Applications of Machine Learning to Statistical Arbitrage Pairs Trading

Nicholas Kim
Princeton University
nk6@princeton.edu

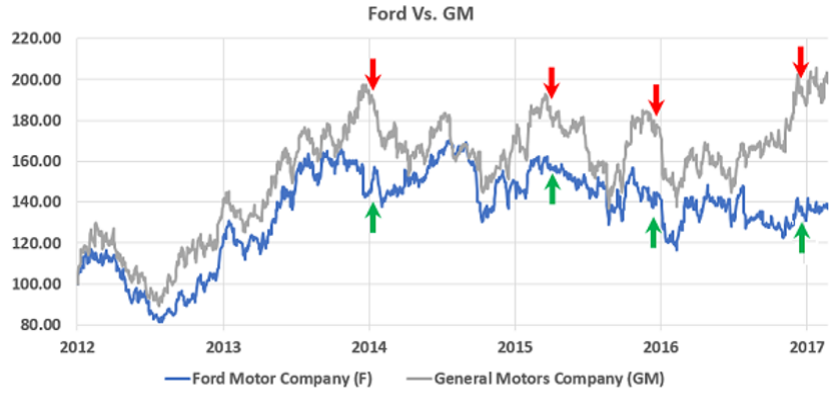
No Partners

Abstract

Trading in financial markets is an ever-present part of modern society, driving the fundamental aspects of price discovery and liquidity. Countless strategies for efficient trades exist today, with one of the most popular being a form of statistical arbitrage known as "pairs trading". A pairs trading strategy aims to find stocks that are highly cointegrated - in other words, stocks that exhibit a consistent difference of mean. Due to the mean-reverting nature of cointegrated series, a significant deviation in price difference relative to the historic price difference thus presents a trading opportunity. In such a scenario, the under-performing stock is bought while the over-performing stock is sold with the expectation that they will converge back to the historic price difference over time. The two primary challenges associated with pairs trading are the finding of stocks that exhibit genuine cointegrated behavior and the development of signals to enter into trading positions. Traditional methods of stock selection are often manual and tedious, comparing graphs of stocks hypothesized to have reason to move together, which not only presents potential for statistical unreliabilities (due to multiple comparison bias) but also reduces the breadth of valid pairs discovered. Traditional signal generations are often overly simplistic, relying on basic binary indicators of manually constructed threshold values of the price difference. We experiment on the Quantopian "Morningstar" dataset [1], which contains comprehensive price, volume, and fundamental company data for thousands of highly traded US equities. We applied clustering methods to create a sub-dataset of highly cointegrated stocks, and then tested machine learning applications to difference halving and velocity reversing trading strategies. We find that both strategies have difficulty outperforming a buy and hold position in favorable market conditions, though the decision tree velocity reversing method strongly outperformed the difference halving strategy. In non-favorable market conditions, both strategies consistently outperform the buy and hold, indicating robust potential for use.

1 Introduction

There is a persistent demand for successful trading strategies in industry, and a gamut of approaches exists. From purely discretionary procedures to the most rigorously technical and quantitatively-driven models, there's no shortage of individuals seeking to test their market-intuition against others. Pairs trading is a popular strategy, but often falls short due to highly manual implementations that use coarse, non-statistically-validated indicators of when to trade or not trade. For reference, we show a picture that visually depicts an optimal execution of pairs trading for Ford and GM.



-

As one can see, significant deviations in their expected price difference are usually followed by a return to the that historical expected price difference. Profit is achieved by selling the overpriced and buying the under-priced in the pair, with returns proportional to the magnitude of the initial anomaly of price separation. While obviously trivial to determine highly profitable entry and exit times when all the data is present, in actual trading, one only has information up until the present day. The fundamental problem then becomes how to determine quantitatively when the stock pair will begin to mean revert. A basic strategy might simply enter whenever the deviation exceeds a certain manual threshold, though such methods are prone to many shortcomings. For one, there is no validation of future price movements, and a price deviation of a certain percentage may have been caused by non-anomalous movements that will not immediately mean revert. Additionally, such manual thresholds are prone to entering into positions too early, losing out on a good deal of potential returns from waiting until the proper peak of separation.

We are curious to see whether or not machine learning can improve upon such issues.

