Swing Trading Strategies

Gustavo Campos Hoang Dinh Jared Marks

July 28, 2020

Abstract

In this paper, we research and implement swing trading strategies in order to test the efficacy of such strategies in a systematic trading environment. Portfolio managers and analysts face many challenges in developing systematic trading systems. With advances in technology, financial engineering, and machine learning, financial markets have become more competitive than ever. Many markets are dominated by high-frequency algorithmic trading and large-market participants with billions of dollars in trading capital. There is also an argument that traditional financial fundamental analysis has become less effective due to quantitative easing, negative interest rates, and other macroeconomic factors. Thus, investors are forced to rely on other techniques such as technical analysis and systematic trading methods to help make decisions and manage their portfolios. Quantitative trading systems have their own set of difficulties such as infrastructure and data management, strategy design, optimization and implementation, risk management, etc. The goal of the project is to research cutting-edge technical analysis techniques and implement them in a reproducible, systematic, and optimized way using the latest technology and best practices to achieve profitability. To analyze the proposed swing trading strategies, a universe of liquid large-cap U.S. stocks will be selected. Then, we develop an infrastructure to collect data, perform the research, and back-test; This will allow us to research and compile the selected swing trading strategies to understand the differences therein and optimize each strategy parameters using contemporary methods and best practices. Finally, we deploy the trading strategy in a simulated live market environment using C# and NinjaTrader, that will allow us to analyze the performance, identify areas for improvement, summarize and draw conclusions.

Keywords: Swing trading, technical analysis, strategic analysis, quantitative analysis, optimization, machine learning.

Contents

List of Figures

List of Tables

1 Introduction

Swing trading is the art and science of profiting from securities' short-term price movements for a few days to a couple of weeks but can be longer. These strategies are focused on taking small gains in short-term trends and cutting losses quickly. As a result, these small gains made consistently over time can compound into excellent annual returns.

Swing traders can be institutions such as hedge funds or individuals. Usually, they rarely invest 100% in the market at any time. Rather, they wait for low-risk opportunities and try to make profits from a significant move up or down. When the overall market is riding high, they buy (or go long) more often than they sell (or go short). When the overall market is weak, they do the opposite. And if the market isn't doing all that much, they sit patiently on the sidelines and wait for the right opportunity to appear.

On the other hand, the idea of using technical analysis to profit from the stock market has been popular for a long time. Although this idea is controversial and contradicts the Market Efficiency Hypothesis, evidence shows that technical analysis can produce profits during a certain period or for a specific market. Loh (Loh 2005) uses a simple moving average and time series forecasts to construct a strategy that can profit from five Asian-Pacific stock markets. However, it is much more difficult to apply similarly simple techniques to profit from the U.S Stock Market since these markets are more efficient and mature compared to other developing countries. Therefore, new strategies were constructed based on a combination of technical analysis and statistical models. Pesaran (Pesaran and Timmermann 1995) used an ARIMA model to predict the performance of S&P500.

However, as technical analysis becomes more popular, and the number of technical indicators increases, traders have an increasingly difficult time deciding on which one to use. This is where Artificial Intelligence is useful because the patterns or features can help determine which indicators are more reliable than others. Weckman (Weckman and Lakshminarayanan 2004) used neural networks to select the appropriate technical indicators to help predict the stock market performance of Ford Motor Company during 2002-2003. Bogullu, (Bogullu, Enke, and Dagli 2002) applied similar techniques for a larger sample size (7 companies) where they used neural networks to construct a stock trading signal system based on MACD, William R%, and RSI as the input data. However, since technical indicators are built based on historical price and volume, they seem to lack important information needed to predict the market (Zhong and Enke 2019).

2 Theoretical Framework

2.1 The Efficient Market Hypothesis

Research shows mixed results when using technical analysis as the basis for making investing decisions. Supporters of the Efficient Market Hypothesis argue that technical analysis methods are of no use because there can be no useable information in things such as trend lines or other indicators since the market is efficient. However, other investors believe one reason that technical analysis works is that it is a "self-fulfilling prophecy", meaning that since many investors are all making their decisions based on the same charts or patterns, it ultimately leads to the market moving in that direction. This is because since markets are supply and demand based if these technical analysis methods cause groups of people to enter the market or exit the market at the same time, it can influence the price, leading to "inefficient" or unexplainable price movements (like a stock's price seemingly following trend lines on a chart).

2.2 Fundamental Vs. Technical Analysis

Historically, investors have had two very different analysis methods that they can use to help make investment decisions; fundamental and technical. Fundamental analysis is the traditional "Warren Buffet" style of analysis where the investor reviews things such as a company's balance sheet, their competitors, macroeconomics, etc., as the basis for predicting the value of a stock or commodity. Technical analysis is a relatively newer, and hotly debated, style of analysis where the analyst uses indicators involving price, volume, trend lines, and other methods in order to estimate the value or direction of a security. Many systematic swing trading strategies are based on the automation of trading based on predefined technical analysis signals.

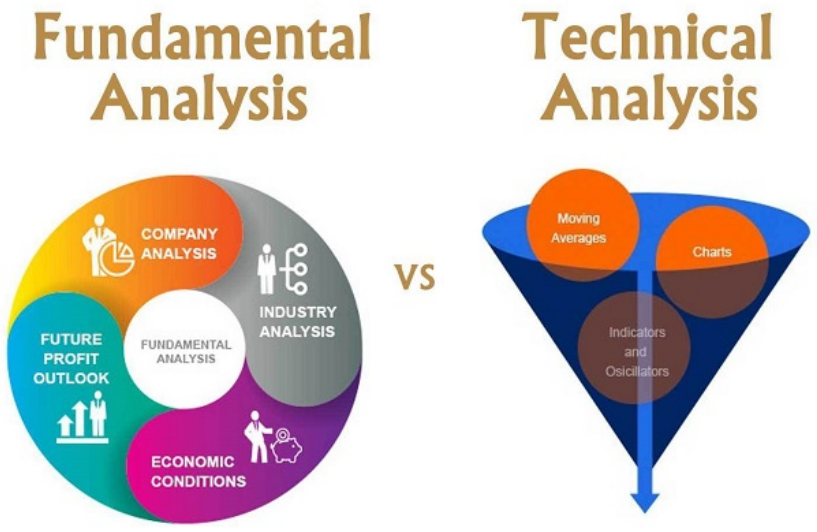


Figure 1: Fundamental Vs. Technical Analysis (Andy 2019)

2.3 Machine Learning in Swing Trading

With advances in technology such as compute capacity, internet speed, etc., some investors have started to improve their technical analysis methods and trade signal generation with machine learning techniques. During our literature review, we noticed mixed results with these methods as well since single indicators like MACD and RSI are not very predictive on their own. However, more advanced models or ensemble methods may be more predictive, however, they also might be more prone to overfitting and require more expertise and care on the part of the analyst.

2.4 Challenges of Systematic Trading

Systematic trading can be just as difficult as trading in real life. This is due to the knowledge, expertise, and technology needed to successfully build and deploy successful trading strategies. Also, the challenge of data-mining bias and data snooping is very problematic because of the ease of back-testing and optimization for a given strategy. Trades have to take extra precautions to avoid introducing bias and selecting the best back-test or changing their parameters to make the back-tests turn out better. Similarly, having too many rules can also make a back-test look promising, only for investors to find out later that they have overfitted their strategy and it does not work out of sample in the real world. It is better to limit the degrees of freedom used in parameter optimization to reduce the chances of overfitting. Additionally, the analyst can utilize strategies like Monte Carlo, resampled trades, cross-validation, and other techniques to check for the robustness of the strategy they have developed.

2.5 Opportunities

Most research on swing trading has been conducted on large, liquid securities like stocks, commodities, and global indices. One opportunity is to research and implement swing trading strategies in newer markets like cryptocurrencies or less liquid markets like micro-cap stocks. In theory, these markets may be less efficient than large, mature markets, which provides an opportunity to profit from this. Additionally, swing trading utilizing machine learning techniques appears to still be a relatively new area of study. There is an opportunity to improve results by utilizing these methods.

3 Methodology

3.1 Research Study

In this project, we will focus on two main research topics: swing trading and applied machine learning classification problems. Our hypothesis is that a combination of technical analysis and statistical methods could potentially produce a profitable trading strategy. However, due to the changing dynamics of the stock market, it would be very difficult to have a strategy that performs well under all market conditions. As a result, in addition to researching different trading strategies, we will also apply machine learning to detect market sentiment and choose the appropriate set of parameters.

3.2 Swing Trading Techniques

The first step towards deploying a profitable trading strategy is to develop the signal generating criteria. There are a plethora of strategies that one could develop and use to generate trade signals, however, we will focus on implementing and optimizing the hyperparameters on one or more of the following strategies.

- Dip Trip
- Bear Hug
- Power Spike
- Coiled Spring
- Finder Finder
- Hole-in-the-Wall

These strategies provide a diverse set of frameworks for us to research and each strategy is unique in how it attempts to profit from the market i.e. (momentum, buying dips, breakouts, etc.)

3.3 Frameworks and Libraries

We chose Python as our programming language of choice for our analytic, back-testing, and optimization needs. Python is an excellent choice given its large open-source libraries and data science tools. Specifically, we will be leveraging the backtrader library to run and analyze our strategy back-tests. Backtrader is a feature-rich open-source library for back-testing and trading. Their goal is to provide an intuitive API for traders so they can focus on writing strategies instead of infrastructure. One of the critical components to this framework is the idea of the "indicator" which can be a traditional technical indicator or a custom indicator as we will explain in the following sections. We have chosen to develop our own indicators used in this framework to generate trade signals. Our data needs will be met by utilizing the free AlphaVantage™API and the python wrapper library alpha_vantage to source historical U.S. equity data for our back-tests.

3.4 Machine Learning

The objective of this part is to build a system that can predict the future market environment of the S&P 500 Index. Since this is a classification problem, there are four categories that we need to label: Bull High Variance, Bull Low Variance, Bear High Variance and Bear Low Variance. The dataset includes different fundamental data and the performance of 10 sectors indicators from 1999 to 2019. We utilize a Recurrent Neural Network (RNN) construct and train the Machine Learning algorithm. The model delivers acceptable accuracy on both training and validation data. The average categorical accuracy on the validation dataset is 80%. This model will be implemented in our trading framework as a filter to help improve our strategy's performance. We use daily data from 1999 to 2019 to build, train, and evaluate the Machine Learning Model.

3.4.1 Fundamental Data

Our hypothesis is that monetary policy can be used to predict the state of the economy and market conditions. Therefore, we use the following fundamental data: Inflation Rate, T-Bills and Gold price as the inputs to the algorithm. Below is the summary of fundamental data used:

Table 1: Fundamental Data[1]

Symbol	**Description**
`T_bill_3`	3-month T-bill rate, secondary market, business days, discount basis
`T_bill_6`	6-month T-bill rate, secondary market, business days, discount basis
`T_bill_60`	5-year T-bill constant maturity rate, secondary market, business days
`T_bill_120`	10-year T-bill constant maturity rate, secondary market, business days
`XAU`	Gold Spot Price (in USD)
`CTB3M`	Change in the market yield on US Treasury securities at 3-month constant maturity, quoted on investment basis
`CTB6M`	Change in the market yield on US Treasury securities at 6-month constant maturity, quoted on investment basis
`CTB5Y`	Change in the market yield on US Treasury securities at 5-year constant , quoted on investment basis
`CTB10Y`	Change in the market yield on US Treasury securities at 10-year constant maturity, quoted on investment basis

3.4.2 Stock Return Data

According to a white paper published by Fidelity (Lisa Emsbo-Mattingly 2020), sector performance can also help predict the business cycle as well as economic conditions. Therefore, we use the daily return of 10 sectors as potential attributes. Below is a summary of the stock data used.

