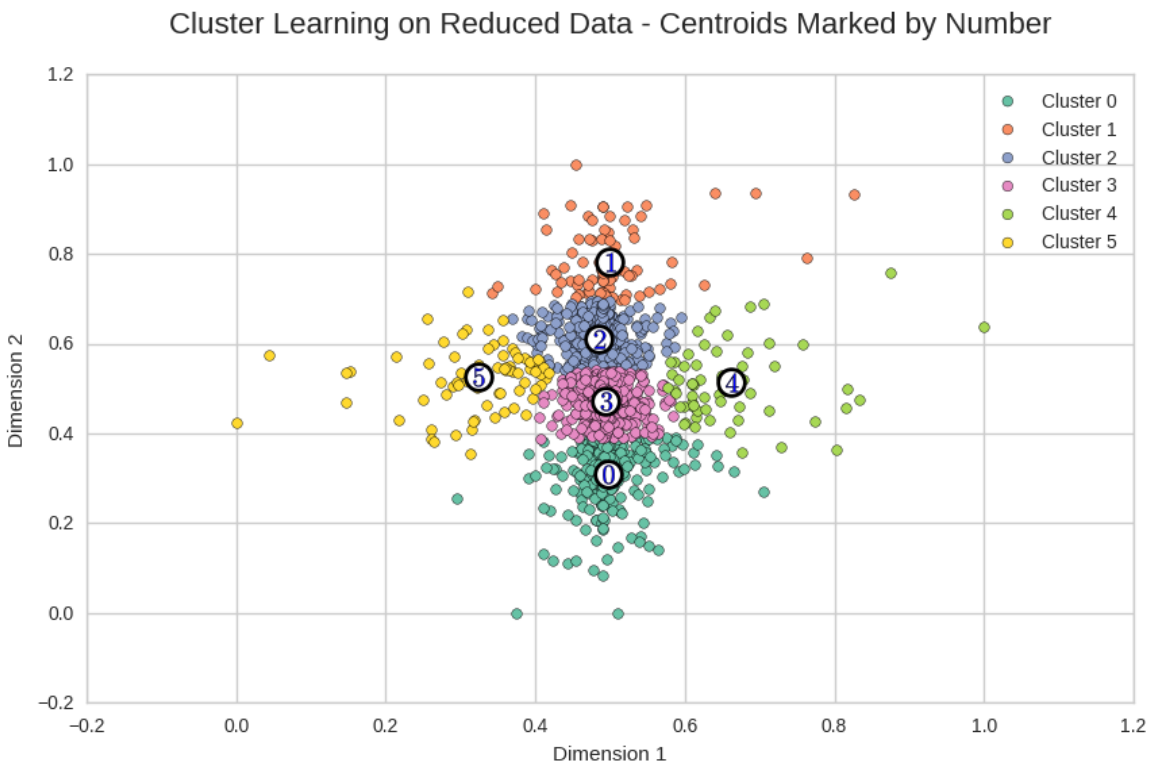


Figure 8.2: K-Means discretizes our data into 6 clusters

8.4 Reward and Penalty Policy

The task of the agent is to learn a policy π that maps each state to an action ($\pi : S \rightarrow A$), selecting its next action a_t based solely on the current observed state s_t, that is $\pi(s_t) = a_t$. The optimal policy produces the greatest possible cumulative reward over time. So, we can state that:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + ... = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

where $V^{\pi}(s_t)$ is represents the cumulative value achieved by following an policy π from an initial state s_t. $\gamma \in [0, 1]$ is a constant that determines the relative value of delayed versus immediate rewards.

If we set $\gamma = 0$, only immediate rewards is considered. As $\gamma \rightarrow 1$, future rewards are given greater emphasis relative to immediate reward. The optimal policy π^* that maximizes $V^{\pi}(s_t)$ for all states s is:

$$\pi^* = argmax_{\pi}\ V^{\pi}(s)\ \ \forall s$$

However, learning $\pi^* : S \rightarrow A$ directly is difficult because the available training data does not provide training examples of the form (s, a). Instead, as Mitchel[19] explains, the only available information is the sequence of immediate rewards $r(s_i, a_i)$ for $i = 1,\ 2,\ 3,\ ...$

So, as we are try to maximize the cumulative rewards $V^*(s_t)$ for all states s. The agent should prefer s_1 over s_2 wherever $V^*(s_1) > V^*(s_2)$. Given that the agent must choose among actions and not states, and it isn't able to perfectly predict the immediate reward and immediate successor for every possible state-action transition, we also must learn V^* indirectly.

Thus, we define a function $Q(s,\ a)$ such that its value is the maximum discounted cumulative reward that can be achieved starting from state s and applying action a. So::

$$Q(s,\ a) = r(s, a) + \gamma V^*(\delta(s, a))$$

As $\delta(s, a)$ is the state resulting from applying action a to state s (the successor) chosen by following the optimal policy, V^* is the cumulative value of the immediate successor state discounted by a factor γ. Thus,

$$\pi^*(s) = argmax Q(s,\ a)$$

This implies that the optimal policy can be obtained even if the agent just uses the current action a and state s and chooses the action that maximizes $Q(s,\ a)$. Also, it is important to notice that the function above implies that the agent can select optimal actions even when it has no knowledge of the functions r and δ.

8.5 Training Policy

According to Mohri [22], the optimal state-action value function Q^* is defined for all $(s,\ a) \in S \times A$ as the expected return for taking the action $a \in A$ at the state $s \in S$, following the optimal policy. It can be written as

$$V^*(s) = argmax\, Q(s,\ a')$$

A recursive definition of Q function:

$$Q(s,\ a) = r(s, a) + \gamma \max Q(\delta(s,\ a),\ a')$$

The recursive nature of the function above implies that our agent doesn't know the actual Q function. It only estimates Q, which we refer to as $\hat{Q}$. It represents the hypothesis $\hat{Q}$ as a large table that attributes each pair $(s\ ,\ a)$ to a value for $\hat{Q}(s,\ a)$.