2 Related Work

In general, machine learning in application to finance is a relatively new field. While simple forms of machine learning have been present in the industry for some time, the applications have generally been simple and in the interest of very supplementary/general information as opposed to creating core models and trading strategies. Of the recent attempts to apply machine learning to raw, purely automated trading/quantitative modeling, the general trend has been one of unsuccessful. The financial world and its unpredictability proves to be difficult for algorithms to explain well

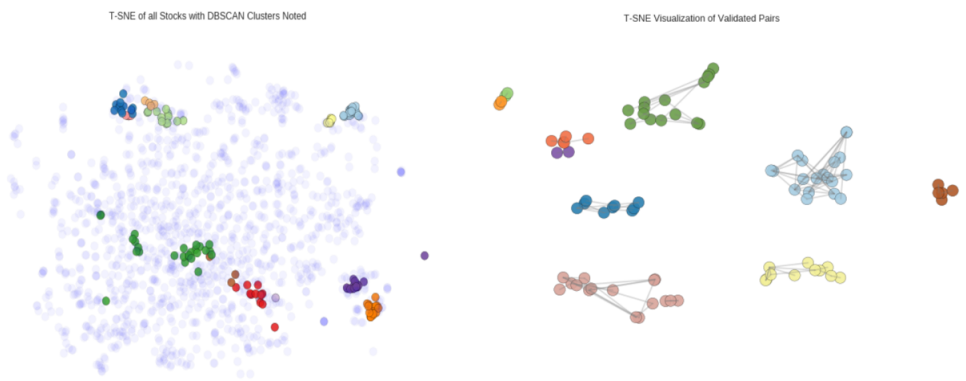
Machine learning applied to pairs trading seems quite new, with the majority of searched papers being from the last few years with a significant amount of student theses present. A student group from Stanford [3] applied Kalman filters alongside an EM algorithm and achieved returns of up to %10.41 when smoothed. One student from Erasmus University [2] applied neural networks to pairs trading and was able to achieve returns of %11 with ”high Sharpe and Sortino” ratios, indicating strong risk-adjusted performance.

2.1 Data Processing

The given dataset contains numerical data for thousands of equities, listing open, close, price movements, and fundamental company data, such as market cap and industry. First, we describe the clustering procedure outlined in Jonathan Larkin’s Quantopian notebook [4], which we applied to create a dataset of highly-cointegrated stocks to perform tests on. Then, we describe derivation of features for purposes of trading features.

Cointegration Clustering - We want to create a set of descriptive features for equities that will allow an unsupervised algorithm to reliably cluster them based on similarity of price movements. First, from the price movements of equities from day to day, we derive a percentage price movement metric, known as "returns". This is to account for the fact that many stocks that move similarly have wildly different prices, which would be difficult for many unsupervised learning models to account for. We then add three key fundamental company health indicators, 'Financial Health', 'Industry', and 'Market Cap' (total value of all company shares).

From here, we apply PCA to a reduction of n=50 components, as daily return data over a few years leads to quite high dimensionality. Then, we apply the DBScan clustering algorithm, which has two convenient properties 1) It automatically determines an optimal number of clusters 2) It does not cluster a data point if it does not fit into any of the discovered clusters particularly well. While the latter may be of issue if attempting to use the clustering as a final classification step, since we are only using this to create a dataset, the fact that some equities do not get clustered is irrelevant (so long as we end up with enough to form a reasonably sized dataset, which we do). The end result is visualized with the t-SNE algorithm, which reduces higher dimensional data into 2D visual-space.



Stock Data Derivations - From here, we added key metrics to the data of each stock. First, we applied log transformations to the returns in such a way that the difference between any two values of log returns equated to the percentage change in the value of stock (this simplifies many calculations regarding profits and returns). Then, we added (dependent on the trading strategy being tested) additional metrics, ranging from the difference between the log returns of the two stocks to velocities, accelerations, and 20-day windows of price movements. We describe these in more detail in the specific trading methods descriptions.

2.2 Evaluation Metrics

We evaluate the cointegration stock clusters by analyzing cointegration p-values for out-of-set data after initially applying cointegration methods. A p-value of 0.05 was used for validating cointegration.

