Logic and Computations behind Algorithmic Trading System: From Moving Average to Dynamic Random Forest Triple Barrier Algorithm

Zihan Guo
Carnegie Mellon University
zihang@andrew.cmu.edu

Kun Zhang (Faculty Advisor)
Carnegie Mellon University
kunz1@cmu.edu

Abstract

Cryptocurrency is a virtual currency that does not rely on any centralized organization. It is built on blockchain, a decentralized database system, perhaps one of the most controversial inventions in the 21st century not only because of its potential to fundamentally change the economic structure but also because of its intricate relationship with Artificial Intelligence. One of the most utilized approach in Artificial Intelligence, Machine Learning, relies significantly on large amount of data. Data needs to be stored somewhere in the database either centralized or decentralized. Therefore, there are great expectations on the interactions between machine learning and blockchain technologies. Our focus only address one such interaction from an algorithmic trading perspective: how well does machine learning algorithm perform in forecasting cryptocurrency's trends and movements. We have collected real data from live order books since February 2018 and analyzed various machine learning model's performances in forecasting. In additions, we have outlined the steps of constructing algorithmic cryptocurrency trading system to provide some practical guides to those who are interested. We have found the most challenging part of constructing such trading system is building statistically meaningful evaluations. At the end, we will build a winning algorithmic strategy using random forest and triple barrier method [1] to outperform others.

1 Introduction

1.1 Motivation

Machine Learning has proved successful across industries. Financial Machine Learning, especially in algorithmic trading field, has advanced with growth of natural language processing and easily accessible packages like Google's Tensor Flow. However, there are few literacy about how these algorithmic system are being made and what features/alphas hedge funds are using to make powerful predictions better than human experts. There are even fewer publications about algorithmic trading of cryptocurrency like Bitcoin and Etherum because first Bitcoin was not created until 2009 and the first academic journal about cryptocurrency and blockchain, LEDGER, was established only on September 2015, not even three years ago. Yet, blockchain technology does reveal its potential to revolutionize the entire economic structure because of its transparency, peer to peer nature.

Therefore, it is of great importance for quantitative researchers and data scientists to look at properties of crytocurrency's markets to understand it better and hopefully this paper will inspire more researchers to build and share more complicated algorithmic trading strategies and algorithms and discuss new findings about cryptocurrency's market's properties. The insights of the market will not only help economists better understand blockchain and cryptocurrency market but also help regulators to understand and monitor malicious market behaviors.

Hence, there are mainly three motivations: the first motivation is to illuminate the dark path in building a machine learning driven algorithmic trading system on crptocurrency. The second motivation is to gain a clarity of mind: understand the cryptocurrency market better and recognize some obstacles and solutions when building such a system. The third and last motivation is to compare and examine the complexity and difficulties of machine learning application in trading and investment. On one hand, we have quantitative researchers using beautiful and elegant mathematical formulas to construct features that provide little empirical results. On the other hand, we have quantitative traders who construct features that are highly over-fitted and statistically unexamined. It is of our hope, in our last motivation, to provide both statistical and meaningful features that can make investment no longer gamble but based on logic and computation.

1.2 Philosophical Point of View

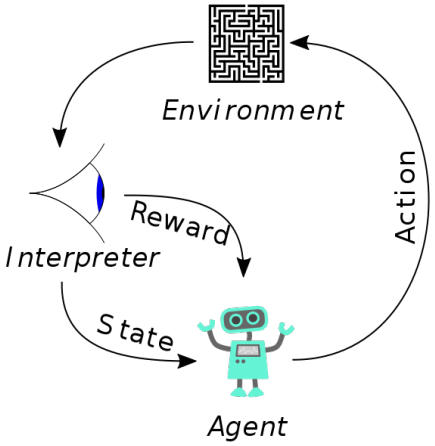


Figure 1: Illustration of Reinforcement Learning Flow

From a philosophical point of view, an algorithmic trading bot is an intelligent agent who can learn statistical models from vast amount of historical data. We hope that this intelligent mechanism is able to use such learning from the past to make some predictions about the future. Machine Learning has three major branches: supervised machine learning, unsupervised machine, reinforcement learning. Each branch has its own focus. Supervised Machine Learning focuses on learning the function $f(x)$ with data-sets that consisted with X and Y. Unsupervised Machine Learning focuses on clustering data X into different groups. Reinforcement Learning, however, is more dynamic because the agent learns by acting and interpreting. Similarly to the general picture about reinforcement learning above, a trading bot would initially place bid and ask at random to cause a small change of the market environment; then, the agent observes the environment and interprets the state changes in the market and concludes a reward to adjust its strategy. However, with small amount of trading size, the market's response might be minimal. The key assumption in the reinforcement learning strategy is that the action may causes a change in the state of the environment, just like placing a new piece on the Go board or move Knight on Chess-board. However, this assumption might not be fulfilled in our context when our trading size is small and does not possess market making ability.

1.3 Trading System Overview

A successful implementation of an algorithmic trading system should be a duet between development phase and deployment phase (live testing). Because no strategy, not moving average cross-over nor relative strength indicators and certainly not a fixed machine learning model, can always be a winning strategy. It is similar to martial arts where there are a set of rules to follow but when combating in practice winners are those who can quickly adjust to the environment. Similarly, when building a winning algorithmic trading system, one should avoid developing one single strategy. Instead, constructing a monitoring program that constantly examines the current most valid strategy is of great importance for both risk management and dynamically adjusting to the market's environment[3].