The agent could over commit to actions that presented positive $\hat{Q}$ values early in the simulation, failing to explore other actions that could present even higher values. Mitchell [19] proposed a probabilistic approach to select actions, assigning higher probabilities to action with high $\hat{Q}$ values, but given to every action at least a nonzero probability. So, we implement the following relation:

$$P(a_i \mid s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

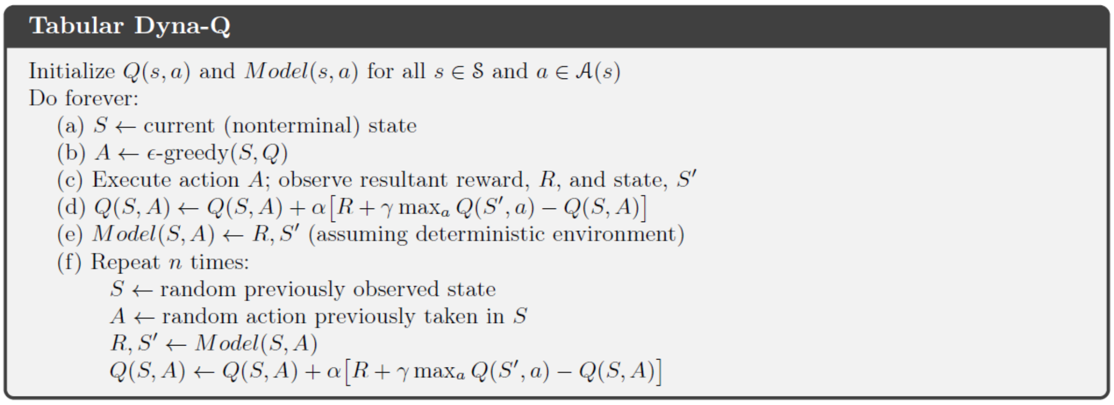
Where $P(a_i \mid s)$ is the probability of selecting the action a_i given the state s. The constant k is positive and determines how strongly the selection favors action with high $\hat{Q}$ values.

8.6 Dyna-Q

Even after discretization the problem space, convergence was extremely slow. Dyna-Q seems to help with this problem. Dynamic Q essentially creates another table that counts the number of times each state is reached after an action in a given state. All values are initialized to 0.00001 to avoid division by 0 errors. Dyna-Q then updates a state's expected reward as weighted sum of reward in each s' weighted by their relative probability.

$$\frac{T_c[S,a,s']}{\sum_i T_c[s,a,i]}$$

where the numerator is the number of times s, a leads to each s'. The denominator is the sum of $T_c[s,a,:]$ (normalizes each s' to its probability)



```
Tabular Dyna-Q
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
    (a) $S \leftarrow$ current (nonterminal) state
    (b) $A \leftarrow \epsilon$-greedy$(S,Q)$
    (c) Execute action $A$; observe resultant reward, $R$, and state, $S'$
    (d) $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
    (f) Repeat $n$ times:
            $S \leftarrow$ random previously observed state
            $A \leftarrow$ random action previously taken in $S$
            $R, S' \leftarrow Model(S,A)$
            $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
```

Figure 8.3: Tabular Dyna-Q Pseudocode as outlined from Sutton's book[32]

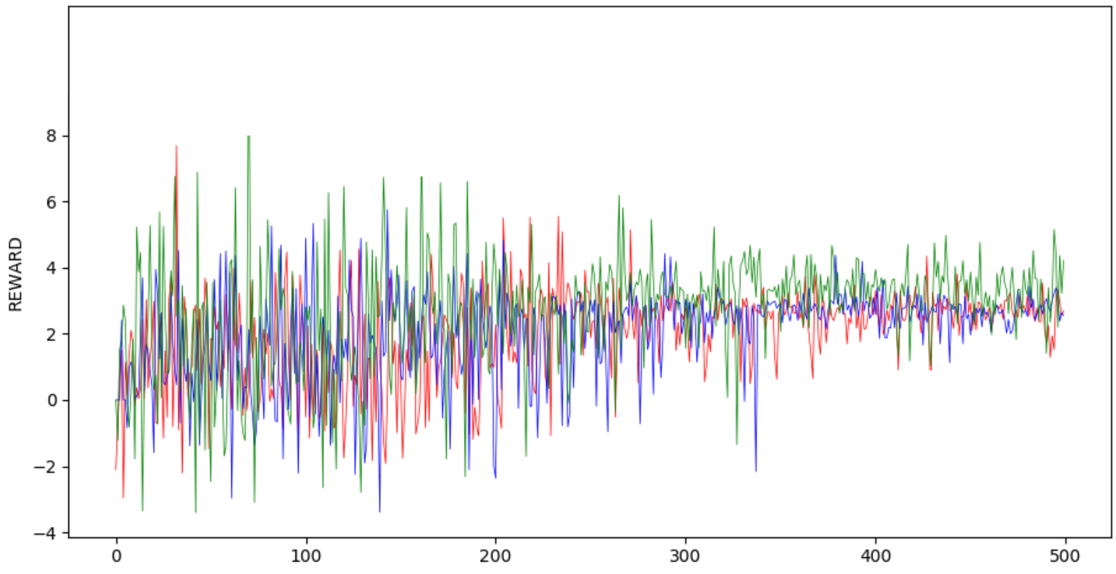


Figure 8.4: Reward vs Steps. Figure shows convergence progress of 3 different epochs (blue, green, red) after 500 steps. Note that the agent struggles with finding positive rewards early, but eventually converges to what it perceives to be an optimum around 3

8.7 Result Discussion and Validation

8.7.1 Results

γ is the discount factor. A higher value towards 1 means we care more about long-term results. A value near 0, makes it a greedy.

Discount Rate	Signals	MCC	SR
Q 0.9	890	0.5125	0.5663
Q 0.8	911	0.5032	0.4853
Q 0.7	1022	0.4836	0.4999
Q 0.6	921	0.5376	0.5223
Q 0.5	953	0.5131	0.5673
Q 0.4	1232	0.4555	0.5562
Q 0.3	1246	0.4639	0.5583
Q 0.2	1356	0.4732	0.3334
Q 0.1	1321	0.4356	0.2458
Q 0.6 Ensemble	221	0.4923	0.5484
BM	N/A	N/A	0.5591

Table 8.1: Q-Learning Results for various discount rates and ensemble

Notice that there is a slight increase in signals as gamma decreased. This means that search deep into the past has a slight effect in smoothing out noise. However, results were very inconsistent due to the probabilistic exploration. It appears that even our ensemble could not help Q beat the benchmark, despite almost cutting its signals down by 5 fold.

8.7.2 Confusion Matrix

	Actual Buy	Actual Sell
Predicted Buy	32	63
Predicted Sell	51	75

Table 8.2: Confusion Matrix for Q-learning Ensemble with with $\gamma 0.6$

Accuracy	Precision	Recall	F1
0.48	0.33	0.38	0.36

Table 8.3: Q γ 0.6 Matrix Statistics

8.7.3 Welch's Unequal Variances t-test

t statistic of -1.93 and p-value > 0.05 suggests that we can not reject the null hypothesis with 95% confidence that the Q-learner does not beat the random agent in risk-adjusted return.. This is the only instance in which a learner underperformed the random agent.

8.7.4 Possible Reasons for Failure

Markov chains are designed to work on a discrete set of possible states. While price history can be discretized by segmenting the price range in a predetermined fashion, this might have greatly reduces the potential precision of the model.