Table 2: Stock Return Data[2]

Index Symbol	**Description**
`S5INFT`	Information Technology Sector Daily Return
`S5HLTH`	Health Care Sector Daily Return
`S5FINL`	Financial Sector Daily Return
`S5TELS`	Telecommunication Service Sector Daily Return
`S5COND`	Consumer Discretionary Sector Daily Return
`S5INDU`	Industrial Sector Daily Return
`S5CONS`	Consumer Staples Sector Daily Return
`S5ENRS`	Energy Sector Daily Return
`S5UTIL`	Utilities Sector Daily Return

3.4.3 Labels

To label the data, we need to categorize the market conditions into different groups. There are two popular market conditions: bull market and bear market based on the stock return. However, using only these two labels seem to miss important information about the market volatility i.e. variance. Therefore, we use a combination of market return and variance to create four categories: Bull/ High Variance, Bull/ Low Variance, Bear/ High Variance, Bear/Low Variance. To do this, we chose to use 63 days (3-months trading days) as our evaluation range. We look at the future data and calculate the mean and variance of the return during this evaluation window. Below are the label's logic and formulas that we used, where $n = 63$, and $m = 4901$ is the number of days in the dataset.

[1]Source: Bloomberg Terminal
[2]Source: Bloomberg Terminal

Table 3: Data Labels

Label	**Definition**	**Condition Formula**
Bull	Average Return of Evaluation $Period > 0$	$\frac{1}{t+n}\sum_{t=1}^{t+n} r_t > 0$
Bear	Average Return of Evaluation $Period < 0$	$\frac{1}{t+n}\sum_{i=t}^{t+n} r_t < 0$
High Variance	Average Standard Deviation of $EvaluationPeriod <$ $AverageStandardDeviationOfAllPeriod$	$\frac{1}{t+n}\sum_{t=1}^{t+n} \sigma_t > \frac{1}{m}\sum_{i=1}^{m} \sigma_i$
Low Variance	Average Standard Deviation of $EvaluationPeriod >$ $AverageStandardDeviationOfAllPeriod$	$\frac{1}{t+n}\sum_{t=1}^{t+n} \sigma_t < \frac{1}{m}\sum_{i=1}^{m} \sigma_i$
Bull/High Var	Bull Market, High Variance	Bull & High Variance
Bull/Low Var	Bull Market, High Variance	Bull & Low Variance
Bear/High Var	Bull Market, High Variance	Bear & High Variance
Bear/Low Var	Bull Market, High Variance	Bear & High Variance

After we apply the above logic, we can create a single Label column where each row is labeled based on the future conditions of the next 63 trading days. Here is the summary of the frequency for each category:

Table 4: Market Conditions Frequency

Market Condition	**Frequency**
Bull/LowVar	2,689
Bull/HighVar	513
Bear/LowVar	810
Bear/HighVar	889

3.4.4 Create attributes

Next, we need to create attributes from the raw data. We decide to not apply any transformation to the fundamental data. However, we will transform the sector returns to create two attributes: Mean of Correlation and Standard Deviation of Correlation. To do this, we use 21 days (1 month trading days) as our evaluation period. For a single row, we calculate the mean and standard deviation of the last 21 data points. The descriptions and formula to create these attributes are summarized below, where $n = 21$.

Table 5: Attributes

Label	**Definition**	**Formula**
Correlation Sets	Sets of Correlation between two sector x and y	$r = \frac{\sum_{t=1}^{T}(x_t-\overline{x})(y_t-\overline{y})}{\sqrt{\sum_{t=1}^{T}(x_t-\overline{x})^2(y_t-\overline{y})^2}}$
Mean_Correlation	Mean of correlation sets	$\frac{1}{n}\sum_{t-n}^{t} r_t$, where t is timestamp starts at $t = 21$
StdDev_Correlation	Standard Deviation of correlation sets	$\sqrt{\frac{\sum_{t-n}^{t}(r_t-\bar{r})}{n}}$ where t is timestamp starts at $t = 21$

3.4.5 Construct RNN Model & Train Model

One of the most challenging problems when dealing with time-series data is that the order of data matters. As a result, we cannot randomly shuffle the data as can be done in other applications which can lead to a biased model. For example, given a dataset from 2000 to 2010, if we use data from 2000 to 2008 to train and from 2009 to 2010 to evaluate, this could be problematic because the market conditions during these two periods are different. To approach this problem, if we had enough data we could split the data into smaller batches. However, this approach could shrink the data very quickly. For example, a dataset of 210 days will shrink to 10 data points if we use 21 days periods. To avoid this problem, we construct the data in 3 dimensions instead of 2 dimensions so that each data point contains the historical data of the latest 63 days (3-month trading days). Therefore, now we can shuffle the data randomly which gives us abundance data to train and evaluate.

3.4.6 Evaluate and Update Parameters

Once we train the model, we can evaluate the performance using the validation loss and categorical accuracy. From these performance data, we can update the parameters accordingly to enhance the performance of the model. We use the `callback` functions in Keras to save the best model that delivers the lowest validation loss. Then, once we train the model and update the optimal parameters, we apply the model to the validation data. The model shows no sign of overfitting. Below is the chart of the training and validation loss by epoch:

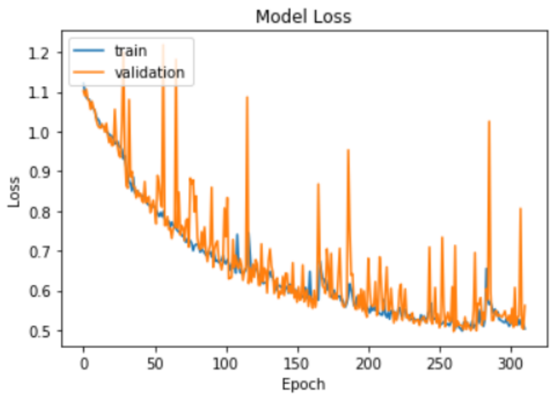


Figure 2: Model Loss

The highest categorical accuracy on the validation set is 84%. Below is the chart of the training and validation categorical accuracy by epoch:

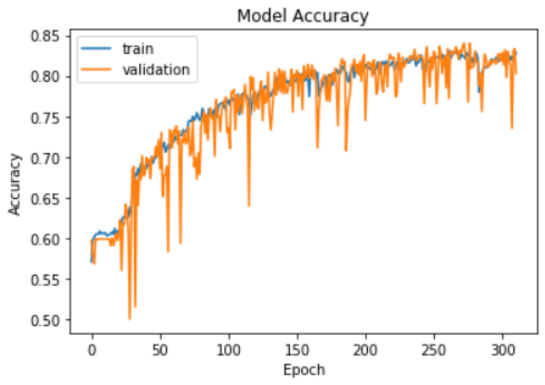


Figure 3: Model Accuracy

3.5 NinjaTrader Framework

NinjaTrader is a trading platform designed to have great performance and flexibility using modern design architecture. The latest version contains advanced trading features for discretionary and automated traders of all levels and supports trading stocks, futures, forex, and cryptocurrencies. It offers a standards-based (.NET) programming environment for indicators and strategies that allow developers to build incredibly rich and integrated applications.

As an open framework for developers, NinjaTrader provides a C# framework for developers to build integrated indicators, drawing tools, automated strategies, and more.

This platform also provides features such as advanced order types, automated trading, and back-testing. One premium feature is Order Flow which allows you to analyze trade activity using order flow, volume bars, and market depth (Folger 2020).

- Pros
  - Excellent charting, great technical analysis tools, as well as partial and full strategy automation.
  - The NinjaTrader Ecosystem offers thousands of apps and add-ons from third-party developers.
  - Learn the platform and practice trading before risking real money.
  - Platform guides, video library, and free daily webinars.
- Cons
  - Basic platform features are free with a funded account, but you'll need to pay to access premium features.
  - Easy setup for futures and forex traders, but you'll have to use a supporting broker to trade equities.
  - NinjaTrader brokerage clients can use the CQG mobile app, but there's no app (yet) if you're using another broker.

In this project, we'll be implementing strategies using NinjaScript (C#) and setting up the parameters to perform back-testing and optimization. With these results we can have a point of reference to compare with the machine learning technique.

3.6 Data Analysis

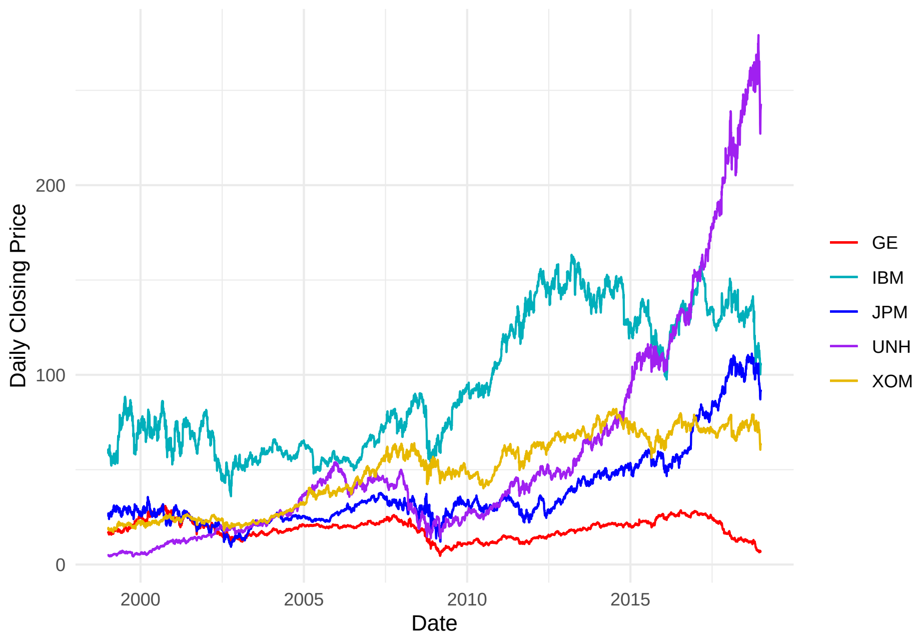
For this project, we will need to collect historical stock prices used in the research and development of our trading strategies. We will also consider live data for validating the system. As we hope our trading strategy is robust enough to be applied to any market, we will use data for two main market categories:

- Index (SP&500, Dow, Nasdaq)
- Large companies by market cap
  - General Electric `GE`
  - International Business Machines Corporatio `IBM`
  - JPMorgan Chase & Co `JPM`
  - UnitedHealth Group Incorporated `UNH`
  - Exxon Mobil Corporation `XOM`

We have selected the following U.S. Large Cap equity stocks to use in our research: `GE, IBM, JPM, UNH`, and `XOM`. We use approximately 20 years of data to analyze, the period used for all of the stocks is from **Jan-01-1999 to Jan-01-2019**.

The historical price data of our set of assets is obtained using an API provided from `quantmod` package. Using the function `getSymbols()` we get the daily Open, High, Low, Close and Adjusted prices and Volume for the desired period.

Below, we generate a time plot of all five price series on the same graph and give a brief discussion of the similarities and differences we observed.



In the above graph, we can divide stock performance into 4 periods:

- From 1989 -2000:
  - The stock price of all 5 companies increased during this period due to the strong economic conditions. IBM yielded the highest growth, while the growth of GE, JPM, and XOM are quite close to each other. UNH stock prices were in side-ways trend during this period.
- From 2000 - 2008:
  - During this period, GE and JPM were in side-ways trend.
  - The stock of XOM was soaring during this period due to the benefit of high oil price conditions
  - UNH stock increased with a high growth rate and relatively low volatility. This might be a sign of latency due to the nature of the healthcare sector.
  - IBM stock went into a down-trend due to the poor performance of the firm during the 2000s.
- From 2008-2009:
  - This is the Global Financial Crisis Period, all of the stocks declined dramatically.
- From 2009 to 2019:
  - Most stock prices sharply rose following the crisis.
  - Shortly after the recovery, XOM and GE were in a side-ways trend.
  - IBM stock plunged after its robust recovery.
  - Meanwhile, JPM and UNH maintain their uptrend movements.

3.7 Data Tools

3.7.1 Collect Historical Data

There are many tools available to collect historical market data. However, for this project we will use NinjaTrader as this tool also offers a wide range of capability. Due to the time constraint, we prefer a tool that has as many functions as we need, so that we do not need to spend time on different tools.

We have chosen to test our strategy on daily, historical U.S. equities data by utilizing the AlphaVantage™API. All U.S. equity data and the respective Open, High, Low, Close (OHLC) values have been adjusted for dividends and splits. We have chosen to run our back-tests and optimizations over 80% of the data and use the remaining 20% for validation testing. In all cases, we attempted to source the "full" dataset from the AlphaVantage™API for the life of the security, with most data going back to at least the early 2000s.

