Statistical Arbitrage by Pair Trading using Clustering and Machine Learning

Anonymous
Department of XXX
University of XXX
City, Province, Zipcode
[example@email.com]

Anonymous
Department of XXX
University of XXX
City, Province, Zipcode
[example@email.com]

Abstract

Pair trading is one of the most popular trading strategies since 1980 for finding statistical arbitrage opportunities in the stock market. The logic behind pairs trading is to trade pairs of stocks belonging to the same industry or having similar characteristics, such that their historical returns move together and are expected to continue to do so in the future. In this project, we investigate the application of machine learning methods to find statistical arbitrage opportunities in the stock market using pair trading strategy. Pairs are recognized using clustering methods, while trading signals are predicted by multiple supervised learning algorithms. Comparative analysis is carried out based on returns, Sharpe Ratio, and other performance indicators of different models.

1 Introduction

Pairs trading is an investment strategy of constructing a portfolio with matching stocks in terms of market risk factors with a long/buy position in the stock we are optimistic and a short/sell position in the stock we are pessimistic. The approach is designed to eliminate market risk and exploit temporary discrepancies in the relative returns of stocks.

The first step in constructing the portfolio is to identify pairs of stocks based on fundamental analysis, i.e. having common factors such as being in the same industry and having similar market capitalization.

The second step is to track return spread between each pair of stocks . A pair of cointegrated assets is selected, that are known to historically move close to each other and are expected to continue to do so. Under the assumption that the spread, defined as the difference in price between the paired assets, is mean-reverting, deviations from the mean can be exploited.

When the spread is abnormally wide and deviation from the mean reaches some pre-determined threshold, the outperforming stock is sold short, and the under-performing stock is purchased. As soon as the spread converges back to its mean, the investor liquidates both positions, resulting in a profit. Although this may sound intuitive and trivial, sophisticated machine learning techniques can be used at every step of the pairs trading process. The key to successful pairs trading is the ability to detect patterns in spreads and correctly identify when a spread has become abnormally large and is likely to converge back to it's mean.

Our approach to pairs trading will be to apply PCA for dimension reduction on a large set of features in order to ease computation of DBSCAN filter for clustering stocks. Afterwards, we use cointegration test to extract all possible combinations of stocks in each cluster that are within 5% significance level. For each candidate pair, under some predetermined threshold, we trade the pair whenever the absolute value of spread is greater than the threshold. We supplement trade decision-making with supervised learning by using neural network classifier to predict next-day spread of the pair. If the prediction is in favor of the spread mean-reverting, we execute the trade. The goal is to take bets that will result in statistical arbitrage.

2 Related Work and Novelties

The basis of our pairs trading algorithm begins with Ernest Chan's [2] implementation of a pairs trading strategy involving two pairs of stocks from the alternative energy sector: Abengoa (ABGB) and First Solar (FSLR), and China Sunergy (CSUN) and Ascent Solar Technologies (ASCI). The strategy assumes both pairs of stocks are cointegrated enough such that one can profit by simultaneously longing or shorting the spread according to a two-standard deviation threshold. We build upon this simple, yet effective strategy, by assuming we have no a prior knowledge of what pairs of stocks will be profitable in the future and allowing an unsupervised learning model to do the job of picking stock pairs for us. In this case, we find that the PCA and DBSCAN clustering approach by [6] to be an effective candidate model for generating cointegrated pairs of stocks based on common factors, such as market cap, industry, and financial health. The paper by [8] uses LSTM network for pairs spread forecast and the multilayer perceptron (MLP) network for stock signal prediction in order to decide whether a trade should be executed when an upper/lower threshold is crossed.

Lu, Parulekar, and Xu [7] instead supplement their stock data by assigning stock-ETF pairs a score based on Johansen's test for cointegration, which outperformed compared with PCA and K-means clustering methods. Then, the authors used LSTM to predict trading signals under 40-day trading window, compared its performance to one lag Auto-Regressive model AR(1). Have [4] investigated intra-sector ETF data in NYSE and also used cointegration for pair selection, then built the trading strategy based on Neural Network. To measure algorithm performance, they make use of financial indicators including Sharpe ratio, Sortino ratio, and Maximum-drawdown.

In our research, we seek to apply machine learning to supplement pairs trading. The novelties we explore are as follows,

1. Apply clustering methods to exclude unqualified candidate stocks from the universe, based on a set of hand-picked financial factors as features in the clustering process.
2. Implement machine learning techniques in an effort to predict the spread movement direction among all candidate pairs, rather than predicting the scale of movement.
3. Classification techniques will supplement our decision-making process, by supplying us with buy/sell signal when the spread crosses an upper or lower threshold.