The memoryless-ness of the MDP means that they work well if the only state influencing future states is the current one. This may not be the case with stocks, as there may be many longer term dependencies and patterns in the market fluctuations that Markov chains could simply not pick up on due to lack of memory.[13]

Chapter 9

Long Short Term Memory

9.1 Overview

While in theory RNNs can account for relationships over arbitrarily long sequences, in practise the influence of predictions more than a few steps in the past becomes too weak to have any meaningful impact on subsequent predictions. Due to the distorting nature of commonly used activation functions such as the sigmoid (which condenses its inputs to a 0-1 range), a gradient can gradually diminish over time as it is condensed by consecutive layers. Similarly, a gradient can "explode" if constantly magnified by the recurrent layer. This gradient behavior makes it extremely difficult for an unmodified RNN to distinguish long term dependencies. To address this inadequacy, Long Short Term Memory networks (LSTM) was developed.

9.2 Normalization and Exponential Smoothing

To predict the direction of price movements. We can use SciPy's MinMaxScaler which sales our data from 0 to 1.

$$z_i = \frac{x_i - \min X}{\max X - \min X}$$

where z is the transformed variable. x_i is the variable to be transformed. X = a vector with all x that is transformed.

Exponential smoothing is performed to reduce noise.

$$X_{t+1} = \frac{1}{N} \sum_{i=t-N}^{t} X_i$$

$x_{t+1} = EMA_t = \gamma \times EMA_{t-1} + (1-\gamma) x_t$ where $EMA_0 = 0$ and EMA is the exponential moving average

9.3 Implementation

9.3.1 TensorFlow

Our architecture contains three layers of LSTMs and a linear regression layer (denoted by w and b), that takes the output of the last LSTM cell and output the prediction for the next time step. We used 'MultiRNNCell' in TensorFlow to encapsulate the three 'LSTMCell' objects created.

We then create TensorFlow variables c and h that holds the cell and the hidden state of the LSTM. Then we transform the list of 'train inputs' to the shape of NumUnrollings, BatchSize, D for calculating the outputs with TensorFlow's RNN function. We then compute the LSTM outputs and split the output back to a list of NumUnrolling tensor objects.

Instead of outputting x_t always x_{t+1}, we randomly sample an output from the set $x_{t+1}, x_{t+2}, \ldots, x_{t+N}$ where N is a small window size.

After defining a placeholder for feeding in sample input, we define state variables for prediction (c and h). Compute the prediction with the TF.RNN function and then send the output through the regression layer (w and b). Define the sample state function, that resets the cell state and the hidden state of the LSTM. This is executed at the start, every time before predictions.

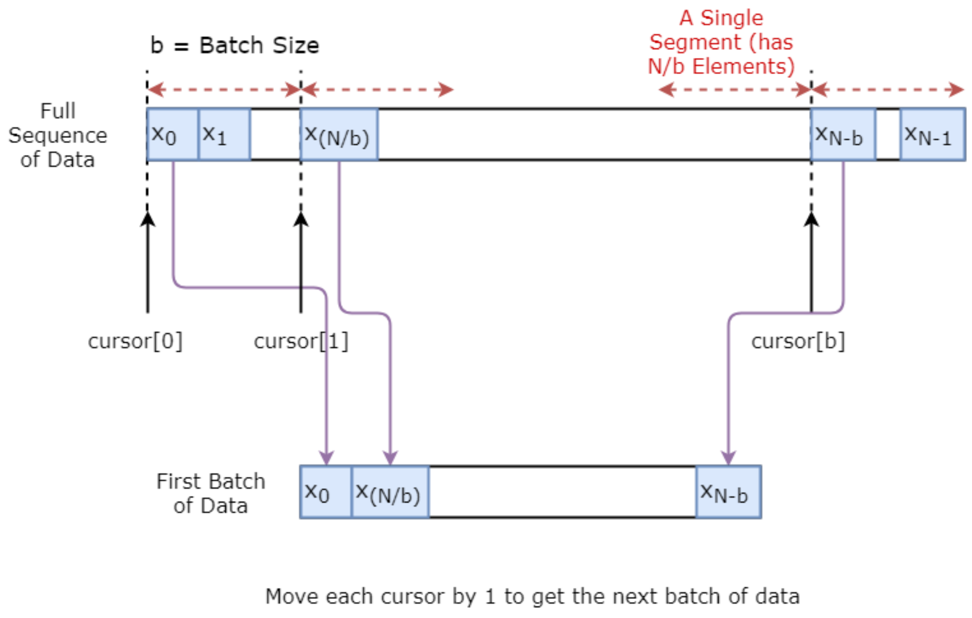


Figure 9.1: The movement and processing of batches in LSTM[9]

Algorithm 3 LSTM Training

```
1:  procedure TRAIN(Data)
2:      Define a test set of starting points
3:      for each epoch do
4:          for full length of training data do
5:              Unroll a set of NumUnrollings batches
6:              Train with the unrolled batches
7:          end for
8:          Calculate MCC
9:          for each point in test set do
10:             Update the LSTM state by iterating through the previous NumUnrollings data points found before the test point
11:             Make predictions for n steps continuously, using the previous prediction as the current input
12:             Return MCC the Sharpe Ratio
13:         end for
14:     end for
15: end procedure
```

9.3.2 Training Algorithm

9.3.3 Optimizing Hyperparameters

Previously, in Random Forest and Q-Learning there were few hyperparameters to optimize. With LSTM however, there are several. Thus, manual trial-and-error strategy becomes infeasible. The following hyperparameters were optimized through Random Search. Random search achieves 95% probability of finding the maximum over 60 epochs[3].

- D: The dimensionality of input or 1 since we matrix-ed our data.
- NumUnrolling: Instead of optimizing the LSTM by looking at a single time step, we can optimize the network by backpropagating through time (BPTT) NumUnrolling number of steps. Higher number means we look farther into the past. Practically, the number of arrays sent from input. Optimized to 50.
- BatchSize: How many data samples per single time steps. Practically, the number of values per array sent from the input. Optimized to 500.
- Dropout: Optimized to 0.1 to reduce overfitting and improve out-of-sample training result
- NumLayer: Searched between 3 to 5. More layers did not improve Sharpe Ratio, but were much slower to train.

