Politechnika Wrocławska

Faculty of Computer Science and Management
Field of study: **COMPUTER SCIENCE**
Specialty: COMPUTER ENGEENERING

Master Thesis

Research and evaluation of algo-trading algorithms.

Hubert Olkiewicz

keywords:
algotrading, strategy implementation
stock market, tests, robustness tests
genetic algorithm, ARIMA

short summary:

Over 85% of market volume is generated by computer driven trades. This thesis aims to show process of implementation, testing and evaluation of algo-trading algorithms. Researched strategies are divided into two groups; strictly indicator driven and using advanced statistical models or optimization and search problem algorithms. Each of eight researched algorithms is described, tested, optimized and discussed.

Supervisor	.....	.....	…….
	Title/ degree/ name and surname	*grade*	*signature*
The final evaluation of the thesis			
Przewodniczący Komisji egzaminu dyplomowego	.....	.....	…….
	Title/ degree/ name and surname	*grade*	*signature*

For the purposes of archival thesis qualified to: *

a) Category A (perpetual files)
b) Category BE 50 (subject to expertise after 50 years)

* Delete as appropriate

stamp of the faculty

Wrocław 2019

Streszczenie

Algotrading, od 2003 roku, szybko przejmuje giełdy na całym świecie. Ponad 85% wolumenu jest generowane przez automatyczny handel. Ta praca ma na celu pokazanie procesu implementacji, testów oraz ewaluacji algorytmów algotradingowych. Algorytmy muszą zostać wyczerpująco przetestowanie. Sprawdzenie, czy tylko logiczna strona implementacji działa prawidłowo jest niewystarczające. Podstawa rozumowania zaimplementowanej strategii musi zostać przetestowana używając historycznych, sztucznie wygenerowanych oraz rzeczywistych danych z naciskiem na to, czy przynosi zyski przy możliwie najmniejszym ryzyku. Badane strategie są podzielone na dwie grupy: używające tylko i wyłącznie wskaźników (proste algorytmy) i używające zaawansowanych modeli statystycznych lub przeszukujących przestrzeń w celu znalezienia najlepszych rozwiązań i optymalizacji algorytmów.

Każdy badany algorytm jest opisany, przetestowany, zoptymalizowany i omówiony, wskazując na wady i możliwe ulepszenia. Niektóre algorytmy działają lepiej, gorzej bądź nadzwyczajnie. Ostatnie są poddane dodatkowym testom pokazującym czy ich nienormalne osiągi są tylko kwestią szczęścia w doborze parametrów algorytmu, czy też istnieje solidne uzasadnienie.

Wyniki pokazują, iż techniczne wskaźniki nie radzą sobie dobrze z inwestowaniem długoterminowym, jednakże działają one znacznie lepiej niż statystyczne modele prognozujące przyszłą cenę. Średnio, bazując na optymalizacji algorytmów, najlepiej działającą strategią była bazująca na idei odwrócenia trendu. Jednak testy niezawodności oraz „80/20” pokazały, że dobranie parametrów były tylko kwestią szczęścia. Testy pokazały, że średnio algorytm nie był stratny, ale też nie przynosił znaczących zysków. Algorytm genetyczny udowodnił, że jest dobrym narzędziem optymalizacyjnym, ale tylko do znajdowania mniej więcej najlepszych konfiguracji parametrów. Największym minusem jego używania jest to, iż nie dostarcza on szerszych informacji jak średnie wyniki. Algorytm bazujący na uczeniu maszynowym nie działa najlepiej bazując tylko na technicznych wskaźnikach, potrzebuje on więcej zewnętrznych atrybutów jak wiadomości mechanizm bądź rozpoznawania wzorców. Dalsze znaczące ulepszenia można osiągnąć poprzez zastosowanie technik zarządzania ryzykiem lub analizy fundamentalnej.

Abstract

Algorithmic trading, since 2003, is rapidly overtaking stock market. Over 85% of market volume is generated by computer driven trades. This thesis aims to show process of implementation, testing and evaluation of algotrading algorithms. Algorithms have to be exhaustively tested. Checking if the logic core of implementation works properly is not enough. The backbone reasoning behind implemented strategy needs to be tested against data, historical, artificial generated and real, with emphasis whether it yields profits with as low risk as possible. Researched strategies are divided into two groups; strictly indicator driven (simple algorithms) and using advanced statistical models or optimization and search problem algorithms.

Each researched algorithm is described, tested, optimized and discussed, indicating flaws and possible solutions. Later on simple algorithms are adjusted using proposed ideas. Some of algorithms perform better, worse or exceptional. Latter ones are subjected extra tests in order to see if their abnormal performance is just matter of luck in choosing perfect algorithm parameters or there is robust reasoning behind it.

Results shows that technical indicators does not cope well with long term investing, although they perform much better than statistical prediction models. On average, based on algorithm optimization, the best performing strategy was mean reversion based. Yet, robustness checks and “80/20” tests showed that compositions of chosen parameters was just matter of luck. Mentioned tests showed that on average algorithm was not losing nor making notable profit. Genetic algorithm proved to be good optimization tool but only for finding more or less best parameter configurations. The biggest downside its usage is that it does not provide any wider information about algorithm performance, i.e. average outcomes. Machine learning algorithm does not work well by basing only on technical indicators, it needs additional external features like news or pattern recognition mechanism. Further major improvements could be achieved by incorporating risk management techniques or fundamental analysis.

Table of Contents

Preface

Automated trading systems, also called algorithmic trading, automated trading or system trading, allow traders to set up and maintain specific rules for trade entries and exits that, once implemented, can be automatically executed by a computer. The trade decision rules can be based on simple conditions such as a moving 200 days average crossover, or they can be more complicated strategies that require a comprehensive understanding of statistics or the programming language specific to the user trading platform. These systems work with specific environment, in most cases using broker's external API (Application Programming Interface) to make trade entries or exits.

Fully tested strategy along with automated system which uses it can help traders to make decision or even make it for them to maximize yields and minimize risks. Major advantage of such a system is emotionless decision making (which is main human problem when it comes to trading) so it makes able to consistently use given strategy, without any sentimental or emotional biases.

Automated trading systems are made by converting trading strategies into sequence of machine code. Once run, system uses given rules to find the most corresponding trades which by his rule understanding are most profitable. Hence it's clear that any, even tiny, deviation in strategy understanding may lead to quick losses, therefore huge algorithms testing importance should be emphasized.

Various testing standard practices ought to be taken care of while testing automated system like: back-tests using historical data, stress tests with real time indicators, avoiding data overfitting or biases (ex. survivorship), robustness tests, considering broker fees…

Automated systems underlying strategies are mostly based upon market price moves which is called technical analysis. Another known approach is called fundamental analysis which oftenly involves technical analysis as well and other indicators like economical or financial factors. However it doesn't mean that systems made around for instance news triggered trades don't exist. It should be noted that one strategy might work really well for given conditions but it will never work that well for whole the time since environment properties are changing (not only stock itself but even trading volume or competition which might use same algorithm).

This thesis aims to implement, test and adjust several, already existing, strategies varying from simple to complex ones using different financial instruments. Afterwards system should be adjusted to return more profits than its primary version. It will provide necessary methodology to manage implementation of investment rules for algorithmic decisions and as well as best testing practices which are required to avoid unnecessary losses.

1. Introduction

1.1. Algorithmic trading

Algorithmic trading is the high-speed execution of automated trading strategies, which are conducted in short time in order to profit from pricing or other market inefficiencies. Basically it is use of computers and programs to generate and execute orders in markets faster and more precisely than human. Securities and Exchange Board of India has defined Algorithmic-trading as "any order that is generated using automated execution logic". Institutional clients trade huge volumes in shares, equities, commodities, derivatives and asset classes. These values and volumes are often larger than market could take without impacting price. This is where Algorithmic trading comes with help, by executing even most simple strategies automatically, saving a lot of trader's or firm's time and money. [1]

1.1.1. Common algorithm types

- **Momentum**
  This algorithm type looks for cases where market trend moves sharply in one direction on significant volume. When one is discovered, investor joins this trend. Simple example of this algorithm is to buy a stock when the price is above a moving average and then sell when it's below. If current price is higher
- **Mean Reversion**
  Mean reversion assumes that the price of the stock will revert over time to its long time average price. If the stock price is higher than moving average, it should be sold. If the stock is below its moving average, it should be bought.
- **Sentiment**
  This strategy is based on events, where investors have to be up-to-date on news to anticipate stock reaction. Investors might add sources like Twitter, Bloomberg, Google search trends to their sources which are continuously researched. Artificial intelligence, text recognition are handy tools to help with dealing with such strategies.
- **Technical investing**
  Technical investing researches historical market data to discover meaning of current changes. This strategy tries to anticipate future movements of stock and based upon this make a profit.
- **Pairs trading**
  Pairs trading involves buying a pair of stocks that move together closely by market based similarities. When one stock in pair is performing better than another, the worse performing stock should be bought with expectation that it will outperform partner if near future and profit will be made. [2]

1.1.2. Macro level strategies

The macro level decision is about of choosing an appropriate optimal trading strategy. Whole process consists of: choosing appropriate price (ex. stock price), trading style (ex. risk aversion) and adaptation tactics (how will strategy adopt to changing market conditions). Strategies can be categorized into two major domains:

- **Rule based trading**
  Rule based trading algorithms control the macro-level decisions like order slicing strategy (breaking larger orders into smaller ones over time to reduce market impact). These are based on simple logic and don't provide insight into

potential cost or associated risk. Mostly rule based algorithms are “black-box” approach to trading.

- **Quantitative techniques**
  Quantitative algorithmic trading also controls the macro-level decisions but differs from rule based trading by basing all decisions on statistical and mathematical models. These algorithms manage transaction costs and provide investors with transparency regarding risk and cost by showing how the algorithm will behave with different market conditions. Transparency like this allows traders to adjust algorithms to determine most appropriate for configuration for given task and time. [3]

1.1.3. Micro pricing strategies

Micro pricing regards pricing schemes and order submission rules (ex. size quantities, limits). Goal of micro pricing strategy is: executions follow given macro strategy, achieving reasonable prices without creating too much impact on cost, determine when it would be the best time to deviate from currently optimal macro level strategy. If costs are high, optimizing and micro-management on each exchange can also save costs. For example reserve/iceberg order is common technique to continuously replenish order. For example, a 1000 share order can be entered as 100 but 10 times. After first 100 orders are entered immediately another 100 are replenished and displayed to avoid market impact and fluctuations. It’s very important that micro pricing strategy must be consistent with macro level strategy to achieve implementation goal. Micro level strategy determines when it’s appropriate to deviate from macro level strategy based on liquidity or price movements. [4]

1.1.4. High frequency trading influence

High-frequency trading (commonly abbreviated as HFT) is a program trading platform that uses powerful computers to transact a large number of orders at fractions of a second. It uses complex algorithms to analyze multiple markets and execute orders based on market conditions. Typically, the traders with the fastest execution speeds are more profitable than traders with slower execution speeds. This approach is well described in *Figure 1*.

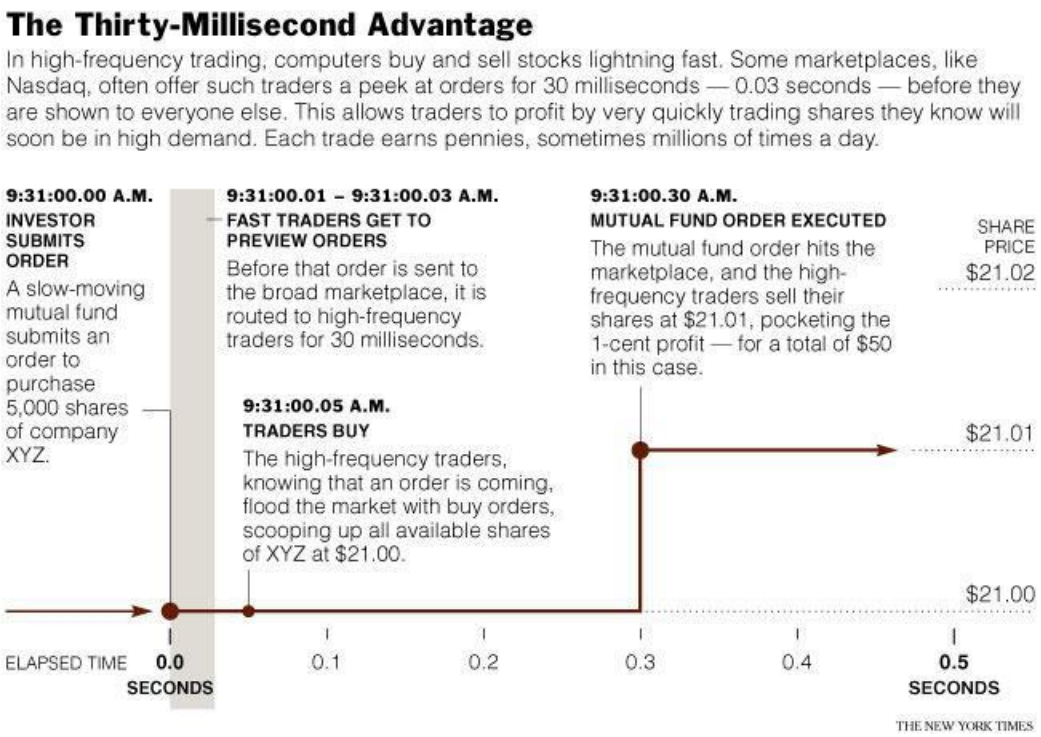

Figure 1 The Thirty-Milisecond Advantage (by New York Times)

"For a time, it looked as if high-frequency trading, or HFT, would take over the market completely. In 2010, HFT made up over 60% of U.S. equity volume. But the trend is declining. In 2009, high-frequency traders moved about 3.25 billion shares a day. In 2012, it was 1.6 billion a day. At the same time, average profits fell from about a tenth of a penny per share to a twentieth of a penny. In 2017, HFT accounted for just under half of all domestic equity volume." [5]

Most of scientific papers don't find negative effects of HFT market quality, they say that it cuts down transaction costs and time and adds liquidity (some of them argue if it's only artificial liquidity) to the market. However it basically acts like another agent which cuts down profits of sellers and buyers. These losses are invisible in first glance but grow significantly over time. Another drawback of HFT are flash-crashes which in matter of seconds can plunge market by even 10% followed by price recovery. [6]

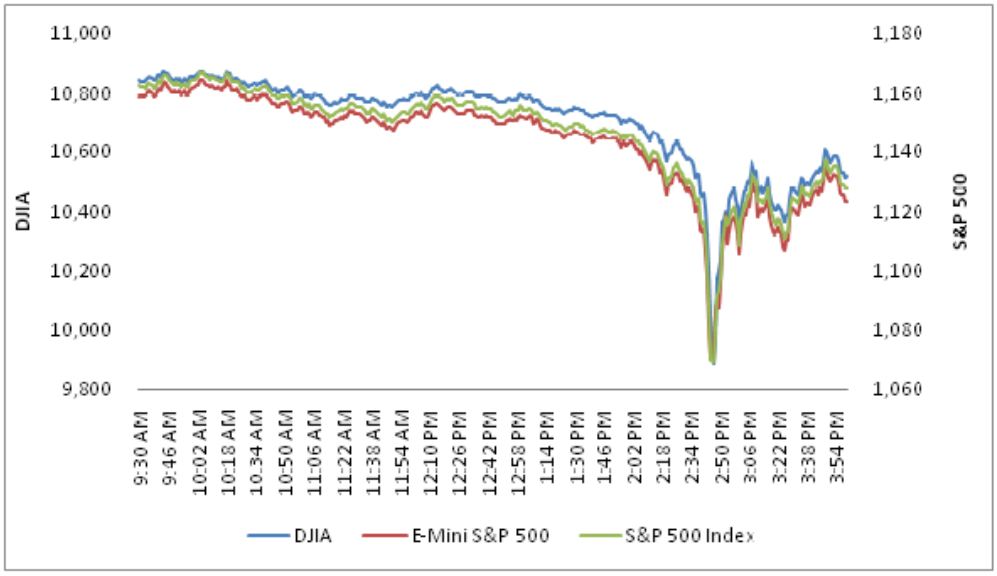

Figure 2 Flash crash example

HFT dominance is slowly diminishing since whole system is too costly to create and maintain and as well, to stay constantly competitive, it requires frequent and expensive upgrades. Markets are dynamic hence every program needs to be updated frequently to yield profit. As well HFT regulations are getting stricter. In 2013 Italy was the first country to introduce a special tax exclusive for high frequency trading and soon was followed by France. Also marketplace is very crowded with same system applications hence systems have to compete with another similar ones.

1.2. Quantitative trading

Quantitative trading is subset of algorithmic trading, where implementations of strategies are made in systematic manner, accordingly to established plan. These strategies are created using complex and exhaustive researches and mathematical computations. Usually trading team consist of market researcher/analysis who looks for profitable trading ideas and tests them, developer who implements idea and trader who supervises working system. Commonly used programming languages for constructing such a systems are *MATLAB*, *R* or *Python* but as trading frequency and algorithm strategy complexity rises, technological aspect becomes more relevant hence *C/C++*, assembly and networking optimization knowledge is of high importance.