3.7.2 Back-testing and Paper Trading

Since the underlying framework for Ninja Trader is C#, all of the default indicators are built-in C# already. It is very convenient for us to build indicators without having to start from scratch. Instead, we can just find similar indicators in NinjaTrader, and then modify the parameters or logic if we need to.

The second benefit of this application is the back-test and simulation capability already built-in. Therefore, we could easily validate our trading system. However, there are technical constraints with this application as not all of our members familiar with C#, and it might be difficult to implement the machine learning system that we build in Python to the Ninja Trader.

3.8 Machine Learning Classification

We will use a machine-learning algorithm to handle classification problems:

- Neural Network

3.8.1 Detect Market Condition

We will use Python language and Jupyter Notebook to build our classification system based on the historical data.

3.8.2 Trading Strategy Customization Platform

As our team members have different backgrounds. We are considering Python (Backtrader) and C# (Ninja Trader) to implement our trading strategy. Since the areas of expertise of our team members are diverse, we need to select the most suitable platform that fits the needs of the projects and the skill-sets of our team members.

3.9 Developing the Strategy

3.9.1 Indicators

The Dip Trip trading strategy utilizes a well-known technical indicator called Fibonacci Retracement levels to identify trade signals. During our research we were not able to find an open-source implementation of this indicator, so we chose to develop our own. Fibonacci levels are the only indicator used in the strategy, and in one variation of the strategy, two sets of Fibonacci levels are used to generate trade signals.

3.9.2 Swing Points

In order to calculate the Fibonacci retracement levels indicator, the analyst typically will look at a price chart, identify important “swing” points, and use these points as the high and low point at which to generate the indicator. This means that there is typically a level of subjectivity as to how the analyst may choose these points and which points they deem to be significant.

Technically a Fibonacci retracement can be drawn between any two points, but it is most common to use a high and low point that is deemed significant. Since this is a systematic trading strategy, we had to implement an indicator to calculate these points for us.

In order to find the critical points that our strategy uses to calculate the Fibonacci levels, we set a threshold value - a percentage.

We use a brute force iteration over the time series and mark “swing” points when the price moves above and below this threshold.

This is the main hyper-parameter that we use to optimize our strategy since the level that we set for this threshold will determine when and where the Fibonacci levels are generated.

For example, using a threshold value of 3% generates these set of “swing” points:

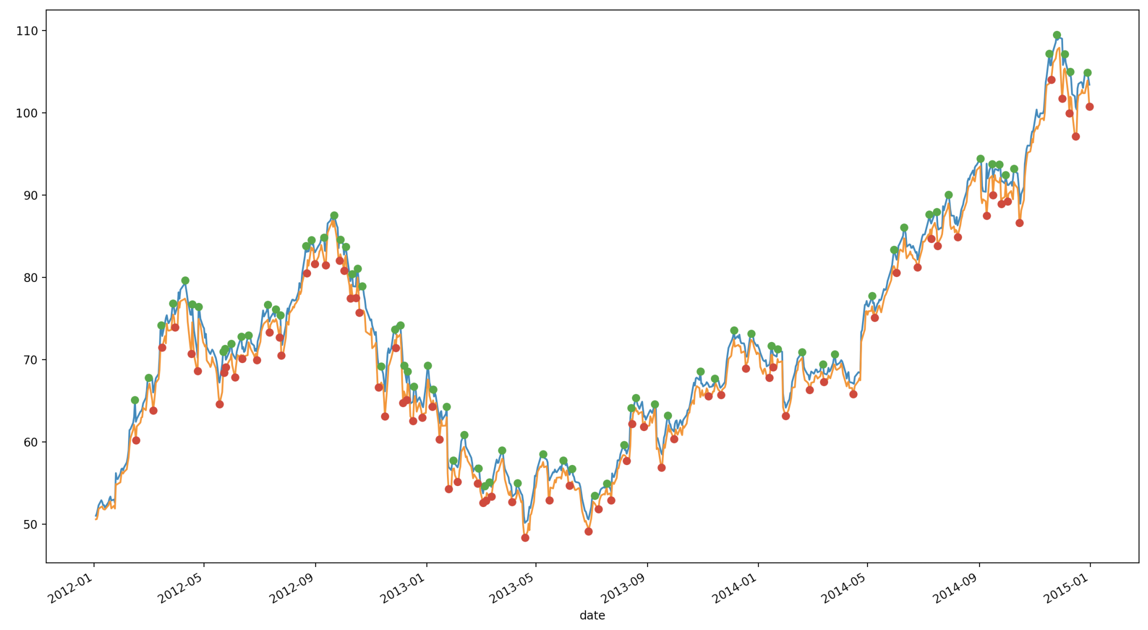


Figure 4: Swing Points with Threshold = 3

Using a threshold value of 8% generates these set of “swing” points:

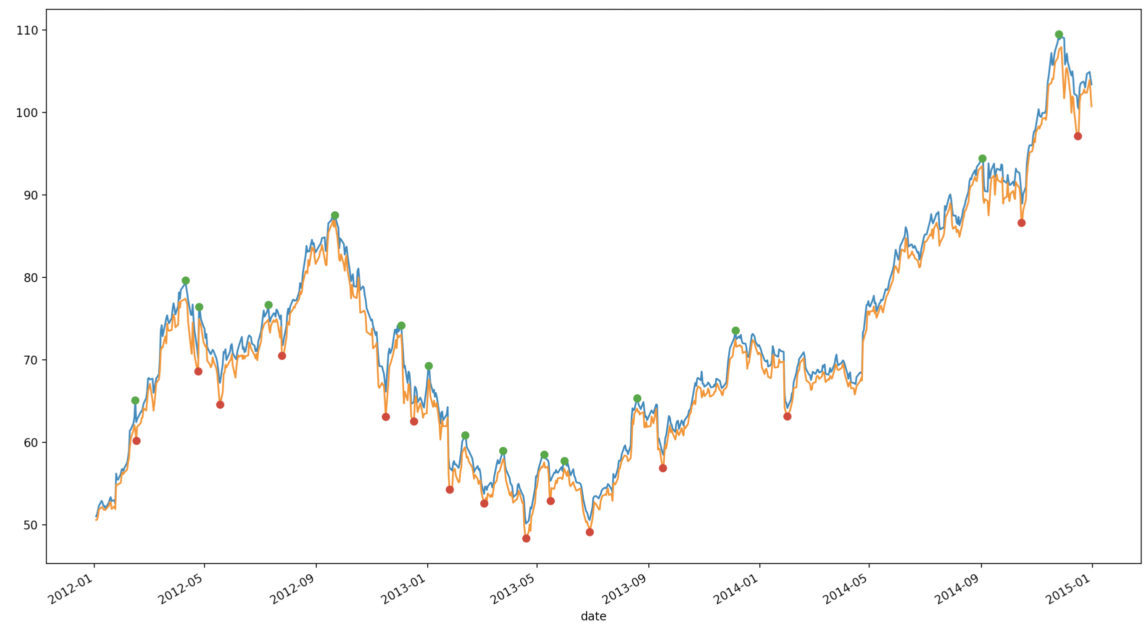


Figure 5: Swing Points with Threshold = 8

By increasing the threshold, fewer points are generated, but they may be more significant and less “noisy”. By using Fibonacci levels calculated on two sets of points, one lower threshold, and one higher, it is theorized that a trade signal generated may be more powerful and yield higher profits.

3.9.3 Fibonacci Retracement

The Fibonacci sequence is an iconic mathematical formula that allegedly is found in nature. It is derived by starting with the integers 1, 1 and then adding the last number n with the number n_{-1} to generate the next value such that you generate a set if numbers as such: $\{1, 1, 2, 3, 5, 8, 13, 21, ...\}$.

In trading, the Fibonacci retracement values are derived from this sequence by dividing the term n by the previous term n_{-1} to generate ratios. The most popular ratios used in trading are 23.6%, 38.2%, 50%, 61.8%, and 78.6%. Fibonacci retracement levels can be used to identify areas of support and resistance, generate trade signals, or position stop-loss orders. In our research, we will be using these levels to generate trades.

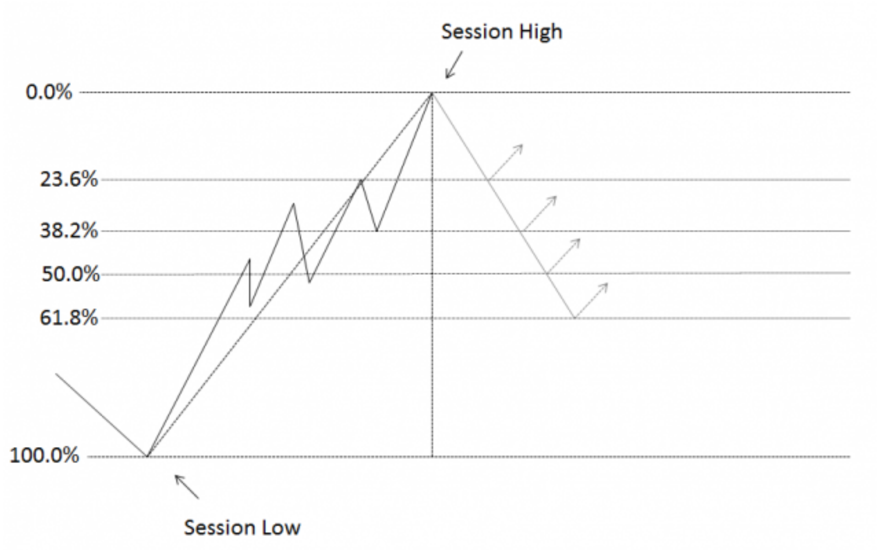


Figure 6: Fibonacci Retracements

3.9.4 Signal Generation

The Dip Trip trading strategy has several variants, each with its method for determining a trade signal. Further, there it is usually up to the technical analyst to provide the final say on whether a given trade setup looks promising. However, in order to make our strategies systematic, we have had to code this logic ourselves. We have several variants of the strategy, but in all cases, signals are generated by seeing if the current day's closing price has crossed through our chosen Fibonacci Retracement level. Other variants of the strategy put other filters in place such as using a second Fibonacci Retracement value or only taking trades when we predict that the market is in an uptrend. Other technical indicators could be used as a filter in further research. However, one should be careful not to use too many indicators as this would likely overfit the strategy and lead to false conclusions about how well it will do out of sample.

3.9.5 Position Sizing

Position sizing is a critical design element both for back-testing consideration and also a live production trading system. Given that we will be back-testing our strategy on a single security at a time, as opposed to a portfolio of securities, we have decided to implement a naive position sizing algorithm. Our algorithm will attempt to invest the entire portfolio into the security upon signal generation. The main advantage of this methodology is that it helps produce a linear equity curve over long periods of time given that stock prices tend to rise, at least historically in the United States. This is opposed to other naive strategies such as investing a fixed number of shares in every trade which can produce exponential equity curves as stock prices rise and those fixed shares represent a larger and larger dollar amount. This will more heavily weight the key performance metrics to the later years of the strategy, which can lead to false conclusions about the robustness of the strategy.

3.9.6 Position Exit

Another important design choice that can heavily influence the strategy is how trade exit signals are generated. In fact, it is our opinion that this may be equally important as how trade entry signals are generated. A strategy could have an excellent ability to find a trade signal but still fail to make money if the trade exit logic is poorly designed.

In our strategy, we have chosen to implement a trailing stop loss to generate our trade exit signals. A trailing stop-loss is a variation of a traditional stop-loss order type that the trader can use to “lock-in” gains as their position appreciates. We will set our initial stop-loss price at 1.8 times the Average True Range of the security calculated over the last 14 days. There is nothing significant about the 1.8 multiplier, although we feel this gives the security enough room to “breathe” and allow the trade to work out. We did not optimize this parameter and this logic warrants further research on its own.