- NumNodes: Represents the number of hidden neurons in each LSTM cell. Optimized to 200,200,150 for each layer respectively.
- α: Learning Rate optimized by Adam
- α $Decay$: Learning Rate Decay optimized by Adam

9.4 Result Discussion and Validation

9.4.1 Results

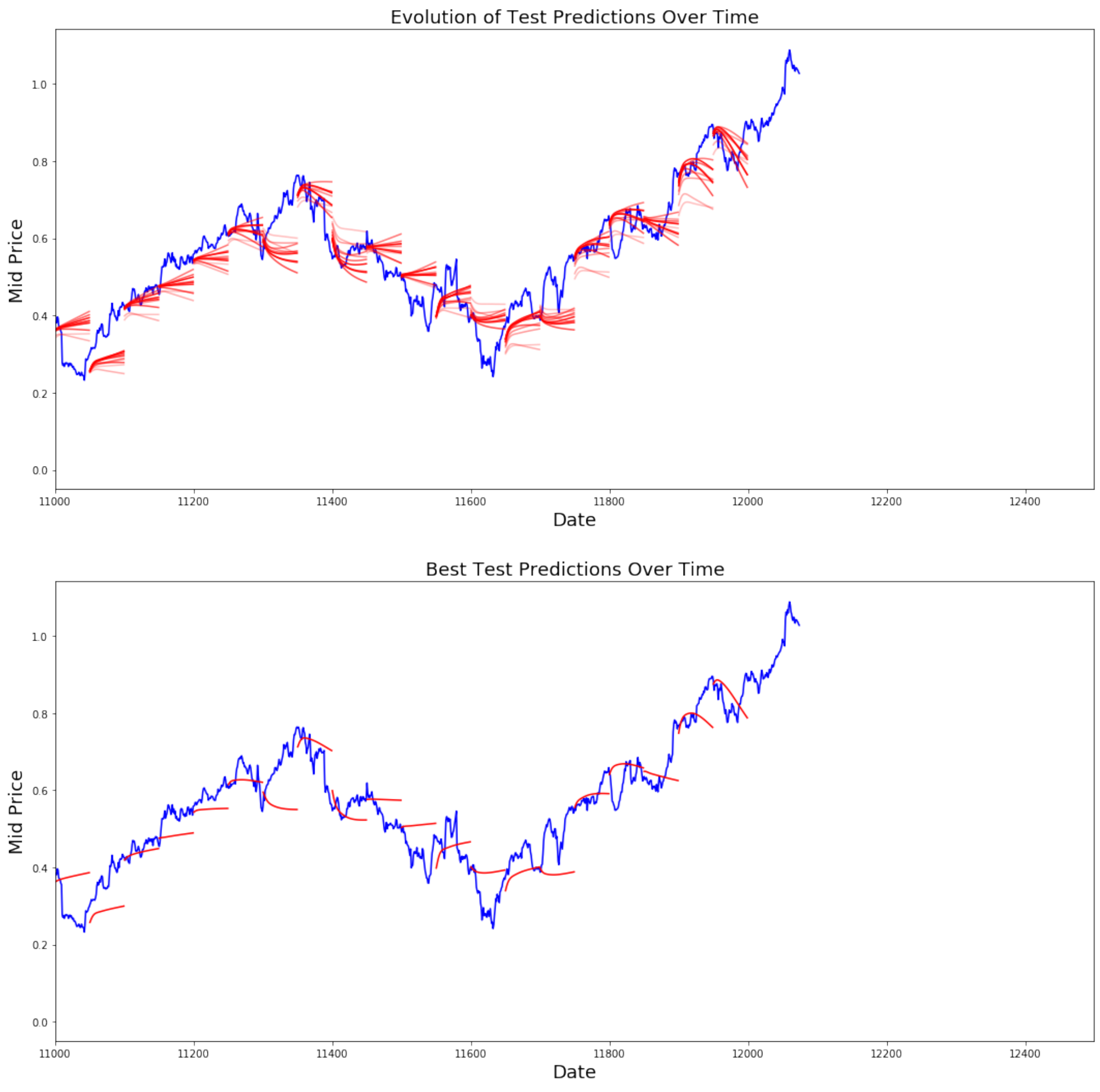


Figure 9.2: Top: All Epochs; Bottom: Best Epoch

Stronger conformity between red (prediction) and blue (actual price) indicates stronger prediction strength. Our model is 64% accurate in terms of MCC classification, out of sample and a 20%. LSTM Ensemble with technical analysis improves performance of the raw algorithm at the expense of reduced signal.

LSTM Type	Signals	MCC	SR
30 Epoch Average	724	0.6274	0.6163
30 Epoch Ensemble	122	0.6423	0.6779
Benchmark NQ	N/A	N/A	0.5591

Table 9.1: Best LSTM Results after 30 Epochs

9.4.2 Confusion Matrix

	Actual Buy	Actual Sell
Predicted Buy	41	28
Predicted Sell	15	38

Table 9.2: Confusion Matrix for LSTM 30 Ensemble

It can be witnessed, that like Random Forest, a high precision (classifying buys correctly) is more important then sells, because it is very difficult to predict sudden changes in the underlying stock price.

Accuracy	Precision	Recall	F1
0.65	0.59	0.73	0.66

Table 9.3: Q γ 0.6 Matrix Statistics

9.4.3 Welch's Unequal Variances t-test

A t-stat of ≈ 6.71 and p-value of < 0.05 suggests that we can reject the null hypothesis with 95% confidence that the Q-learner does not statistically outperform the random agent. Therefore, our results are statistically significant.

Chapter 10

Other Applications – Bitcoin and Ethereum

10.1 Overview

While our model is optimized for Nasdaq Futures. Experimental applications of our models to other assets such as Bitcoin and Ethereum has been made. LSTM works quite well with Ethereum, while Random Forest worked for neither.