Quantitative trading system consists of four major components:

- **Strategy identification**
  This phase considers finding strategy which is consistent with your portfolio and as well checking if the strategy coexists well with your other strategies portfolio. Decision on the trading frequency should be made also in this phase.
- **Strategy back testing**
  It consists of broad wide and exhaustive testing using historical data using various scenarios. Several biases should be taken care of which can affect final outcome profitability.
- **Execution system**
  In the execution system phase connecting to a broker, executing orders and minimalizing costs such as transaction costs are considered.
- **Risk Management**

It's constant and continuous process of inspecting all possible risks (i.e. technology or brokerage risk), biases and recent changes that can decrease profitability of the portfolio. [7]

Overall this topic is very complex since it requires backgrounding knowledges of statistics, econometrics and programming domains. Hence my thesis will focus only on last three components: back testing, system execution and risk management. Already proven strategies will be used.

1.3. Recent growth in Algorithmic Trading

Algorithmic trading was introduced in the 1990s in U.S. equity market and has become common in stock markets in recent years. Since its introduction, interest in understanding potential of markets impact and possible profitability was increasing. Since then computer-driven trading has caused several disturbances on market.

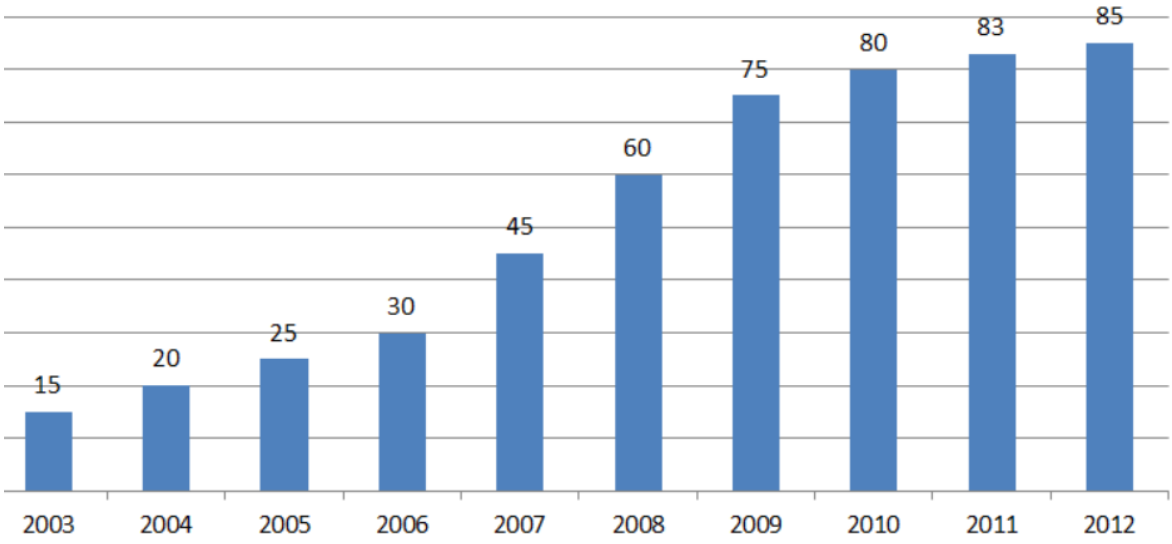

Figure 3 Algorithmic Trading. Percentage of Market Volume

Algorithmic trading resolved problem of major human issue – emotions and reaction time. Every strategy can be precisely implemented and executed with given rules. On the other hand research shows that some strategies like triangle arbitrage, which were profitable mainly because of human action speed, were diminishing since algorithmic trading introduction: "…algorithmic traders improve the informational efficiency of prices by taking advantage of arbitrage opportunities and making them disappear quickly, rather than by posting quotes that prevent these opportunities from occurring (as would be indicated by a strong negative effect on reducing triangular arbitrage opportunities)." [8]

By early 2000s, computer-driven trades had an execution time of few seconds. By 2010 – milliseconds and nanoseconds by 2012. From early 2000s high frequency trading accounted for less than 10% of trading volume. Since then automated trading popularity was growing in leaps and bounds.

In 2010 new idea was introduced to use computers to process and analyze Social Media news into trading signals. News based trading emerged showing that Algorithmic trading doesn't have to rely only on technical analysis.

Nowadays HFT is still dominant way of trading but it's hardly developing. Main advantages over competition comes from latency time or faster and better computers.

1.4. Financial instruments

Financial instruments are contracts between parties which can be traded, modified and settled. It can be currency, share or bonds. Financial instruments can be divided into two types:

- **Cash instruments**
  These instruments value is determined by the markets. It can be loans, deposits, securities, where both parties have to agree on a transfer
- **Derivative instruments**
  Value of derivative instruments is based on the one or more underlining entities like asset, interest rate or index.

Financial instruments may be also categorized according to asset class, which relies on whether they are debt or equity based as table 1 shows: [9]

Table 1 Asset class categorization

Asset class	Instrument type			
	Securities	**Other cash**	**Exchange-traded derivatives**	**OTC derivatives**
Debt (long term) > 1 year	Bonds	Loans	Bond futures Options on bond futures	Interest rate swaps Interest rate caps and floors Interest rate options Exotic derivatives
Debt (short term) ≤ 1 year	Bills, e.g. T-bills Commercial paper	Deposits Certificates of deposit	Short-term interest rate futures	Forward rate agreements
Equity	Stock	N/A	Stock options Equity futures	Stock options Exotic derivatives
Foreign exchange	N/A	Spot foreign exchange	Currency futures	Foreign exchange options Outright forwards Foreign exchange swaps Currency swaps

In context of trading it's important to keep in mind that various strategies might work well on one financial instruments and meanwhile on others not, in given time period.

2. Technical parameters

2.1. Introduction

Every algorithm is based on strategy. In some cases algorithm tries to mimic professional traders strategies and in another it relies on heavy statistical models that requires a lot of processing power hence it can be reliably managed and executed just by human. In these days most prevailing tools in trading algorithms development are *C++* (speed), *R* (created mainly for statistical computation) and *Python* (huge community support, relatively easy). Programming language itself is useless if no reliable trading or statistical frameworks would not be implemented. After algorithm is implemented, usually next step is to optimize it against desired stock since no stock is the same. Although some of them may apply to more or less vague connections with other stocks or instruments. To discover these connections huge CPU power is required, as well as more sophisticated methods than brute force. Final phase of algorithm development is to test it against real or artificially derived prices from real environment to evaluate if our assumptions are right or not and our strategy is properly implemented.

In this chapter all strategies, that are later on evaluated, are described as well as tools, testing and optimization methodologies.

2.2. Tools

Every strategy implementation is written using *Python* [10] programming language which is most prevailing programming language in terms of scientific external libraries developed and maintained as well as dedicated and helpful community.

Strategies developed and evaluated using *backtrader* [11] framework. This framework provides environment for back-testing strategies, including: broker behavior simulation, numerous indicators, analyzers and auxiliary tools like data providers.

Along with latter framework, various additional libraries are used, including: *statsmodels* (statistical models implementation), *matplotlib* (2D plotting library).

2.3. Strategies description

Trading strategies are defining when to buy and sell assets in market based on defined rules. These rules include investing and trading plan that implies specific investing objectives, time horizon, risk and taxes. Ideas needs to be researched and adopted to specific stock. Planning includes for instance that if stocks, bonds, ETFs or more complex financial instruments like options of futures are considered. Trade placement means collaboration with broker, taking into account spreads, fees and commissions. After trade is executed, decision about adjusting or closing determines final return.

Development of trading strategy relies mostly on technical or fundamentals. Technical trading is based on technical indicators to create trading signal are using data only from prices (current and historical) and trends. Fundamental trading, on contrary, relies on fundamental factors like revenue growth, profitability, reports.

Presented strategies are developed accordingly to technical trading. Simple strategies are based on technical indicators and statistical models, whereas genetic algorithm and machine learning rely more on optimization and search problems algorithms.

2.3.1. Simple strategies

2.3.1.1 Mean reversion

Mean reversion strategies are assuming that a stock's price will move to the average price over time. Mean revision analysis requires average price computation using variety of statistical techniques and identifying trading range for specific stock. Thus, if current price is far less than the average price, the stock is considered to be worth purchasing and on contrary, when market price is above average, it is expected to fall.

Main characteristics of mean reversion strategies are:

- Short hold time – this strategy often takes advantage of short-term corrections back to mean.
- Lower risk-reward ratio – prices returning to mean usually will not offer a lot of ticks.
- Higher win-rate

2.3.1.2 Momentum

Momentum investing is based on buying securities that had high returns over past time and selling those that had poor returns. Although there is no consensus among economist regarding validity of momentum investing, this strategy proved to work. Momentum strategies as well may use a historical data of a stock's prices, relative to current ones, to predict expected returns.

There are variety of momentum strategies such as [12]:

- Earnings momentum – it uses the historical earnings to predict future returns.
- Price momentum – it uses the price to create trading indicators which are used in trading strategies (i.e. SMA – Simple Moving Average)
- Relative strength momentum – most common momentum strategy. Is based on buying past securities with good returns and selling those with bad.
- Absolute momentum
- Dual momentum – best performing stock out of few is chosen and then compared to i.e. U.S. Treasury bills. If it performs worse, second best performing is compared and so on.
- Risk adjusted momentum – it standardizes returns by risk measure (i.e. Sharpe Ratio)

2.3.1.3 Statistics prediction based strategy

Statistical strategies rely on mostly mathematical or statistical complex models. They can be used to predict, model volatility, find patterns or analyze time series to show lead/lag effects or standard deviation and variance. Some of them can be strictly used as indicator (in case of prediction), some of them should be treated as supporting tool to resolve market i.e. momentum.

Exemplary statistical prediction models are:

- Linear regression prediction
- ARMA based
- Gaussian models

2.3.2. Genetic algorithm

Genetic algorithm is a metaheuristic based on the process of natural selection that belongs to class of evolutionary algorithms. This algorithm is mainly used to generate optimization and search problems solutions based on operators like mutation, crossover and selection. Optimization is huge case in algorithm development, by de default *backtrader* uses brute force. This approach is very slow but gives a glimpse, on average, how algorithm

could perform. Yet it is very slow hence faster solutions, like genetic algorithm are commonly used.

2.3.3. Machine learning

Machine learning uses advanced models in order to perform task without using explicit instructions, relying on inference and patterns. Machine learning algorithms, based on trading data, build a model in order to make predictions or decisions without being explicitly programmed to do so. Data mining is inherent part of creating machine learning model. Choosing proper features is be or not to be for a model. Most commonly used machine learning algorithms are

- Nearest Neighbor
- Naïve Bayes
- Decision Trees
- Linear Regression
- Support Vector Machines
- Neural Networks

2.4. Optimization

Optimization is about matching best performing attributes to algorithms, like period to be considered when calculating indicator or ranges of oscillators which should trigger trade. It plays huge role in algorithm performance, even best algorithms, with improperly matched parameters, may turn out to be worst ones and conversely, worst ones can behave like very profitable.

Figure 4 and 5 presents graphically how parameters may affect algorithm decision making. Line crossings are delayed in Figure 4 in comparison to figure 4, and sometimes there are no comparable crossings. As well in RSI indicator huge difference is visible where in Figure 4 oscillator line is volatile and sharp, whereas in Figure 5 smooth without notable peaks or declines.

Figure 4 Graphical interpretation of indicator parameters (sma_{short}= 50, sma_{long}= 200, rsi_{period} = 14)

Figure 5 Graphical interpretation of indicator parameters (sma_{short}= 10, sma_{long}= 40, rsi_{period} = 100)

For instance, taking into consideration algorithm from section 4.2.1, mean reversion based, optimization showed that between best and worst pair of parameters there is 731% difference in outcome value (starting value = 10.000).

Table 2 Algorithm described in 4.2.1 section, optimization results,
stock = "FB" 2012.01.01 – 2019.03.29

Rank	Value [10^3]	sma_{long}	sma_{short}	rsi_{top}	rsi_{bot}	rsi_{period}
1	79,11	240	60	75	13	12
2	77,56	210	80	75	21	22
110	29,20	150	80	75	37	32
111	29,04	180	80	65	37	22
260	10.00	240	80	95	29	52
261	9,73	120	40	65	29	22
262	9,51	150	20	65	13	12

As we can see, in case of $sma_{long} = 240$ parameter, it is seen in best performing and as well in worst performing configuration. On average, algorithm using "FB" stock, yielded 32.636 value.

Table 3 Algorithm described in 4.2.1 section, optimization results,
stock = "TSLA" 2012.01.01 – 2019.03.29

Rank	Value [10^3]	sma_{long}	sma_{short}	rsi_{top}	rsi_{bot}	rsi_{period}
1	162,8	120	80	85	29	12
2	118,8	150	80	85	29	12
3	118,0	180	80	75	29	12
153	22,70	150	80	75	37	32
253	12,50	240	60	75	13	12
259	10,00	240	80	95	29	52
262	9,02	240	40	95	37	52

Algorithm tested using "TSLA" stock yielded on average 36.735 value. What is interesting that even though the difference between average yielded values is ~12.5%, difference between best performing configurations is broad, 105%, and worst configurations

are nearly the same. The rank 1 configuration using "FB" stock was one of the worst using "TSLA" (253). In case of rank 110 using "FB" stock, the counterpart in "TSLA" measurement was comparable in rank and as well outcome (153) with 27% value difference. Surprisingly only one configuration yielded same value, 10.000 in both algorithms. This means that in this particular parameters, algorithm did not make any trades at all.

As we can see, optimization is as important as implementing properly working algorithm, although it is very CPU consuming task. By conducting researches we can determine what range of what parameters, on average, performs best on given stock. But parameters are not everything, some strategies may be created with a view to shine on bullish market hence they might not do best while stock is stagnating. But parameters are not everything, auxiliary descriptors like Sharpe ratio or maximum drawdown should as well be considered, because it is very likely that historically best working parameters configurations, might not perform best currently.

2.5. Testing methodology

2.5.1. Tests using historical data

Tests are conducted using *backtrader* framework. Each algorithm is tested using specific timeline and stocks and then compared to most simple and common strategy – buy and hold. Latter strategy is only about to buy a security and keep it for whole time period. It is common performance indicator, if algorithm does not beat its result, it is considered worthless. Initial trading money is set always to 10.000 and commission rate is given at 0.25%. Tests does not only calculate performance indicators but also graphically present all actions and account balance what is helpful while dealing with various bugs.

Comparing algorithms performance by only final outcome does not always show much information hence auxiliary performance measurements are included:

- Sharpe ratio – the ratio is the average returned earned in excess of the risk-free rate per unit of volatility or total risk: $Sharpe\ ratio = \frac{R_p - R_f}{\sigma}$ where R_p – return of portfolio, R_f – risk-free rate, σ – standard deviation of the portfolio's excess return. The better Sharpe ratio, the greater risk-adjusted performance. Usually ratio above 1.0 is considered as good, 2.0 very good, and above 3.0 excellent.
- Max drawdown – maximum observed loss for a peak to trough of a portfolio, before a new peak is reached. It is an indicator of downside risk over a specified time. $MDD\% = \frac{Trough\ Value - Peak\ Value}{Peak\ Value}$. Large drawdown suggests that down movements are volatile.
- Annualized return – the geometric average money earned by investment each year over a given time period.
- Total compound return – rate of return that represents the cumulative effects that a series of trades has on an original amount of capital over a period of time. It is more accurate measure than average returns to assess growth or decline in an investment.

2.5.2. Robustness tests

The robustness checks have different goals. By definition robustness is the share of the probability density distribution of the baseline model that falls within the 95% confidence interval of the baseline mode. More easily is to see how strategy (conclusions) behaves when stock prices (assumptions) change. But common reason for a robustness test is to demonstrate that main strategy definition is well defined. For instance, by historical data and one indicator we assume that one strategy will fit into this stock perfectly. Robustness test

checks if after changed price movement, strategy will still perform or at least is resistant to loses.

There are several approaches of robustness tests ([13]):

- Model variation – model variation tests change one or more model specification assumptions and replace with alternative assumptions. Examples: change in set of regressors, change in sample, change in functional form.
- Randomized permutation – change specification assumptions repeatedly. Examples: sensitivity tests (Leamer 1978), sample split, multiple imputation (King et al. 2001).
- Structured permutation – change a model assumption within a model space in a symmetric way. Changes in the assumption are based on a rule, rather than random. Examples: sensitivity tests (Levine and Renelt), jackknife tests, partial demeaning test.
- Robustness limit – test provides a way of analyzing structured permutation tests. These tests ask how much a model specification has to change to render the effect of interest non-robust. Examples: under and overrepresentation, measurement error, omitted variable correlation.
- Placebo – tests analyze whether a placebo treatment is correlated with the outcome. Examples: temporal lags, random regressors correlation, temporal lags.

The most frequently described and used tests in automated strategy modelling are variation based.

2.5.3. "80/20" tests

This approach is to divide historical data into two separate, training (80%) and test data (20%). Then strategy is tested and optimized, and with best performing parameters, later on tested using test data. By incorporating this approach we can more precisely gauge if strategy can perform well in training and optimization phase and afterwards as well in real time.

3. Risk management

Constant monitoring of the performance of trading strategy is required in order to maintain consistent returns in these days fast moving markets. Risk calculations, based on historical data and other calculations, help us to understand if our strategy takes too much risk. The risk management has to be incorporated already in strategy. There are several approaches and examples of risk management.

- One percent rule - this rule states that never more than 1% of capital or trading account should be put in single trade.
- Cutting losses – using stop-loss is designed to prevent from mentality that stock will eventually bounce back. For example, if a stock plunges below a key support level, traders often sell immediately their shares. The opposite tool to latter is take-profit.
- Diversify – all the money should be never put into one stock. Investments should be diversified among multiple stocks. Days where traders were specialized in one stock, for instance oil, are already gone (due to development of automated trading).
- Low Beta stocks – The higher the beta of a stock, the more volatile it is relative to stock market.