To illustrate our diagram, we will first describe each component below:

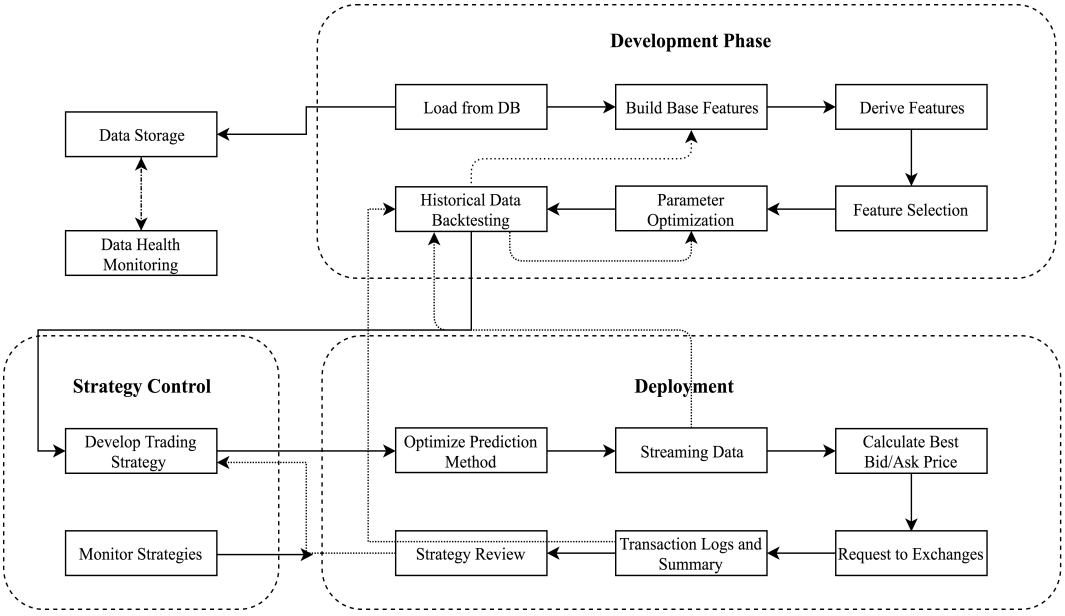


Figure 2: High-level overview of interactions among each components of an algorithmic cryptocurrency trading system[3][4]

Data Storage We store order-book for the most recent 20 bid and ask using cloud storage service. Earlier data always come before later data and each row has time-stamp as its 0th column.

Data Health Monitoring A crucial step in making sure that we are not recording ill-formatted or NA data in our data file. The program should periodically examine if the data is healthy and sending reports to supervisors on its status.

Load from Data Query data from the cloud storage and load it into the program. These data will then be used to construct features.

Build Base Features To generate predictors, we analyze the correlations between each feature and our labels. Here, our labels are Bitcoin's Mid-price which is calculated based on best bid and best ask prices.

Derive Features There are generic ways of generating more features for modeling. Here, we smooth some of our predictors to remove noises and make patterns stand out. These features combining with the base features will then pass into the feature selection.

Feature Selection There are numerous technique for feature selection and we use GINI score from random forest algorithm to examine how important a feature is. In addition, Granger Causality can be used to include only granger-causal variables in the predictor set.

Parameter Optimization Optimize parameters to achieve optimal performance using historical back-testing.

Historical Data Back-testing Build evaluations indicators such as return rate, max gain and loss to examine how well the model performs. Bactesting is one of the most important step in the whole system because it tests how well a model perform without investing real money. Based on the results of back-testing, we might jump back to feature development or parameter optimization.

Develop Trading Strategy Accurate prediction about the future is one thing, while developing a trading strategy using the forecasting model is another. A trading strategy involve entry and exit strategy which are a set of logic that decides when to buy and when to sell.

Optimize Prediction Method Prediction methods should have low latency and it should uses the same features during the development phase. In addition, prediction method incorporates the strategy logic in its implementation.

Streaming Data We keep a copy of live data locally in the program's memory.

Calculate Best Bid/Ask Price There are two types of transmission fee: maker fee and taker fee. A maker adds liquidity by placing a a limit order that will be satisfied only some time later. A taker remove liquidity by placing a market order that are executed immediately. **Request to Exchange** Upon deciding the best bid and ask price, the trading bot sends requests to the exchange to execute trading actions with appropriate stop loss and stop limit.

Transaction Logs and Summary Every order needs to be recorded and summarized into simple statistics such as current return rate, max and min gain and loss, trading frequency, market volatility, number of winning trades and losing trades.

Strategy Review Strategy Review is related to **strategies monitoring**. The trading bot keeps numerous trading strategies ready at the background and are constantly checking which strategy performs the best. Each strategy is given a score. The trading bot will use the best set of strategies for the next cycle.

1.4 Difference between forecasting and strategy

One common misconception about building an algorithmic trading system is thinking that predicting the future price is the only objective. However, in practice, knowing the next second's price is of little importance. First, we need to understand how these exchanges work. Exchange serves as a medium between buyers and sellers where buyers place bid at a lower value than the asks. However, these orders all have their own volume, it is possible to place a limit order that does not pass through until a much later time as the market moves quickly. Second, programs take time to run and there is always a latency between what we expected and what we traded. Third, exchanges often charge a transmission fee to participants of the markets. All of these factors play a role in quantitative trading. Therefore, simply knowing the next second's price is not enough to profit.

2 Background

2.1 Literature Review

The nature of any financial trading is its algorithmic system. What lies at the core of an algorithmic system is its sets of well tested strategies. Because of these strategies' worth to the strategy developers and to the firm, there are little literacy online or in prints that explain up-to-date strategies in details. However, there are numerous online blogs, videos and prints that touch on building a forecasting algorithm, constructing an algorithmic trading system and using advances in machine learning to produce profitable strategies with limited details. We include three in depth literacy review below.

2.2 General Trading Strategies

2.2.1 Mean Reverting Strategy

Mean-reverting strategy is one of the most prevalent trading strategies in financial trading. Dr. Ernest P. Chan commented in his book that "trading strategies can be profitable only if securities prices are either mean-reverting or trending"[2]. However, there are many limitations of mean-reverting strategy. First, competition among traders decreases the opportunity for mean-reverting trading. Second, selecting a time window for computing mean can be tricky. How do we select a time window such that we are not over-fitting historical data depends on rigorous back-testing techniques. Back-testing mean-reverting strategy is prune to error as well. On one hand, many databases contain error data either artificially or by mistake, while on the other hand, survivorship bias might inflate backtesting performance: actions such as acquisition and bankruptcy are rare event. Hence, the sparse and often zero available records of such events could distort the evaluative results[2].