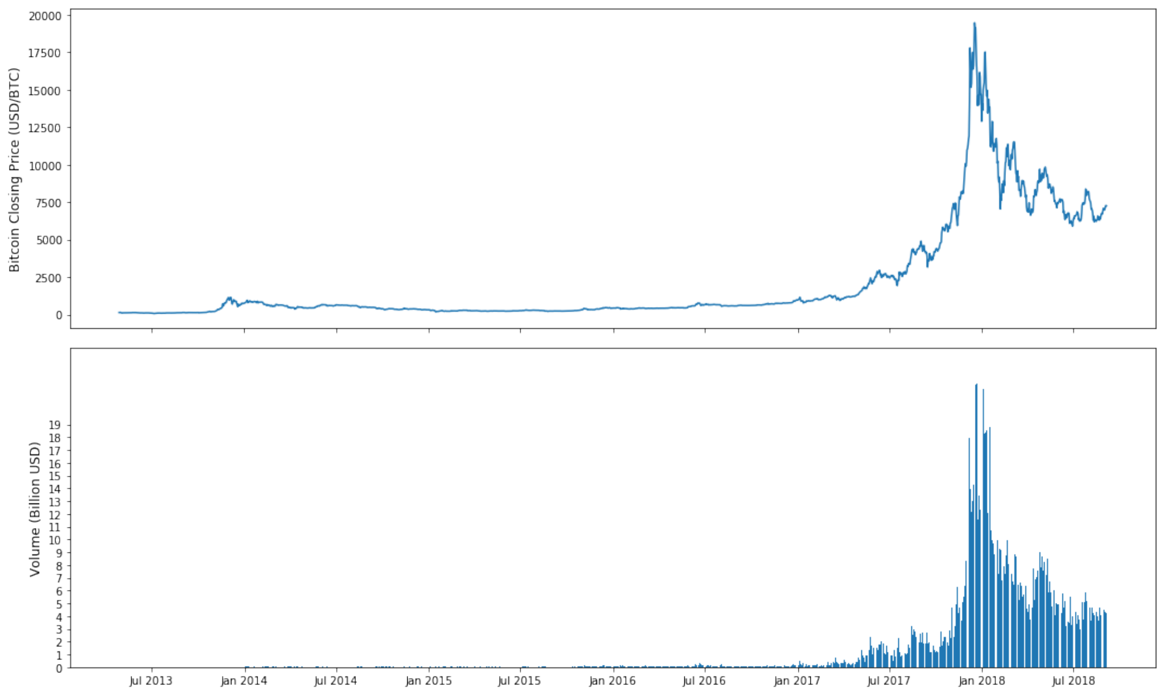


Figure 10.1: Bitcoin Price and Volume since 2013

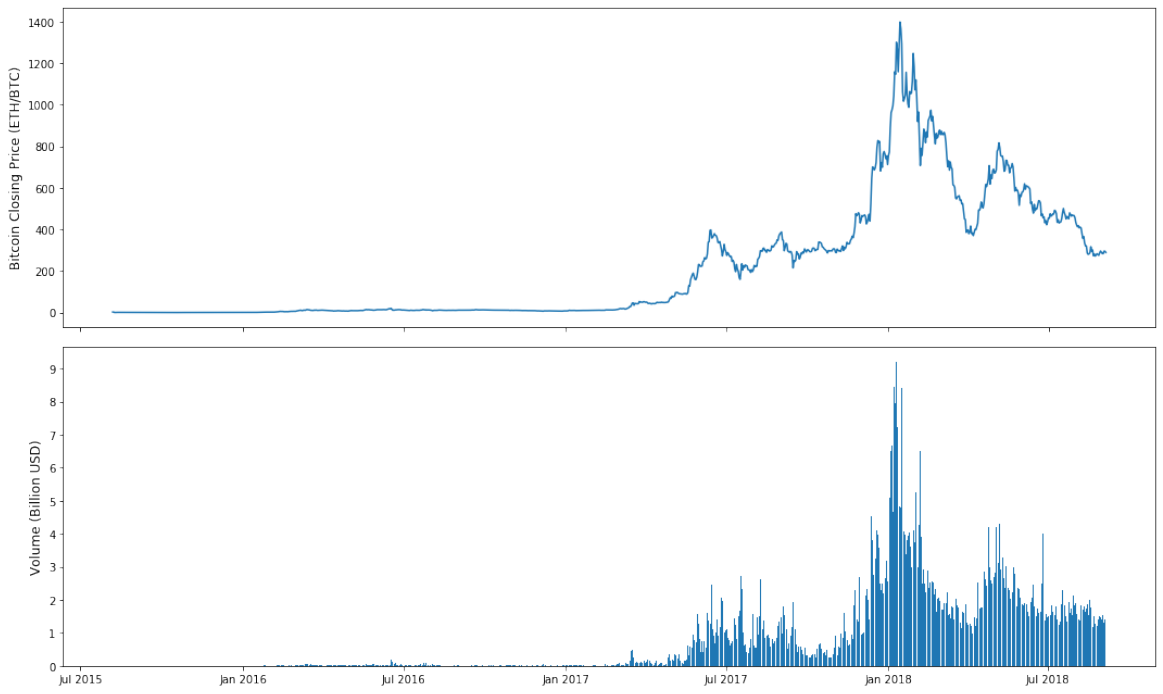


Figure 10.2: Ethereum Price and Volume since 2013

10.2 Result Discussion and Validation

Type	Signals	MCC	SR
RF BTC	953	0.3835	1.1563
Ensemble RF ETH	465	0.4625	1.2573
LSTM BTC	652	0.5234	1.3687
Ensemble LSTM ETH	315	0.5623	1.4234
Benchmark BTC	N/A	N/A	1.6581
Benchmark ETH	N/A	N/A	1.2631

Table 10.1: Vanilla and Ensemble RF and LSTM Results for BTC and ETH

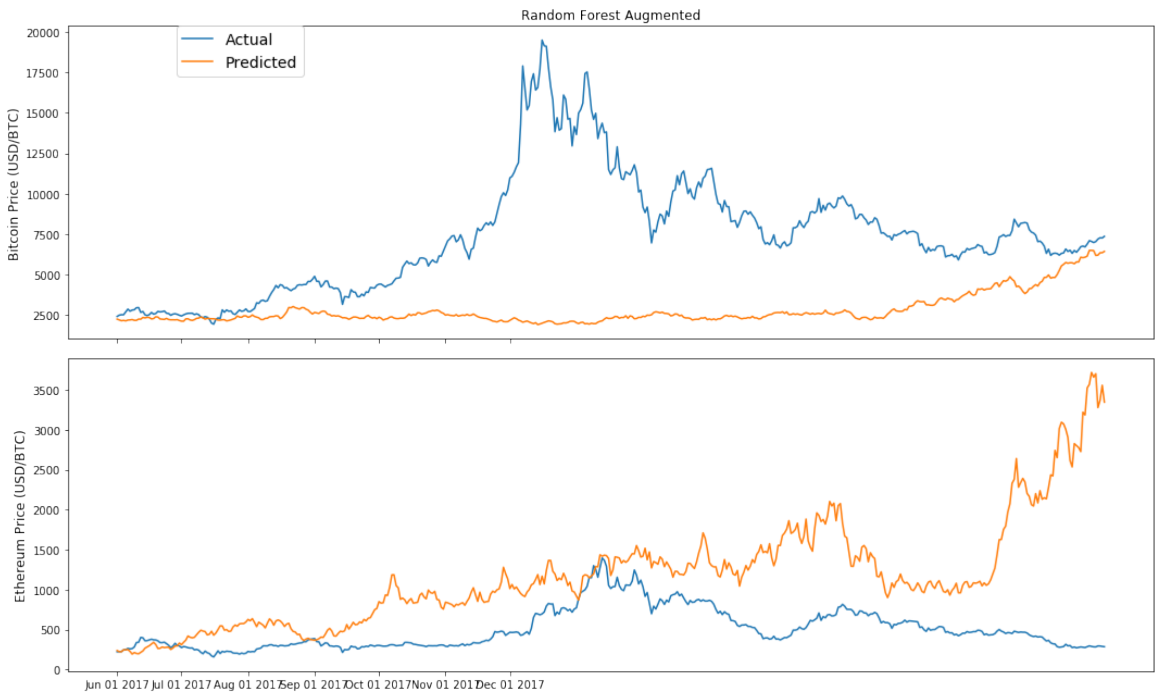


Figure 10.3: Random Forest

As we can see, Ensemble Random Forest performs poorly for both Bitcoin and Ethereum. This is comes as a surprise, since it did so well with the Nasdaq Index. This may indicate that greedy algorithms that depend on next day returns may not do so well. This may also imply a week correlation between next-day returns in the data.