- Kelly Criterion – formula which shows, based on historical win percentage of given strategy and trader overall win/loss ratio, what percent of investor's capital to put into a single trade.
- Entry/Exit – entry and exit rules have to be well defined and respected. Trades should not be prolonged because maybe stock will rise, or fall.
- Record keeping and analysis – by gauging vital statistics like win, payoff, commission rations, largest, lowest and average winning and losing trades, consecutive losses, helpful data can be extracted to adjust strategy. For instance if our strategy makes profit only in 30% of trades, but that profit is relatively big, we can set for instance marginal stop-losses to leverage outcome.

4. Implementation

4.1. Introduction

In this chapter strategies implementation is discussed. Each discussion is broken down into three major parts: strategy description where used strategy and techniques are described, testing environment – in what conditions tests were conducted and finally optimization and summary – performance evaluation and possible flaws discovery. This chapter is divided into two major parts describing simple strategies (based mainly on indicators) and advanced strategies (advanced statistical model and optimization problem).

4.2. Simple strategies

4.2.1. Mean reversion

4.2.1.1 Description

Implemented strategy relies on Golden cross pattern and RSI indicator. Golden cross is a pattern when a short-term moving average (such as 15 or 50 day) crosses above a long term moving average (such as 50 or 200 day). The golden cross indicates a bull market (condition of market when prices are rising or expect to rise). RSI (Relative strength index), on the other, hand is a momentum indicator that calculates magnitude of recent price changes, hence indicates if asset price is over or underpriced. RSI is shown as an oscillator with range from 0 to 100. Most common interpretation of RSI is that values above 70 indicate that security is overvalued and values below 30 – undervalued. No leverages are used and 100% of money is used to buy securities.

Strategy algorithm:
Input: Data array *prices*, *stock* – variable determining which stock are we operating on, rsi_{top}, rsi_{bot} – RSI ranges, sma_{long}, sma_{short} – SMA indicators values, *rsi* – RSI indicator value.

1. **Set** $i \leftarrow 0$
2. **while** $i < length\ of\ prices$ **do**
3. $\boldsymbol{if}$ *no currently opened position* $\boldsymbol{then}$:
3.1. $\boldsymbol{if}\ rsi_i \leq rsi_{bot}\ \boldsymbol{and}\ sma_{short} > sma_{long}\boldsymbol{then}$:
3.1.1. $\boldsymbol{execute}\ order = buy(stock, price_i)$
4. $\boldsymbol{else\ if}\ rsi_i \geq rsi_{top}\ \boldsymbol{and}\ sma_{short} \leq sma_{long}\ \boldsymbol{then}$:
4.1. $\boldsymbol{execute}\ close(order)$
5. $i \leftarrow i + 1$
6. **end while**

4.2.1.2 Testing environment

Tests were conducted using stocks: „FB", "BTC-USD", "CLUB", "JPM", "TSLA", "TUWOY". Each of them represents different or comparable dynamics, behavior and possible patterns. For sake of simplicity, tests are conducted using timeframes from 2013.01.01 to 2019.03.29 (since "FB" stock had its initial public offering by 2012.05).

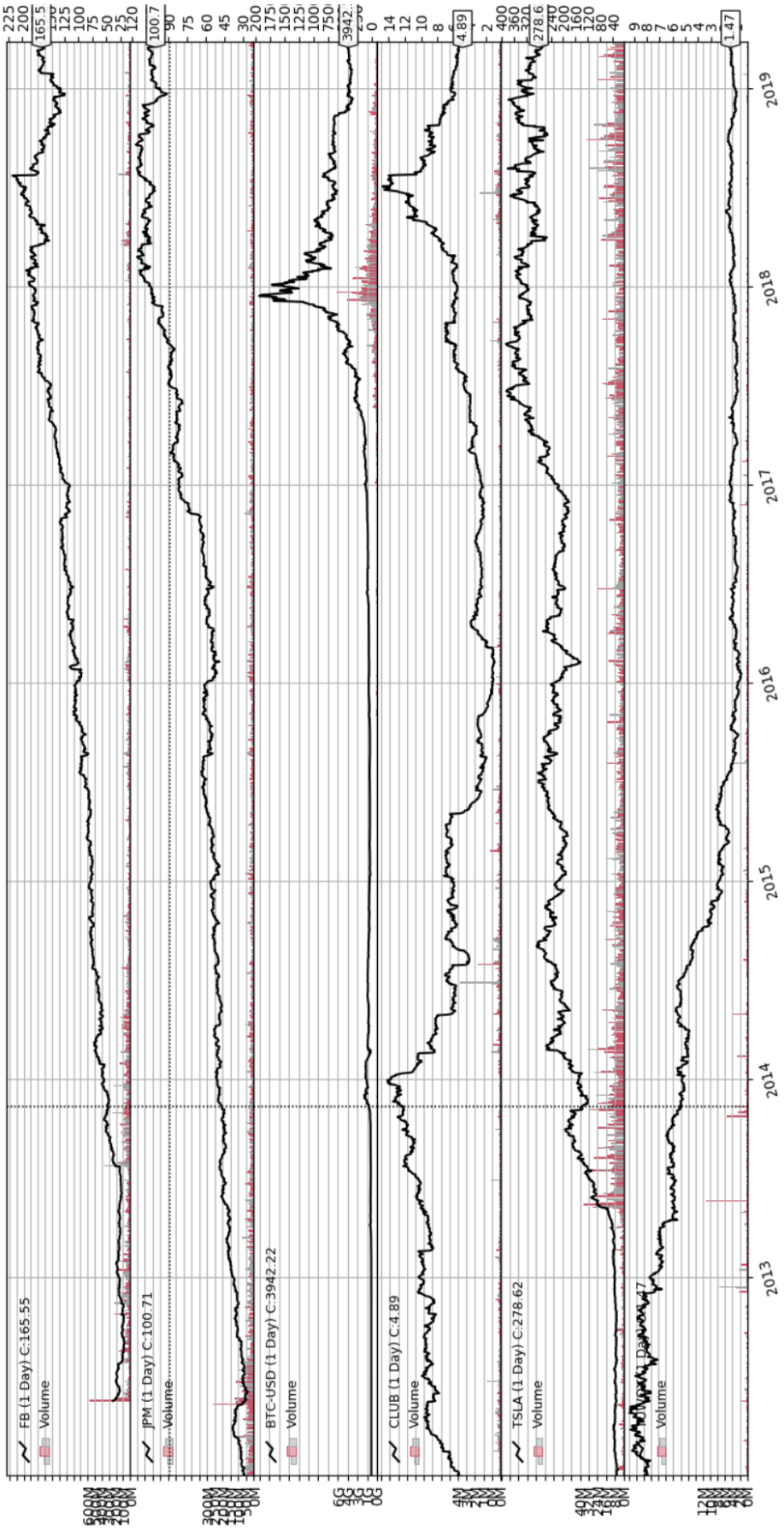

Figure 6 Stocks, used in tests, prices charts 2012.01.01 – 2019.03.29

For given timeframe, buy and hold strategy results are:

Table 4 Mean reversion, buy and hold strategy results, 2013.01.01 – 2019.03.29

Stock	FB	JPM	BTC-USD	CLUB	TSLA	TUWOY
Value [10^3]	58,65	26,31	2934	4,77	78,13	1,98

4.2.1.3 Tests

Recommended algorithm parameters are:
rsi_{top} = 70, rsi_{bot} = 30, sma_{short} = 50, sma_{long} = 200, rsi_{period} = 14.

For given parameters results were not impressive:

Table 5 Mean reversion based algorithm tests results for recommended paramenters, 2013.01.01 – 2019.03.29

Stock	Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return
FB	22,84	0,72	29%	14%	82%
JPM	23,10	0,82	22%	14%	84%
BTC-USD	537,9	0,58	91%	56%	399%
CLUB	3,65	-0.16	84%	-15%	-101%
TSLA	15,91	0,36	41%	8%	46%
TUWOY	4,34	-0,41	69%	-13%	-83%
Average	101,2	0,32	55,9%	10,75%	71,2%

After further investigation of results, it is clear that algorithm, in most of time, just holds money and does not make many any trades what is visible in Figure 7 and 8.

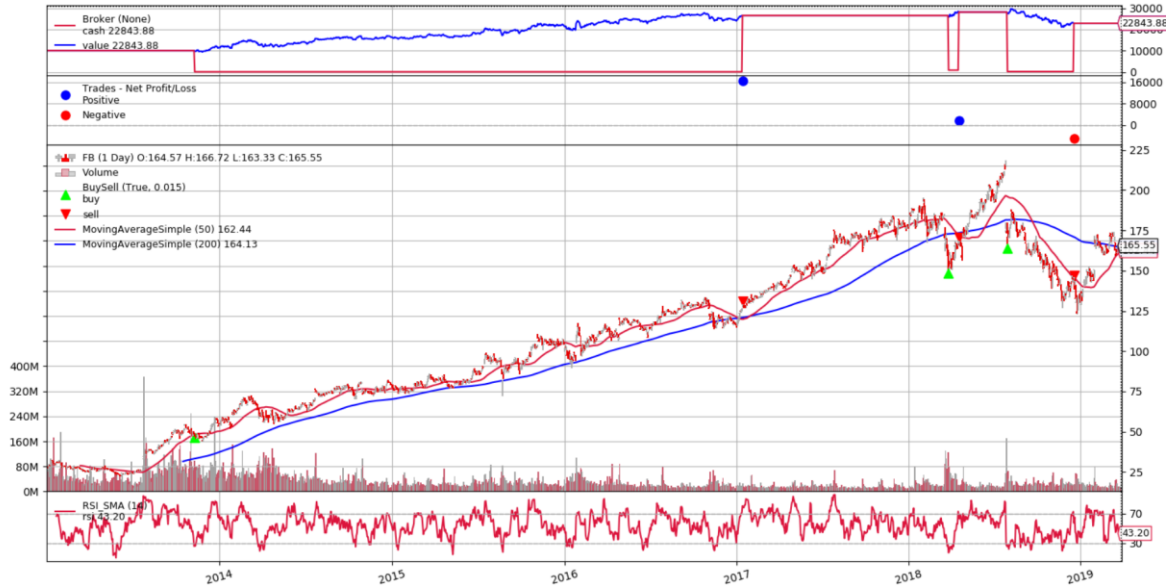

Figure 7 Algorithm graphical recap, stock "FB"

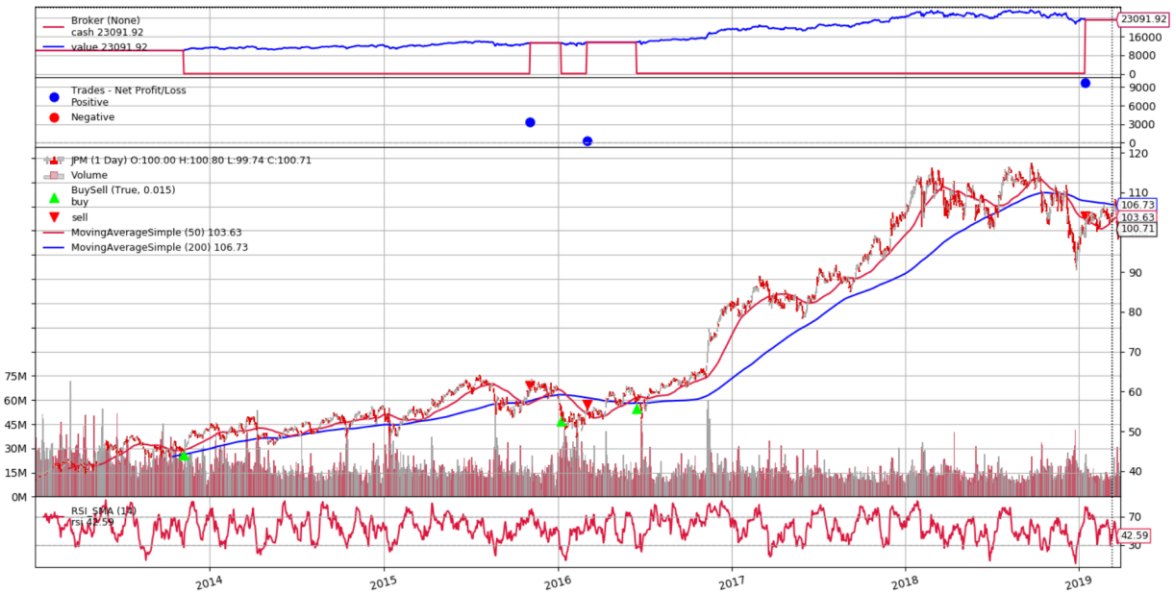

Figure 8 Algorithm graphical recap, stock "JPM"

Using recommended configuration showed that algorithm is just worse version of buy and hold strategy. Only few times underpaid prices were detected and afterwards sold in reasonable time. Sharpe rations are as well abysmal, on average 0,36. Average Max drawdown is huge, roughly 56%. Average value return is biased by BTC-USD outcome, which at first glance seem to be great (~400% compound return!), but considering its Sharpe ratio 0.58 and comparing result to standard B&H strategy unveils real performance.

4.2.1.4 Optimization

Since *backtrader* does not allow to include analyzers like Sharpe ratio in optimization tests, average max drawdown and Sharpe ratio are shown from outcome of most corresponding parameter algorithm outcome to average value of optimization tests.

Optimization was conducted using parameters values:

- $rsi_{top} \in \{65, 75, 85, 95\}$
- $rsi_{bot} \in \{5, 15, 25, 35, 45\}$
- $sma_{short} \in \{20, 40, 60, 80\}$
- $sma_{long} \in \{120, 150, 180, 210, 240\}$
- $rsi_{period} \in \{14, 24, 34, 44, 54\}$

Table 6 Mean reversion optimization results

Stock	Best Value $[10^3]$	Average Value $[10^3]$	Average Max drawdown	Average Sharpe ratio	Best to B&H value difference	Average to B&H value difference
FB	79,53	37,78	32%	0,86	36%	-34%
JPM	39,59	23,12	22%	0,59	50%	-12%
BTC-USD	14076	4089	92%	0,57	379%	39%
CLUB	28,27	5,09	93%	0,22	460%	6%
TSLA	107,4	39,29	60%	0,38	37%	-50%
TUWOY	19,79	5,75	52%	-0.27	900%	190%
Average	2391	700	58,5%	0,39	310%	23%

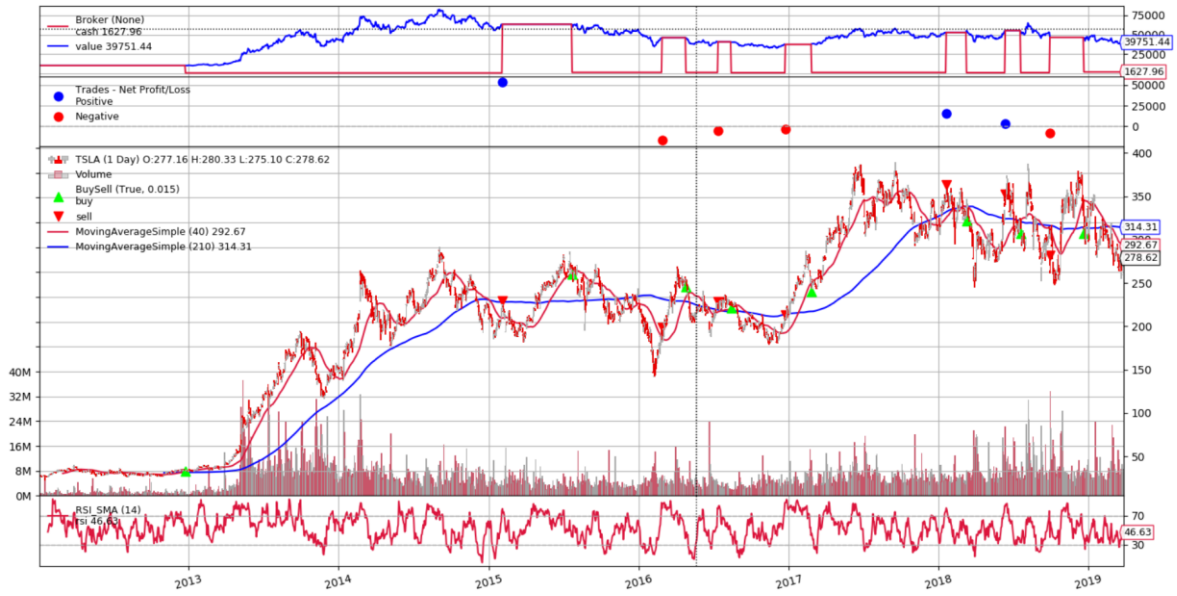

Figure 9 Average algorithm graphical outcome for „TSLA" stock

Yet again averages of average results are biased by "BTS-USD" and "TUWOY" stock. Nevertheless Sharpe ratio is still abysmal, max drawdown huge and overall by comparing to B&H strategy, average value performance is worse. In Figure 9 we can see an average outcome for "TSLA" stock which clearly shows that algorithm sometimes avoids price drops (2015.01 – 2015.06), but the value chart shows that earnings are following stock trends, hence whenever stock is losing, algorithm, in most cases, loses as well.

4.2.1.5 Summary

Strategy implementation and tests have proven high importance of algorithms back testing. If we are lucky, we might guess algorithm best parameter configuration and influenced by positive outcomes, hastily, assume that this is good performing algorithm. For our safety, additional analyzer indicators like Sharpe ratio and maximum drawdown exists, to show that even if algorithm is performing really well, in reality it is not that good as it seems (i.e. "BTC-USD"). Additionally is it worth to check graphical recap of algorithm behavior to look for potential flaws or bugs. In my opinion the only reasonable usage of such algorithm is to divide portfolio into multiple positions and assigning them to best performing algorithm configurations.