3.9.7 Hyper-Parameter Optimization

Our main Dip Trip trading strategy has two inputs; the threshold by which to calculate the swing points used to calculate the Fibonacci Retracement levels. These parameters need not be integer values, but rounding to the nearest 0.10 or 0.50 of a percent should be sufficient to get a good understanding of the stability of the parameter universe. Since the parameter set is also bound by a lower bound of 0 and an upper bound of 100 (although no points may be generated with a threshold of above, say, 20) we chose to use a grid search algorithm to test each parameter combination. Then, we prepared “heatmap” plots for various key performance indicators such as Sharpe ratio, Total Return, Drawdown, etc., in order to get a visual representation of the stability of the parameter set. The goal is to see large regions with very stable results which could suggest that the strategy is robust and stable to changes in the parameters.

3.10 Back-testing

Back-testing will be conducted by utilizing the Backtrader library written in python. Backtrader provides a feature-rich infrastructure to back-test custom trading strategies. We can supply our strategy logic, order sizing parameters, slippage and commissions, data, and other parameters to define our strategy. The framework also provides an API for grid search optimization that we will utilize to test the robustness of our strategy and identify optimal parameter combinations for our Dip Trip trading strategies. A typical Backtrader strategy looks like this:

```python
class DipTripSingle(Strat):
    params = (('short_threshold', 3), ('atrwindow', 14), ('atrmultiplier', 1.8))
    def __init__(self):
        self.atr = bt.ind.AverageTrueRange(period=self.p.atrwindow)
        self.short_fibonacci = Fibonacci(threshold=self.p.short_threshold)
        self.price_data = self.datas[0]
        self.broker.set_coc(True)
    def next(self):
        # Check if we have a position, if we do, skip this iteration
        if self.position:
            return
        # Check to make sure the fibonacci levels are available, if nan, return early
        sixty_two = self.short_fibonacci.sixty_two[0]
        if sixty_two != sixty_two: # evaluates nans
            return
        # If price has cross below our signal, take action
        if self.price_data.close < self.short_fibonacci.sixty_two[0]:
            stop_distance = self.p.atrmultiplier * self.atr[0]
            try:
                self.order = self.buy_bracket(exectype=bt.Order.Market,
                                          stopexec=bt.Order.StopTrail,
                                          stopargs={'trailamount': stop_distance},
                                          limitexec=None)
                self.log('BUY CREATE, %.2f' % self.price_data[0])
            except:
                return
```

We supply the parameters, indicators we need for our strategy and override the `next()` method of this class with our signal generation criteria, and Backtrader takes care of the order execution, management, and performance tracking.

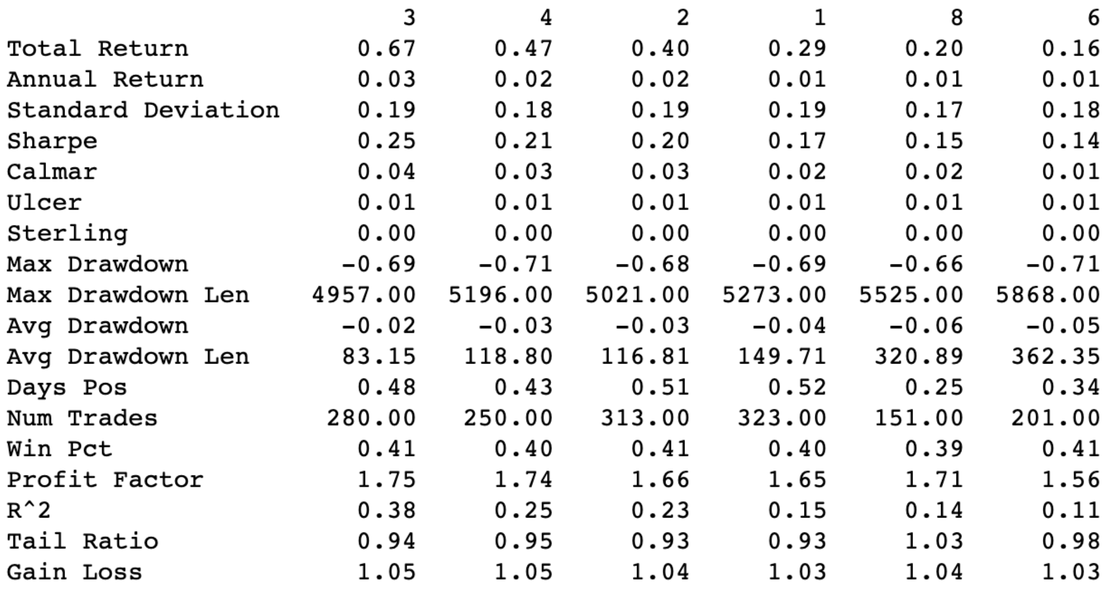
4 Results

4.1 Dip Trip with Single Parameter

Loosely defined, the Dip Trip trading strategy can be defined as a strategy where the investors looks to "buy the dip". This means that the investors look for a large rally and tries to buy the "pullback" to a key level in hopes that buyers will come back into the market and drive the price to a new high. Allen Farley in the book *The Master Swing Trader* states that buyers will naturally wait for a pullback in a large rally before they re-enter the market (Farley 2001). He recommends using Fibonacci retracement levels to identify natural levels where buyers may be looking to re-enter the market.

Specifically, the author recommends the 62% retracement level as a place to look for opportunities to buy the stock. For our first trading strategy, this is exactly what we will do. We will calculate the Fibonacci retracement levels and if the price of the stock has moved through this level, we will buy that stock at the close of the candle. Then, we size and manage our position according to our previously discussed methodology (100% invested and a trailing stop-loss). This is the same methodology we use for all variations of the strategy. Below, we show the key performance metrics from the top 5 back-tests as a result of our test data set.

4.1.1 Key Performance Indicators



```
                          3        4        2        1        8        6
Total Return           0.67     0.47     0.40     0.29     0.20     0.16
Annual Return          0.03     0.02     0.02     0.01     0.01     0.01
Standard Deviation     0.19     0.18     0.19     0.19     0.17     0.18
Sharpe                 0.25     0.21     0.20     0.17     0.15     0.14
Calmar                 0.04     0.03     0.03     0.02     0.02     0.01
Ulcer                  0.01     0.01     0.01     0.01     0.01     0.01
Sterling               0.00     0.00     0.00     0.00     0.00     0.00
Max Drawdown          -0.69    -0.71    -0.68    -0.69    -0.66    -0.71
Max Drawdown Len    4957.00  5196.00  5021.00  5273.00  5525.00  5868.00
Avg Drawdown          -0.02    -0.03    -0.03    -0.04    -0.06    -0.05
Avg Drawdown Len      83.15   118.80   116.81   149.71   320.89   362.35
Days Pos               0.48     0.43     0.51     0.52     0.25     0.34
Num Trades           280.00   250.00   313.00   323.00   151.00   201.00
Win Pct                0.41     0.40     0.41     0.40     0.39     0.41
Profit Factor          1.75     1.74     1.66     1.65     1.71     1.56
R^2                    0.38     0.25     0.23     0.15     0.14     0.11
Tail Ratio             0.94     0.95     0.93     0.93     1.03     0.98
Gain Loss              1.05     1.05     1.04     1.03     1.04     1.03
```

Figure 7: Dip Trip Optimization Results

As you can see from the results above, the trading strategies did in fact produce profitable results, however, it would be difficult to call this a viable trading strategy based on these results. The highest total return over the test period was only 67%, generating a Sharpe ratio of just 0.25. Even worse, we had to endure a drawdown of nearly 70% for nearly 5,000 days! Additionally, not every strategy variation generated a positive return over the period, as seen in the chart below.

4.1.2 Equity Curves

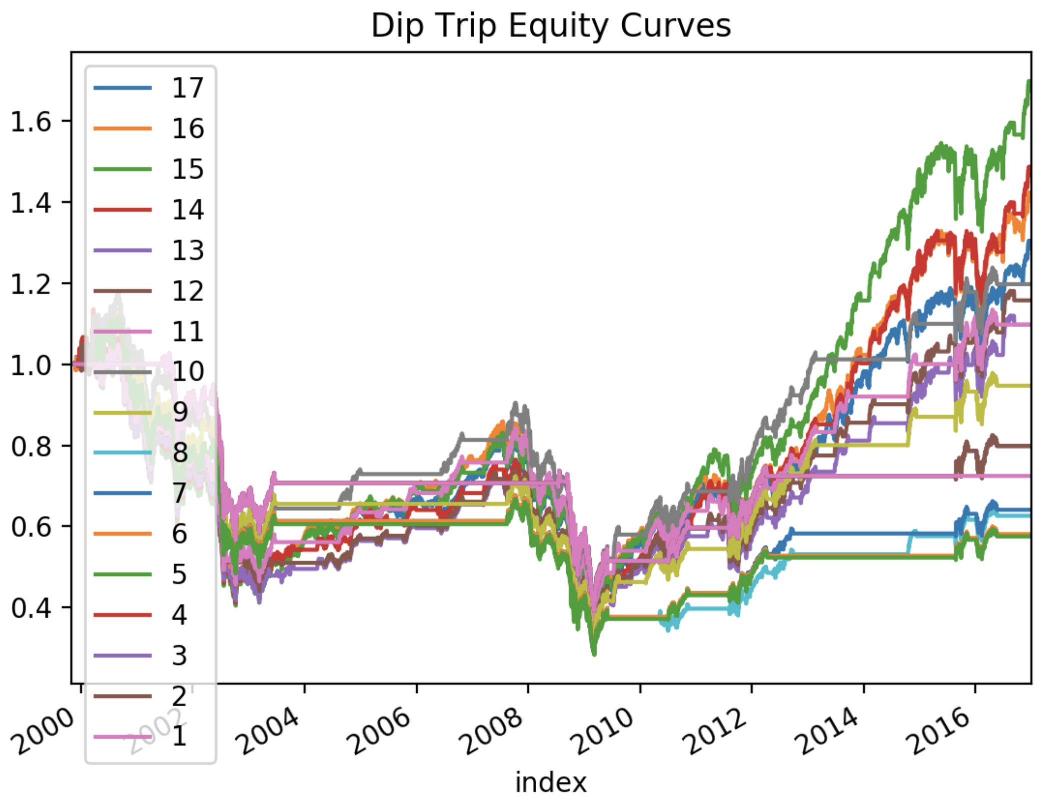


Figure 8: Dip Trip Equity Curves

As you can see, not every strategy generated a positive return. However, the majority of them did, which shows promise and basis to continue developing the strategy to improve its performance.

4.2 62/38 Strategy

Alan Farley (Farley 2001) in his book, recommends improving the Dip Trip trading strategy by using what he calls the 62/38 strategy. This is a variation of the Dip Trip that uses two Fibonacci levels generated from different swing points and only takes trades when the price is moving through both of them. We use one set of Fibonacci points used on short term price variations called the "short_fibonacci" and then another set of points generated on the larger trend called the "large_fibonacci". Then, inside our strategy logic, we identify times when the 62% level of the short fibonacci is within 1% (in price) of the 38% level of the large fibonacci indicator. The idea is that by combining the two levels, we will find a higher probability trade because the underlying stock is finding support at an important level on both a shorter and longer-term trend. The strategy logic is implemented as such:

```
class DipTrip6238(Strat):
    params = (('large_threshold', 5),
              ('short_threshold', 3),
              ('atrwindow', 14),
              ('atrmultiplier', 1))

    def __init__(self):

        self.atr = bt.ind.AverageTrueRange(period=self.p.atrwindow)
        self.short_fibonacci = Fibonacci(threshold=self.p.short_threshold)
        self.large_fibonacci = Fibonacci(threshold=self.p.large_threshold)
        self.price_data = self.datas[0]
        self.broker.set_coc(True)

    def next(self):

        # Check if we have a position, if we do, skip this iteration
        if self.position:
            return

        # Check to make sure the fibonacci levels are available, if nan, return early
        sixty_two = self.short_fibonacci.sixty_two[0]
        thirty_eight = self.large_fibonacci.thirty_eight[0]
        if sixty_two != sixty_two or thirty_eight != thirty_eight:
            return

        # check to see if the large and short fibonacci values are close together
        if abs((self.short_fibonacci.sixty_two[0] - self.large_fibonacci.thirty_eight[0])
                / self.short_fibonacci.sixty_two[0]) < 0.01:

            # If price has cross below our signal, take action
            if self.price_data.close < self.short_fibonacci.sixty_two[0]:
                stop_distance = self.p.atrmultiplier * self.atr[0]
                self.order = self.buy_bracket(exectype=bt.Order.Market,
                                              stopexec=bt.Order.StopTrail,
                                              stopargs={'trailamount': stop_distance},
                                              limitexec=None)

                self.log('BUY CREATE, %.2f' % self.price_data[0])
```