2.2.2 Momentum Strategy

Momentum strategy, on the other hand, is news driven. News incorporates sensitive information about certain events that might have high correlation with directions of price. Chan pointed out that one of the most noticeable momentum strategy is post earning announcement drift which uses earning results to decide if one should buy or sell post earning. The same strategy is not applicable to cryptocurrency like Bitcoin[2]. However, a pivoting version of the strategy might be possible. Cryptocurrency users'

primary social network are reddit and twitter. One approach could be monitoring and computing the semantics score of certain internet celebrities on reddit and twitter when they publish messages to the public, and decides buying and selling time based on these scores.

2.2.3 GMMA Strategy

GMMA is an abbreviation for Guppy Multiple Moving Average Indicator. At its core, it uses two sets of time-horizon based exponentially moving averages. Each set contain 6 different time horizons. One set represents short-term, while the other one long-term. A simple strategy from GMMA indicator looks at the crossover between the sum of six short-term exponentially moving averages and the sum of six long-term exponentially moving averages. However, as Selby commented[15], GMMA works best in long term market tradings. Also, the art of choosing each sets of time-horizon are prune to over-fitting and requires careful backtesting[14].

2.3 Machine Learning Strategies

2.3.1 Data Mining for Regime Switching Prediction

Regime Switching describes when a bullish market reverses direction into a bearish market or when a bearish market reverse into a bullish market. For years, quantitative researchers have been searching for indicators that can predict when could such fulcrum point occurs. One approach, according to Ernest Chan, is to use data mining algorithms to look over massive number of indicators and select ones that might best predict the switch. Furthermore, such machine learning approach could be enhanced when confirming the price moves with the momentum strategy by looking into events and relating semantics score[2].

2.3.2 Supervised Learning with Triple Barrier Labeling Method

We can't build a supervised machine learning without labels. On one hand, we can construct the label as the mid-price calculated using the best bid and ask prices. However, such decision forces us to develop a strategy later. On the other hand, we can build labels that are entry and exit strategies so our learns how to trade directly. In addition, the traditional labeling method should be avoided because traditional method uses fixed-time horizon, which looks like the following[1]:

$$y = \begin{cases} -1 & r_{t_0,t_{0+h}} < r_* \\ 0 & |r_{t_0,t_{0+h}}| \leq r_* \\ 1 & r_{t_0,t_{0+h}} > r_* \end{cases}$$

where r_* is a fixed threshold regardless of observed volatility and

$$r_{t_0,t_{0+h}} = \frac{P_{t_{0+h}}}{P_{t_0}} - 1$$

is not only path independent but also uses the same time horizon h. The proposed solution is as the following:

1. Compute dynamic thresholds by updating daily volatility using exponentially moving average.
2. Construct horizontal barriers using the computed dynamic threshold.
3. Construct vertical barrier of length h and use it as an expiration limit.
4. For each time-stamp, we look at its h time steps ahead. For such path, we label the corresponding row based on which barrier it touches first. If it touches the upper limit first, we give it a 1; if it touches the lower limit first, we give it a -1; if it touches neither upper nor lower limit but hits the vertical expiration limit first, we give it a 0.
5. Based on the consecutive 3 label outputs, we decide buying and selling signals.
6. End.

We plot a red line at 0.3 to highlight when the rolling standard deviation of return rate exceeds 0.3%, which is the highest cost of maker transmission fee on Bitfinex exchange. It means that we will pay

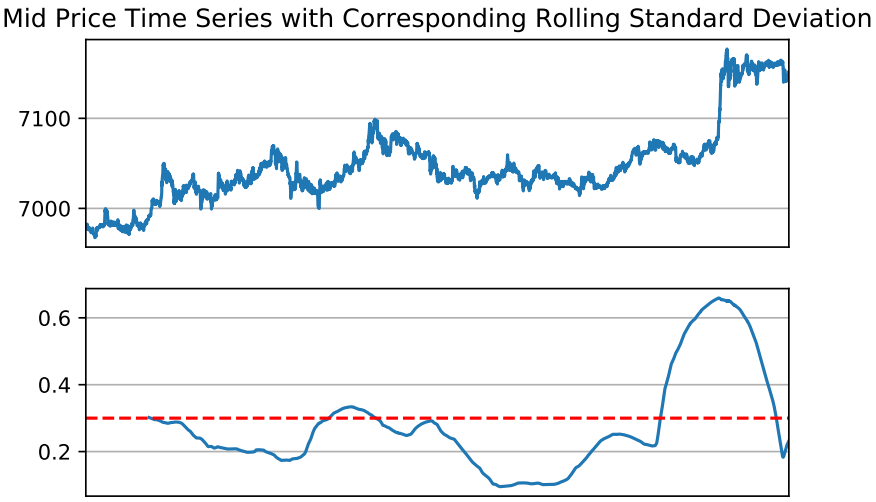


Figure 3: Visualization of rolling standard deviation of mid-price of Bitcoin between 2018-04-02-21:42:46 and 2018-04-04-18:20:26

special attention for the time when the volatility is higher than usual because those are often times with abrupt market changes. By utilizing a dynamic threshold, we should expect the bot to learn market's volatility as well.

3 Algorithm

3.1 Motivation

The motivation behind sharing these strategies is to illustrate several important back-testing techniques and common mistakes when developing algorithms. We intentionally selected the most representative algorithms for illustration because the algorithms themselves are not the focus but the techniques behind developing these strategies. In the end, it is the trading system that matters not one single or one set of strategies. For general strategies, we will go over cross-over of multiple moving averages, auto-regressive moving averages models (and integrated version).We will also cover forward talk, back-testing, Granger Causality during our discussion. In the end, we will propose a Random Forest Dynamic Model using lagging values for interval time-frame point prediction. The focus will be on developing a whole strategy systems using not only algorithms, statistics but also logical reasoning based on computations.