For the trading strategies, we will use the following metrics.
-1)**Return** (raw percentage profit)
-2)**Volatility** (standard deviation of returns)
-3)**Sharpe Ratio** (risk adjusted returns)
-4)**Maximum Drawdown** (largest drop in profit from a trade)
-5)**Percentage Profitability** (what percent of trades were profitable)

In addition, we will compare to three baselines of monetary gain.
-6)**Rudimentary Pairs Trading Algorithm** (percentage profit from initiating a trade whenever the price difference exceeds the historical average by more than 30%)

-7)**Buy and Hold** (percentage profit from simply buying two equities and holding until the end of the testing period)
-8)**Risk-free profit** (percentage profit from buying non-defaultable government treasury bills with expiration date at the end of trading period. For our time period, the corresponding treasury bill interest is 0.66%)

-The rudimentary pairs trading algorithm serves as a reference to see to which extent the machine learning application can improve upon an execution of the conceptual method without significant statistical validation. The buy and hold comparison can be thought of as the baseline comparison against a skill-less investors who is open to taking a level of speculative risk for potential profit. The risk-free profit represents an absolute baseline of "free money" that anyone with money could earn in the given time period. If a trading strategy outperforms the risk-free profit rates, then even if it under-performs buy and hold, if the strategy is robust and safe, it may be actionable under certain conditions. However, it would ideally surpass the buy and hold metric, as actively managed trading strategies tend to have higher risk levels than buy and holds, so their potential profit should compensate this aspect.

3 Trading Strategy Experimentation

We approach the general creation of strategies from a classification approach. The general intention is to create an algorithm that is able to create a binary signal that will tell a trader/investor whether or not it would be prudent to enter into a trade at a given point in time (using only data that would be accessible from the current point in time, so model inputs cannot include any future data).

There are two principle tasks to tackle here. The first is the fact that there are no inherent labels on training data for what constitutes a valid time to enter a trade. The second is the potential breadth of derived features that one could use as inputs for the model. The creation of target labels in the trading data is perhaps one of the most challenging aspects of this project, as it requires one to create an ad-hoc reduction of the deceptively simple pairs trading imagery into a quantitative signal. While it's fairly easy to look at a paired stock graph and say "they look wide here, close there, so that's an easy trade", it is remarkably difficult to create an objective set of rules that extends into the majority of tradeable cases. For instance, if one simply put some kind of manual threshold of decrease in price difference to initiate a trade, this would be prone to inadequately capturing full profits when the initial price difference anomaly is large. The bottom-line here is that since the advantage of machine-learning lies in its way to abstractly derive future predictions, there would be no point in applying an algorithm that does not make use of future information, since there would be no reason to use machine learning to act on a simple indicator who's basis is not future data. As such, creating labels for good trading times with machine learning algorithms is quite new/uncharted territory, as it goes against the standard process of trading strategy development which does not assume the existence of future validation data hard-embedded into the final model. We proceed in the following sections to describe two approaches we developed independently of explicit outside reference. First however, we describe a procedure to convert a set of valid trading indicators into a testable trading strategy

How to convert trading signals into a testable trading strategy?
So imagine we have a model that is able to tell us whether or not it would be smart to enter into a trade at a given point in time. How do we convert this into a testable trading strategy? There are an infinite number of things a trader with x amount of capital could decide to do with this information (for instance, they could choose not to trade, or choose to only put in a certain portion of their capital, etc.), so we outline an original algorithm that comes with set of carefully considered and acknowledged assumptions to simplify the testing process. The algorithm functions as follows -
1) On any given day, check to see if the model gives a valid binary indicator.
2) If valid, we assume the trader pours all of their capital into the position.
3) If the exit-label used for the strategy as defined by the strategy's target label is met (also within a time-period specific by the trading), the trader exits out of the position entirely.
4) This process is repeated continuously until the end of the testing period.

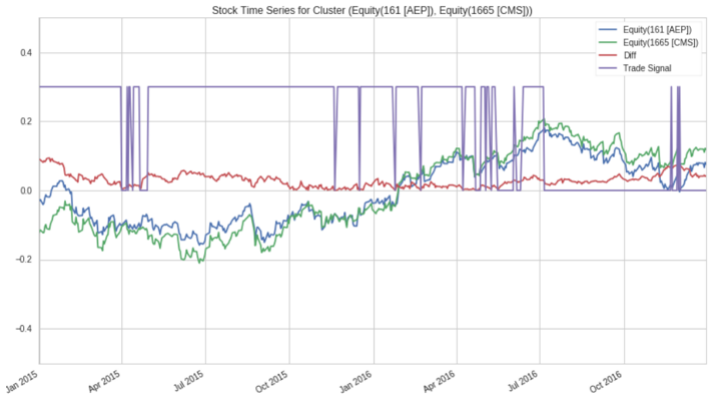
This gives a convenient way to evaluate a rough image of the potential performance of the strategy. We now proceed into describing the two trading strategies tested.