4.2.1 Key Performance Indicators

```
                        1-4     2-4     1-7      1-5      2-5      2-6
Total Return           3.19    1.94    1.92     1.91     1.87     1.50
Annual Return          0.09    0.06    0.06     0.06     0.06     0.05
Standard Deviation     0.09    0.08    0.07     0.08     0.08     0.07
Sharpe                 0.96    0.79    0.89     0.80     0.84     0.78
Calmar                 0.43    0.42    0.62     0.29     0.35     0.35
Ulcer                  0.00    0.00    0.00     0.00     0.00     0.00
Sterling               0.01    0.01    0.01     0.01     0.01     0.01
Max Drawdown          -0.20   -0.15   -0.10    -0.22    -0.18    -0.16
Max Drawdown Len     510.00  743.00  960.00  1321.00  1210.00  1747.00
Avg Drawdown          -0.02   -0.02   -0.02    -0.02    -0.02    -0.02
Avg Drawdown Len      43.86   60.44   67.56    69.45    70.11    79.01
Days Pos               0.16    0.12    0.08     0.11     0.10     0.08
Num Trades           173.00  138.00   73.00   119.00   101.00    79.00
Win Pct                0.53    0.51    0.66     0.53     0.55     0.62
Profit Factor          2.23    2.19    2.13     2.40     2.46     2.23
R^2                    0.89    0.93    0.86     0.86     0.90     0.90
Tail Ratio             1.33    1.37    3.12     1.60     1.76     2.13
Gain Loss              1.37    1.33    1.53     1.36     1.40     1.44
```

Figure 9: 62/38 Optimization Results

Interestingly, the top 5 parameter combinations show much better results than the single Dip Trip strategy. Our Sharpe ratio jumps to nearly 1 and our max drawdown reduced to just 20% in the highest performing strategy. However, by looking at the equity curves, we see many strategies that have small wins and losses, and ideally, we would like to see a large number of parameter combinations performing extremely well in order to validate the robustness of the strategy.

4.2.2 Equity Curves

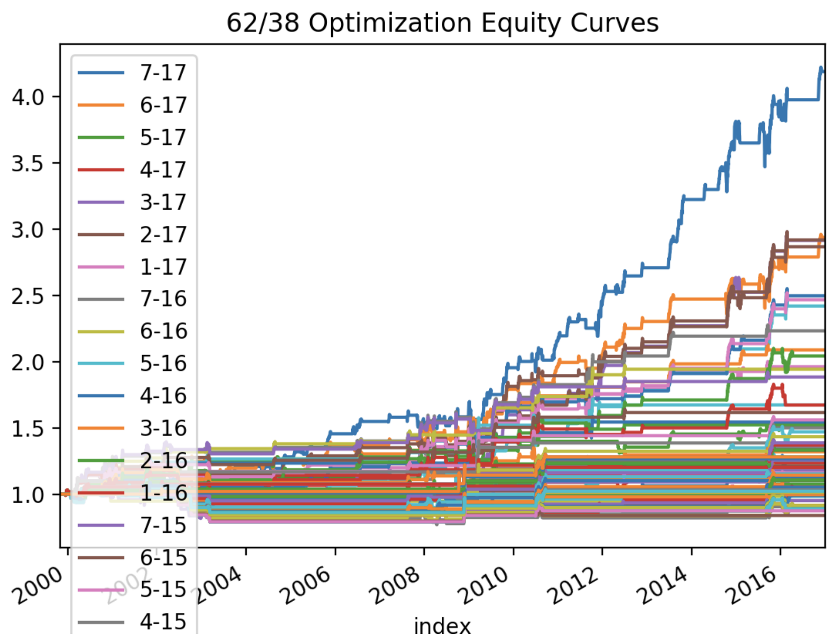


Figure 10: 62/38 Optimization Results

4.2.3 Heatmap

A heatmap generated by using the Annual Return of the back-test reveals a large portion of the parameter universe that returns close to 0 annual return. However, there is also a range that shows stability and annual returns of 4-7% during the back-test. We also plotted heatmaps for every key performance indicator and those results are available in the notebook/results directory of the project repository.

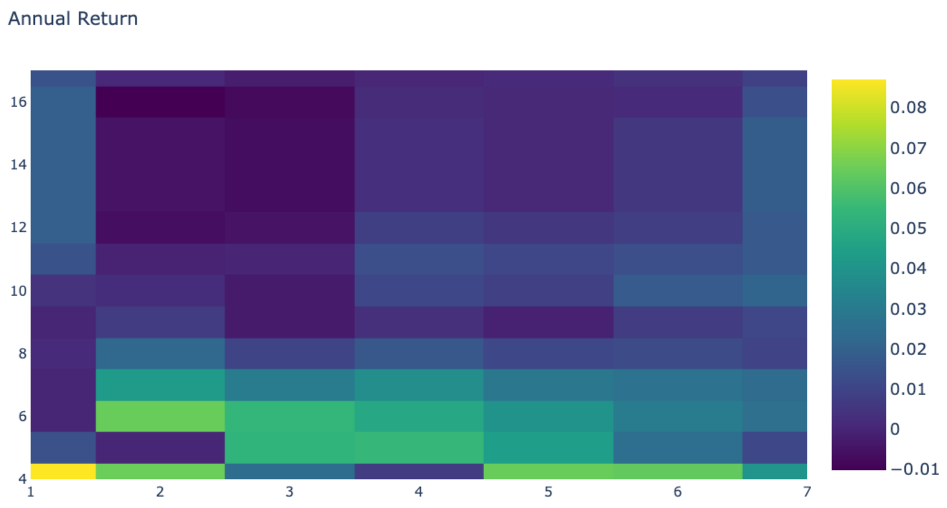


Figure 11: Annual Return Heatmap

4.3 62/38 Machine Learning Overlay Strategy

In order to further improve the strategy, we have decided to implement a machine learning overlay to filter trade signals that are generated in bear market conditions. This was not suggested by the author, but rather an idea that we had to improve the systematic nature of the strategy. In theory, it is not wise to buy dips during long-term bear markets because there may be more selling pressure in the market and not as many investors willing to buy at these key support levels. If we can accurately predict which market environment we are in, we can make sure that we only make trades when we are in a bull market. The strategy is implemented as such where `MarketConditionNum` contains an integer from 0-4 as the result of our machine learning implementation. We only take trades where this value is equal to 3 or 4 which indicates a bull market.

```python
class DipTripBull(Strat):
    params = (
        ('large_threshold', 5),
        ('short_threshold', 3),
        ('atrwindow', 14),
        ('atrmultiplier', 1)
    )

    def __init__(self):
        self.atr = bt.ind.AverageTrueRange(period=self.p.atrwindow)
        self.short_fibonacci = Fibonacci(threshold=self.p.short_threshold)
        self.large_fibonacci = Fibonacci(threshold=self.p.large_threshold)
        self.price_data = self.datas[0]
        self.broker.set_coc(True)
```

```
def next(self):
    # Check if we have a position, if we do, skip this iteration
    if self.position:
        return

    # predicts a Bear Market
    if self.price_data.MarketConditionNum < 3:
        return

    # Check to make sure the fibonacci levels are available, if nan, return early
    sixty_two = self.short_fibonacci.sixty_two[0]
    thirty_eight = self.large_fibonacci.thirty_eight[0]
    if sixty_two != sixty_two or thirty_eight != thirty_eight:
        return

    # check to see if the large and short fibonacci values are close together
    if abs((self.short_fibonacci.sixty_two[0] - self.large_fibonacci.thirty_eight[0]) /
           self.short_fibonacci.sixty_two[0]) < 0.01:
        # If price has cross below our signal, take action
        if self.price_data.close < self.short_fibonacci.sixty_two[0]:
            stop_distance = self.p.atrmultiplier * self.atr[0]
            self.order = self.buy_bracket(exectype=bt.Order.Market,
                                          stopexec=bt.Order.StopTrail,
                                          stopargs={'trailamount': stop_distance},
                                          limitexec=None)
            self.log('BUY CREATE, %.2f' % self.price_data[0])
```

4.3.1 Key Performance Indicators

As we can see, our machine learning overlay did increase the performance of our strategy. Not only was the total return of our top-performing back-test higher, but the trade win percentage is higher, and we see an increase in the Sharpe ratio to 1.35. Additionally, the maximum drawdown is further reduced to only 9%. This suggests that our strategy successfully avoided taking bad trades in bear markets and focused more on trades that were more likely to be successful and yield higher profits.

```
                         1-4      1-5      2-5      2-4      1-7      1-6
Total Return            3.28     2.40     2.00     1.91     1.61     1.59
Annual Return           0.09     0.07     0.07     0.06     0.06     0.06
Standard Deviation      0.06     0.06     0.05     0.06     0.05     0.05
Sharpe                  1.35     1.28     1.26     1.13     1.14     1.17
Calmar                  0.98     1.11     1.33     0.67     0.87     0.86
Ulcer                   0.00     0.00     0.00     0.00     0.00     0.00
Sterling                0.02     0.01     0.01     0.01     0.01     0.01
Max Drawdown           -0.09    -0.07    -0.05    -0.10    -0.07    -0.07
Max Drawdown Len     1016.00  1036.00  1036.00  1016.00  1114.00  1952.00
Avg Drawdown           -0.01    -0.01    -0.01    -0.01    -0.01    -0.01
Avg Drawdown Len       47.50    64.74    72.46    68.86    76.62    87.22
Days Pos                0.11     0.08     0.07     0.08     0.05     0.06
Num Trades             93.00    58.00    48.00    68.00    34.00    40.00
Win Pct                 0.65     0.74     0.75     0.62     0.82     0.82
Profit Factor           2.01     2.16     2.37     2.19     2.15     2.08
R^2                     0.88     0.88     0.91     0.93     0.85     0.86
Tail Ratio              2.28     7.04      inf     3.05      inf      inf
Gain Loss               1.73     1.86     1.86     1.68     1.95     1.95
```

Figure 12: 62/38 ML Overlay Optimization Results

4.3.2 Equity Curves

Overall, we see similar, but slightly better equity curves among all parameter combinations in the universe. Many strategies still yield close to 0 profit, but more strategies are generating consistent profits.

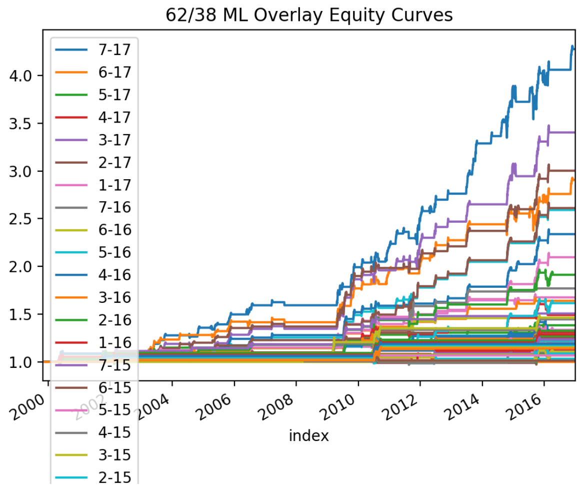


Figure 13: 62/38 ML Overlay Optimization Results

4.3.3 Heatmap

The heatmap however tells a different story. Although we have fewer strategies near the 0 or negative return space, our stable zone of strategies generating 4-7% annual returns seems to have slightly shrunk. This suggests that only some parameter combinations perform poorly during bear markets and others may benefit from it. Further analysis could be conducted on this front.