4.2.2. Momentum based

4.2.2.1 Description

Implemented strategy uses B&H strategy but equally divides portfolio into top 10 best performing stocks of S&P 500 (this index includes top 500 best market capitalization large companies listed on NYSE or NASDAQ). The geometric average rate of return is used to sort and chose desired stocks. This strategy is based on logic to buy winners and sell losers that are less likely to be in the final stages of overreaction.

GARR ratio formula is as follows:

$$R_{month} = \frac{close\_price_{month} - open\_price_{month-1}}{open\_price_{month-1}}$$

$$GARR_{last\ 12\ months} = (1 + R_1)^{\frac{1}{12}} \cdot (1 + R_2)^{\frac{1}{12}} \dots (1 + R_{11})^{\frac{1}{12}} \cdot (1 + R_{12})^{\frac{1}{12}} - 1$$

$$GARR_{last\ 12\ months} = (1 + R_{last\ month})^{\frac{1}{12}} - 1$$

$$GARR_{ratio} = \frac{GARR_{last\ month}}{GARR_{last\ 12\ months}}$$

The algorithm is as follows: by $GARR_{ratio}$ given stocks are sorted and top 10 bought. Every month stocks are sorted again by current values and portfolio is reassigned, thus stock which have dropped out of top 10 list are sold, and new best anticipated bought.

4.2.2.2 Testing environment

For testing, nearly all 500 stocks, included in S&P 500 index, were chosen with exception to around 30 which were flawed by data inconsistency or not existing from 2012-01-01.

As comparison, B&H strategy is shown, using S&P index. Final B&H value outcome for 2013.01.01 to 2019.03.01 is 18,400 (from date one year shift is due to itself GARR formula, which requires prices from previous year). As reminder, commission rate is 0.25%, and starting value 10.000.

4.2.2.3 Tests

Recommended parameters are: $number\ of\ stocks\ to\ hold\ =\ 10$ – number of stocks that are held for one month.

Table 7 Momentum strategy results, stocks to hold = 10, 2013.01.01 – 2019.03.01

Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return	Hold swaps	Total fees value [10^3]
8,18	-0,22	45,53%	-44%	-20%	693/750	3,49

As we can see, outcome is just bad. But what is interesting that 693 out of 750 possible hold swaps occurred (hold swap – finding superior $GARR_{ratio}$ stock to one currently held) hence enormous total fees paid to broker (35% of initial value).

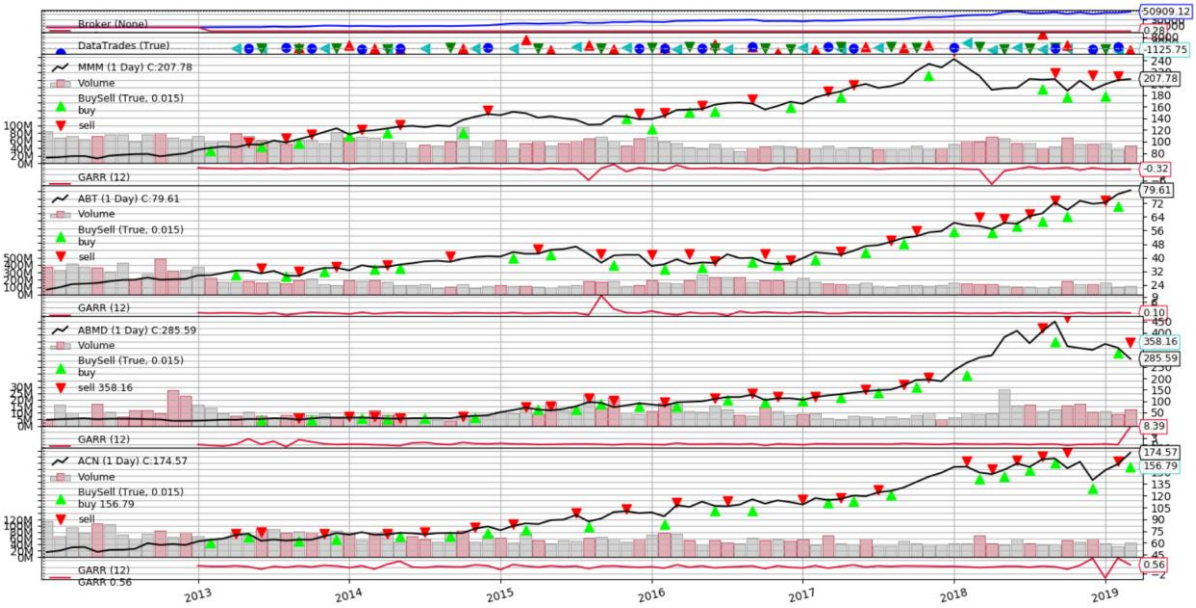

Figure 10 Exemplary graphical recap. Number of stock to hold = 2, „MMM", „ABT", „ABMD", „ACN" stocks included. Value outcome = 50.909

4.2.2.4 Optimization

Optimization was run with range of parameters: $number\ of\ stocks\ to\ hold\ =\ 2-30$.

Table 8 Momentum strategy optimisation results, 2013.01.01 – 2019.03.01

Rank	Number of stocks	Value [10^3]	B&H difference
1	23	52,26	181%
2	24	50,06	170%
3	25	47,62	154%
27	3	11,58	-37%
28	6	11,47	-38%
29	2	10,92	-41%
Average	-	21,90	18%

On average, the more stocks we held, the better average value outcome is. As usually there is big difference between best and worst value, 138%.

Additionally tests using descending number of random stocks were conducted to check, if really relationship between amount of stocks and value is valid.

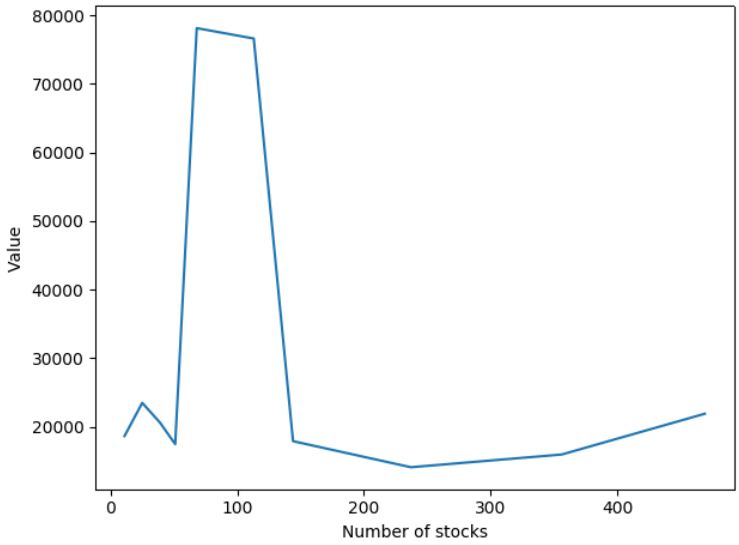

Figure 11 Momentum strategy, number of included stocks dependence

That assumption was wrong. The less stocks we include, the more final outcome is biased by single stocks, whereas more stocks included average value. However, top performing *number of stocks* parameter, among tests results, averaged 17,5.

4.2.2.5 Summary

The overall idea of momentum trading is clever. Instead of separating portfolio equally (or like in this case, one S&P 500 index), we are constantly looking for most promising stock based on historical data, hence averaging risk. This strategy clearly works well with ascending stocks prices without plunges, outperforming traditional buy and hold strategy. This strategy would need another, more prediction like indicator (or fundamental analysis), serving as auxiliary determinant if stocks is worth to hold for next time period. Nevertheless recommended default values for algorithm have failed to yield positive final value.

4.2.3. Statistics prediction based strategy

4.2.3.1 Description

Implemented strategy relies on linear regression, relative strength index and Bollinger bands. Linear regression is a model that allows us to summarize relationship between two variables. It simply shows trend over given period of time. It is especially useful when prices are highly volatile. Bollinger bands is an indicator that is defined by two sets of lines which are standard deviations (negative and positive), away from, simple moving average indicator. Strategy uses linear regression to predict when to make trades. If linear prediction is below bottom Bollinger band, it should buy and if, on contrary, it is above top Bollinger band it has to close position. In addition, RSI is used as confirming tool, if buy signal is triggered, it is checked if RSI indicator is below desired value and if it is, securities are bought with 2.0 leverage multiplier, if not, with 1.0.

Strategy algorithm:
Input: Data array *prices*, *stock*– variable determining which stock are we operating on, *available_money*, rsi_{bot} – RSI range, *rsi* – RSI current indicator value, $boll_{top}$, $boll_{bot}$ – Bollinger bands values, *linearReg* –linear regression prediction.

1. **Set** $i \leftarrow 0$
2. **while** $i < length\ of\ prices$ **do**
3. $\boldsymbol{if}$ *no currently opened position* $\boldsymbol{then}$:
3.1. $\boldsymbol{if}$ $linearReg_i < boll_{bot}$ $\boldsymbol{then}$:
3.1.1. **Set** orderMark $\leftarrow True$
3.1.2. $\boldsymbol{if}$ $rsi \leq rsi_{bot}$ $\boldsymbol{then}$:
3.1.2.1. $\boldsymbol{execute}$ $order = buy(stock, price_i, 2.0)$
3.1.2.2. **Set** orderMark $\leftarrow False$
3.2. $\boldsymbol{else\ if}$ $orderMark$
3.2.1. $\boldsymbol{execute}$ $order = buy(stock, price_i, 1.0)$
3.2.2. **Set** orderMark $\leftarrow False$
4. $\boldsymbol{else\ if}$ $linearReg_i > boll_{top}$ $\boldsymbol{then}$:
4.1. $\boldsymbol{execute}$ $close(order)$
5. $i \leftarrow i + 1$
6. **end while**

4.2.3.2 Testing environment

Testing environment is the same as in Testing environment(4.2.1.2) section where only timeframe is changed to 2016.05.18 – 2019.03.29

For given timeframe, buy and hold strategy results are:

Table 9 Statistic prediction based strategy, buy and hold strategy results, 2016.05.18 – 2019.03.29

Stock	FB	JPM	BTC-USD	CLUB	TSLA	TUWOY
Value [10^3]	14,07	16,95	88,63	15,72	12,62	11,23

4.2.3.3 Tests

Recommended parameters are: $rsi_{period} = 12$, $boll_{period} = 20$ – Bollinger bands period, $boll_{dev\_factor} = 1.0$ – Bollinger bands standard deviation multiplier, $rsi_{bot} = 40$ – RSI value requirement to trigger leverage buy, $linear\_regression_{period} = 60$ – linear regression period.

Table 10 Statistical prediction based algorithm tests results for recommended paramenters, 2016.05.18 – 2019.03.29

Stock	Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return
FB	13,93	0,37	52%	12%	33%
JPM	12,13	0,37	25%	7%	19%
BTC-USD	2,01	-0.02	99%	-32%	-160%
CLUB	6,71	0.12	87%	-13%	-40%
TSLA	13,10	0,38	52%	10%	27%
TUWOY	13,90	0.37	70%	12%	33%
Average	10,29	0,26	64%	-0.5%	-14,5%

As we can see, B&H strategy turn out to be superior over strategy with two small exceptions, “TSLA” and “TUWOY” stocks. Recommended parameters results are abysmal. But it proved that it works under high volatile conditions, as we can see in Figure 12, we just have one peak, and pull, hence bad outcome result, whereas in Figure 13 stock moves very volatile, yielding positive outcome value.

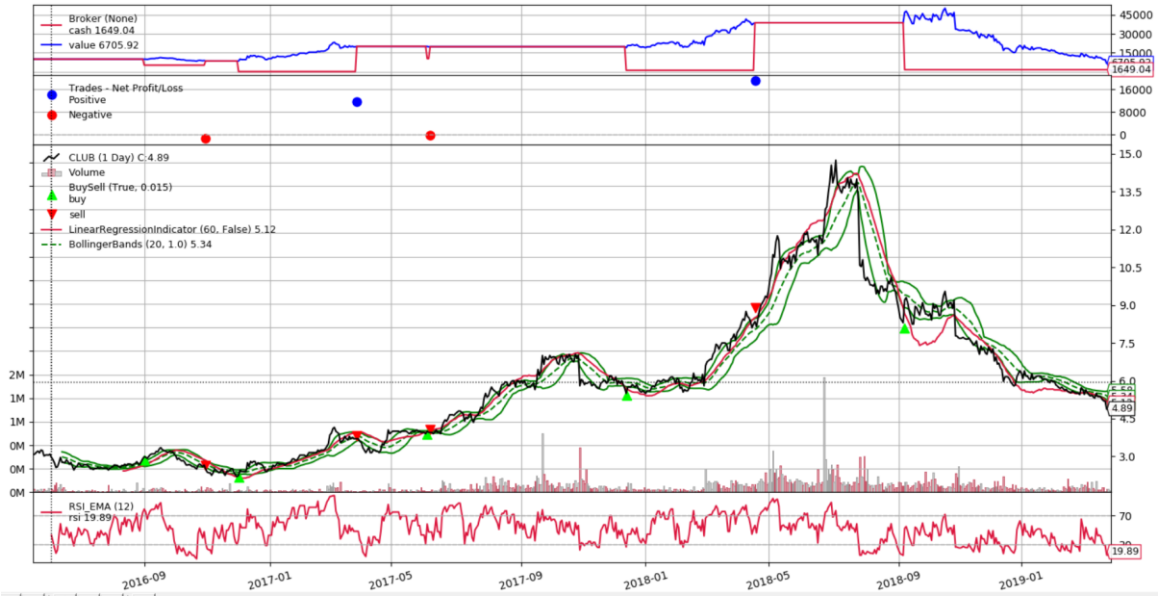

Figure 12 Algorithm graphical recap, stock „CLUB”

Figure 13 Algorith graphical recap, stock „TUWOY"

It is noticeable as well that algorithm tends to wait too long to enter position. It is due to **3.1** algorithm logic line (described in section Description(4.2.3.1)), where it just waits until RSI reaches desirable value in order to buy leveraged securities.

4.2.3.4 Optimization

Optimization was conducted using parameters values:

- $rsi_{period} \in \{12,15,18,21,24,27\}$
- $boll_{period} \in \{20,30,40,60,90\}$
- $boll_{dev\_factor} \in \{0.7,0.8,0.9,1.0,1.1,1.2,1.3\}$
- $rsi_{bot} \in \{20,25,30,35,40\}$
- $linearRegression_{period} \in \{20,40,60,80,100\}$

Table 11 Statistical prediction based algorithm, parameters optimization, 2016.05.18 – 2019.03.29

Stock	Best Value [10^3]	Average Value [10^3]	Average Max drawdown	Average Sharpe ratio	Best to B&H value difference	Average to B&H value difference
FB	24,01	15,27	42%	0,43	71%	7%
JPM	15,87	12,02	34%	1.19	-6%	-29%
BTC-USD	90,37	7,00	76%	-0,26	2%	-92%
CLUB	9,80	3,12	90%	-0.29	-37%	-80%
TSLA	63,02	23,91	19%	0.60	404%	92%
TUWOY	58,70	27,26	53%	0.81	422%	142%
Average	43,63	14,66	58,5%	0,41	285%	6,66%

Optimized algorithm, on average, beats by 6,66% B&H strategy, hence it's the worst one from previously tested strategies.

4.2.3.5 Summary

This algorithm has some flaws. First and foremost, it delays buying security it order to wait until it reaches desired rsi_{bot} level to buy leveraged stock. Secondly, closing trigger frequently does not or triggers too quickly. Highly volatile stocks are preferred to use this strategy, since it has proved its usefulness on those. Linear regression is reliable if it comes to predicting more generally, like trend, although it is too unprecise in forecasting next day price.

4.3. Advanced Strategies

4.3.1. Genetic algorithm

4.3.1.1 Description

Algorithm is based on [14] strategy, which was meant to be used on future contracts. It involves 5 different indicators, giving in total 12 parameters to be optimized, hence is it good candidate for genetic algorithm usage. Since algorithm is using equities, not futures, it is adjusted to stocks environment. Change incorporates keeping opened position as long as strategy formula says to (unenhanced version was constantly buying and selling every day, not considering currently opened orders).

Strategy algorithm:
Input: Data array *prices*, *stock* – variable determining which stock are we operating on, roc - rate of change indicator parameter value, roc_i – current rate of change, rsi_{change} – is difference between previous and current RSI indicator value, rsi – RSI indicator value, $percD, percK$ are stochastic oscillator indicator parameter values described in [15], $macD$ – MACD indicator parameter value described in [16], $macD_{i-1}$ – previous day MACD value.

1. **Set** $i \leftarrow 0$
2. **while** $i < length\ of\ prices$ **do**
3. ***if*** $roc_i > roc$ ***and*** $rsi_{change} < rsi$ ***and*** $percD_{i-1} < percD$ ***and***
4. $oscAD_{i-1} < oscAD$ ***and*** $macD_{i-1} < macD$ ***and*** $percK_i < percK$ ***then***:
4.1. ***if*** *no opened order* ***then***:
4.1.1. ***execute*** $buy(stock, price_i)$
4.2. ***else if*** $order.type == SELL$
4.2.1. ***execute*** $close(order)$
4.2.2. ***execute*** $buy(stock, price_i)$
5. ***else***:
5.1. ***if*** *no opened order* ***then***:
5.1.1. ***execute*** $sell(stock, price_i)$
5.2. ***else***:
5.2.1. ***execute*** $close(order)$
5.2.2. ***execute*** $sell(stock, price_i)$
6. $i \leftarrow i + 1$
7. **end while**

4.3.1.2 Testing environment

For sake of clearer comparison to prior algorithms results, it is tested on stocks from Testing environment(4.2.1.2). For genetic optimization 25 generations, each of 150 population was used with mutation rate of 0.1 (operator – Gaussian) and crossover – 0.7 (operator – two-point crossover). To select best individuals among population, tournament selection is used.