3.2 Moving Average Cross-Over Strategy

3.2.1 Description

One of the most widely used trading strategy is moving average cross-over strategy. The strategy uses two moving averages with different degrees of smoothing to forecast market trends. The faster moving average has shorter memory while the slower moving average has longer memory. Because a short-term smoothing considers only prices within a short period of time, it is more reactive to short-term price changes. Similarly, long-term smoothing moves slower. When short-term moving average crosses above the slower one, it represents a bullish upward trend which tells a trader that it is time to buy. During the opposite situation, the market shows down-ward trend and a trader should sell at that time. However, because moving average cross-over is so simple, it gives equal opportunities to traders so it probably does not work well in long trend. It is however meaningful to examine one simple alternation to such strategy: placing an insurance bid into the trading system. During our back-testing, we will buy a stock if and only if we have just sold our stock or we have just

started the testing; in addition, we will only sell a stock if and only if we have just bought a stock and have reached more than $(1.001)^2 = 0.2\%$ increase comparing to the lastly bought price because the back-testing exchange platform we are using is binance which charges a 0.1% transmission fee for every buying or selling actions.

3.2.2 Back-testing result

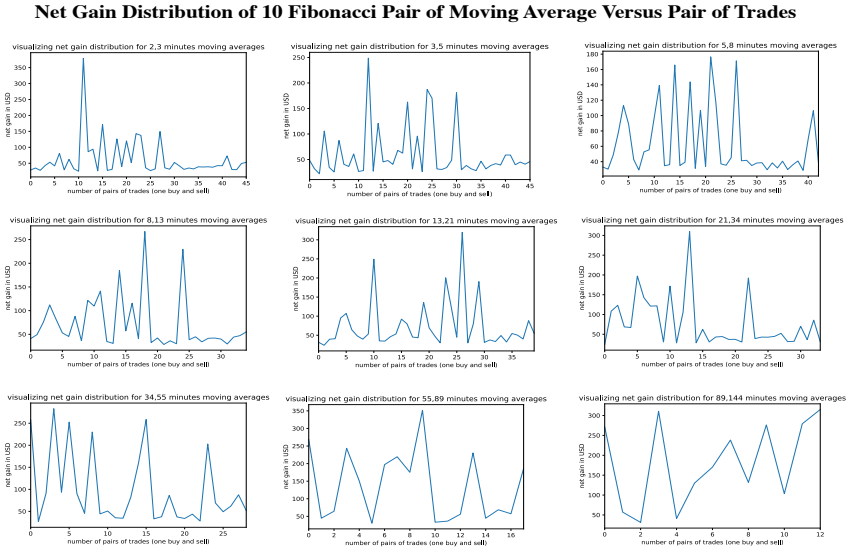


Figure 4: Net gains versus number of pair of trading actions with insurance

During our historical back-testing, we examined how would different window-sizes of moving averages affect return rate. We found that using the moving average with insurance and without insurance have very distinct behaviors. However, the discussion about their behaviors are not meaningful unless we incorporate real price movements. Looking at the price time series chart below, a buy and hold strategy would profit no more than $2000\$$ between April 2nd, 2018 and April 26th, 2018. In addition, we have circled one period of abrupt increase after several days of decline and side-way movements. Now, looking at the net gain distribution without insurance, the one single abrupt net gain is obvious and that is corresponding to the same increase as we have circled in the mid-price chart. We also noticed that across-all pairs of moving averages, the general shapes of net gain distribution are surprisingly similar. However, the moving average strategy with insurance constraint have very different shapes as showed above.

The trading frequencies for the two strategies also differ in magnitude. The most frequent trading for strategy without insurance has more than 6000 pairs of trading actions within 22 days. That is 2.64 trading actions on average per minute. However, the most frequent one with insurance is only 90 pair of trading actions in 22 days or around one single trade per three hours on average.

Trading without insurance has much higher standard deviation and much lower average return as showed in the side by side box-plot below. In addition, we noticed some extreme outliers for method without insurance. When time window size is a tuning parameter, we can use this method to compare if an additional constraint will increase the return rate or not.

3.2.3 Conclusion and Code Appendix

We offer the most basic strategy to start with. However, trading system is not only consisted with strategy. There are logical inference built upon strategies to formalize trading actions. We have seen that with a simple twist of logic: adding a constraint to selling action, we are already capable to generating statistically significant return rates based on the side-by-side box-plots. Therefore, similar tests should be performed when adding new constraints and prior to live testing, it is only through historical data back-testing can we know how well a new element will affect the model.