3 Method and Approach

For constructing a Pairs Trading strategy, the first task is to find valid, eligible pairs which exhibit unconditional mean-reverting behavior. One simple approach is to iterate through all stock pairs in the universe of stocks, but this would not be computationally feasible. Instead, PCA can be used to compress the daily stock price data into several components, which will be regarded as price movement features for a stock. Then, we incorporate feature engineering for each stock by combining the price movement features generated by PCA with the features from economic prior knowledge. Afterwards, we apply DBSCAN clustering algorithm to cluster the stocks into different groups, and stock pairs with p-values less than 5% are selected from each of the groups.

Once the candidate pairs are identified, we apply multiple machine learning algorithms to predict the spread between each pair, in an effort to catch the trading signal whenever the spread exceeds some predefined threshold.

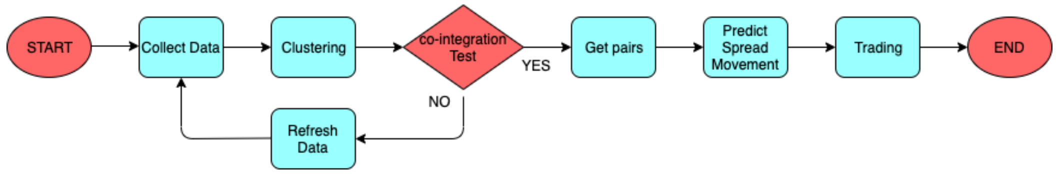


Figure 1: Flow Chart of the this project

The particular machine learning methods we incorporate in our analysis are discussed below.

3.1 PCA

Principal Component Analysis (PCA) is a mathematical procedure that transforms a large number of variables into a smaller number of uncorrelated variables called principal components. For this paper, PCA will reduce 500 daily stock prices to 50 variables while trying to keep as much variance as possible. Each of the resulting principal component can be seen as representing a risk factor, and the stocks will be clustered based on these components.

3.2 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996 [3]. It is a density-based clustering non-parametric algorithm which can be described in the following steps [9]:

(1) Find the points in the ϵ neighborhood of every point, and identify the core points with more than $\min P_{ts}$ neighbors, where $\min P_{ts}$ is a parameter to be tuned.

(2) Find the connected components of core points on the neighbor graph, ignoring all non-core points.

(3) Assign each non-core point to a nearby cluster if the cluster is an ϵ neighbor, otherwise assign it to noise.

Compared with K-Means, DBSCAN has advantages in our use case. Specifically, DBSCAN does not cluster all stocks,i.e. it leaves out stocks which do not neatly fit into a cluster, and the number of clusters does not need to be specified.

3.3 t-SNE

Once the data is clustered, we need a way to visualize the high dimensional data and its clusters into a two-dimensional graph. One approach to this visualization is using the nonlinear dimensionality technique known as t-Distributed Stochastic Neighbor Embedding (t-SNE) [1].

We denote the original high dimensional data set as X, with each data point as x_j, and denote the mapped (transferred) low dimensional data set as Y, with each map point as y_i. Then, let $|x_i - x_j|$, $|y_i - y_j|$ be the distances among the original data points and map data points respectively. To maintain the relative structure of the data, if two data points in the original space are close, the corresponding map data points also need to be close. We can now define the conditional probability of the data points as,

$$p_{j|i} = \frac{\exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-|x_i - x_k|^2/2\sigma_i^2)} \tag{1}$$

Equation 1 means that the probability distribution is constructed such that similar objects have a high probability of being picked, while non-similar objects have a low probability of being chosen. Then, we define the joint probabilities $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$, and define a similar matrix of map points $q_{ij} = \frac{\exp(|y_i-y_j|^2)}{\sum_{k\neq i}\exp(|y_i-y_k|^2)}$. The objective is to minimize the distance between the two probabilities. This process is done through the Kullback-Leiber divergence with a gradient descent,

$$KL(P||Q) = \sigma_{i,j} p_{ij} \log(\frac{p_i j}{q_i j}) \tag{2}$$

Equation 2 tells us that if two map points are distant to each other, while the corresponding data points are not, they will be drawn together. The process iterates until a final mapping is procured and equilibrium is attained.