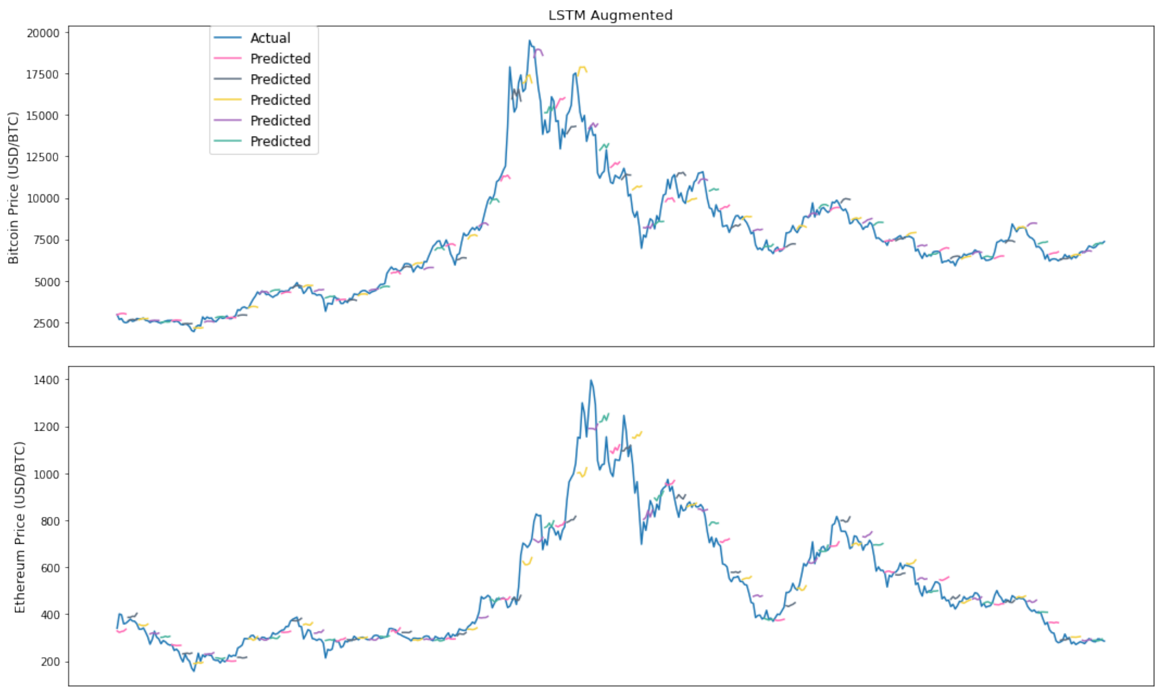


Figure 10.4: LSTM

Random Forest underperformed on both. LSTM fared much better, especially with Ethereum, with significant gains in classification, and Sharpe Ratio. This is a strong indication that this model cannot learn the underlying patterns in Bitcoin. Nonetheless, without further investigation, we cannot conclude that no patterns exist. The Bitcoin underperformance may be a result of the high number of hyperparameters that remain unoptimized in LSTM.

Additionally, the lack of scaling in random forest may have contributed greatly to its underperformance given the huge absolute price swings experienced by Bitcoin. The Ethereum graph appears smoother, thus perhaps more suitable for LSTM.

Chapter 11

Nasdaq Meta Analysis – Trade and Hold

We can now analyze aggregated results from all three ensemble machine learning strategies.

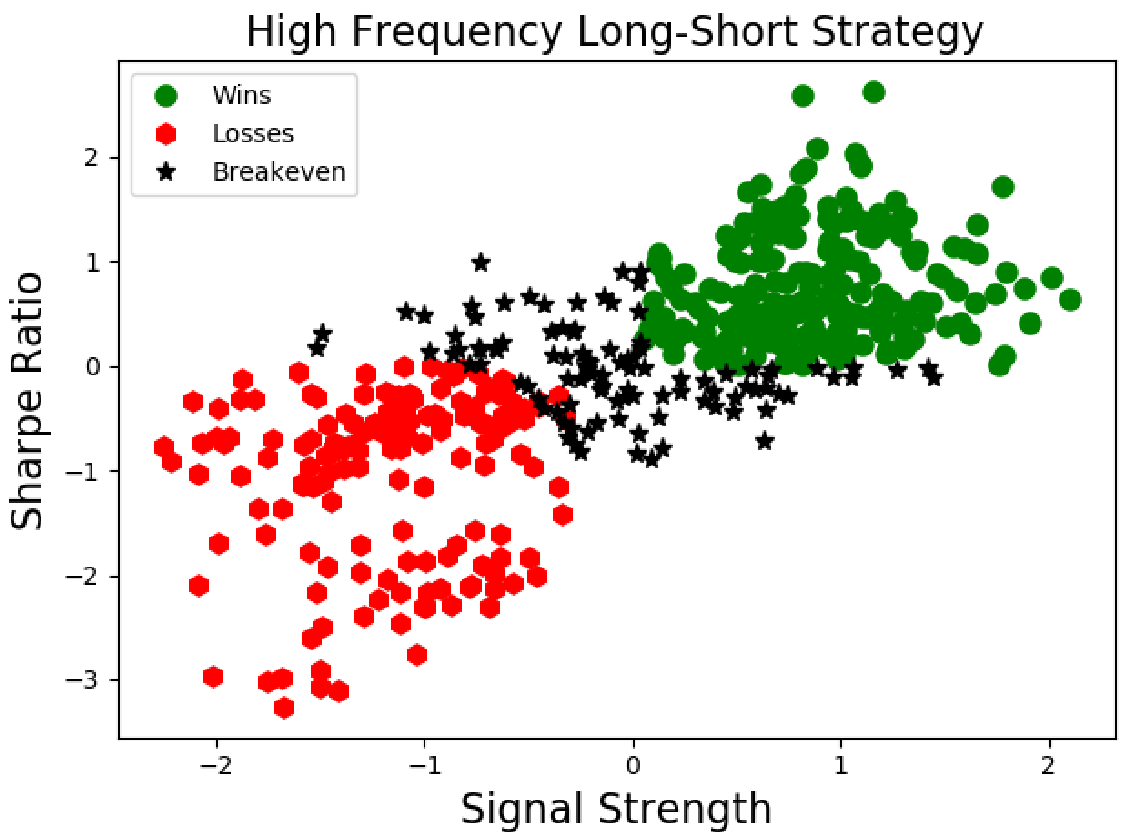


Figure 11.1: High frequency trades tend to use signals with low strength. Strength is the consensus measure among all signals for this trade, similar to entropy.