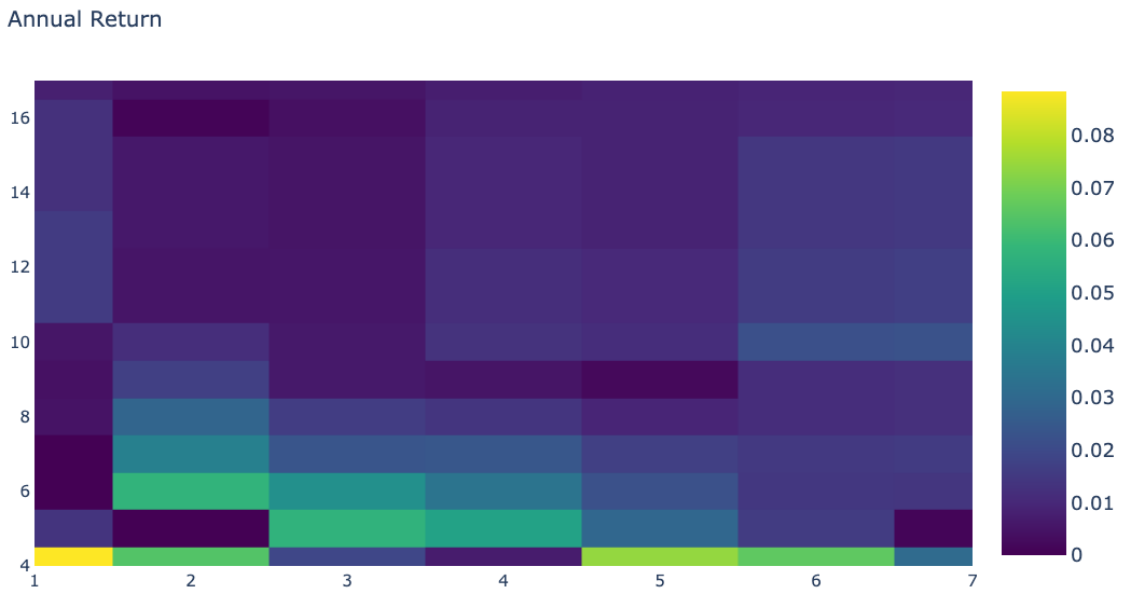


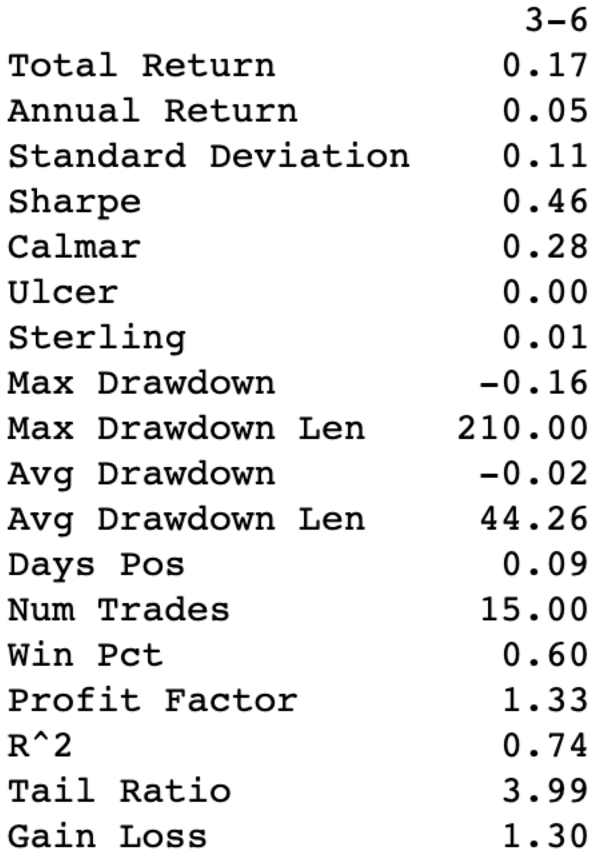
Figure 14: 62/38 ML Overlay Annual Return Heatmap

4.4 Out of Sample Testing

Next, we want to test this strategy out of sample to see how well it performs during realistic market conditions. We saved 20% of the dataset to be used for validation. Then, we will test the strategy on other stocks as well as a validation test of our strategies logic and robustness. We have chosen to use 62/38 strategy with the parameter set to 3, 6 as our out of sample test because this parameter combination appears to be in the middle of the stable zone.

4.4.1 Comparison of Results

Judging by the performance metrics, the strategy continues to perform well out of sample, with a win percent higher than the in-sample back-test. However, the Sharpe ratio of 0.46 is still a concern and could cause the strategy to be rejected for production use.



```
                         3-6
Total Return            0.17
Annual Return           0.05
Standard Deviation      0.11
Sharpe                  0.46
Calmar                  0.28
Ulcer                   0.00
Sterling                0.01
Max Drawdown           -0.16
Max Drawdown Len      210.00
Avg Drawdown           -0.02
Avg Drawdown Len       44.26
Days Pos                0.09
Num Trades             15.00
Win Pct                 0.60
Profit Factor           1.33
R^2                     0.74
Tail Ratio              3.99
Gain Loss               1.30
```

Figure 15: 62/38 Out of Sample Performance

Additionally, the equity curve is upward sloping but does suffer a relatively severe drop near the end of the period. Overall, it looks like the strategy identifies several good trades and is able to profit from them in the out of sample tests. In our opinion, this is sufficient to continue researching the strategy for production use and go to the next step of testing the strategy on other stocks or asset classes.

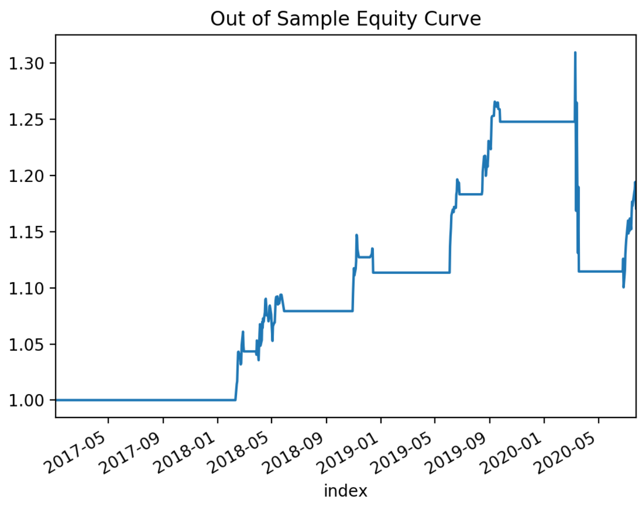
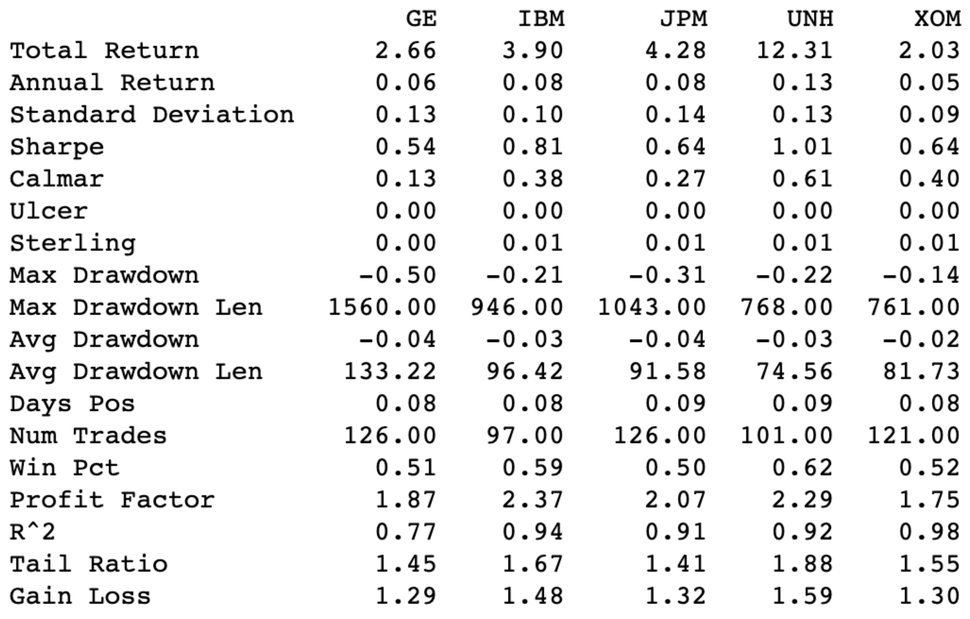


Figure 16: 62/38 Out of Sample Equity Curve

4.5 Testing Other Stocks

We selected five additional large-cap U.S. equity stocks to use for validation testing of our strategy. These include the ticker symbols ‘GE’, ‘IBM’, ‘JPM’, ‘UNH’, and ‘XOM’. This is a diverse group of stocks from different industries and should allow us to test the efficacy of our strategy logic. Interestingly, the performance looks respectable on the validation tests. Sharpe ratios vary from 0.54 to 1.01 and the win percentage is above 50 for all symbols. The number of trades is similar across stocks and overall, each strategy seems to have similar performance. This suggests that the strategy is not overfitted to the sample stock and that it may perform well for a variety of U.S. large-cap equities.



	GE	IBM	JPM	UNH	XOM
Total Return	2.66	3.90	4.28	12.31	2.03
Annual Return	0.06	0.08	0.08	0.13	0.05
Standard Deviation	0.13	0.10	0.14	0.13	0.09
Sharpe	0.54	0.81	0.64	1.01	0.64
Calmar	0.13	0.38	0.27	0.61	0.40
Ulcer	0.00	0.00	0.00	0.00	0.00
Sterling	0.00	0.01	0.01	0.01	0.01
Max Drawdown	-0.50	-0.21	-0.31	-0.22	-0.14
Max Drawdown Len	1560.00	946.00	1043.00	768.00	761.00
Avg Drawdown	-0.04	-0.03	-0.04	-0.03	-0.02
Avg Drawdown Len	133.22	96.42	91.58	74.56	81.73
Days Pos	0.08	0.08	0.09	0.09	0.08
Num Trades	126.00	97.00	126.00	101.00	121.00
Win Pct	0.51	0.59	0.50	0.62	0.52
Profit Factor	1.87	2.37	2.07	2.29	1.75
R^2	0.77	0.94	0.91	0.92	0.98
Tail Ratio	1.45	1.67	1.41	1.88	1.55
Gain Loss	1.29	1.48	1.32	1.59	1.30

Figure 17: Validation Test Performance Metrics

Reviewing the equity curves shows a similar story. Although we see outperformance of one stock, UNH, all stocks seem to have stable, linear uptrends. This reinforces our idea that the strategy is robust to various stocks and underlying sectors of the economy.

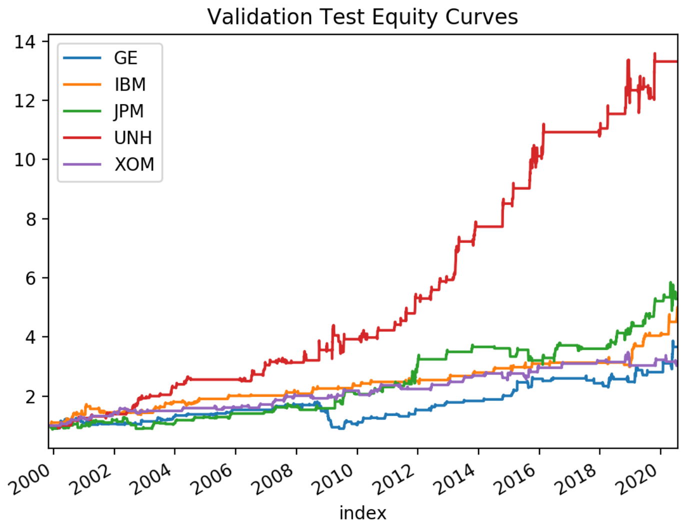


Figure 18: Validation Test Equity Curves

5 Discussion

To review, we tested three variations of the Dip Trip trading strategy coined by Alan Farley in his book *The Master Swing Trader*. We performed grid search optimization on all three strategies and reviewed key performance metrics and plotted heatmaps. All three strategy variations produce parameter combinations that produce profitable results. Further, our out of sample test was successful and our validation tests on other stocks also showed profitability in every scenario.

5.1 Analysis of Results

Despite successful out of sample and validation tests, there is still some concern with the performance of the strategy in general. First, in all cases, the strategy can endure years of drawdown, in some cases up to 50% or higher. This would be difficult for many investors to handle and they may abandon the strategy altogether in favor of something else.