3.4 Gradient Boosting

Gradient boosting is typically used with decision trees of a fixed size as base learners. Generic gradient boosting at the m-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, ..., R_{J_m m}$ and predicts a constant value in each region. We specify two hypothesis spaces of Gradient Boosting[10]:

- base hypothesis space: $\mathcal{H}$, which consists of real-valued functions
- combined hypothesis space: $\mathcal{F}_M = \left\{ \sum_{m=1}^{M} v_m h_m(x) | v_m \in R, h_m \in \mathcal{H} \right\}_{m=1,...,M}$

Using the indicator notation, the output of $h_m(x)$ for input x can be written as

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x),$$

where b_{jm} is the value predicted in the region R_{jm}. Given data $\mathcal{D} = ((x_1, y_1), ..., (x_n, y_n))$, fitting a gradient boosting model is equivalent to choosing $v_1, ..., v_M \in \mathbb{R}$ which represent our lags of return changes, and $h_1, ..., h_M \in \mathcal{H}$, which represents the decision tree basis functions. The ERM objective function is defined as

$$J(v_1, ..., v_M, h_1, ..., h_M) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, h_1(x), ..., h_M(x)).$$

Finally, we minimize the objective function by FSAM, an iterative optimization algorithm for fitting adaptive basis function models.

3.5 Random Forest

Before presenting the random forest, we first introduce the bagging algorithm. Suppose there are B independent training sets from the same distribution. The learning algorithm produces B decision functions: $f_1(x), f_2(x), ..., f_B(x)$, then the average prediction function is defined as $f_{\text{avg}} = \frac{1}{B} \sum_{b=1}^{B} f_b$. In practice, however, the training sets are usually not independent, so bootstrapping is used to generate samples $D^1, ..., D^B$ from original data D. The bagged decision function is then a combination of $\{f_b(x)\}_{b=1}^{B}$, i.e.

$$f_{\text{avg}} = \text{Combine}(f_1(x), f_2(x), ..., f_B(x))$$

The main idea behind random forest is to reduce the variance of the decision tree models without incresing bias by using bagging. Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the prediction accuracy and control over-fitting.

3.6 LSTM

RNN is usually used for training time series data, but a significant shortcoming of RNNs is their difficulty in handling “long-term dependencies.”. Hochreiter and Schmidhuber [5] introduce LSTM in 1997, a model that is explicitly designed to avoid this long-term dependency problem.

The key to LSTM is the cell state, which is the main output. Next, h_t in this model can be seen as filtered version of the cell state. The first step for LSTM is the “forget gate layer”, which determines the old information that should be kept. It is built by f_t as, shown in Figure 2, where f_t is defined as,

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

The next step is to determine the new information for the cell state. This part is built up by $i(t)$ and $\tilde{C}(t)$,

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

The new cell state is obtained by,

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t$$

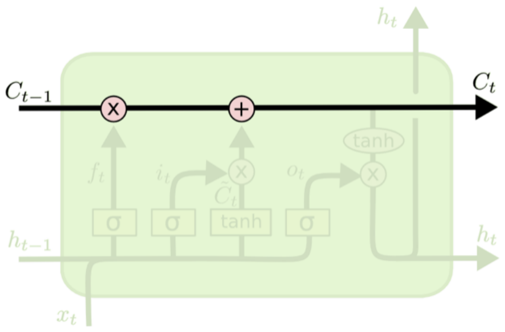


Figure 2: Structure of the hidden layers of LSTM : The yellow rectangle represent nonlinear functions where the new input x_t and output from the previous hidden layer h_{t-1} are processed through. The red circles containing either a multiplication or addition sign represent the corresponding linear operation. Furthermore, the variable C_t denotes the cell state at time t.[4]
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The last step is to generate the new hidden layer

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

In particular, we use LSTM as a time-series binary classifier, so the objective function is set to be the log-loss. Also, the more efficient ADAM algorithm is used for optimization.

3.7 Trading Methodology

In the following, we assume \$10 million initial capital, the size of each bet will be equal to a tenth of our portfolio, and the maximum number of pairs that our portfolio holds a position in will be 10. Our goal is to construct a pair trading algorithm that could be used in practice. Below we describe the criteria for executing trades and generating profit.

3.7.1 Spread

Let S_t^i be the price of asset i at time t. Define $y_t^i := \log\left(S_t^i\right)$. The cointegration relation between i and j is given by,

$$y_t^i = \alpha + \beta y_t^j + \epsilon_t, \qquad \epsilon_t \sim N\left(0, \sigma^2\right)$$

Where α and β are estimated by OLS. Thus the spread is calculated as,

$$y_t^i - \beta y_t^j = \alpha + \epsilon_t \tag{3}$$

If cointegration holds, then the spread is stationary. Furthermore, α can be interpreted as the mean level of the spread.

3.7.2 Thresholds

We set a two-standard deviation threshold for the z-score, computed by,

$$z_t = \frac{s_t - \mu_{s,20}}{\sigma_{s,20}}$$

where $\mu_{s,20}$ is the average of the spread over a 20-day rolling window and $\sigma_{s,20}$ is the standard deviation of the spread. Then we would enter a long position on the spread if $z_t < -2$ and short position if $z_t > 2$. To be more precise, entering a long position means we anticipate the value of a particular asset to rise and entering a short position means we anticipate the value to drop.