3.1 Strategy 1) Return Difference Velocity

In observing the standard patterns of pairs trade, we thought a good way to capture the movement would be based on a ratio of the differences changing by a certain amount in a given time frame. The intuition is that the pattern of mean-reversal is not determined by a static threshold, but rather a fluid one that allows the pattern to occur at varying scales. As such, we expect that the ratio will be able to capture both large and small actionable trades under the pairs trading logic.

-Target Label - **Return Difference halves in the next 80 trading days**
-Observe the difference between the log returns of the two equities in the pair
-If this difference halves in the next 80 trading days, then we consider the given day to be labeled as valid for entering a trading position.

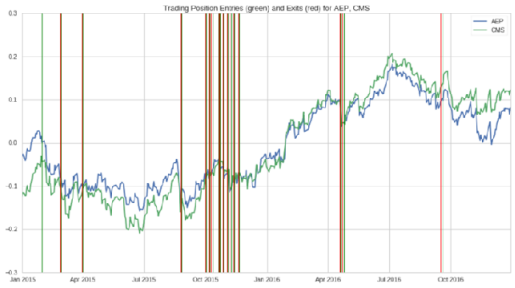
Now we decide on an appropriate feature. Since the label is based on movements of return differences, we thought it fit to use the velocity of that difference (derived as a feature), as it would be likely to contain information about potential future movements in the difference itself. We use a 20-day trailing window of return difference velocities as the data input for this model to adequately capture data from previous days. We show an outline of how the derived features and labels look for a random pair of stocks in our cointegrated dataset



We used an 80:20 split for training and testing data of our cointegrated stock dataset, and tested **SVM, Logistic Regression, and Decision Tree** classifiers to the task. The accuracy in matching our trading labels was as follows -
SVM - 77.67%
LR - 69.82%
DT - 66.26%

SVM provides the highest accuracy, so we proceed with that as the basis of our remaining testing. We apply our aformentioned algorithm to test the returns of trading strategies, which for this strategy functions as follows 1) Enter trade if model prediction is valid 2) Exit when difference halves from the initial entry position, or after 80 trading days have passed. We emerge with the following results. We also depict an example of our generated traing signals during the testing algorithm in one example of stocks (green lines are entries, red lines are exits, where an entry is buying the underperforming stock and selling the overperforming, and an exit is vice versa).

- **Trading Strategy Avg Returns** 1.0671
- **Buy and Hold Avg Returns** 1.2196
- **Risk-Free Returns** 1.0066
- **Percentage of Profitable Trades** 0.9071
- **Volatility of Returns** 0.04733
- **Worst 10 Trades Avg** -0.05806
- **Single Worst Trade** -0.1067
- **Sharpe Ratio** 1.4149
- **Simple Algorithmic Pairs Returns –** 1.0821



This strategy implementation manages to outperform the risk-free return rate, though it starkly underperforms the buy and hold return rate. We note however that the buy and hold return rate is abnormally high, perhaps being an anomalous characteristic of the favorable economic conditions of our time window. The strategy maintained a high percentage of profitable trades, at around 0.9071, with a healthily low volatility of returns. The sharpe ratio is quite passable despite the low average returns, which suggests that risk-adjusted performance may have some potential with further testing and modification. Despite this, the single worst trade (-0.1067) is quite severe for any automated trading strategy, so while on average the strategy seems quite safe, it seems to also harbor potential for rare sever downward shocks. Compared to the rudimentary algorithmic pairs returns, the strategy underperforms by around 1.5%. Overall, this strategy is not particularly profitable given its potential for heavy downsides in the worst case scenarios, and investors would appear to be able to do better following simple manual rules or, in favorable market conditions, buying and holding.

So what were the shortcomings? Two main ones were identified. 1) The signal sometimes generates signals for times when the price difference is extremely small - this leads to the trade position getting ”stuck” for a long time and exiting when the time period expires, often at a loss. 2) The signal is general and slow to act, meaning that there is enough time for overall drifts in the stock prices to outweigh the mean reversion of halving. In general, we also noticed that the trading label itself was likely over-eager to label valid trading entries, as the general frequency of valid trading entry days was dense to the point of meaninglessness. We seek to rectify these in the next iteration of the strategy’s development.