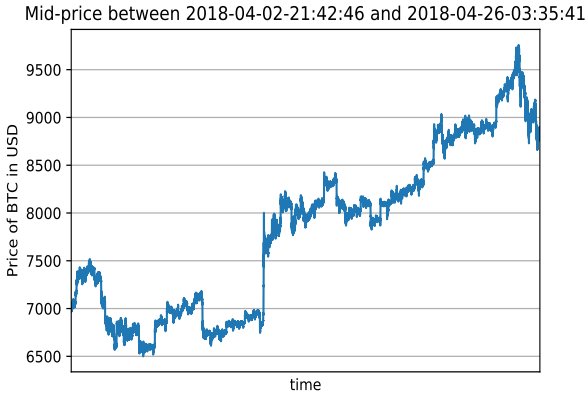


Figure 5: mid-price time series plot between April 2nd 2018 and April 26th, 2018

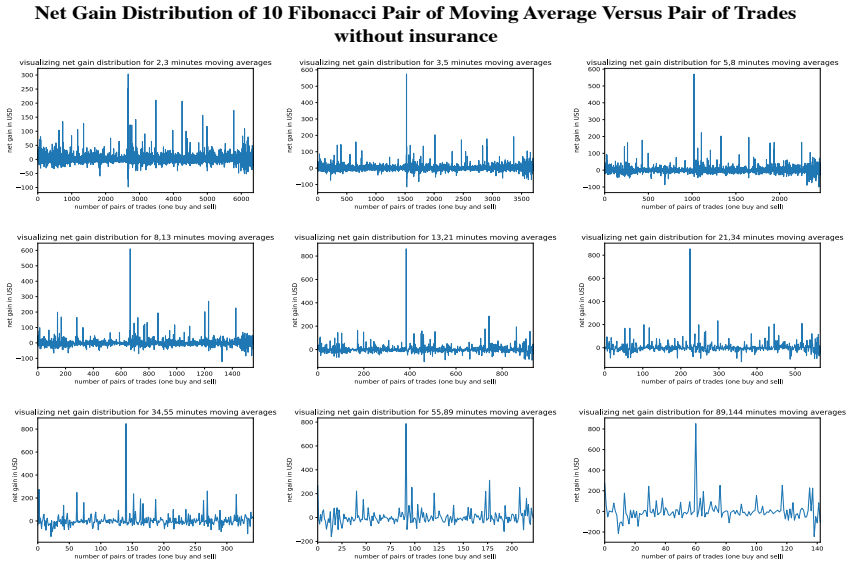


Figure 6: Net gains versus number of pair of trading actions without insurance

For clarity of the algorithm, we share a small portion of the code below for readers to study.

```
    for i in range(nrows—1):
        if short_ma[i] < long_ma[i] and short_ma[i+1] >= long_ma[i+1] and sold:
            trading_log.append(mid_price[i] * —1 * current_ratio)
            bought, sold = not bought, not sold
        if short_ma[i] > long_ma[i] and short_ma[i+1] <= long_ma[i+1] and bought:
            assert(len(trading_log) != 0)
            last_buy_price = abs(trading_log[—1])
            if mid_price[i] * current_ratio > (1 + 0.3/100) * last_buy_price:
                trading_log.append(mid_price[i] * 1 * current_ratio — last_buy_price * (0.001/100) ** 2)
                bought, sold = not bought, not sold
                current_ratio = trading_log[—1] / abs(trading_log[—2])
                assert(current_ratio >= 1)
```

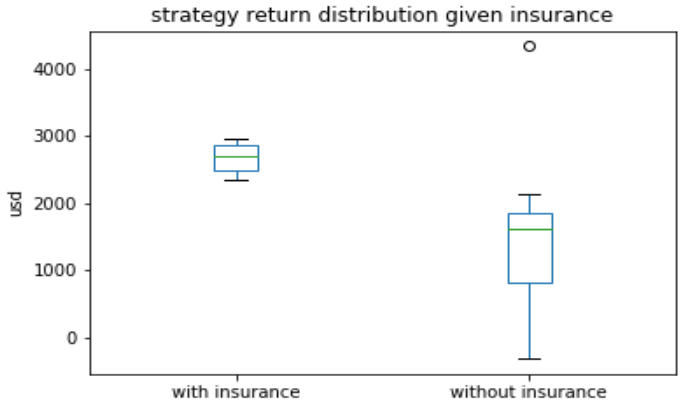


Figure 7: statistically significantly different distribution between two methods

3.3 Autoregressive Moving Average Strategy

3.3.1 Description

ARMA models has two components: autogression which includes using lag values and moving average which models linear combinations of errors at different times. In more formal notation, we have the following:

$$ARMA(p,q) : X_t = c + \epsilon_t + \sum_{i=1}^{p} \psi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

Where the autogressive model is $AR(p) : X_t = c + \sum_{i=1}^{p} \psi_i X_{t-i} + \epsilon_t$ and the moving average model is $MA(q) = X_t + \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$. ψ, θ are all coefficients for X ane ϵ respectively. The parameters: q and p can be optimized by examining partial autocorrelation and autocorrelation plots.

In addition, because often real-world data are non-stationary, we can use a revised version of ARMA: Autoregressive Integrated Moving Average or ARIMA. The difference between ARMA and ARIMA is differencing, a transformative techniques applicable to time-series data so make the data stationary.

3.3.2 Autocorrelation

We set the maximum limit of lag to be 60 and observe that the first ten lags are positive with the first five lags demonstrating statistical significance based on the two plots below with different details. Therefore, we choose $p = 5$

In addition, we want to examine if there would be no correlation for lag values after $p = 5$.

Based on partial correlation result, the first three lags are clearly significant and while four and five are somewhat outside of the insignificance zone, it is clear that almost all other lags beyond five have insignificant correlations. After modeling using ARIMA regression package, we have the statistics summary as below:

In addition, we want to examine the residuals errors. Most of errors are centered about 0 with some sparse outliers. The density of residual error plot clearly indicates errors are Gaussian Distributed.

Here we model the density distribution of residual errors:

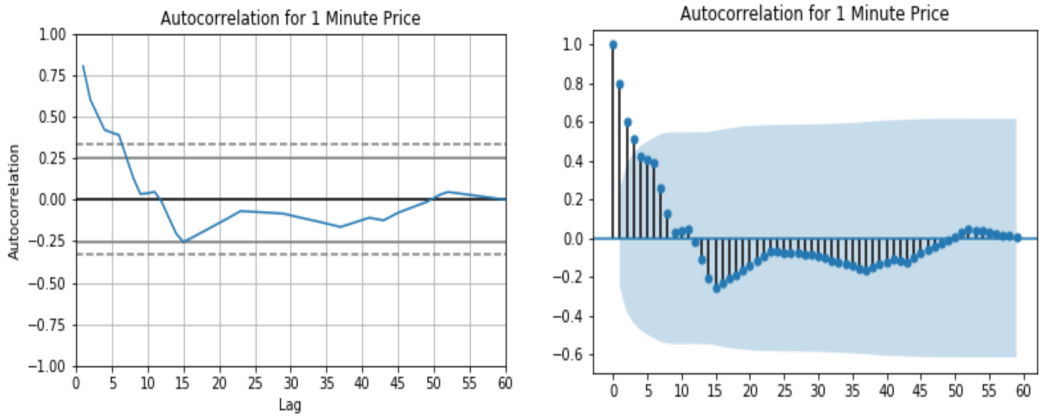


Figure 8: Autocorrelation Plot using two different packages to bring out details

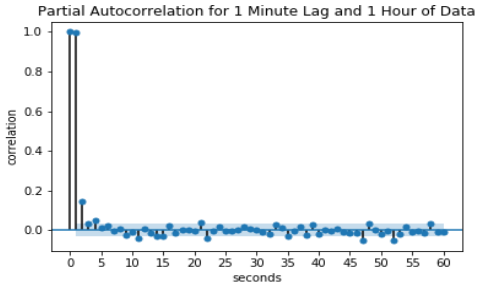


Figure 9: Partial Correlation to examine lag values beyond p = 5

3.3.3 Granger Causality

We live in a complicated world where one thing might have many causes whether directly or not. Therefore, a univariate ARIMA only tells a part of the story of fortune telling. From a philosophical point of view, how much can history tell about the future if our focus is no singular? In the context of predicting future price of Bitcoin, many other cryptocurrency might have high co-integrations with it. Granger causality came in just to solve this problem. The essence of Granger causality is simple: say we have three variables X_t, Y_t, E_t. First, we build a model that predicts Y_t using some lag values of Y: Y_{t-p} and E_{t-p} where p is positive integer. Second, we bring a new variable X_{t-p} into the set of predictors, so collectively we use Y_{t-p}, E_{t-p} and X_{t-p} to predict Y_t. If the second attempt exceeds the first one, we then conclude that the past of X contains information that are useful in forecasting Y_t that are not present in neither the history of Y nor the history of E. Hence, we conclude that X grangerly causes Y.

Recall ARMA's formula:

$$ARMA(p,q) : X_t = c + \epsilon_t + \sum_{i=1}^{p} \psi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

A bivariate ARMA model would look something like the following:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \psi_i X_{t-i} + \sum_{k=1}^{m} \phi_k Y_{t-k} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

We can abbreviate the model as:

ARIMA Model Results

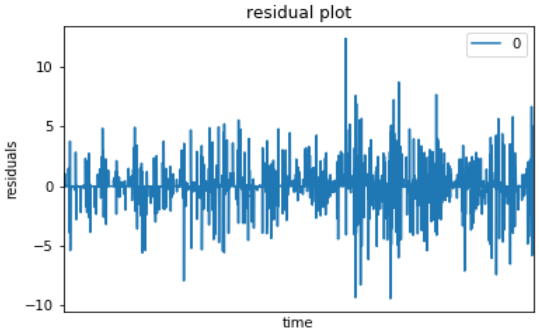
Dep. Variable:	D.y	No. Observations:	3599
Model:	ARIMA(5, 1, 0)	Log Likelihood	-5656.578
Method:	css-mle	S.D. of innovations	1.165
Date:	Tue, 15 May 2018	AIC	11327.156
Time:	00:36:26	BIC	11370.475
Sample:	04-02-2018 - 04-02-2018	HQIC	11342.594

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0076	0.015	0.516	0.606	-0.021	0.037
ar.L1.D.y	-0.1692	0.017	-10.156	0.000	-0.202	-0.137
ar.L2.D.y	-0.0466	0.017	-2.760	0.006	-0.080	-0.014
ar.L3.D.y	-0.0509	0.017	-3.003	0.003	-0.084	-0.018
ar.L4.D.y	-0.0189	0.017	-1.118	0.263	-0.052	0.014
ar.L5.D.y	-0.0282	0.017	-1.689	0.091	-0.061	0.005

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.4060	-1.4414j	2.0136	-0.1270
AR.2	1.4060	+1.4414j	2.0136	0.1270
AR.3	-1.9244	-0.0000j	1.9244	-0.5000
AR.4	-0.7796	-1.9840j	2.1317	-0.3096
AR.5	-0.7796	+1.9840j	2.1317	0.3096

Figure 10: summary after fitting the model using 1 hour of univariate data

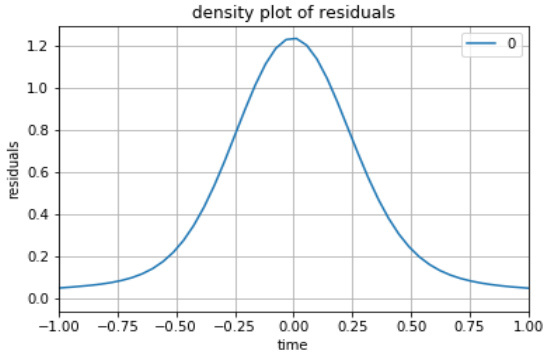


$$X_t = \sum_{i=1}^{p} \psi_i X_{t-i} + \sum_{k=1}^{m} \phi_k Y_{t-k} + E_t$$

If including Y in the modell reduces the variance of E, we then say Y granger-causes X. Linearly regressing on both models, we can obtain R^2 values for both the full model and the reduced model (the one without Y as predictor). Here, we use Etherum's price as Y and Bitcoin's price as X. We found that the R^2 increased from 0.91 to 0.92 while the MSE decreases from 0.16 to 0.14. Now, we compute the F-statistics which is:

$$F = \frac{\frac{(R_F^2 - R_R^2)}{1}}{\frac{(1-R_F)^2}{50-2-1}}$$

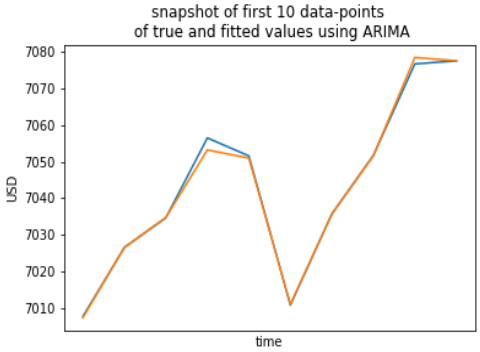
We found that $F > F^*(1, 50-2-1)$. Hence, we reject the null hypothesis that Y does not contain useful information in predicting future of X and conclude that etherum price granger-causes bitcoin price. However, be aware that granger causality is not true causality. In fact, we can not know if there are any co-founding variables between etherum and bitcoin (there probably are). In addition, there are two important assumptions behind granger causality: first it assumes co-variance stationary; second, linear model sufficiently describes it. We might have already realized that these assumptions might not be met especially the second assumption. However, granger causality for feature selection is useful in our context and does shows improvements in term of mean square of errors in practice.