3.7.3 Bet Amount

Denote π_t^j as the number of shares that we buy (sell) of asset j for every share of asset i that we sell (buy), define π_t^j is defined as, $\pi_t^j := \frac{S_t^i}{S_t^j}$. In other words, to long the spread 3, our purchase amount is given by $1 * S_t^i - \pi_t^j S_t^j = 0$. If we short the spread, then our purchase amount is given by $\pi_t^j S_t^j - 1 * S_t^i = 0$ Thus, the overall value of each bet is 0 regardless of long or short.

3.7.4 Open Position Criteria

A trade will be triggered in the following scenarios:

1. Baseline method: When the spread between two stocks reaches lower (upper) threshold, we take a long (short) position in the spread.

2. Alternative method: when Baseline method is met, we use the prediction of our neural network with logistic loss classification model to have the final call on the trading decision. If both stocks show opposite signals (i.e. one stock shows long position and the other one short position) and the trade signals are in line with the position we would take at the boundary, then we execute the trade.

3.7.5 Stop Loss Criteria

If an upper (lower) threshold is crossed by our spread before reverting back to the mean spread after opening a position, then we close the position of our trade and incur a loss. In this case we avoid a rampant loss due to the spread never reverting back to the mean.

3.7.6 Close Position Criteria

We will close the position of our trade, when the spread crosses the mean-level of spreads, provided that the stop loss was not invoked. For a graphical illustration of this trading methodology, see Figure 3 from Ospino Lopez 2019 [8]. In Figure 4 is a visual depiction of our trading strategy.

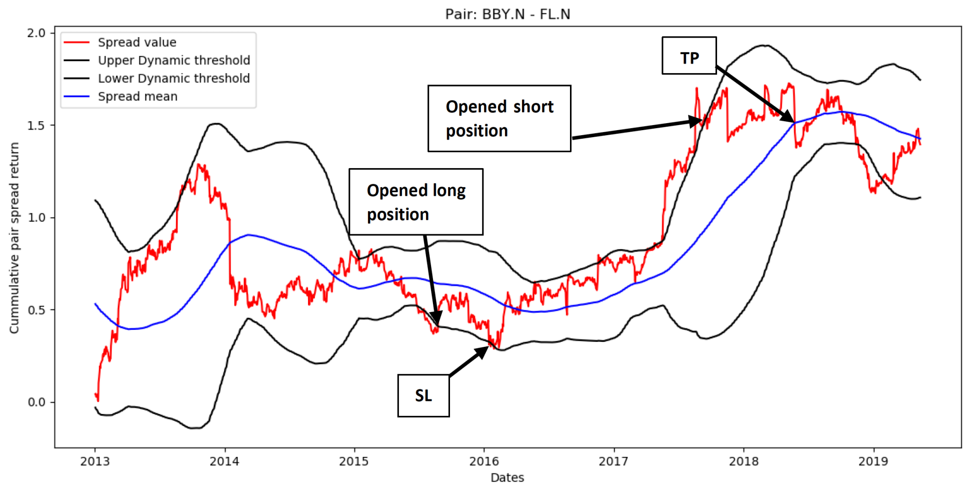


Figure 3: Graphical illustration trading the spread: Open long/buy position after the spread has hit the lower threshold from down-up. SL (Stop Loss) is triggered when the lower threshold is hit for the third time, before the mean was reached. Open short/sell position after the spread has hit the upper threshold from up-down. TP (take profit) is triggered when the mean spread is reached.