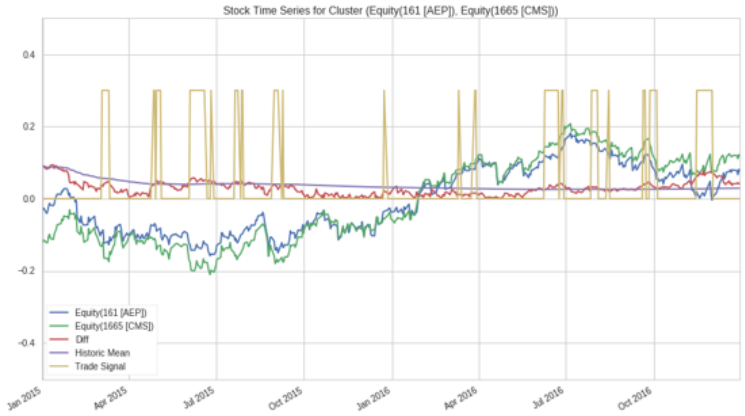
3.2 Strategy 2) Difference Velocity Moving Average Reversal

As per the last strategy, it would seem that a strategy based purely around the raw ratio of the trading metric is not capturing the full or precise picture of the phenomenon we want to trade around. We realized that the fundamental interest unique to pairs trading as a strategy basis were not focused on the strict ratios of differences, but rather on their relative velocities. As such, we wanted to conceive of a derived feature to capture the idea that at an optimal point in a pairs trade, the recent past features a rapid burgeoning of the price difference while the coming future features a rapid constriction of the price difference. A seemingly ideal metric for this is a moving average of the difference’s velocity.

-Target Label - **20-day trailing price difference velocity is positive while the 20-day ensuing price difference velocity is negative**
-Observe the difference between the log returns of the two entities to generate a price difference velocity metric
-Create 20 day moving average windows, both for the forward and backwards direction
-If at any point, the 20 day moving average towards the past is positive and the 20 day moving average towards the future, create a valid trading label.

Now we decide on appropriate features. Since the label is based on movements of price velocities, it is obvious to include the 20-day trailing price difference velocity as a feature. We also include the *acceleration* of price difference movements in another derived feature in the hopes that this

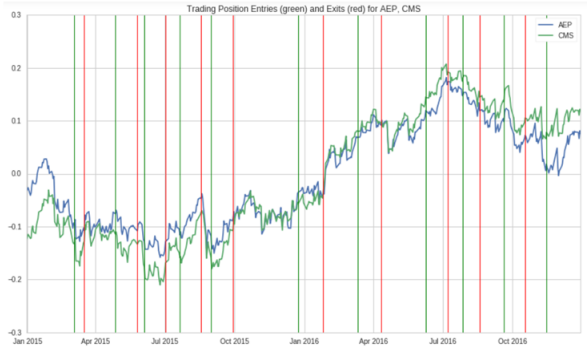
contains meaningful information on the propensity for a 20-day trailing price difference velocity to gradually go from positive to negative. We also include a historical mean difference data point to assist in the acceleration's calibration of this relation. We show an outline of how a few of the derived features and labels look for a random pair of stocks in our cointegrated dataset



Immediately, we see a sparser set of entry/exit labels, which is a good sign for the meaningful extraction of information from this algorithm. As with the previous algorithm, we apply SVM, Logistic Regression, and Decision Tree methods. The accuracy in matching our trading labels was as follows -
SVM - 86.05%
LR - 88.90%
DT - 91.90%

The decision tree classifier performs best, so we proceed with the ensuing analysis using it as the basis. We apply our aforementioned algorithm to test the returns of trading strategies, which for this strategy works as follows 1) Enter trade if prediction is valid 2) Exit if the 20-day trailing velocity becomes positive (a reversal from the 20-day forward velocity being negative in the target label). We emerge with the following results. We also depict an example of our generated traing signals during the testing algorithm in one example of stocks.

- **Trading Strategy Avg Returns** 1.114
- **Buy and Hold Avg Returns** 1.2196
- **Risk-Free Returns** 1.0066
- **Percentage of Profitable Trades** 0.915
- **Volatility of Returns** 0.04023
- **Worst 10 Trades Avg** -0.00515
- **Single Worst Trade** -0.00882
- **Sharpe Ratio** 2.54
- **Simple Algorithmic Pairs Returns –** 1.0821