3.3.4 Rolling Forecast ARIMA

The most important thing about an algorithmic trading system whether it is machine learning driven or not is the trading strategy system. Even if we are able to accurately predict the future, it does not guarantee that we will profit. Many things should be taken into considerations. For example, how long into the future are we predicting. In real life, knowing the next second's stock price does not give trader enough time to trade nor would it be a large profit: just think about how much does a stock price fluctuate in one second. Therefore, when predicting about a future value, we should always looks several steps ahead. Theoretically, we can build multiple layers of linear regression to predict a sequence of future values. However, it is not entirely useful if we keep a record of historical fitted points. In other words, if we are predicting about the bitcoin price in the future X seconds, and we have already made prediction at $X - 1$ seconds one second ago and, therefore, we have all fitted values from now all the way to the future X seconds. Based on this path, we can then decide when to buy and when to sell.

For the scope of ARIMA, we won't be discussing too much on formalizing the strategy as we will cover that in the Random Forest section. On the other hand, it is important to point out when cross-validating errors, we should use walk-forward back-testing or sliding-window techniques to examine how accurate our predictions are. We took a snap-shot of the first ten data-point true values and fitted values below. In addition, we found the MSE to be 1.31 which is sufficiently small.



Let's visualize how to conduct time series cross-validation. Notice that the blue points are training data, whereas the red points are prediction data. Essentially, we grow the size of training data and observe how does evaluative statistics change over time.

3.3.5 Conclusion and Code Appendix

Up to this point, we have examined general trading strategies such as multiple moving average cross-over and auto-regressive integrated moving average. Regression has been in financial field for quiet some time and there are several reasons: first, it is easy to understand and interpret and second it is amazing fast when making predictions. These two characters make regression ideal for fast-paced

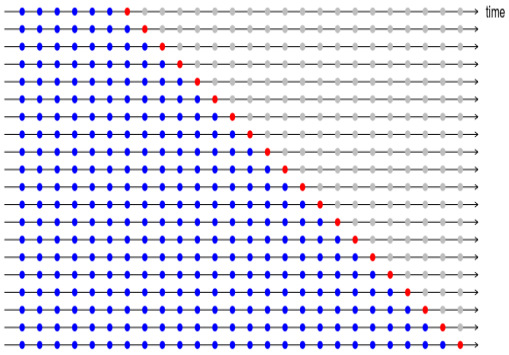


Figure 11: This Image is From: https://robjhyndman.com/hyndsight/tscv/

prediction environment like financial trading bot. However, it lacks the complexity and often faces over-fitting challenges. We will next demonstrate our last strategy which shows significantly better results when comparing with other strategies we have seen so far during a dynamic random forest skip forecast algorithm with triple barrier labeling method.

```
fitted_values = []
true_values = []
for i in range(3600, 3600*24, prediction_interval):
    model = ARIMA(X[:i], order = (5, 1, 0))
    fit = model.fit(disp=0).forecast()
    yhat = fit[0]
    fitted_values.append(yhat)
    obs = list(X)[i]
    true_values.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
```

3.4 Winning Algorithm: Dynamic Random Forest and Triple Barrier Labeling Method

3.4.1 Background

Decision Tree algorithm tends to have low bias but high variance. Random forest solves decision tree's over-fitting tendency by building many decisions trees with random features. Let training set $X = x_i$ and $Y = y_i$ where $1 \leq i \leq n$. First, we sample with replacement K times from the n training samples, and then train a decision tree $f^k_{decision}$ on X^k and Y^k where X^k is sampled from X. Then, for new unseen data, we can predict it by takes averages of these K decision trees (regression case, majority voting for classification):

$$\hat{f} = \frac{1}{K}\sum_{k=1}^{K} f_k(x_{new})$$

The variance is reduced in the following manner:

$$\sigma = \sqrt{\frac{\sum_{k=1}^{K}(f_k - \hat{(f)})^2}{K-1}}$$

3.4.2 Description

In our algorithmic trading system, we dynamically update our model every two minutes in order to extract micro-structural features out of the data through random forest training. As parts of predictors, we took averages of two important lessons learned from moving averages and ARIMA analysis: we include etherum and etherum/btc ratio inside our $X_predictors$. In hope to take advantages of MA and ARIMA's predicted power we compute several important econometric such as imbalance, spread, differencings and exponentially moving averages. In addition, we autoregress our predictors to create lag values with maximum lag $p = 10$ or using maximum ten seconds of history. The point labels we want to predict is BTC^{price}_{t+k} where t is a time variable and $k = 60$ seconds. I.E.: we want to use ten seconds of historical data (laggs) along with the power of econometric from various sources of Grangerly Causal variables to forecasts ten seconds of future bitcoin price seamlessly (make prediction every single second) so that we could have a full path in $k - p = 60 - 10 = 50$ seconds of future. Using this path, we then utilize triple barrier methods as proposed by de Prado to compute entry and exit signals. The final touch is, of course, adding a stop loss and stop limit insurance mechanism to maximize our Sharpe Ratio.

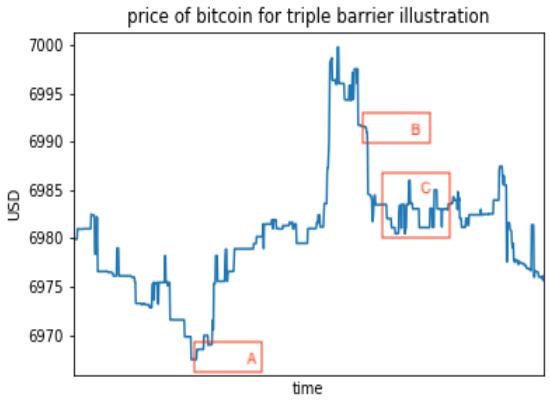


Figure 12: Triple Barrier Method Illustration

In Figure.12, the barrier box marked by A is an example where we exited out of upper limit first, indicating a buying signal. At point B, the price first touches the lower limit triggers a selling request. However, our last box at point C touches neither upper limit nor lower limit but exit out of expiration

horizon directly in the end which indicates a HOLD signal. Also, it is worth to point out that the upper limit and lower limit height is calculated dynamically using the past three hours' volatility. Therefore, we incorporate volatility of the most current cryptocurrency market when constructing this strategy.