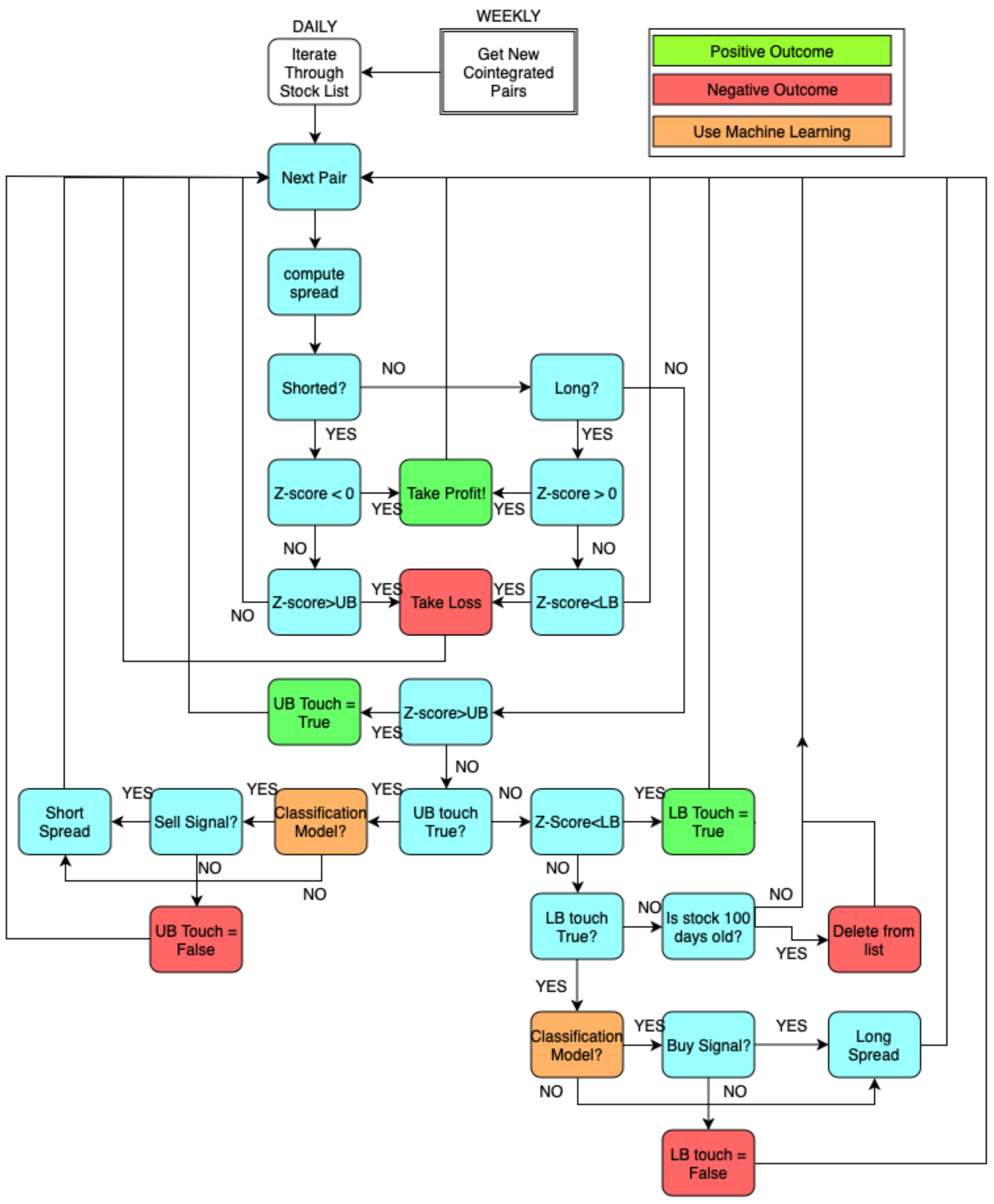


Figure 4: Flow Chart illustrating the logic behind our trading strategy.

4 Experiments and Results

4.1 Pair Discovering

We start by specifying the stock universe constrained to the S&P 500 index components and US1500 universe stocks, which are considered to be large and liquid enough. Next, we combine historical daily pricing data with fundamental and industry/sector data in an effort to classify stocks into clusters, after which co-integration test is done on stocks within clusters with the goal of finding pairs with strong mean-reverting relationships.

To illustrate the pair selection process, we consider daily data with time horizon, Nov 2017 to Nov 2019. In the stock clustering process, we take daily prices as features, which results in approximately 500 dimensions. PCA helps cope with the curse of dimensionality. For a full list of additional features, see Appendix.

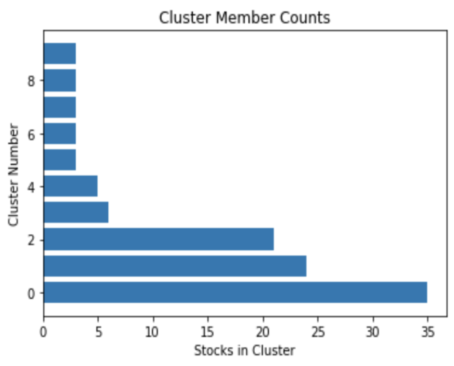
Afterwards, we use the DBSCAN unsupervised clustering algorithm available in `scikit-learn`. The clustering algorithm will output sensible candidate pairs.