4.3.1.3 Optimization and tests

Optimization was run in range of parameter values:

- *Rate of change* $\in < -99{,}99 >$
- $RSI_{change} \in < -99, 99 >$
- $percD \in < 0.00, 100.00 >$
- $percK \in < 0.00, 100.00 >$
- $oscAD \in < 0.00, 100.00 >$
- $MACD \in < 0.00, 100.00 >$
- $RSI_{period} \in < 3, 100 >$
- $Stoch_{period} \in < 3, 100 >$
- $oscAD_{period} \in < 3, 100 >$
- $MACD\_me1_{period} \in < 3, 100 >$
- $MACD\_me2_{period} \in < 3, 100 >$
- $Rate\ of\ change_{period} \in < 12, 36 >$

Table 12 Genetic algorithm, parameters optimization results, 2012.01.01 – 2019.03.29

Stock	Best Value $[10^3]$	Max drawdown	Sharpe ratio	Best to B&H value difference
FB	69,58	29%	1,83	25%
JPM	27,97	29%	0,71	6%
BTC-USD	482,2	114%	0,52	-83%
CLUB	18,10	129%	0,32	279%
TSLA	147,3	45%	0,45	89%
TUWOY	130,0	41%	0,94	6465%
Average	2628	64%	0,80	1130%

The results on average were much better than B&H strategy and as well previous algorithms. As it turned out, this algorithm performed very well on declining stocks ("CLUB" and "TUWOY), where previously tested algorithms are struggling. On contrary, on ascending stocks ("FB", "JPM", "TSLA") it performed decently. If it comes to the "BTC-USD" stock, this strategy didn't work well. Although it yielded reasonable outcome value, optimization was finding only solution which would buy at earliest and hold as long as possible, hence it wasn't constantly trading. As we can see in figures 14, 15 and 16, algorithm, in this cases, was opening and closing positions very frequently and for short time (mostly 1 - 3 days). This strategy proved to be efficient, having ~70% accuracy, hence there is possible improvement on adding leverage which could improve outcome a lot.

Figure 14 Algorithm graphical recap, stock „FB”

Figure 15 Algorithm graphical recap, stock „JPM”

Figure 16 Algorithm graphical recap, stock „TUWOY"

Optimization using genetic algorithm is quite CPU consuming, hence all indicators have to be calculated, where some of them use basic formula containing few mathematic operations, whereas other can be particularly sophisticated. Population of 150 was the best compromise between outcome quality and time cost. The negative side of using genetic algorithm for optimization is that it does not show whole context since only best (or local best minimums) are considered, showing us only the "bright" side of algorithm. As we could see in previous optimization tests, averaging values showed what we can expect and looking only by best outcomes we could fell into a "trap", what would result in sudden money plunge, if one algorithm would be used in real market.

4.3.1.4 Summary

This strategy performed really well on plunging "CLUB", "TUWOY" markets and decently (in comparison to B&H strategy) in inclining ones – "TSLA", "JPM", "FB". Yet again on "BTC-USD", the best strategy was just to buy and hold as soonest as possible, therefore I assume that this strategy was not working as expected, especially considering that is should trade frequently.

Genetic algorithm proved to be efficient in finding best solutions over big space of possible outcomes, nevertheless it does not show averages. Local minima are another problem because sometimes the best outcome might be that average one, hence genetic algorithm parameters should be chosen carefully and genetic optimization run at least few times.

4.3.2. Machine learning based

4.3.2.1 Description

Strategy was derived from [17] – "Stage 1" from Table 3. Parameters from "Stage 1" are incorporated into algorithm and random forest regression (*RFR*) and boosted regression tree (*BRT*) are tested since they performed the best out of all machine learners' algorithms mentioned in publication. In addition, to show learner when to buy or sell, simple strategy was included, which was looking into future prices to determine if to sell, buy or hold stocks. Training (2012.01.03 – 2016.08.09) and testing data (2016.08.10 – 2019.03.29) was divided as it is visible in Figure 17 which shows exemplary strategy run. Not all trades were positive,

since for testing purpose, randomized closing transaction time was used, which will be explained later on.

Figure 17 Machine learning, strategy for feature extraction, graphical recap

4.3.2.2 Testing environment

Tests are conducted using standard six stock list, described in Testing environment(4.2.1.2). For training purposes, prices from 2012.01.03 to 2016.08.09 are incorporated, and for testing 2016.08.10 – 2019.03.29.

One strategy with two different close trade executions are tested: first one with random delay between time when order should be closed and day period, what is checking when it should be closed in order to yield positive outcome, of ¼ and second of ¾. For example, if day checking period is 20 and close trade is detected, real trade close day is $currentDay + random(\left(20 * \frac{1}{4}\ or\ \frac{3}{4}\right),\ 20)$. Learned algorithm was tested using from 50 to 500 estimators with jump of 50.

4.3.2.3 Tests

In total, as features, indicator values are taken:

- Linear Regression – *period* = 20
- Bollinger Bands, top mid and bottom – *period* = 20, *devfactor* = 1,5
- Relative Strength Index – *period* = 20
- Moving Averages – first *period* = 25 and second *period* = 100
- Moving Average Convergence / Divergence – $period_{me1}$ = 12, $period_{me2}$ = 20, $period_{signal}$ = 9
- On Balance Volume indicator, described in [18].
- Close price
- Volume

Table 13 Machine learning, learner results in comparison to base learner strategy outcome, RFR learner, ¼ close delay, "BTC-USD", 2016.08.10 – 2019.03.29

Base Value [10^6]	Average Value [10^3]	Average max drawdown	Average Sharpe ratio
432	6,74	51%	-0,98
963	5,31	55%	-1,05
1856	7,83	40%	-0,79
2992	5,35	51%	-1,06
5538	10,2	48%	-0,01
6122	6,96	52%	-0,92
13938	11,01	57%	-0,24

Table 14 Machine learning, learner results in comparison to base learner strategy outcome, RFR learner, ¼ close delay, "CLUB", 2016.08.10 – 2019.03.29

Base Value [10^3]	Average Value [10^3]	Average max drawdown	Average Sharpe ratio
355	18,72	24%	0,31
392	19,24	29%	0,40
788	30,38	30%	0,60
1024	19,90	43%	0,42
1640	13,18	47%	0,31
1672	27,26	29%	0,66
1840	20,66	32%	0,51

Table 15 Machine learning, learner results in comparison to base learner strategy outcome, RFR learner, ¾ close delay, "BTC-USD", 2016.08.10 – 2019.03.29

Base Value [10^6]	Average Value [10^3]	Average max drawdown	Average Sharpe ratio
138	26,00	34%	0,77
216	25,82	32%	0,68
249	24,31	35%	0,70
361	19,85	30%	0,54
378	5,69	53%	-0,91
410	8,16	39%	-0,59
467	28,27	30%	0,70

Table 16 Machine learning, learner results in comparison to base learner strategy outcome,

RFR learner, ¾ close delay, “CLUB”, 2016.08.10 – 2019.03.29

Base Value [10^3]	Average Value [10^3]	Average max drawdown	Average Sharpe ratio
304	31,80	26%	0,52
386	28,32	21%	0,47
786	18,98	33%	1,08
843	18,35	58%	0,36
934	16,65	45%	0,39
1220	37,82	24%	0,62
1312	14,42	25%	0,46

Adding randomness factor to closing trade day showed that (results visible from Table 13 to 16) there is no explicit correlation between training strategy outcome and test data value. Sometimes best training learner was performing the worst, and sometimes the best. The same conclusion appears to best, worst and average base values.

Table 17 Machine learning, results,

BRT, ¾ close delay, 2016.08.10 – 2019.03.29

Stock	Average Value [10^3]	Average max drawdown	Average sharpe ratio	Best to B&H value difference
FB	12,36	31%	0,32	-12%
JPM	8,52	28%	-0,49	-50%
BTC-USD	34,51	67%	0,60	-61%
CLUB	13,56	63%	0,35	-13%
TSLA	6,88	69%	-0,37	-45%
TUWOY	10,93	51%	0,19	-2%
Average	14,46	51%	0,10	-30%

Table 18 Machine learning, results,

RFR, ¾ close delay, 2016.08.10 – 2019.03.29

Stock	Average Value [10^3]	Average max drawdown	Average sharpe ratio	Best to B&H value difference
FB	8,88	45%	-0,32	-36%
JPM	6,25	43%	-1.01	-63%
BTC-USD	34,42	59%	0,59	-61%
CLUB	7,89	61%	-0,28	-49%
TSLA	5,30	65%	-0,47	-58%
TUWOY	10,24	47%	0,19	-10%
Average	12,15	54%	-0,12	-46%

Tests that results are visible in Table 17 and 18 are averaged from five runs, each consisting of using from 50 to 500 estimators with 50 jump. On average BRT learner performed better that RFR, although both average outcome is much worse than B&H strategy provides.

4.3.2.4 Summary

Machine learning based learners proved to be worse than ordinary, indicator based, strategy, even having near perfect backbone strategy as feature extractor. That shows that, there is no strict correlation between price movement, indicators and final outcome. Machine learning algorithms are reacting more or less always the same, based on previously learned schemas. Stock market mostly follows patterns, and meanwhile moves more or less randomly. Adding more features (like stop-loss) or media news could probably enhance algorithm a lot. Although based on poor and random performance, I do not think that it could improve much more (for example having at least B&H strategy results on average).

5. Algorithms adjustments

5.1. Introduction

In this days markets are changing rapidly, hence in most cases algorithms performance depletes over time. Ideas which were working before might not be current for variety of reasons like common adoption and usage of underlying strategy, hardware performance or software issues. Algorithms performance has to be measured and evaluated every day. In this chapter previously described algorithms are adjusted by proposed or new ideas. Every algorithm is tested using the same environment for a sake of clearer comparison. Backbone of every strategy stays more or less the same.

5.2. Simple strategies

5.2.1. Mean reversion

5.2.1.1 Description

The basics of strategy are the same as in section Mean reversion(4.2.1), but we are taking an advantage of downtrend, signaled by negative golden cross. When shorter SMA indicator is above longer one, buy state is set and on contrary, when shorter SMA line is below longer, state is set to sell. In addition we use Acceleration Deceleration Oscillator as auxiliary determinant, in addition to RSI, to close trades.

Strategy algorithm:
Input: Data array *prices*, *stock* – variable determining which stock are we operating on, rsi_{top}, rsi_{bot} – RSI ranges, *rsi* – RSI indicator value, sma_{long}, sma_{short} – SMA indicators values, *osc*, osc_{top}, osc_{bot} – acceleration deceleration oscillator parameters

1. **Set** $i \leftarrow 0$
2. **while** $i < length\ of\ prices$ **do**
3. $\boldsymbol{if}\ sma_{short} > sma_{long}\ \boldsymbol{and}\ state == SELL\ \boldsymbol{then}$:
3.1. $\boldsymbol{Set}\ state \leftarrow BUY$
3.1.1. $\boldsymbol{if}\ position\ is\ opened\ \boldsymbol{then}$:
3.1.1.1. $\boldsymbol{execute}\ close(position)$
4. $\boldsymbol{else\ if}\ sma_{short} \leq sma_{long}\ \boldsymbol{and}\ state == BUY\ \boldsymbol{then}$:
4.1. $\boldsymbol{Set}\ state \leftarrow SELL$
4.1.1. $\boldsymbol{execute}\ close(position)$
5. $\boldsymbol{if}\ no\ position\ is\ opened\ \boldsymbol{then}$:
5.1. $\boldsymbol{if}\ osc \leq osc_{bot}\ \boldsymbol{then}$:
5.1.1. $\boldsymbol{if}\ state == BUY\ \boldsymbol{then}$
5.1.1.1. $\boldsymbol{execute}\ buy(stock, price_i)$
5.1.2. $\boldsymbol{else\ if}\ state == SELL$ **then**:
5.1.2.1. $\boldsymbol{execute}\ sell(stock, price_i)$
6. $\boldsymbol{else}$:
6.1. $\boldsymbol{if}\ (osc_i \geq osc_{top}\ \boldsymbol{or}\ rsi_i > rsi_{top})\ \boldsymbol{and}\ state == BUY\ \boldsymbol{then}$:
6.1.1. $\boldsymbol{execute}\ close(position)$
6.2. $\boldsymbol{if}\ (osc_i \leq osc_{bot}\ \boldsymbol{or}\ rsi_i < rsi_{bot})\ \boldsymbol{and}\ state == SELL\ \boldsymbol{then}$:
6.2.1. $\boldsymbol{execute}\ close(position)$
7. $i \leftarrow i + 1$
8. **end while**

5.2.1.2 Testing environment

The testing environment is exactly the same as it is in Testing environment(4.2.1.2). Stocks used: „FB", "BTC-USD", "CLUB", "JPM", "TSLA", "TUWOY". Timeframe: 2013.01.01 to 2019.03.29

5.2.1.3 Tests

Previous recommended values are used:

rsi_{top} = 70, rsi_{bot} = 30, sma_{short} = 50, sma_{long} = 200, rsi_{period} = 14.

Additionally, new parameters are introduced with given default values:

osc_{period} = 4, osc_{top} = 2 (0.1 for "TUWOY"), osc_{bot} = -1 (-0.05 for "TUWOY")

Oscillator parameter values osc_{bot} and osc_{top} have to be lowered because of low share price.

Table 19 Mean reversion adjusted algorithm test results, 2012.01.01 – 2019.03.29

Stock	Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return	Difference to non-adjusted algorithm
FB	13,93	0,37	52%	12%	33%	-39%
JPM	12,13	0,37	25%	7%	19%	-11%
BTC-USD	2,01	-0.02	99%	-32%	-160%	-99,6%
CLUB	6,71	0.12	87%	-13%	-40%	83%
TSLA	13,10	0,38	52%	10%	27%	-17%
TUWOY	8,32	0.37	70%	12%	33%	91%
Average	10,29	0,26	64%	-0.5%	-14,5%	1,3%

On average algorithm performed worse than non-adjusted, with exception to "CLUB" and "TUWOY" stocks, which were both declining since beginning of testing timeframe. That shows that adding shorting has helped this algorithm to deal with bear market, whereas in times of bull market, aggravated final outcome value in comparison to non-adjusted algorithm.

5.2.1.4 Optimization

Optimization was conducted using parameters values:

- $rsi_{period} \in \{14, 24, 34, 44\}$
- $osc_{period} \in \{4, 6, 8, 10\}$
- $rsi_{top} \in \{75, 80, 85, 90, 95\}$
- $rsi_{bot} \in \{5, 10, 15, 20, 25\}$
- $osc_{top} \in \{1, 2, 4, 6\}$ ("TUWOY" $\in \{0.009, 0.01, 0.03, 0.005\}$)
- $osc_{bot} \in \{-1. -2, -4\}$ ("TUWOY" $\in \{-0.009, -0.01, -0.03, -0.005\}$)
- $sma_{short} \in \{10, 30, 60, 90, 110, 130\}$
- $sma_{long} \in \{50, 80, 120, 150, 180, 210\}$

Table 20 Mean reversion adjusted algorithm, parameters optimization, 2012.01.01 – 2019.03.29

Stock	Best Value [10^3]	Average Value [10^3]	Average Max drawdown	Average Sharpe ratio	Best to B&H value difference	Average to B&H value difference
FB	42,50	21,21	47%	0,46	-23%	-62%
JPM	22,52	14,51	21%	0,32	-14%	-45%
BTC-USD	28,24	1,92	93%	-0,01	-99%	-99,9%
CLUB	9,21	6,54	36%	-0,46	93%	37%
TSLA	37,05	6,40	79%	0,01	-52%	-92%
TUWOY	12,81	4,64	67%	-0.21	546%	134%
Average	43,63	14,66	58,5%	0,41	75%	-21%

Adjusted algorithm performed much worse on average than B&H strategy. As we can observe in Figure 14 and 15, it seems unstable. In some cases it buys a lot (and wastes a lot value on commissions with these day trading trades), and sometimes not. This problem regards acceleration deceleration oscillator since his output values are shown directly as price movements from moving average. On the positive side is the fact that it performed better against B&H and non-adjusted algorithm using stocks "CLUB" and "TSLA" proving that shorting addition in algorithm works but decreases final yielded value.