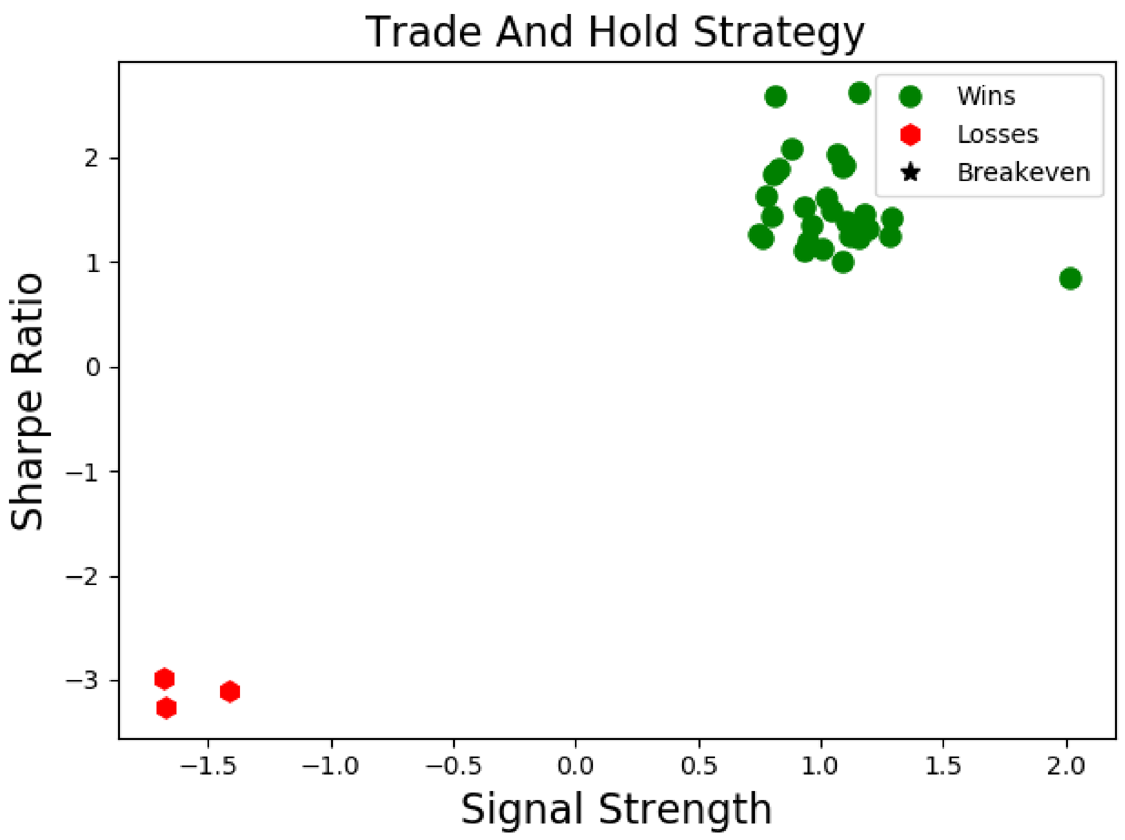


Figure 11.2: Low Frequency trades tend to use signals with high strength. Strength is the consensus measure among all signals for this trade, similar to entropy.

As seen from the two strategies, Trade and Hold in low frequency can produce extremely profitable trades. In contrast, high frequency long-short strategies are actually quite well distributed in terms of wins and losses, with a particular large tail on the left side. This is due to volatility skew, or the fact that financial markets which have huge volatility when selling. This most likely contributed to the enhancement of our Sharpe Ratio in technical indicator ensemble strategies.

Chapter 12

Conclusion

This dissertation has achieved its objective of using technical analysis and machine learning to produce trading strategies that outperforms a buy-and-hold strategy on a risk-adjusted and statistically significant basis. In particular, a LSTM strategy ensembled with 4 technical indicators produced the most consistent results. These results entail that learnable patterns exist in Nasdaq Futures.

The model performed exceptionally well in out-of-sample testing with the Nasdaq Futures and satisfactorily well on Etherum. Ensemble Random Tree performed well on Nasdaq Futures but not cryptocurrencies. Ensemble Q-learning did not perform well in any strategy. Ensemble improved performance of all strategies involved at the expense of reducing trading signal count.

12.1 Summary of Achievements

- Designed and built a market simulator from scratch to take in data, technical indicator signals, machine learning signals, sort the signals, backtest the strategy and produce tangible trading instructions.
- Designed and implemented a novel technical indicator – Max Drawdown Indicator, which allows a strategy to further enhance its performance by not trading on potentially high drawdown days.
- Concluded that Moving Average Divergence Convergence, Relative Strength Index, and On Balance Volume, in descending order are the strongest individual predictors for technical analysis.
- Principle Component Analysis was utilized to find the most important factors among popular technical indicators to enhance learning and test performance, speed and relevance.

- Extensive data preprocessing was done to reduce data noise.
- A Random Forest model was implemented to test if there is next-day
- A Q-learning model was implemented with Discretization to reduce problem space, and Dyna-Q to speed up convergence. This disappointing result may stem from the fact that there may be longer term dependencies in the market data that Markov chains could not pick up due to lack of memory. Discretization alleviates this problem but may have reduced the precision of our model because we have broken up the data set.
- A LSTM model was implemented using grid search to optimize its hyperparameters. Data augmentation was used to produce directional instead of absolute predictions.
- Found that Volume indicators enhances signal accuracy at the expense of signal reduction.
- Discovered why classification rates for trading strategies differ from their Sharpe Ratio through confusion matrix analysis. The biggest factors being blindness to magnitude of moves, and the ability to predict good buying opportunities than selling ones, which tend to move faster and more unpredictably.
- Conducted meta analysis to unveil the importance of signal strength to profitable strategies that derive their success from improbable events.

12.2 Future Work

12.2.1 Signal Weighting Improvement

Currently, signal strength for each indicator is determined by their individual test results. Each indicator signal would be more robust if their individual weights were optimized through machine learning itself. The veto right of the machine learning algorithm is also fairly crude. The weighting is currently done by trial-and-error. The goal of this method, is so that Machine Learning's decision is never overturned by technical indicators, so learning can occur. However, the net effect, is that technical indicators only contribute potentially half of the time. While these parameters worked for this study, they can nevertheless be further improved.

Secondarily, it would be useful to know how much of our model's performance derived from machine learning, and how much was from eliminating noisy signals.

12.2.2 More Data Variety

Given the constraints and scope of this project, only price and volume data were used. In reality, there are many more data points that can be used to improve our index future model. Ideas include:

Economic Data

- GDP
- Unemployment
- Wages
- Manufacturing Index
- Housing Starts

Market-specific Data

- Order Book
- Trading
- Equity Research Recommendations

Sentiment Analysis

- Twitter using VADER
- Web scrapping code repositories like Github and forums like Reddit
- Citigroup Global Surprise Index, and Investor Intelligence Newsletter are sentiment surveys among investors which is considered to be a contrary indicator. This means that when investors are bullish, stocks are likely to go down in the future. We can design a technical indicator whereby if bulls increase above 70%, a sell signal can be issued. A buy signal can be issued when bulls drop below 30%.
- Money Flow Index, Advance-Decline Line, and William R% are great indicator of net inflow or outflow of money into the markets. While long-term money inflow is necessary for the stock markets to go up, stocks tend to perform better during steady inflows. A burst of inflow or outflow tend to be ominous for prices.