4.2 Data Collection

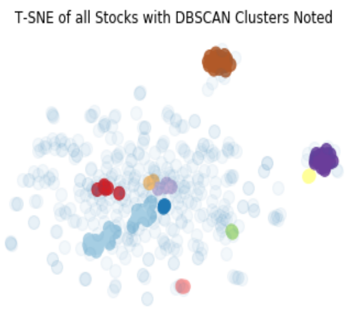
We download the S&P500 component stocks data from FactSet and US1500 Universe stock data from Quantopian. For our model training and local experiments, we use the S&P500 stock data, while the US1500 stocks data is used for simulating a real-world trading application on the Quantopian backtesting platform.

4.2.1 Data Processing

After applying DBSCAN to cluster the stocks based on the 50 principal, we were able to find 10 clusters.



(a) cluster hist



(b) DBSCAN clusters

Figure 5: Cluster results

We visualize the discovered pairs to help us gain confidence that the DBSCAN output is sensible; i.e., we want to see that T-SNE and DBSCAN both find our clusters. From Figure 5b, we see that the stocks that are clustered together by the DBSCAN are also close with each other from the T-SNE plot.

4.3 Quantopian Backtesting Simulator

A backtest is a simulation to determine how our algorithm would have behaved in the past by feeding it historical data. Backtesting is used by hedge funds and other researchers to test strategies before real capital is applied. We use Quantopian backtesting platform in this paper for backtesting. For a summary on the benefits of Quantopian, see Appendix.

4.4 The evaluation methodology

For this part, we will use the supervised machine learning models to predict the stock market, which will be used for the pair trading in the final part of the paper. In this paper, Gradient Boosting Machine, Random Forest, Logistic Regressionn and LSTM are used for fitting the stocks price difference of the two stocks in pair. The data before 2019-08-11 will be used for train the model and the data after 2019-08-11 will be used for test the model. The evaluation metrics for the fitting is prediction accuracy in this paper.

4.4.1 Performance Metrics

1. **Sharpe Ratio**: $\text{SR}_i = \frac{R_i - R_f}{\sigma_i}$. Here, R_i denotes the return of a stock pair, and R_f the risk-free rate. The standard deviation of a pair is denoted with σ_i . The Sharpe ratio helps us determine whether a portfolio's excess returns are due to investment decisions that do not incur too much risk. A Sharpe ratio greater than one is considered acceptable.

2. **Total Return**: $\text{Ret}_t = \frac{V_t}{V_0} - 1$, where V_t is the value of our portfolio at time t and V_0 is the initial value of our portfolio

3. **Volatility**: Volatility $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \bar{r})}$. Defined as the standard deviation of daily returns. The volatility serves as a measure of uncertainty of returns.

4. **Max Drawdown**: Let S_t be the value of an asset at time t. Its running maximum M_t is given by: $M_t = \max_{u\in[0,t]} S_u$. The Maximum drawdown, MaxDD_t, is defined as the largest drop of the asset price from the running maximum up to time t, $\text{MaxDD}_t = \max_{u\in[0,t]}(M_u - S_u)$ Ideally max drawdown should be smaller than the total return.

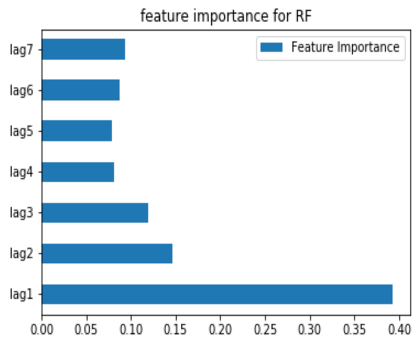
4.5 Results

Model	NDAQ-US	ICE-US	CME-US
Baseline	0.502	0.506	0.513
Random Forest	0.63	0.67	0.58
Gradient Boosting	0.61	0.67	0.63
Logistic Regression	**0.69**	**0.76**	**0.70**
LSTM	0.56	0.53	0.61

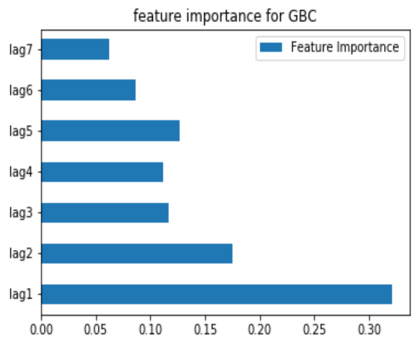
Table 1: Accuracy of spread movement direction prediction

Table 1 in Appendix, represents the prediction results of the models we used. From the result we can see that the simplest model, Logistic Regression performs best against the other ensemble and deep learning models. A main reason may be that complex models are more likely to cause over-fitting.

Next, we visualize the feature importance and find that the most recent lag variable (return difference) is most important for determining the future direction of the stock return.