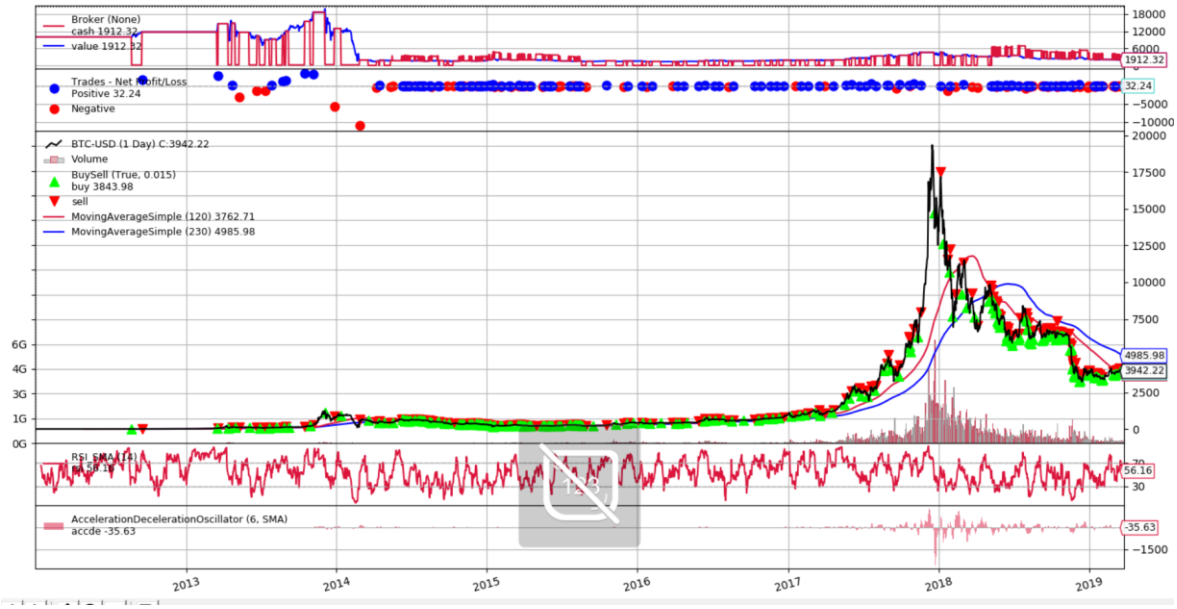

Figure 18 Algorithm graphical recap, stock „BTC-USD"

Figure 19 Algorithm graphical recap, stock „FB"

To overcome algorithm instability regarding acceleration deceleration oscillator, additional tests were conducted, but using dynamic calculation of parameters osc_{top} and osc_{top}, depending on current closing value. The formula was as follows: $osc_{top} = close_{price} * \frac{osc_{top\_input\_param}}{1000}$ and $osc_{bot} = -close_{price} * \frac{osc_{bot\_input\_param}}{1000}$

Best configurations for each stock were tested with change to parameters:

- $osc_{top} \in 0 - 400$ with jump 20
- $osc_{bot} \in 0 - 400$ with jump 20

Table 21 Mean reversion adjusted algorithm, oscillator parameter optimization, 2012.01.01 – 2019.03.29

Stock	Best Value [10^3]	Average Value [10^3]	Average Max drawdown	Average Sharpe ratio	Best to B&H value difference	Average to B&H value difference
FB	60,90	39,17	42%	0,70	9%	-29%
JPM	33,80	21,50	27%	0,71	29%	-18%
BTC-USD	15629	3370	96%	0,50	432%	14%
CLUB	2,36	-3,3	268%	-0,41	-50%	-169%
TSLA	25,20	12,21	86%	0,27	-67%	-84%
TUWOY	18,01	6,34	119%	0,21	809%	220%
Average	2628	574	106%	0,41	193%	-11%

Comparing to non-adjusted mean reversion algorithm results, Optimization(4.2.1.4), average and best values are worse (excluding "BTC-USD" which is roughly even). These results are derived from best parameter configuration without dynamic parameter values, although results proved that dynamic price dependent parametrization (if it is logically explainable), can improve result (which can still be improved by running another optimization, with whole range of parameters). In Figure 16 and 17 we can see that it sometimes does a lot of transactions, whereas sometimes it goes for longer period of time

with one (what is more preferable). This is major flaw of this implementation. In Figure 18 we can see, that algorithm suddenly stopped to make any transactions. This shows that parameters and optimization are playing highly important role in algotrading.

Figure 20 Algorithm graphical recap, stock „FB”

Figure 21 Algorithm graphical recap, stock „JPM”

Figure 22 Algorithm graphical recap, stock „CLUB"

5.2.1.5 Summary

It turned out that logical adjustment to algorithm (not only longing but as well shorting) did not imply better results on average. It was worse than B&H strategy on average but, in case of "TUWOY" stock, it proved that, as well, on declining prices algorithm is able to make value. Implementation process shown that even small neglect in code can break the algorithm, resulting in abysmal results hence the importance of back testing and possibly unit tests.

5.2.2. Momentum based

5.2.2.1 Description

Algorithm has been adjusted by adding Beta indicator. Stock are chosen now by having best average score given by GARR and Beta indicator. Wages can be added, by default scores are equal. Additionally algorithm can now long top stock, or divide into longing top half of declared portfolio capabilities and shorting bottom half of stocks.

Beta indicator regards risk, high-beta (risky) should deliver lower risk-adjusted returns than low-beta stocks. Beta formula is as follows:

$$\beta_i = \frac{covariance(R_i, R_{market})}{Variance(R_{market})}$$, where R – stands for returns

Since formula involves market returns, Wilshire 5000 Total Market Index, (which covers all stocks traded in USA), is used.

5.2.2.2 Testing environment

Testing environment is the same as in non-adjusted environment Testing environment(4.2.2.2). In addition new parameters are available to test. As reminder, commission rate is 0.25%, and starting value 10.000.

Recommended values remain the same as in non-adjusted algorithm:

$number\ of\ stocks\ to\ hold\ =\ 10$

New parameters default values:

$GARR_{wage}$ = 1.0, $Beta_{wage}$ = 1.0

5.2.2.3 Tests

Table 22 Adjusted momentum algorithm results, half long, half short
2013.01.01 - 2019.03.29

Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return	Hold swaps	Total fees value [10^3]
10,46	-0,22	15,53%	0,63%	4,5%	630/750	1,67

Table 23 Adjusted momentum algorithm results, only buying,
2013.01.01 - 2019.03.29

Value [10^3]	Sharpe ratio	Max drawdown	Annualized return	Total compound return	Hold swaps	Total fees value [10^3]
13,56	0.31	20%	4%	30%	615/750	3,58

It is not clearly noticeable but there is some kind of problem with algorithm which is shoring half and longing half which can be deduced from total paid fees which is way too low (only buying version has comparable fee value with non-adjusted algorithm, whereas half long, half short version has over 2 times less fees). Overall algorithm based on two indicators is better than non-adjusted algorithm. With default parameters, adjusted algorithm performed 27% and 65% better than non-adjusted version.

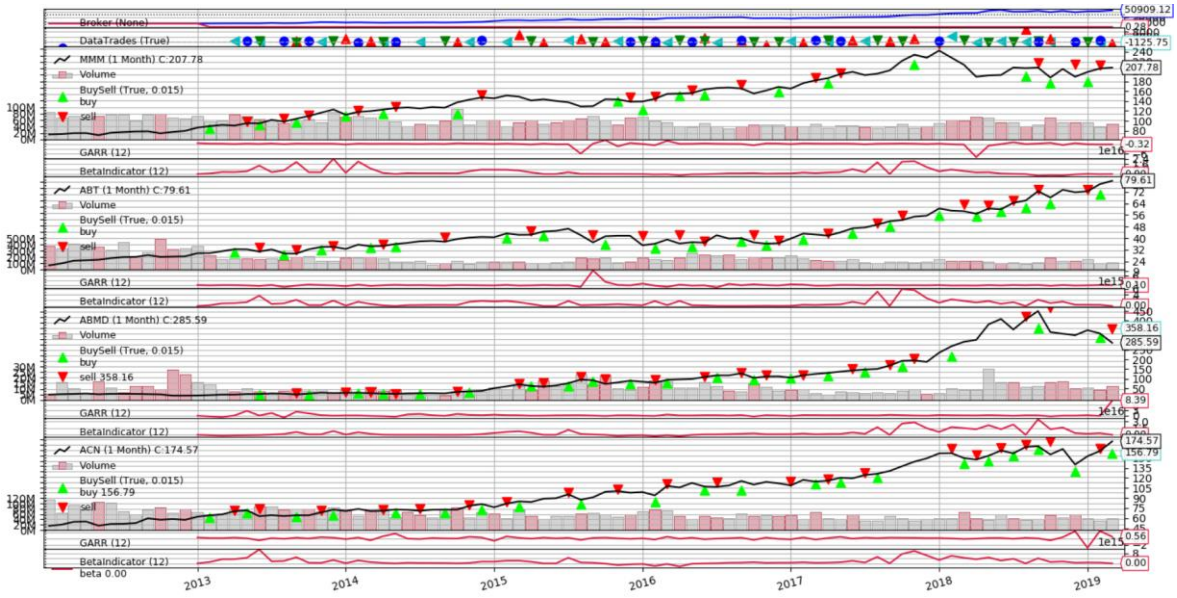

Figure 23 Exemplary algorithm graphical recap, only buy algorithm version

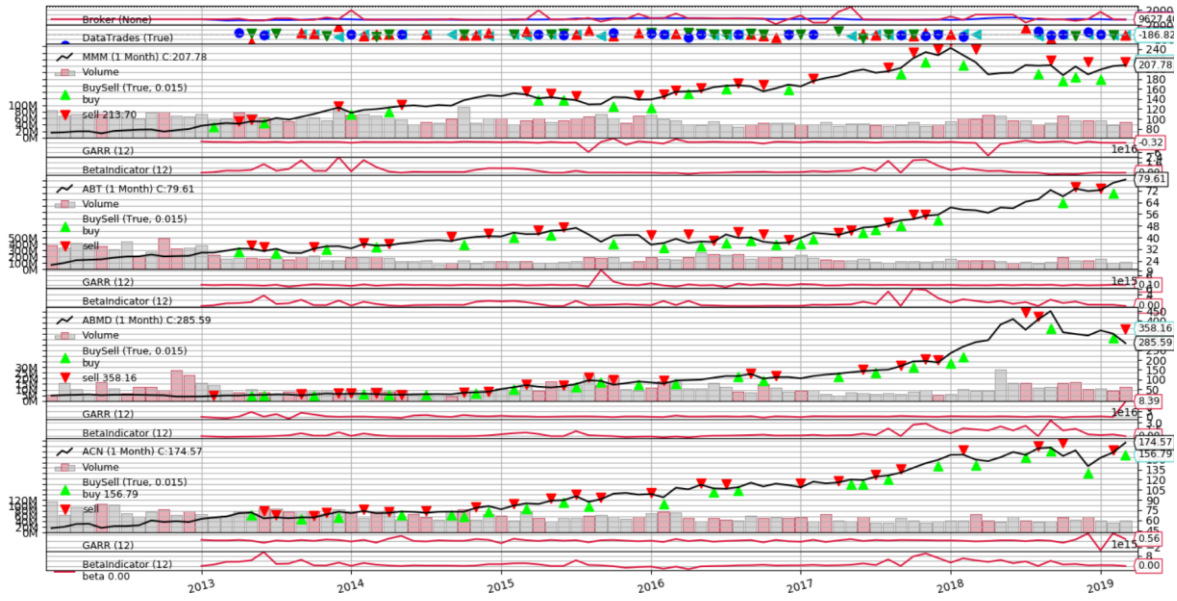

Figure 24 Exemplary algorithm graphical recap, half short, half long version

5.2.2.4 Optimization

Optimization was run in range of parameter values:

- $number\ of\ stocks\ to\ hold\ = 6 - 30$ with jump 2
- $GARR_{wage} \in [0.2, 0.5, 0{,}8, 1.0, 1{,}3, 1{,}5]$
- $\text{Beta}_{wage} \in [0.2, 0.5, 0{,}8, 1.0, 1{,}3, 1{,}5]$

Table 24 Adjusted momentum algorithm optimization results, half long, half short, 2013.01.01 - 2019.03.29

Best Value $[10^3]$	Average Value $[10^3]$	Best to B&H difference	Average to B&H difference
13,46	3,21	-26%	-82%

Table 25 Adjusted momentum algorithm optimization results, only buying, 2013.01.01 - 2019.03.29

Best Value $[10^3]$	Average Value $[10^3]$	Best to B&H difference	Average to B&H difference
41,35	17,21	121%	-7%

As we can see, both improvements did not performed well against B&H strategy nor non-adjusted algorithm. However all best results were using the biggest GAAR wage value (5.0) and lowest Beta (0.2). It proves GARR and Beta indicators cooperation did not perform well.

5.2.2.5 Summary

The worst problem to overcome is in *backtrader* itself and its cash management implementation. Whenever security is bought, cash is subtracted and when is sold, added.

That is quite problematic when in one iteration all transactions must be done (buy, sell, close trade). It wouldn't be a problem if *backtrader* allowed to mix timeframes since it is natural to close all transactions firstly and then wait for confirmation which will show how much value is left. *Backtrader* just goes within one timeframe, hence transactions are done (in this case) only once per month, so we have to do all calculations by ourselves and put transactions into queue in right order. *Backtrader* as well does not allow to load data while not using it in algorithm, hence Beta indicator had to be implemented using external data loader tools, what had bad influence over optimization tool performance (which is already slow since it works using brute force way of solving optimization problem). Using multiple indicators is good idea overall, even though results are showing differently (because of those two does not work well together). As results shown, sometimes we can discover algorithm faults, just by comparing obvious algorithm parameters results (like fee), so it is worth to check them by this prism (another alike example is when some optimization cases are over performing over average by thousands of percent's, which as well was case here because of historical data flaws).

5.2.3. Statistics prediction based strategy

5.2.3.1 Description

Strategy was enhanced by usage of Autoregressive integrated moving average (ARIMA) model. It shares same flaws, as previous strategy, hence it will always lose money on declining stocks. This strategy is will not be optimized since ARIMA model calculations are very CPU consuming (~3 predictions per second). ARIMA model is described in [19].

Strategy algorithm:
Input: Data array *prices*, *stock*– variable determining which stock are we operating on, *available_money*, rsi_{bot} – RSI range, *rsi* – RSI current indicator value, $boll_{top}$, $boll_{bot}$ – Bollinger bands values, *linearReg* –linear regression prediction, *arima* – ARIMA model next day prediction value.

1. **Set** $i \leftarrow 0, trend \leftarrow None$
2. **while** $i < length\ of\ prices$ **do**:
3. ***if*** $linearReg_i > linearReg_{i-1}$ ***or*** $arima_i > price_i$ ***and*** $arima_{i-1} > price_{i-i}$ ***then***:
3.1. **Set** $trend \leftarrow BULL$
4. ***elif*** $linearReg_i < linearReg_{i-1}$ ***or*** $arima_i < price_i$ ***and*** $arima_{i-1} < price_{i-i}$ ***then***:
4.1. **Set** $trend \leftarrow BEAR$
5. ***else***:
5.1. **Set** $trend \leftarrow VOLATILITY$
6. ***if*** *no currently opened position* ***then***:
6.1. ***if*** $trend == BULL$ ***then***:
6.1.1. **Set** orderMark $\leftarrow True$
6.1.2. ***if*** $rsi \leq rsi_{bot}$ ***then***:
6.1.2.1. ***execute*** $order = buy(stock, price_i, 2.0)$
6.1.2.2. **Set** orderMark $\leftarrow False$
6.2. ***else if*** $orderMark$ ***and*** $linearReg_i < boll_{bot}$ ***and*** $trend == VOLATILITY$
6.2.1. ***execute*** $order = buy(stock, price_i, 1.0)$
6.2.2. **Set** orderMark $\leftarrow False$
7. ***else if*** $linearReg_i > boll_{top}$ ***then***:
7.1. ***execute*** $close(order)$
8. $i \leftarrow i + 1$
9. **end while**

5.2.3.2 Testing environment

Testing environment is the same as in Testing environment(4.2.3.2) section.

5.2.3.3 *ARIMA model tests*

ARIMA model was tested using the simplest prediction strategy: if next price prediction is higher than current one, buy, else sell. Strategy was from point of having 50 previous day prices, in order to calibrate ARIMA model prediction, all further prices were added do model to increase precision. In Figure we can see graphical recap of that strategy. Although this strategy predicted roughly 49% of price trends, after commission cut, only 22% yielded positive outcome (without commission cut – 27%), what resulted in 3400 final value.

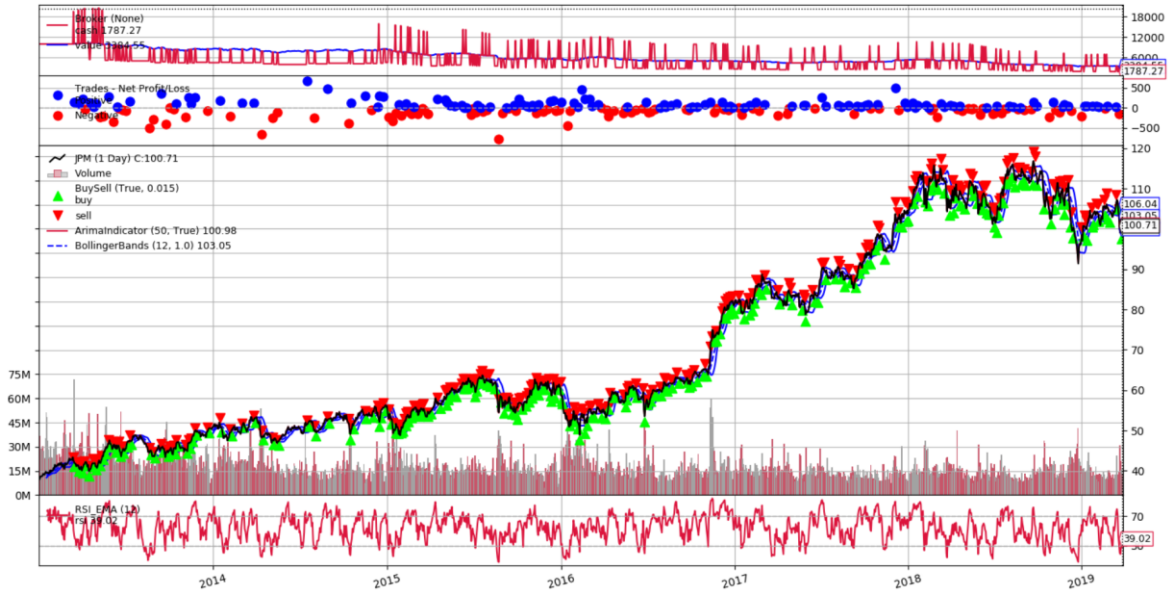

Figure 25 ARIMA accuracy test, graphical recap

"JPM" 2013.01.01 – 2019.03.29

Table 20 shows how properly ARIMA model predicted prices on "JPM" stock chart, from 2013.01.01 to 2019.03.29. Trend measurement is as follows: if previous day price is lower and predicted price was bigger than current, then trend is bullish, else bearish. Second row shows if price was predicted but in right direction (if prediction properly predicted if overall price increased or decreased).