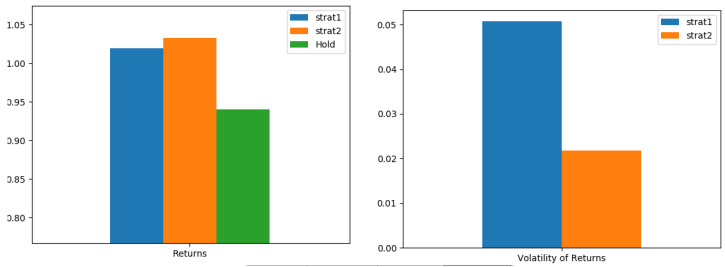


Immediately, we see many strong improvements over the previous trading strategy. The returns are much higher at 1.114, and while still below the buy and hold average returns, now significantly outperforms the prior strategy and the simple algorithmic pairs returns. The sharpe ratio indicates strong risk-adjusted performance, and the percentage of profitable trades is at a healthy 0.915. Interestingly, this strategy seems to do exceedingly well when it comes to avoiding bad trades, with the average of the 10 worst trades being only -0.00515, and the single worst trade being only -0.00882. This is practically a factor of 10 safer than the prior strategy in this regard, and could become one of the core strengths of this strategy if deployed.

Despite the strengths, there remain apparent areas to improve. In observing the sample graphs of trading patterns, while the trades are often profitable and safe, they often fail to maximize on the profit of holding on just a little longer. This implies that while the mechanism to enter trades may be doing relatively well, the mechanism to exit trades is too risk-adverse, and some slight modifications to its duration window could potentially lead to a greater return rate.

3.3 Comparison in less-favorable Market Conditions

-A defining characteristic of the chosen window was that the buy and hold strategies on average performed quite strongly. Then, it is important to what extent our strategies were affected by such influences. Do our strategies outperform buy and hold in less favorable market conditions? Do they perform profitably only because we test in favorable market conditions? We filter the testing range dates to create a relatively flat buy and hold average and observe the results from doing such. We choose a time window consisting of the first half of 2015, where the average buy and hold return would turn a 0.94 return at a loss.



	Buy and Hold	Strategy 1)	Strategy 2)
returns	0.94003	1.019	1.11375
vol returns		0.05071	0.0479

The results are certainly interesting, and demonstrate that both of our strategies actually outperform the buy and hold in relatively neutral market conditions. While the relative returns of our strategies compared to the favorable market condition period decreased, the fact that they maintain a robust lead in more ”standard” conditions certainly bodes well for their effectiveness. The second strategy outperformed the first fairly by a much smaller window in this margin than in the favorable market condition, which may indicate that it capitalizes on aspects of upwards trends better than flat ones. The second strategy however gave a much lower volatility on its returns, indicating its risk-adjusted performance is quite a bit stronger. The simple algorithmic pairs strategy returned **1.016**, indicating that both our algorithms outperformed it

4 Conclusion

Machine learning shows strong promise in being able to derive profitable trading strategies.The results, while tending to be safe with high percentages of profitable trades, do not generate staggering returns. However, they outperform buy and holds in standard market conditions, despite underperforming in favorable market conditions. Trading with machine learning applied to a velocity reversal indicator seems far superior to pair halving, with higher returns, safer maximum drawdowns, better risk-adjusted returns, and a higher percentage profitability. We note the incredibly low levels for maximum drawdown as well. Interesting areas of potential exploration include applications of optimization linear programs, since current labels focus on profitability, but not on degree of profitability. Using optimization objectives, one could theoretically create entirely machine-generated labels that are intended not only to simply be positively returning, but to also maximally so. Additionally, the use of networks commonly applied to time-series could also be explored. RNNs and CNNs (absent from sklearn, which was my main impediment to testing in the scope of this project) could likely derive more valuable information from trailing data than is capturable in a simple derived statistic like a moving average, and as such could make more informed inferences about the future. Additional studies could include modifications for general assumptions made, both about the financial world and the trade return testing algorithm. Key flawed assumptions to explore include – full entry into each trade, lack of transaction costs, dividend payments.

References

[1] Morningstar Quantopian Pipeline *Quantopian*. 2017. https://www.quantopian.com/posts/getting-pipeline-to-work-with-morningstar-earnings-ratios

[2] Van Der Have. Pairs Trading Using Machine Learning: An Empirical Study. *Erasmus University*. 2017. https://thesis.eur.nl/pub/41548

[3] Chen, Ren, Lu. Machine learning in Pairs Trading Strategies. *Stanford University*. 2012. http://cs229.stanford.edu/proj2012/ChenRenLu-MachineLearningInPairsTradingStrategies.pdf

[4] Jonathan Larkin *Pairs Trading Clustering with Machine Learning*. 2017. https://www.quantopian.com/posts/pairs-trading-with-machine-learning