(a) RF feature importance



(b) GBC feature importance

Figure 6: Feature importance for lags

The results of our backtest (11/30/2014 - 11/28/2019) are shown in Figure 7 and performance metrics in Table 2. Indeed in the alternative model, we achieved favorable total returns of 10.62% with a small volatility over this five-year period. Given that volatility is small and Sharpe Ratio is close to one, this implies that the excess return of our strategy is close to the risk-free return. In other words, one would benefit similarly by investing in US bonds instead. On the other hand, the backtest of the baseline model in Figure 8 shows that the pair trading strategy falls apart. Clearly, we incur a loss and the maximum drawdown is more severe than the loss from total returns. Given that the only difference between the two models is the implementation of the logistic classifier, we conclude that the classification model is effective and makes the pairs trading strategy profitable.

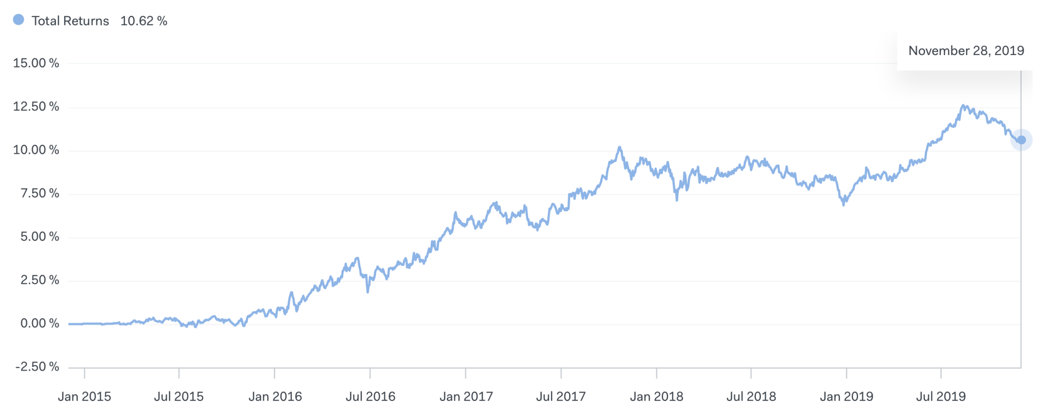


Figure 7: Backtest of pair-trade algorithm with classifier from 11/30/2014 - 11/28/2019.

Metrics	Baseline	Alternative
Total Returns	-7.55%	10.62%
Sharpe Ratio	-0.86	1.07
Max Drawdown	-10.13%	-3.06%
Volatility	0.02	0.02

Table 2: Performance Metrics from backtests

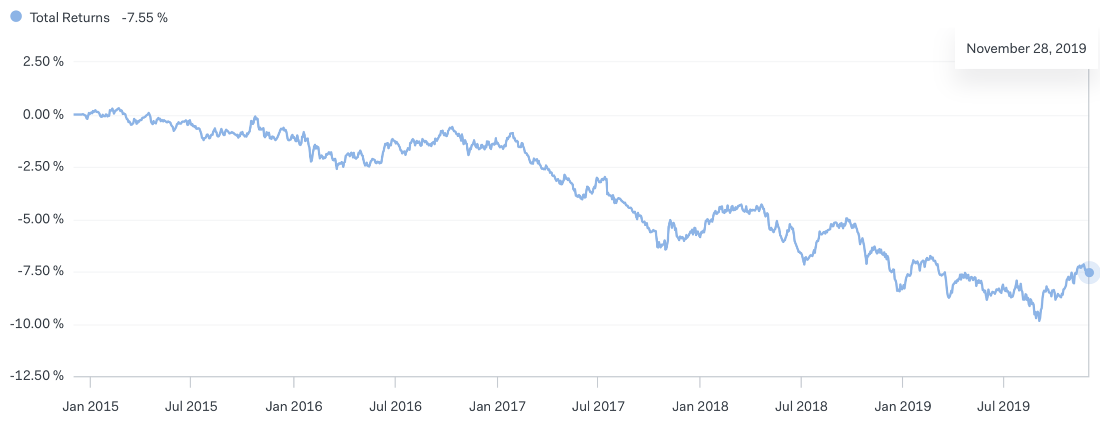


Figure 8: Backtest of pair-trade algorithm without classifier (Baseline model) from 11/30/2014 - 11/28/2019.

4.6 Link to Code and Data

Source to Github Repo: https://anonymous.4open.science/repository/cc3047bb-b0e1-4442-82f8-43501909704a/Code/

5 Conclusions

In our research, we conclude that pair trading is still a feasible trading strategy, but only when machine learning methods are incorporated. Additionally, we gained experience on training neural networks, dimensionality reduction, and supervised learning on time-series data. In future works, we can explore other neural networks, with different number of layers, neurons, and learning parameters, which may improve results.