Table 26 ARIMA model price prediction

Deviations	2,0%	1,75%	1,5%	1,25%	1,0%	0,75%	0,5%	0,25%	0,1%
Price prediction	88%	85%	79%	73%	65%	53%	39%	19%	8%
Price and trend prediction	44%	43%	41%	38%	33%	28%	22%	13%	6%

As we can see, ARIMA model prediction from first glance seems to be mediocre tool (2% deviation price with 88% prediction correctness). Although if we take into consideration trend prediction, we can observe that the bigger price deviation the less trend predictive this model is. This model is not particularly good as a backbone of whole algorithm, but it can stand well as auxiliary determination tool.

5.2.3.4 Tests

Recommended parameters are: $rsi_{period} = 12$, $boll_{period} = 20$ – Bollinger bands period, $boll_{dev\_factor} = 1.0$ – Bollinger bands standard deviation multiplier, $rsi_{bot} = 40$ – RSI value requirement to trigger leverage buy, $linear\_regression_{period} = 12$ – linear regression period, $arima_{period} = 20$ – ARIMA minimum required prices to produce prediction.

Table 27 ARIMA enhanced strategy, results, 2016.05.18 – 2019.03.29

Stock	Value [10^3]	Max drawdown	Sharpe ratio	Non-adjusted strategy difference	B&H strategy difference
FB	23,97	46%	1,14	72%	70%
JPM	21,17	36%	1,14	74%	24%
BTC-USD	4,87	80%	-0,33	142%	-95%
CLUB	9,16	74%	0.17	36%	-41%
TSLA	29,69	50%	1,35	126%	135%
TUWOY	7,91	76%	0,00	-43%	-29%
Average	16,12	60%	0,66	68%	11%

As we can see in the Table 21, usage of ARIMA model prediction payed off. On mostly inclining stocks ("FB", "JPM", and "TSLA") it performed much better than predecessor what value and Sharpe ratio are showing. As expected, on declining stocks ("CLUB", "TUWOY") and "BTC-USD" it performed comparable. In Figure 25 and 26 we can observe that until stocks are rising, in all cases, algorithm is performing very well. In case of "CLUB" stock we can see that strategy performs really well, until stock starts to plunge, what results in huge drawdown. Surprisingly in volatile period of "TSLA" stock (in Figure 27) strategy performed decently.

Figure 26 ARIMA enhanced strategy, graphical recap,

„JPM”, 2016.05.18 – 2019.03.29

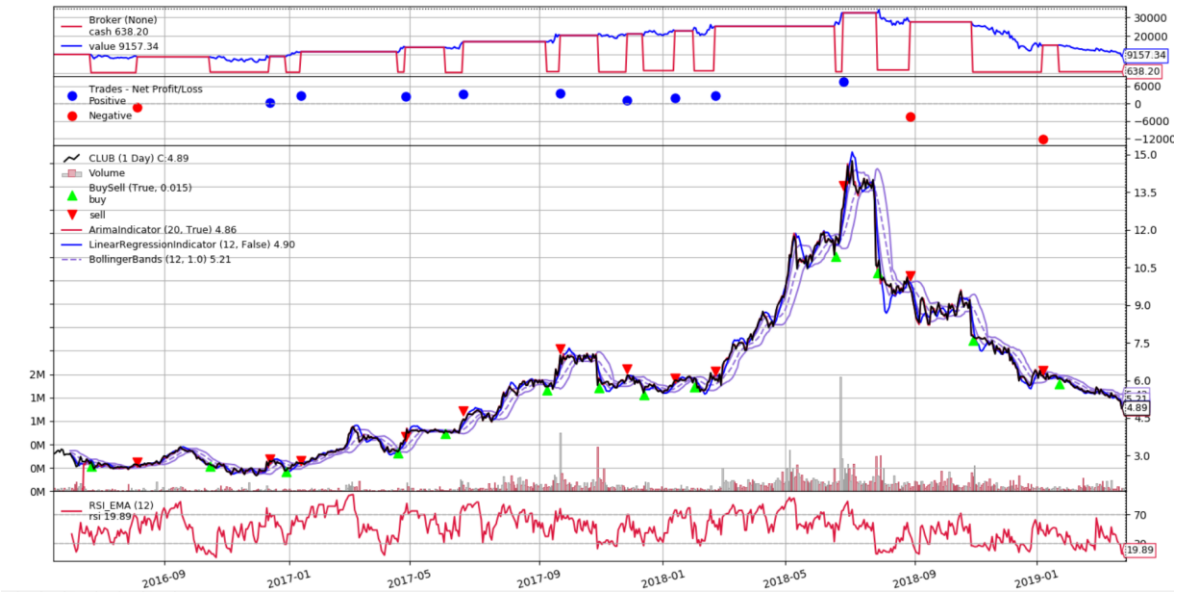

Figure 27 ARIMA enhanced strategy, graphical recap,

“CLUB”, 2016.05.18 – 2019.03.29

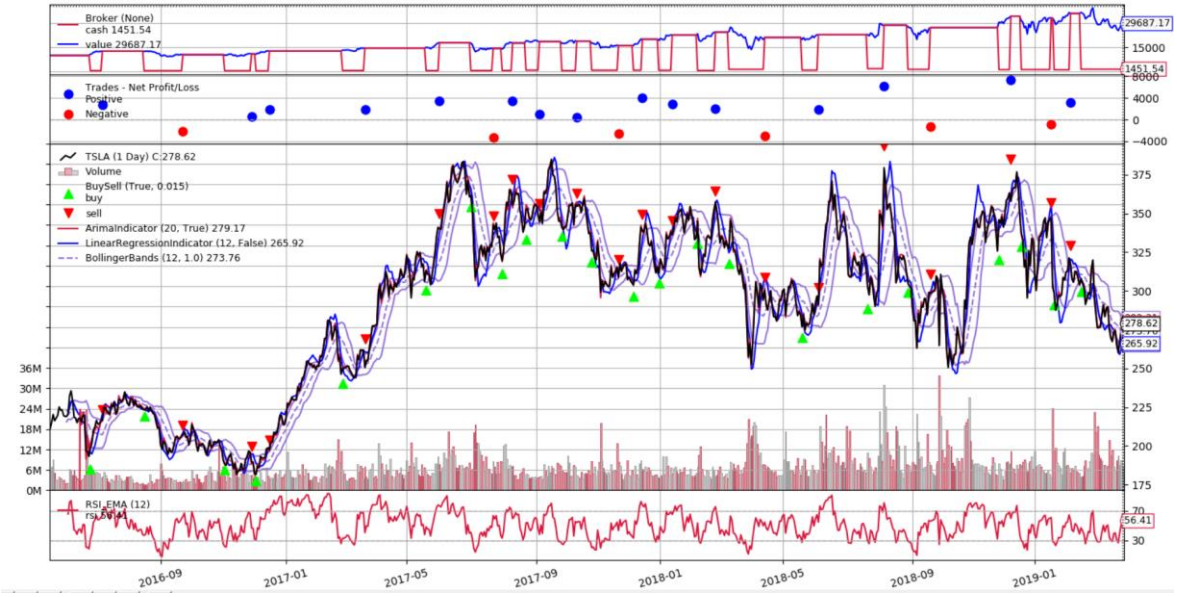

Figure 28 ARIMA enhanced strategy, graphical recap,

“TSLA”, 2016.05.18 – 2019.03.29

5.2.3.5 Summary

ARIMA model proved to be average prediction tool, which cannot be purely nor consistently relied on. As an auxiliary indicator, properly waged, it can boost strategy performance. We can observe that around 50% of price and trend prediction are valid, hence we can assume that this model is still random model but with mediocre (2% deviation – 88% predictions) price prediction estimation. ARIMA model was used, in addition to linear regression, as trend determinant. Since shorting strategy was not included, it did not perform well in declining periods. Interestingly, ARIMA addition, lowered strategy risk (much better Sharpe ratio, in three cases above 1.0), what supports use of leveraged security trades. As

predecessor, this strategy, in case of longing, proved to be efficient, whereas in case of shorting, it lacks any assumptions, which could be great improvement field.

5.3. Extra tests

Tests are inherent step in making and evaluating every strategy. Algorithms optimization usually exhibits and underlines the best performers. The catch is that if these best performers can keep their outstanding results or it was only matter of perfect adjust to historical data.

Two following proposed methods are one of most common, in trading world, to answer that question.

5.3.1. Robustness tests

5.3.1.1 Description

For robustness test, Monte Carlo model is used. This model simulations are used to show different probable outcomes in a process that cannot be easily predicted. Results of such a simulations help to understand risk and potential flaws in algorithm. To generate predictions simple formula is used: $price_i = price_{i-1} * (1 + GaussianDistribution(0, Standard\ Deviation\ of\ price\ volatility$ where price volatility is difference between highest and lowest daily price.

5.3.1.2 Testing environment

Tests were conducted using stocks: „FB", "BTC-USD" and "CLUB". Starting price for Monte Carlo simulation was from 2017.10.13. One hundred predictions were generated for time period from 2017.10.16 (Monday) to 2019.03.29 for each stock. These prediction charts were used against algorithm from section Mean reversion(4.2.1), and later on compared to real chart outcome against same algorithm from same period. The only change to tables' columns is those are using profit, instead of final value. Algorithms were frequently not trading, hence that would bias look at the results. Starting value = 10.000.

5.3.1.3 Results

Nine, from worst, to best performing algorithm, based on historical data performance of mean reversion strategy from 2012.01.01 to 2017.10.13, parameters are chosen and tested. In Figure 29, 30 and 31 we can see Monte Carlo simulations for given period and stock.

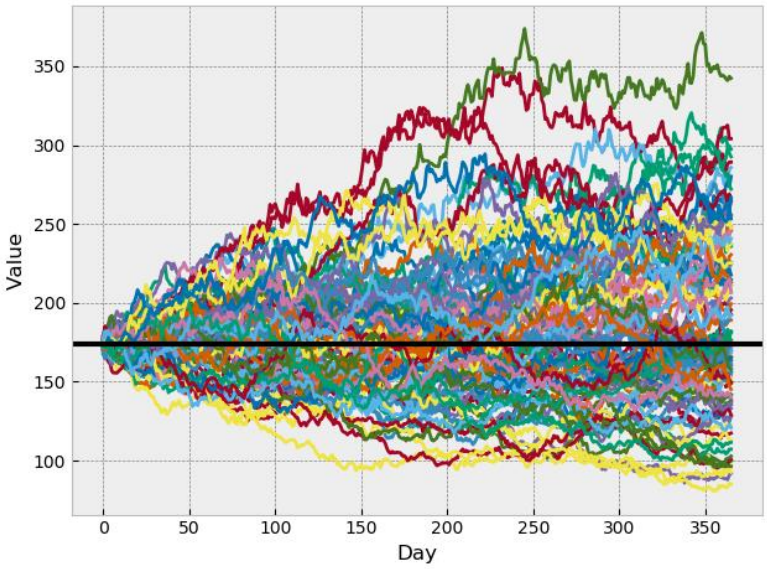

Figure 29 Monte Carlo simulation, starting price 2017.10.13, "FB"

Table 28 Robustness tests, 2017.10.13 – 2019.03.29, „FB”

No.	Optimization Value [10^3]	Average profit against predictions	Min profit against predictions [10^3]	Max profit against predictions [10^3]	Real algorithm profits for given time period [10^3]
1	89,25	-55,46	-5,14	5,66	0,00
2	67,52	30,91	-2,48	3,40	2,28
3	43,36	211,08	-3,81	5,89	-0,25
4	37,67	86,22	-2,38	4,58	0,00
5	31,30	9,62	-3,46	5,09	-1,45
6	26,25	-21,01	-5,06	5,02	0,00
7	22,59	75,12	-2,82	4,58	0,00
8	16,96	-129,29	-4,72	4,16	0,00
9	15,35	62,46	-5,15	4,57	-1,45

As we can see in Table 28, there is no explicit correlation between optimization value and prediction profits nor real historical data algorithm profits, although, by looking at minimum profits, we can see that the more average results, the less maximum drawdown, and the better maximum profits. Algorithm, against “FB” stock, behaved very evenly and we can assume that on average it did not yield any notable profit nor loss.

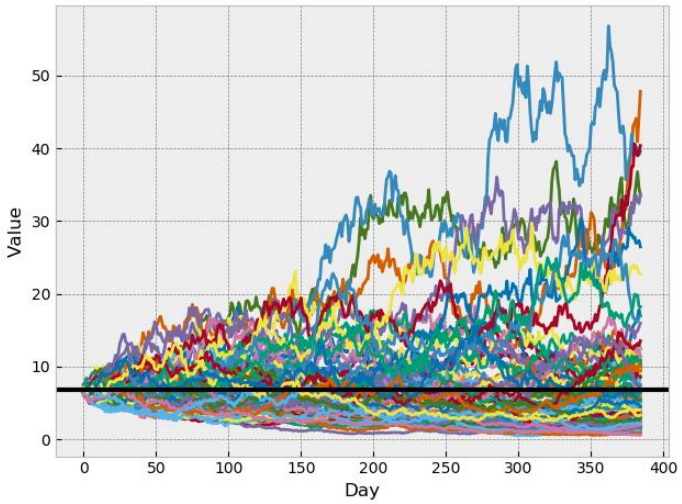

Figure 30 Monte Carlo simulation, starting price 2017.10.13, "CLUB"

Table 29 Robustness tests, 2017.10.13 – 2019.03.29, „CLUB”

No.	Optimization Value [10^3]	Average profit against predictions [10^3]	Min profit against predictions [10^3]	Max profit against predictions [10^3]	Real algorithm profits for given time period [10^3]
1	45,35	0,00	0,00	0,00	0,00

2	32,62	-4,28	-10,00	-4,91	-4,72
3	20,36	-4,79	-10,00	-5,41	-5,34
4	17,96	0,00	0,00	0,00	0,00
5	14,83	0,00	0,00	0,00	0,00
6	14,56	-4,76	-10,00	-5,25	-5,03
7	11,08	-4,76	-10,00	-5,25	-5,03
8	7,62	-4,76	-10,00	-5,25	-5,03
9	6,20	0,00	0,00	0,00	0,00

In Table 29, we can see that there are a lot of zero values. That sates, that algorithm, using specific parameters based on previous predictions, was not trading at all. This shows, that one algorithm was over optimized, just to reach as high value as possible in given time period.

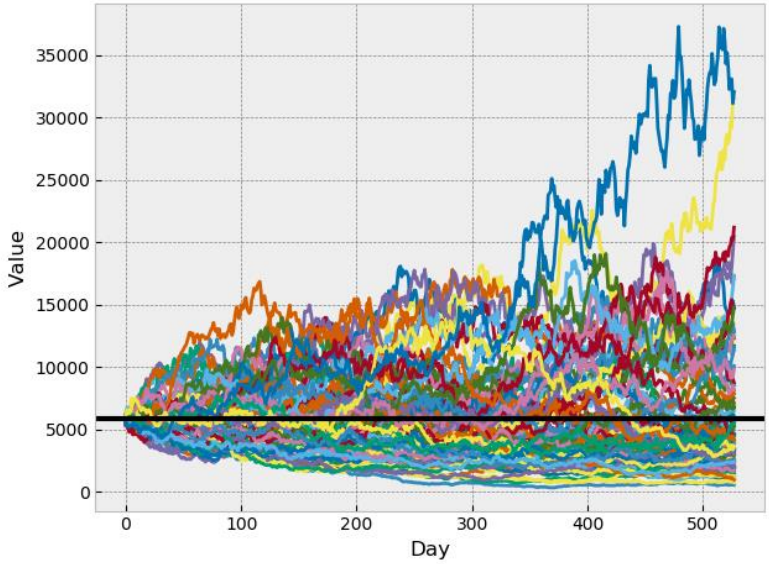

Figure 31 Monte Carlo simulation, starting price 2017.10.13, "BTC-USD"

Table 30 Monte Carlo simulation, starting price 2017.10.13, "CLUB"

No.	Optimization Value [10^3]	Average profit against predictions [10^3]	Min profit against predictions [10^3]	Max profit against predictions [10^3]	Real algorithm profits for given time period [10^3]
1	14312	0,00	0,00	0,00	0,00
2	7725	0,00	0,00	0,00	0,00
3	5045	0,00	0,00	0,00	0,00
4	3134	0,00	0,00	0,00	0,00
5	1573	-2,68	-3,27	0,00	-3,27
6	763	-2,55	-3,71	0,00	-3,71
7	231	0,00	0,00	0,00	0,00
8	134	-1,59	-2,90	0,00	-2,90
9	90	0,00	0,00	0,00	0,00

As in previous example, as it is seen in Table 30, algorithm barely trade and even if it did, it was not making any profits. In this case it is not even about over optimized parameters but more that this algorithm performs abysmal for this kind of stock.