3.4.3 Evaluation and Comparison

Figure 13: This image is from: https://www.tradingview.com/chart/eBoxBhdW/

LIVE TEST RESULT	2 months return rate	daily return std	max daily gain	average daily return rate	longest lock-in time
Buy and Hold (Benchmark)	-2.1%	NA	NA	NA	62 days
MA	3.74%	$34.9	0.0903%	0.0592%	42 hours
ARMA	5.31%	$119.1	0.11%	0.08348%	21 hours
ARIMA	6.75%	$42.9	0.13%	0.1054%	18 hours
Dynamic RF	21.4%	$74.9	1.9%	0.3132%	5 hours
LSTM	4.9%	$133.4	0.2%	0.0771%	7 hours

Figure 14: Live Trading using 1 BTC Statistics

Since our coding system has been examined through rigorous back-testing protocols, we invested real monetary assets: bitcoin for each strategy since February. However, due to the order to implementation, the longest live back-testng duration is two months span of time. Based on these statistics, we found that the dynamic random forest triple barrier strategy outperforms all other implemented strategies by a lot: with 21.4% two months increase since March 11th, and it should come no surprise that the moving average cross-over strategy generates the least amount of return but it still is better than bench-mark which will generate loss. The benchmark strategy uses a buy and hold method that can be easily computed using historical data and not through live testing (it would be wild to invest money in buy and hold strategy, but it is a good benchmark comparison to be included). In addition,we have witnessed that both ARMA and ARIMA performs with equal strength. Surprisingly, the recurrent neural network LSTM model generates about the same return rate as ARMA and ARIMA. Among all modeles, ARMA strategy has the most volatile daily return rate possible because of over-fitting on non-stationary data. Across all strategies, we have placed insurance: if the buying price does not exceed the expected value, we will convert the selling signal into a holding signal. Because of such insurance, there are times when we are locked in for some time. Dynamic RF also

outperforms other algorithms in term of the longest lock-in time. Moving Average strategy has the longest lock-in time as 42 hours which is almost 2 days without zero trading actions.

4 Conclusion and Future Work

From this paper, we have examined numerous aspects of building an algorithmic trading system. At the beginning, we presented an overview of the algorithmic system. It is worthy our efforts to reiterate the point one more time: it is the trading system that are extremely valuable not one single strategy. One single strategy is a very small part of the story despite of its complexity and importance. We should also keep in mind that forecasting and strategy are two very different concepts: knowing about the price for the next second with outstanding accuracy does not generate monetary return because of latency, volumes and transmission fee involved in real tradings.

Throughout our algorithm section, we have also looked at the power of adding constraints such as insurance to a trading strategy. We clearly observed improvements and decreased variance of return distribution for various Fibonacci sequences of moving average windows. We have also looked at auto-regressive moving average and discussed the significance of using granger causality test to decide informative features to build multivariate predictors. However, we should keep in mind that Granger Causality is not true causality but it demonstrates if the history of one variable contains predictive power of another variable in the future time.

At last, we utilized all of our learning and optimized a dynamic random forest algorithm that predicts the future 50 seconds trends. We further uses triple barrier methods which dynamically computes barrier threshold using past hours volatility to generate buying and selling signal. No historical data back-testing can out-perform the validity of real testing results. For the past several months, we have been collecting the trading logs and statistics from the trading bot. The conclusion, which is beyond satisfactory, that we are able to achieve a very good Sharp Ratio using our winning algorithm with 21.4% return rate in two months.

For future works, as the trading size grows large, it would be interesting to learn techniques of dividing the order size. Also, layering predictions that uses both macro-economic prediction and micro-economic prediction is also a fascinating direction. We have discussed the obstacle of implementing an reinforcement learning trading system because of the assumption that agent's action should alter the state of the environment. However, there are newly published articles that uses adjustments of angle to substitute changes of states of environment. Therefore, it might still be possible to experiment with Reinforcement Learning. Optimistically, if we can trade with larger volume, becoming market maker might not be entirely a dream; hence, RF still has its usage in the domain.

5 Reference

[1] Prado, M. L. (2018). Advances in financial machine learning. Hoboken: Wiley.

[2] Chan, E. P. (2009). Quantitative trading: How to build your own algorithmic trading business. Hoboken, NJ: John Wiley & Sons.

[3] Dalio, R. (2017). Principles. New York: Simon & Schuster.

[4] Davey, K. (2014). Building Algorithmic Trading Systems: A Traders Journey From Data Mining t. John Wiley & Sons.

[5] Fabozzi, F. J., Focardi, S. M., & Kolm, P. N. (2006). Trends in quantitative finance. Charlottesville, VA: Research Foundation of CFA Institute.

[6] Hannan, Edward James (1970). Multiple time series. Wiley series in probability and mathematical statistics. New York: John Wiley and Sons.

[7] Brownle, J., Dr. (2017, February 21). How to Create an ARIMA Model for Time Series Forecasting with Python. Retrieved from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[8] Brownle, J., Dr. (2017, February 6). A Gentle Introduction to Autocorrelation and Partial Autocorrelation. Retrieved from https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

[9] Seth, A. (2007) Granger Causality. Retrieved from http://www.scholarpedia.org/article/Granger-causality

[10] Hyndman, R. (2016, December 05). Cross-validation for time series. Retrieved from https://robjhyndman.com/hyndsight/tscv/

[11] Kakushadze, Zura., Dr. (2015, December 9). 101 Formulaic Alphas, Quantigic Solution LLC.

[12] Pandas: Powerful Python data analysis toolkit. Retrieved from https://pandas.pydata.org/pandas-docs/version/0.20/

[13] Documentation of scikit-learn. Retrieved from http://scikit-learn.org/stable/documentation.html

[14] Investopia (2018, March 01). Guppy Multiple Moving Average - GMMA. Retrieved from https://www.investopedia.com/terms/g/guppy-multiple-moving-average.asp

[15] Selby, A. (2016, May 06). Using the Guppy Multiple Moving Average (GMMA) Indicator. Retrieved from http://www.onestepremoved.com/using-the-guppy-multiple-moving-average-gmma-indicator/