References

[1] Håkon Andersen and Håkon Tronvoll. "Statistical arbitrage trading with implementation of machine learning: an empirical analysis of pairs trading on the Norwegian stock market". MA thesis. 2018.

[2] Ernie Chan. *Algorithmic Trading: Winning Strategies and Their Rationale*. 1st. Wiley Publishing, 2013. ISBN: 1118460146, 9781118460146.

[3] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *KDD*. Ed. by Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad. AAAI Press, 1996, pp. 226–231. ISBN: 1-57735-004-9. URL: http://dblp.uni-trier.de/db/conf/kdd/kdd96.html#EsterKSX96.

[4] R.W.J. van der Have. "Pairs Trading Using Machine Learning: An Empirical Study". In: 2018.

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

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

[7] Anran Lu, Atharva Parulekar, and Huanzhong Xu. *Cluster-Based Statistical Arbitrage Strategy*. URL: https://www.readkong.com/page/cluster-based-statistical-arbitrage-strategy-2644892.

[8] Jose Fernando Ospino López. "Improving Pairs Trading Using Neural Network Techniques and Fundamental Ratios". PhD thesis. 2019. URL: https://www.uv.es/bfc/TFM2019/JoseFernando_Ospino.

[9] Erich Schubert et al. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". In: *ACM Trans. Database Syst.* 42.3 (July 2017), 19:1–19:21. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: http://doi.acm.org/10.1145/3068335.

[10] Wikipedia contributors. *Gradient boosting — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-December-2019]. 2019. URL: https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=922411214.

A Appendix

A.0.1 Features of PCA

1. **Market Cap**: Price * Total SharesOutstanding
2. **Industry**: Industries are mapped into one of 69 industry groups based on their common operational characteristics.
3. **Financial Health**: Financial health score is estimated by distance to default, which is the value of the firm's liabilities obtained from the firm's latest balance sheet and incorporated into the model. We then rank the calculated distance to default and award 10% of the universe A's, 20% B's, 40% C's, 20% D's, and 10% F's. Morningstar calculates this figure in-house on a daily basis.
4. **Book Value Per Share**: Common Shareholder?s Equity / Diluted Shares Outstanding
5. **Value Score**: A high value score indicates that a stock?s price is relatively low, given the anticipated per-sharing earnings, book value, revenues, cash flow, and dividends that the stock provides to investors. A high price relative to these measures indicates that a stock?s value orientation is weak, but it does not necessarily mean that the stock is growth-oriented.
6. **Net Margin**: Net Income / Revenue
7. **Sector**: Restrict to the following sectors,

- Basic Materials
- Consumer Cyclical
- Financial Services
- Real Estate
- Consumer Defensive
- Healthcare
- Utilities
- Communication Services
- Energy
- Industrials
- Healthcare

A.0.2 Tables

Tuning parameters	NDAQ-US	ICE-US	CME-US
# estimators	100	100	100
criterion	entropy	entropy	gini
max depth	2	3	3

Table 3: Random Forest parameters

Tuning parameters	NDAQ-US	ICE-US	CME-US
# estimators	150	50	150
subsample rate	0.5	0.7	0.6
max depth	3	3	3

Table 4: Gradient Boosting parameters

Tuning parameters	NDAQ-US	ICE-US	CME-US
optimizer	LBFGS	LBFGS	LBFGS
fit intercept	True	False	True
inverse of regularization strength	0.8	0.7	0.7

Table 5: Logistic Regression parameters

Tuning parameters	NDAQ-US	ICE-US	CME-US
loss	binary cross entropy	binary cross entropy	binary cross entropy
activation	sigmoid	sigmoid	sigmoid

Table 6: LSTM parameters

A.0.3 Quantopian Summary

Quantopian can account for nuances in trading such as:

1. Stock splits, which involve a change in the total number of shares outstanding in an asset
2. Dividends, distribution of reward from a portion of the company's earnings and is paid to its shareholders.
3. Mergers and acquisitions, when companies merge or one is acquired
4. Our trades move the market, the very act of trading affects price. Historical pricing data does not include our trades and is therefore not an accurate representation of the price we would get if we were trading.
5. Transaction costs
6. Trade timing, trade at market open vs market close
7. Large allocations of money that is worth more than the stock of a small company is not allowed.

Another major advantage of Quantopian stock price data is that we can trust it is free of survivorship bias. A common example is using present day's list of S&P 500 stocks as our candidates for model training and backtesting. Survivorship bias is present since we use later lists of stocks in our data as the basis for historical evaluations over earlier periods.