Chapter 13

Ethics

13.1 Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues/cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS/TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓

Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?	✓	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓

Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
Section 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?	✓	

13.2 Leverage and Speculation

Although indirectly, the leveraged nature of futures can have a negative impact on people's lives. Speculation on derivative instruments can cause significant appreciation in asset prices. The 2000 Dot-com Bubble, and the 2008 Financial Crisis of 2008 were both partially a result of rampant speculation. The earlier of which was speculated using Nasdaq futures. The latter unethically traded mortgage backed securities which caused major foreclosures. The use of leveraged instruments like futures can result in margin calls, and can have disastrous consequences for the wider economy. I would advise users of this research to restrict their leverage ratio to below 3.

13.3 Privacy

If further work on sentiment analysis is done on this project, there may exist significant privacy concerns. Twitter and web scraping of forums for data, may contain personal information such as user name and location. That may put those users' identity at risk. Although I have no intention of doing so, compromise of such information through a hack, or if usage of this model by others somehow uncover some of these data, ethical issues may arise. Attempts will be made to encrypt the data or replace names with unique IDs.

Bibliography

[1] Energy Information Administration. *Quantile Derivatives and Risk Management in the Petroleum, Natural Gas, and Electricity Industries*. U.S. Department of Energy, Commodity Futures Trading Commission, 2002.

[2] Saurabh Aggarwal and Somya Aggarwal. "Deep Investment in Financial Markets using Deep Learning Models". In: *International Journal of Computer Applications* 165.2 (2017), pp. 0975–8887.

[3] A. Craig MacKinlay Andrew W. Lo. *A Non-Random Walk Down Wall Street*. Princeton University Press, 1999.

[4] L Breiman. "Random Forests". In: *Machine Learning* 45.5 (2001), pp. 5–32.

[5] Interactive Brokers. *Margin Cost*. URL: `https://www.interactivebrokers.co.uk/en/index.php?f=18069` (visited on 08/28/2018).

[6] Nicholas Tung Chan and Christian Shelton. "An electronic market-maker". In: AI-MEMO-2001-005 (2001).

[7] Norman T L Chan. *Excessive leverage – root cause of financial crisis*. Hong Kong Economic Summit 2012, 2011.

[8] CoinMarketCap. *Bitcoin and Bitcoin Cash Data API*. URL: `https://coinmarketcap.com/` (visited on 08/28/2018).

[9] colah. *Understanding LSTM Networks*. URL: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (visited on 08/28/2018).

[10] Istvan Erlich, Ganesh Venayagamoorthy, and Worawat Nakawiro. "A Mean-Variance Optimization algorithm". In: *IEEE Congress on Evolutionary Computation 2010* (Aug. 2010), pp. 1–6.

[11] EUGENE F. FAMA and KENNETH R. FRENCH. "Size and Book-to-Market Factors in Earnings and Returns". In: *The Journal of Finance* L.1 (1995).

[12] CME Group. *Understanding Stock Index Futures*. URL: `https://www.cmegroup.com/education/files/understanding-stock-index-futures.pdf` (visited on 08/28/2018).

[13] Chien-Yi Huang. "Financial Trading as a Game: A Deep Reinforcement Learning Approach". In: *National Chiao Tung University* arXiv:1807.02787v1 (2018).

[14] Leonid Kopman and Scott Liu. "Maximizing the Sharpe Ratio and Information Ratio in the Barra Optimizer". In: *MSCI Barra Research* (2009).

[15] Michael Lewis. *Flash Boys. A Wall Street Revolt*. W. W. Norton & Company, 2014.

[16] TA-Lib. *TA-Lib : Technical Analysis Library*. URL: `https://ta-lib.org/hdr_dw.html` (visited on 08/28/2018).

[17] Sudeepa Roy Dey Luckyson Khaidem Snehanshu Saha. “Predicting the direction of stock market prices using random forest”. In: *Applied Mathematical Finance* (2001), pp. 5–32.

[18] Roberto Merli, Ilaria Massa, and Maria Lucchetti. “HIGH FREQUENCY TRADING: TECHNOLOGY, REGULATION AND ETHICAL ISSUES”. In: *Polish Society of Commodity Science* (2014), pp. 179–192.

[19] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc, 1997.

[20] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). URL: `http://arxiv.org/abs/1312.5602`.

[21] Vivek Mohan, Jai Govind Singh, and Weerakorn Ongsakul. “Sortino Ratio Based Portfolio Optimization Considering EVs and Renewable Energy in Microgrid Power Market”. In: *IEEE Transactions on Sustainable Energy* 8 (2016).

[22] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 026201825X, 9780262018258.

[23] moneychimp. *The Sharpe Ratio*. URL: `http://www.moneychimp.com/articles/risk/sharpe_ratio.htm` (visited on 08/28/2018).

[24] David Montague. *Algorithmic Trading of Futures via Machine Learning*. Stanford, 2015.

[25] Rohan. *Confusion matrix*. URL: `https://alearningaday.com/2016/09/14/confusion-matrix/` (visited on 08/31/2018).

[26] Mohammed El-Anbari Sabri Boughorbel1 Fethi Jarray. “Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric”. In: *Boughorbel et al* 12.6 (2017).

[27] scikit-learn. *Clustering*. URL: `http://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient` (visited on 08/28/2018).

[28] SeasonAlgo. *Continuous Data and Futures Spread Trading*. URL: `https://www.seasonalgo.com/continuos-data-in-futures-spread-trading` (visited on 08/28/2018).

[29] Jeremy J. Siegel and Jeremy D. Schwartz. “Long-Term Returns on the Original S&P 500 Companies”. In: *Financial Analysts Journal* 62.1 (2006), pp. 81–31.

[30] David M. Smith et al. “Sentiment and the Effectiveness of Technical Analysis: Evidence from the Hedge Fund Industry”. In: *Journal of Financial and Quantitative Analysis (JFQA)* 51.6 (2016), pp. 1991–2013.

[31] Kuldeep Kumar Sukanto Bhattacharya. “A computational exploration of the efficacy of Fibonacci Sequences in Technical analysis and trading”. In: *Social and Behavioral Sciences: Economics* J.4 (2018).

[32] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning 2nd Edition*. MIT Press, 2017. ISBN: 0262193981.

[33] Alpha Vantage. *Nasdaq Data API*. URL: `https://www.alphavantage.co/documentation/` (visited on 08/28/2018).

[34] Koupin Lv Xin Du Jinjian Zhai. “Algorithm Trading using Q-Learning and Recurrent Reinforcement Learning”. In: *Stanford* (2001).

[35] Ömer Kaan Baykan Yakup Kara Melek Acar Boyacioglu. “Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange, Expert Systems with Applications”. In: *International Journal* 38.5 (2011), pp. 5311–5319.