Further, the original Dip Trip strategy does not perform very well on its own and required us to add additional overlays in order to make it more profitable. This is always a concern in trade strategy design and back-testing because of the risk of overfitting the strategy to the data. Luckily, we did not test the strategy on any other stocks until it was time for the validation test. This gives us confidence that we did not overfit our strategy to the original data.

While we did see some improvement to the best performing parameter sets in our Machine Learning overlay strategy, it did not benefit all back-tests equally. This is a concern, however, it appears that some parameter sets may benefit more than others from bear markets; or at least are not hurt by them. This would require further testing to see if this phenomenon is observed when applied to other data sets. If this is the case, we may be able to accept the fact that the machine learning overlay is a viable addition to the strategy in a certain subset of cases.

5.2 Real-World Implementation

In this section, we implement the Dip Trip strategies already explored, in a real-time system using C# and NinjaTrader. First, let's talk about some concepts about this implementation and framework.

5.2.1 NinjaScript

Is the language or script base on C# and the NinjaTrader framework, technically is a C# class that inherits from NinjaTrader. NinjaTrader provides a NinjaScript editor which is a powerful scripting editor that allows you to create custom indicators and strategies efficiently. The NinjaScript Editor includes powerful coding assistance and advanced debugging tools to help you custom build your indicator, strategy or any other supported NinjaScript type. This is the simple and direct way to add, create, and modify different components in the program, including indicators and strategies.

Another way to create scripts is by using Visual Studio. There is a custom folder where you get access to a visual C# project (.csproj) file that can be open in Visual Studio, for easy development, build, debugging, version control, package (Nuget), and all the features that this IDE provide. It give you a tool to distribute custom indicators and strategies to any user of NinjaTrader.

5.2.2 Strategy Builder

An easy way to create the skeleton of your strategy is by using the Strategy Builder tool. These tools open the NinjaScript Wizard that is used to generate the minimum code to get started programming any supported NinjaScript type. This wizard allows us to define any default properties, add custom input parameters, add additional data series, and add any relevant event methods. There are several different properties and options available in the NinjaScript Wizard depending on the type of NinjaScript object you are creating (NinjaTrader 2020).

In order to implement this strategy in C#, we decide to use a different approach to how the swing points are calculated. Let's present a well known technical indicator that can provide, in a different approach, this high and low point of a trend.

5.2.3 Donchian Channels

The Donchian channel is an indicator used in market trading developed by Richard Donchian. It is formed by taking the highest high and the lowest low of the last n periods. The area between the high and the low is the channel for the period chosen.

The Donchian channel is a useful indicator for seeing the volatility of a market price. If a price is stable the Donchian channel will be relatively narrow. If the price fluctuates a lot the Donchian channel will be wider. Its primary use, however, is for providing signals for long and short positions. If a security trades above its highest n periods high, then a long is established. If it trades below its lowest n periods low, then a short is established (Wikipedia contributors 2013). Career futures trader Richard Donchian developed the indicator in the mid-twentieth century to help him identify trends. He would later be nicknamed *"The Father of Trend Following"* (James Chen 2019).

The Formula for Donchian Channels are:

- UC = Highest High in Last N Periods
- $MiddleChannel = (UC - LC)/2$
- LC = Lowest Low in Last N periods

where:

- UC = Upper channel
- N = Number of minutes, hours, days, weeks, months
- $Period$ = Minutes, hours, days, weeks, months
- LC = Lower channel

5.2.4 Donchian Channels Calculations

Channel High:

- Choose time period (N minutes/hours/days/weeks/months).
- Compare the high print for each minute, hour, day, week or month over that period.
- Choose the highest print.
- Plot the result.

Channel Low:

- Choose time period (N minutes/hours/days/weeks/months).
- Compare the low print for each minute, hour, day, week or month over that period.
- Choose the lowest print.
- Plot the result.

Center Channel:

- Choose time period (N minutes/hours/days/weeks/months).
- Compare high and low prints for each minute, hour, day, week or month over that period.
- Subtract the highest high print from lowest low print and divide by 2.
- Plot the result.

Using this channel we can set up a continuous Fibonacci retracement that is calculated with the lower and higher band. With this channel, we set up our entries and exits when the conditions are met. Also, in this case, we don't set up a trailing stop, instead we enter and exit our strategy using the Fibonacci retracement key points, long when close price cross above R1 (38%) and short when price cross below R3 (62%) retracement. Notice that the center channel is always equivalent to the R2 (50%).

5.2.5 Dip Trip Ninjatrader Implementation

```
//This namespace holds Strategies in this folder and is required. Do not change it.
namespace NinjaTrader.NinjaScript.Strategies.SwingTrading
{
    public class DipTrip : Strategy
    {
        private MAX max;
        private MIN min;

        protected override void OnStateChange()
        {
            if (State == State.SetDefaults)
            {
                Description = @"Dip Trip Swing Trading Strategy";
                Name = "DipTrip";
                Calculate   = Calculate.OnBarClose;
                EntriesPerDirection = 1;
                EntryHandling   = EntryHandling.AllEntries;
                IsExitOnSessionCloseStrategy= true;
                ExitOnSessionCloseSeconds= 30;
                IsFillLimitOnTouch = false;
                MaximumBarsLookBack = MaximumBarsLookBack.TwoHundredFiftySix;
                OrderFillResolution = OrderFillResolution.Standard;
                Slippage = 0;
                StartBehavior = StartBehavior.WaitUntilFlat;
                TimeInForce = TimeInForce.Gtc;
                TraceOrders = false;
                RealtimeErrorHandling = RealtimeErrorHandling.StopCancelClose;
                StopTargetHandling = StopTargetHandling.PerEntryExecution;
                BarsRequiredToTrade = 20;
                IsInstantiatedOnEachOptimizationIteration    = true;
                IncludeTradeHistoryInback-test = true;
                IsOverlay = true;
                Period = 14;
                R1 = 38;
                R3 = 62;
            }
            else if (State == State.Configure)
            {
                AddPlot(Brushes.DarkOrange, "ChannelHigh");
                AddPlot(Brushes.Blue, "FR1");
                AddPlot(Brushes.Red, "FR2");
                AddPlot(Brushes.White, "FR3");
                AddPlot(Brushes.DarkOrange, "ChannelLow");
            }
            else if (State == State.DataLoaded)
            {
                max = MAX(High, Period);
                min = MIN(Low, Period);
            }
        }

        protected override void OnBarUpdate()
        {
            double max0 = max[0];
            double min0 = min[0];
            double range = max0 - min0;
            ChannelHigh[0] = max0;
            ChannelLow[0] = min0;
            FR1[0] = max0 - range * R1 / 100;
            FR2[0] = max0 - range * 0.5;
            FR3[0] = max0 - range * R3 / 100;

            if (BarsInProgress != 0)
                return;

            if (CurrentBars[0] < 1)
                return;

            if (Close[0] > FR1[0])
            {
                EnterLong();
            }
            if (Close[0] < FR3[0])
            {
                EnterShort();
            }
        }
    }
}
```

In `SetDefaults` state, we set up the parameters of our strategies and define the default values. In 'configure' we add, define the name and colors of our plots. In `DataLoaded` we perform a basic calculation (min and max) over a period of time for the historical data. Finally, `OnBarUpdate` we estimate our key points and perform the entry and exit logic of our strategy. Here an example of how the strategy is plotted visually.

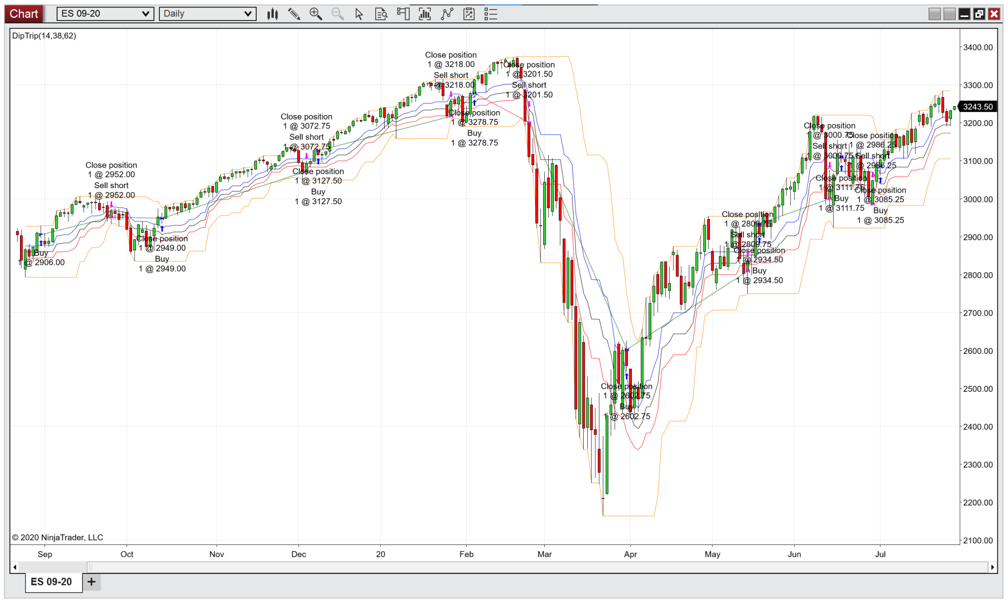


Figure 19: NinjaTrader Strategy Plot

5.2.6 Optimization

We obtain fine tune input parameters of a strategy through optimization. Optimization is the process of testing a range of values through iterative back-tests to determine the optimal input values over the historical test period based on your optimization fitness. First, let's find the optimal values of Dip Trip applied to `SPY` daily prices from 1999 to 2019, and see the results.

Table 6: NinjaTrader Strategy Analyzer Summary

Performance	All trades	Long trades	Short trades
Total net profit	$11,713.00	$12,068.00	($355.00)
Gross profit	$29,139.00	$19,274.00	$9,865.00
Gross loss	($17,426.00)	($7,206.00)	($10,220.00)
Commission	$0.00	$0.00	$0.00
Profit factor	1.67	2.67	0.97
Max. drawdown	($5,449.00)	($2,925.00)	($6,606.00)
Sharpe ratio	0.05	0.07	-0.02
Sortino ratio	0.16	0.34	-0.04
Ulcer index	0.11	0.05	0.19
R squared	0.84	0.82	0.03
Probability	12.49%	7.12%	53.05%
Start date	1/1/1999		
End date	1/1/2019		
Total # of trades	34	17	17
Percent profitable	35.29%	52.94%	17.65%

Performance	All trades	Long trades	Short trades
# of winning trades	12	9	3
# of losing trades	22	8	14
# of even trades	0	0	0
Total slippage	0	0	0
Avg. trade	$344.50	$709.88	($20.88)
Avg. winning trade	$2,428.25	$2,141.56	$3,288.33
Avg. losing trade	($792.09)	($900.75)	($730.00)
Ratio avg. win / avg. loss	3.07	2.38	4.50
Max. consec. winners	3	4	1
Max. consec. losers	4	2	7
Largest winning trade	$6,922.00	$6,922.00	$4,678.00
Largest losing trade	($1,500.00)	($1,500.00)	($1,273.00)
Avg. # of trades per day	0.01	0.00	0.00
Avg. time in market	213.85 days	309.65 days	118.06 days
Avg. bars in trade	147.18	213.35	81.00
Profit per month	$49.13	$50.76	($1.49)
Max. time to recover	614.00 days	1473.00 days	3416.00 days
Longest flat period	0.00 min	946.00 days	1358.00 days
Avg. MAE	$679.18	$643.71	$714.65
Avg. MFE	$1,709.35	$2,049.88	$1,368.82
Avg. ETD	$1,364.85	$1,340.00	$1,389.71

The optimal period value was 229 days.