5.3.1.4 Summary

Monte Carlo is useful in analysis of algorithm performance since it shows how our model will behave under different circumstances. Strategy might perform very well in past simulations but that does not mean that will continue to do so in future. Robustness tests show perfectly what algorithm is up to and what drawdowns and on average results should be expected but most importantly shows if certain strategy can blow up ones account in short time period. It can exhibit over optimized strategies, what in past were over performing, and when it comes to real trading, they do not make any transactions.

5.3.2. "80/20" tests

5.3.2.1 Description

This test is to show how optimized algorithm will perform in future. Data is divided into two parts – training / optimization and test data. This simple method shows if algorithm is not over optimized and how it behaves in alike simulation of real price movements.

5.3.2.2 Testing environment

Tests were conducted using stocks: „FB", "BTC-USD", "CLUB", "JPM", "TSLA" and "TUWOY". In this tests, historical data is divided into two parts, training and test data. From 2013.01.01 to 2017.10.13 (Friday) – training data and 2017.10.16 (Monday) to 2019.03.29 – test data. For given time frame Buy and Hold strategy results are visible in Table 28. Starting value = 10.000.

Table 31 B&H strategy results, 2017.10.16 - 2019.03.29

Stock	FB	JPM	BTC-USD	CLUB	TSLA	TUWOY
Value [10^3]	9,45	10,56	7,04	7,01	7,93	12,51

Three best performing algorithms are tested: mean reversion with averages: B&H difference 23% and Sharpe Ratio 0,39 (Mean reversion(4.2.1)), momentum based with averages: B&H difference 18% (Momentum based(4.2.2)) and genetic algorithm with averages: B&H difference 1130% (with exclusion of "BTC-USD" over enormous difference 64%) and Sharpe Ratio 0,80 (Genetic algorithm(4.3.1)) . They are chosen using following formula $score = SharpeRatio_{average} * B\&H\_Difference_{average}$. Since momentum analysis did not include $SharpeRatio$, it is omitted.

5.3.2.3 Tests

Separated tests are conducted for each of chosen algorithms. Three best performing algorithm parameters combination from optimization / test period are confronted against test data.

5.3.2.3.1 Mean reversion

Table 32 "80/20" tests, Mean Reversion

Stock	I Best Value [10^3]	II Best Value [10^3]	III Best Value [10^3]	I Outcome	II Outcome	III Outcome

				Value [10^3]	Value [10^3]	Value [10^3]
FB	43,86	36,86	37,10	9,59	8,16	9,37
JPM	20,18	20,81	21,13	9,56	9,50	10,00
BTC-USD	707	633	500	10,00	10,00	10,00
CLUB	33,36	38,02	24,88	5,12	5,06	10,00
TSLA	39,65	31,02	29,23	10,80	10,00	10,00
TUWOY	14,23	12,91	12,65	9,38	10,82	9,82

Figure 32 "80/20" tests, Mean Reversion, graphical recap of „TUWOY" test data trading

As we can see in Table 32, algorithm barely trades and if so, nearly always results in loss. This is mostly due to fact, what is seen in Figure 32 that one of the Moving Averages has long calculation period hence over half of test period was used just to gather data for indicator required for trading algorithm. Nevertheless, there is no strict correlation between past and future strategy performance.

5.3.2.3.2 Momentum based

Since our test data backbone relies on six different charts, Buy and Hold strategy is equivalent to holding equal shares of all available stocks, in our case – six.

Table 33 "80/20" tests, Momentum based

Number of stocks	Value [10^3]	Outcome Value [10^3]
1	7,42	12,26
2	16,46	9,82
3	17,41	10,70
4	13,18	9,50
5	13,43	9,40
B&H (6)	17,85	9,40

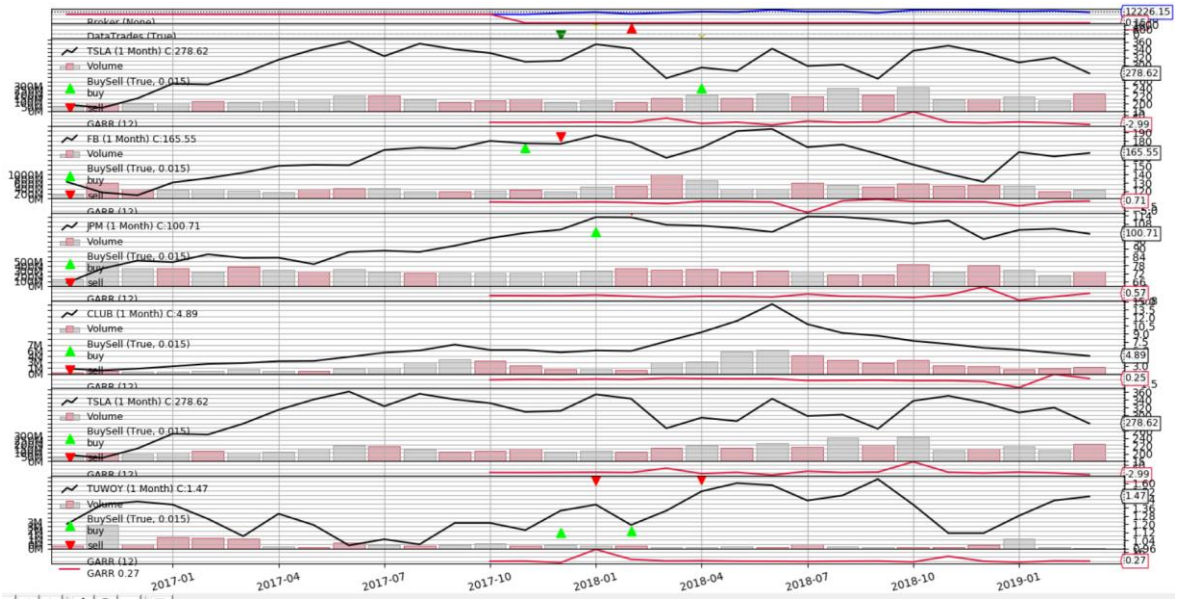

Figure 33 "80/20" tests, Momentum based, graphical recap of test data trading, number of stocks = 3

Yet again there is no explicit correlation between training and test data outcome. It is mostly due to fact of limited stocks amount, hence as we can see in Figure 33, algorithm was barely making any adjustments to portfolio. Although from that tiny sample we can observe that algorithm performance against test data tends to be better the lower amount of stocks we hold.

5.3.2.3.3 Genetic algorithm

Table 34 "80/20" tests, Genetic algorithm based

Stock	I Best Value [10^3]	II Best Value [10^3]	III Best Value [10^3]	I Outcome Value [10^3]	II Outcome Value [10^3]	III Outcome Value [10^3]
FB	30,18	26,86	25,47	9,62	10,22	10,88
JPM	18,86	18,12	17,28	10,22	10,17	10,41
BTC-USD	42,32	37,21	36,23	10,12	3,23	11,56
CLUB	12,68	12,01	11,59	1,49	1,41	1,17
TSLA	38,21	33,17	32,90	12,34	9,87	11,32
TUWOY	46,36	42,43	30,35	6,83	6,09	8,62

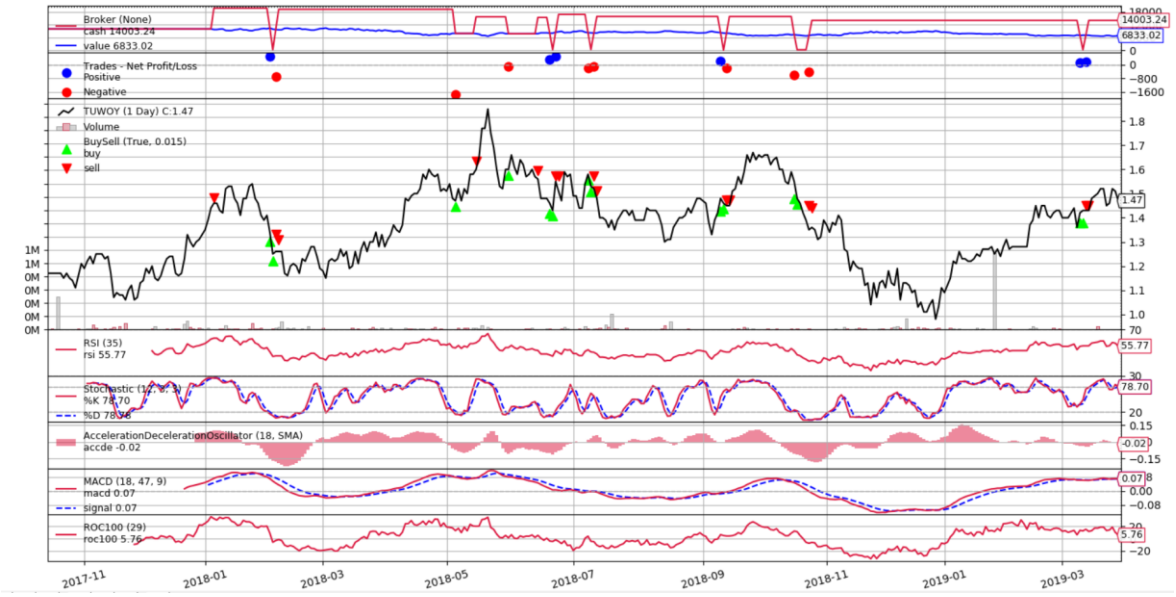

Figure 34 "80/20" tests, Genetic algorithm based, graphical recap of "TUWOY" test data trading

Genetic optimization shows very specific and in most cases over performed results but since algorithm uses a lot of indicators, even these even over performed parameters make trades. On rising stocks, on average result was positive and on declining – negative. In Table 34 we can see that the worse performance parameters combination has achieved the better real test data results tended to be.

5.3.2.4 Summary

These tests proved to excellent tool to most accurately test performance of our algorithm with given parameters. Although it does not cover every aspect like average performance nor what on average can we expect where in this matter robustness test shines. This test should be used in conjunction with robustness tests to discover potential flaws, risks and anticipated result.

Conclusions

Researched algorithms average performance was not particular satisfying, although it showed a lot. On contrary to traders who incorporate strategies to trade and their biggest issue is time reaction, algorithms are written using current programming languages which makes them prone to various bugs (not only implementation but as well used framework and libraries issues) and data issues (empty values, omitted trading days). Graphical recaps are handy to discover bugs and flaws of backbone strategy hence they should not be ignored.

Some strategies work well under different conditions and circumstances; statistics prediction worked well on volatile stocks whereas mean reversion or genetic based algorithm on bullish periods. Algorithms optimization showed that usually the best working parameter configurations are these with perfect align to data and averaging optimization results is not good method to anticipate future profits. All algorithms using "BTC-USD" historical data vividly showed the problem. Most, not only best parameter configurations, were not trading using test data at all. On average algorithms with higher Sharpe Ratio and lower max drawdown performed better. Indicators does not perform well in long term investing. Frequently prices sharply changes when explicit event occurs and that ordinary indicator will not catch immediately. Sometimes logically obvious adjustments to base algorithm, like adding selling implementation to not rely only on buying and closing positions, worsen overall performance like it was in case of mean reversion algorithm enhancement. Momentum strategy exhibited important profit and loss factor – trading fees. Even though strategy at first glimpse was profitable, frequent portfolio changes degraded final outcome because of commissions. Implementation of this strategy was quite complex and hard to maintain. Having concurrently multiple opened positions made program hard to debug. If it comes to statistical models and their prediction performance, it is mediocre. Linear regression coped with trend prediction on contrary to ARIMA model which did not work well on long periods but could be relied on, on shorter time prediction, although not solely. Its trend prediction accuracy tends to increase with price prediction; within 2% deviation, 88% prices were properly predicted, although with proper trend only half of it, 44%, whereas within 0,1% deviation it was 8% and 6% respectively. Genetic algorithm proved to be faster in terms of finding more or less best parameter configurations but because of the way it works, it does not provide any other important data, like average outcome since everything is biased toward local minima in which algorithm frequently gets stuck. Although it is handy tool if it comes to optimization of algorithm relied on multiple variables. Machine learning approach show that basically there is no strict correlation in price movements and possible predictions, more external variables are required like news or domain techniques like pattern recognition mechanism would certainly increase model accuracy.

Some of the strategies worked better, worse or exceptional. In case of latter ones, there is always initial excitement that we could have discovered potentially profitable strategy. That thought might mislead us poorly; if algorithm was beating Buy and Hold strategy by 100%, in real time it should perform well. Luckily there are already invented practices (beyond demo account trading) that can check our assumptions in matter of seconds. Robustness tests show perfectly how our algorithm will behave in different situations. Artificially generating multiple price data, and incorporating them into tests is the best tool to show how robust algorithm is and what performance we can expect on average as well as its boundaries. This test is immediately showing if our parameters are over optimized, or we did not taken care of risk management. In our previous tests, Mean Reversion showed to be the best algorithm, beating on average Buy and Hold strategy by 23%. Robustness checks showed that realistically we can expect that this algorithm will just not lose money nor make any notable profit. It showed as well that this algorithm is very prone to drawdowns which

we have to avoid. On the other hand “80/20” tests, are quickly showing how our parameter composition could have worked in real time, based on historical data. In every case, using best performing parameter configurations based on training data, algorithms did not performed anywhere close to base results. In some cases, like “BTC-USD” they could not trade at all. Robustness checks and “80/20” tests proved to work well and should be used in conjunction.

Used strategies were overall to simple. In order to implement reliable algorithm, multiple data sources and features are required. The more cases algorithm covers, the better it works. Of course the easiest algorithms are these which are trying to mimic traders’ performance, yet we should as well use potential of huge computation power which nowadays hardware gives by utilizing more and more logic. That leads to another important issue which is code performance and maintainability. All tested strategies should have incorporated more risk management, even by just utilizing widely available tools like stop-loss or take-profit. Another improvement could be incorporating dynamic fundamental company analysis or at least taking it into consideration as some sort of variables. This work showed as well that technical indicator does not cope well with long term investing. Most of scientific researches focus on short time trading and prediction using variety of complex statistical models.

Bibliography

[1] Kannan, Sriram, 'Algorithmic Trading and Its Implications on Capital Markets' (December 19, 2014). 2nd International Conference on Business Analytics & Intelligence (ICBAI) held at the Indian Institute of Science, Bangalore (India). pp. 2-4

[2] "blog.quantopian.com" 2019.01.[Online] Available: https://blog.quantopian.com/common-types-of-trading-algorithms/

[3] Kissell, Robert L, "Algorithmic trading strategies" (2006). ETD Collection for Fordham University. AAI3216918. pp. 18-20

[4] Kissell, Robert L, "Algorithmic trading strategies" (2006). ETD Collection for Fordham University. AAI3216918. pp. 20-26

[5] "Investopedia.com" 2019.01.[Online] Available: https://www.investopedia.com/articles/active-trading/081215/new-alternatives-highfrequency-trading.asp

[6] Gomber, Peter and Arndt, Björn and Lutat, Marco and Uhle, Tim Elko, High-Frequency Trading (2011).

[7] "quantstart.com", Beginners Guide to Quantitative Trading, 2019.01.[Online] Available: https://www.quantstart.com/articles/Beginners-Guide-to-Quantitative-Trading

[8] Chaboud, Alain and Chiquoine, Ben and Hjalmarsson, Erik and Vega, Clara, Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market (July 5, 2013). Journal of Finance, 69, pp. 2045-2084.; FRB International Finance Discussion Paper No. 980.

[9] "Wikipedia", Financial instrument, 2019.01.[Online] Available: https://en.wikipedia.org/wiki/Financial_instrument

[10] "Python programming language" 2019.05.[Online] Available: https://www.python.org/

[11] "Backtrader" 2019.05.[Online] Available: https://www.backtrader.com/docu/inddev.html

[12] "QuantsPortal.com", Momentum Strategies, 2019.05.[Online] Available: http://www.quantsportal.com/momentum-strategies/

[13] Neumayer, Eric and Plümper, Thomas (2017) Robustness Tests for Quantitative Research. Methodological Tools in the Social Sciences. Cambridge University Press. ISBN 9781108415392

[14] Ramavathu, Lakshman & Ramesh, D & Bairam, Dr. Manjula & Govardhan, Dr. (2019). Prediction of Stock Market Index Using Genetic Algorithm.

[15] "Wikipedia" Stochastic Oscillator, 2019.04.[Online] Available: https://en.wikipedia.org/wiki/Stochastic_oscillator

[16] "Wikipedia" MACD, 2019.05.[Online] Available: https://en.wikipedia.org/wiki/MACD

[17] Weng, Bin & Lu, Lin & Wang, Xing & Megahed, Fadel & Martinez, Waldyn. (2018). Predicting Short-Term Stock Prices using Ensemble Methods and Online Data Sources. Expert Systems with Applications. 112. 10.1016/j.eswa.2018.06.016.

[18] "Investopedia.com" On Balance Volume Indicator, 2019.05.[Online] Available: https://www.investopedia.com/terms/o/onbalancevolume.asp

[19] J. Conejo, Antonio & Plazas, Miguel & Espinola, Rodrigo & B. Molina, Ana. (2005). Day-Ahead Electricity Price Forecasting Using the Wavelet Transform and ARIMA Models. Power Systems, IEEE Transactions on. 20. 1035 - 1042. 10.1109/TPWRS.2005.846054.

List of Figures

List of tables

Attachments

- CD disk containing all implemented algorithms and their optimization and tests results.