5.2.7 Cumulative Net Profit

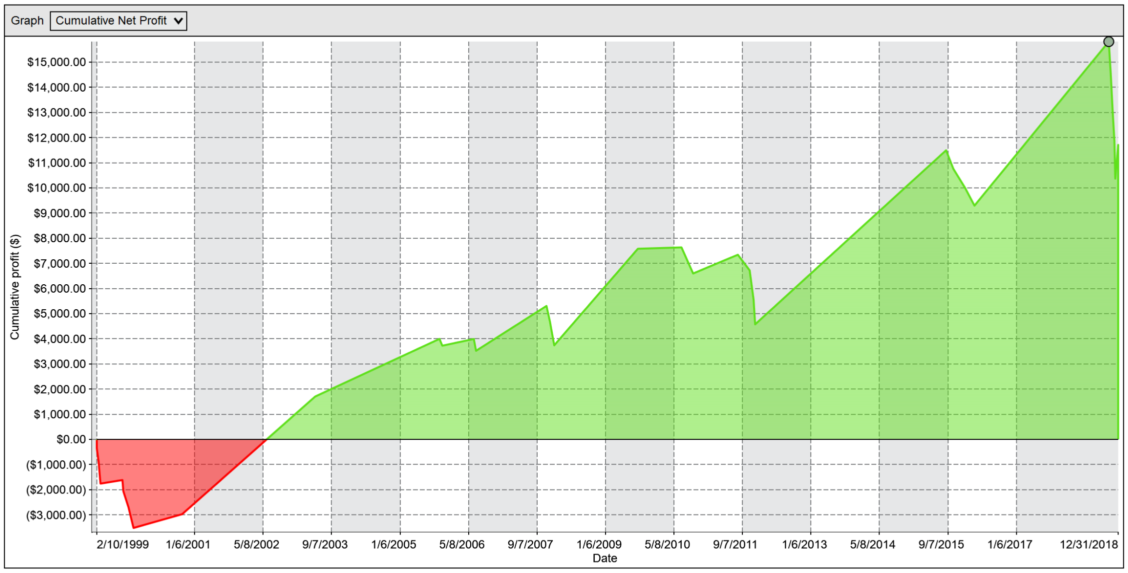


Figure 20: Cumulative Net Profit

5.2.8 Cumulative Max. Drawdown

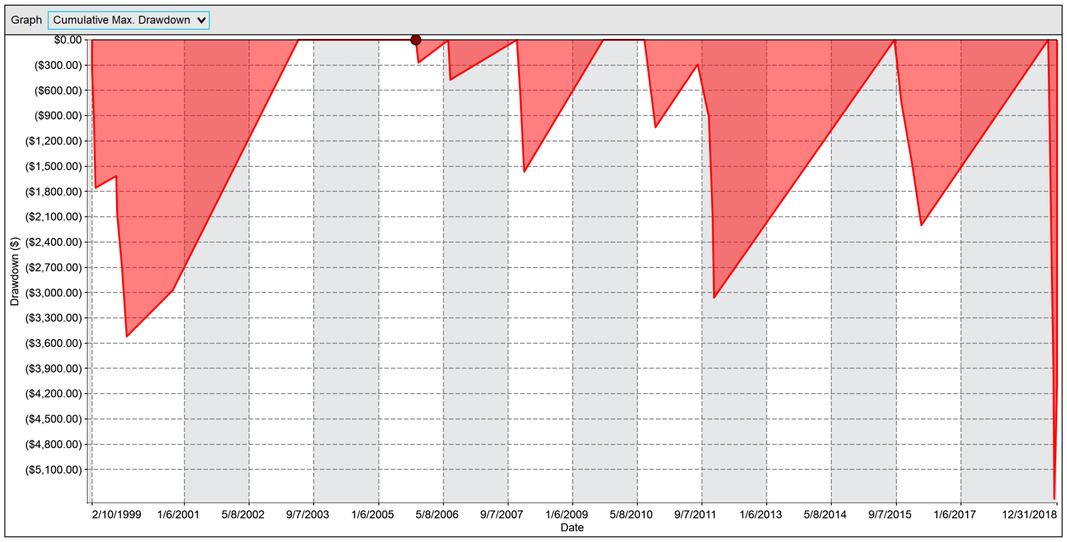


Figure 21: Cumulative Max. Drawdown

5.2.9 Optimization Graph

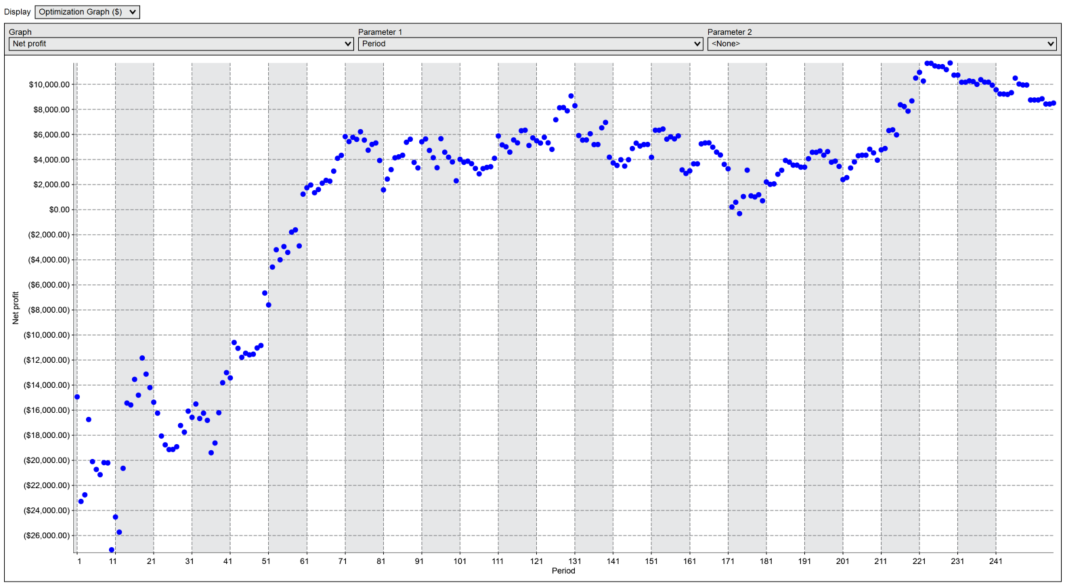


Figure 22: Optimization Graph

In this example, the pure form of the strategy performs well using a simple optimization technique (iterative) with a Max. Profit Factor target. This shows us how we can implement a strategy in a real-world trading system NinjaTrader and move our research to a ready to trade platform in real-time. More research has to be performed to improve the results and the implementation of the swing trading system using machine learning market condition detector has to be done. Using Visual Studio, we are able to implement this piece using NuGet's Machine learning libraries, like TensorFlow.NET, to detect the market condition and adjust the parameters of the swing strategies automatically via NinjaScript. Finally, this technique can be applied to intraday time frames to explore its efficacy.

6 Conclusion

In conclusion, like any other type of trading, swing trading tends to suit those who are willing to do thorough research. Technical analysis is controversial and against the Market Efficiency Hypothesis, but can be used to profit during a certain period or for a specific market. Many systematic swing trading strategies are based on the automation of trading based on predefined technical analysis signals. More advanced models or ensemble methods may be more predictive, however, they also might be more prone to overfitting and require more expertise and care on the part of the analyst. Systematic trading needs high levels of knowledge, expertise, and technology to successfully build and deploy trading strategies. We have to be aware that data-mining bias and data snooping can be challenging. However, there is an opportunity to improve results by applying machine learning techniques in Swing Trading. We hypothesize that a combination of technical analysis and statistical methods could potentially produce profitable swing trading strategies.

The purpose of the theoretical framework was to understand the swing trading field to which we will be contributing and provide a clear path where we'll be focusing on this study proposing a dynamic way to alter the strategy parameters depends on the market condition. Finally, we will build a system using classification techniques to decide what strategy is best suited for the current market.

In the methodology section, we outlined the research process, details of the swing trading techniques, frameworks, libraries, and machine learning implementation, and the development of the strategy itself including back-testing. This gave us the necessaries tools to deploy our system and obtain the necessary results for posterior analysis. Also, we exposed the constraints that we have encountered as well as how we were able to manage these constraints, one of the more important constraints was the short period of time to implement the system and the challenges that this represented.

Finally, we presented graphically the key performance indicators, equity curves, and heatmap of the results of the strategies. While we successfully tested the strategy out-of-sample, there is still some concern with the performance of the strategy due to years of drawdown, in some cases up to 50% or higher. Also, we tested the strategies with other stocks, obtaining a stable, linear uptrend, outperformance on `UNH`, giving us the idea that the strategy is robust to various stocks and underlying sectors of the economy. The final implementation of the system in NinjaTrader using C# closed the gap between pure research and real-world implementation, giving us the ability to deploy the strategies in a simulation or live mode in a trading platform that can be connected with our trading accounts (TD Ameritrade, Interactive Broker, etc) and also multiple sources of real-time data feeds.

6.1 Future Work

In order to further test and refine the strategy, we would recommend testing the strategy on other global asset classes such as foreign exchange, commodities, and cryptocurrencies. These are markets where other traders may be closely looking to buy pullbacks and they may already be using Fibonacci in their analysis, which would help our strategy benefit from the buying pressure of others.

Additionally, other "overlays" could be researched to further refine the strategy. We could add other technical indicators such as RSI and MACD to our analysis or other logic, like only trading stocks that have a "buy" rating at the time. The analyst should be careful to consider the consequences of data mining and back-test overfitting, however, we believe there may be simple ways to identify higher probability trade signals than what we have discovered here.

6.2 Final Thoughts

This project has taught us much in the areas of trading, back-testing, system design/infrastructure, and a host of other topics. We have challenged ourselves to apply rigor in our analysis and think like a practitioner. While our results may require additional scrutiny to be applied to real-world production, we believe we have met our original goal of implementing a cutting-edge technical analysis trading strategy in a systematic, optimized way using the latest technology to achieve profitability. We look forward to a lifetime of additional study and application in this field.

7 References

Andy, W. 2019. "Fundamental Analysis Vs Technical Analysis in the Forex Markets," June. https://www.andywltd.com/blog/fundamental-analysis-vs-technical-analysis-in-the-forex-markets/.

Bogullu, V. K., David Enke, and C. H. Dagli. 2002. "Using Neural Networks and Technical Indicators for Generating Stock Trading Signals." *Intelligent Engineering Systems Through Artificial Neural Networks* 12 (January): 721–26.

Campbell, Sean D, and others. 2005. "A Review of Backtesting and Backtesting Procedures."

Farley, A. S. 2001. *The Master Swing Trader: Tools and Techniques to Profit from Outstanding Short-Term Trading Opportunities*. McGraw-Hill Education. https://books.google.com/books?id=oOrsPCUzyO4C.

Folger, Jean. 2020. "NinjaTrader Review," June. https://www.investopedia.com/ninjatrader-review-4706930.

Gaucan, Violeta, and others. 2011. "How to Use Fibonacci Retracement to Predict Forex Market." *Journal of Knowledge Management, Economics and Information Technology* 1 (2): 1–1.

James Chen. 2019. "Donchian Channel Definition." https://www.investopedia.com/terms/d/donchianchannels.asp.

Lisa Emsbo-Mattingly, D. H. 2020. "The Business Cycle Approach to Equity Sector Investing." https://www.fidelity.com/webcontent/ap101883-markets_sectors-content/20.06.0/business_cycle/Business_Cycle_Sector_Approach_2020.pdf.

Loh, Elaine YL. 2005. "Profiting from Moving Averages and Time-Series Forecasts: Asian-Pacific Evidence." *The Asia Pacific Journal of Economics & Business* 9 (1): 62.

NinjaTrader, Customer Service. 2020. "Ninja Trader Help Guide." https://ninjatrader.com/support/helpGuides/nt8/.

Pesaran, M. Hashem, and Allan Timmermann. 1995. "Predictability of Stock Returns: Robustness and Economic Significance." *The Journal of Finance* 50 (4): 1201–28. http://www.jstor.org/stable/2329349.

Spörer, Jan. 2020. "Backtesting of Algorithmic Cryptocurrency Trading Strategies." *Available at SSRN 3620154*.

Weckman, Gary R, and Sriram Lakshminarayanan. 2004. "Identifying Technical Indicators for Stock Market Prediction with Neural Networks." In *IIE Annual Conference. Proceedings*, 1. Institute of Industrial; Systems Engineers (IISE).

Wikipedia contributors. 2013. "Donchian Channel — Wikipedia, the Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Donchian_channel&oldid=552334721.

Zhong, Xiao, and David Enke. 2019. "Predicting the Daily Return Direction of the Stock Market Using Hybrid Machine Learning Algorithms." *Financial Innovation* 5 (December). https://doi.org/10.1186/s40854-019-0138-0.