Timeframe Trading Algorithms

Student Name: A.L. Gillies

Supervisor Name: M. R. Gadouleau

Submitted as part of the degree of MEng Computer Science to the

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

***Abstract* —** The abstract must be a Structured Abstract with the headings **Context/Background**, **Aims**, **Method**, **Results**, and **Conclusions**. This section should not be longer than half of a page, and having no more than one or two sentences under each heading is advised.

Context/Background - Algorithmic trading is characterised by an entirely hands off approach to stock market trading. All data manipulation, mathematical inference, machine learning and trade execution is done autonomously. With this approach, how much of an improvement can be gained over a standard interest rate provided by a high street bank, in the time frame given?

Aims - Using the average interest rate calculated from British banks, the aim of this paper is to show, through implementation of statistical and machine learning techniques that algorithmic trading can improve the annual return on investment over a given time frame.

Method - This paper will consider two possibilities for implementation of the system, a purely statistical method, relying on known practices and techniques, and a hybrid system incorporating both statistical reasoning and machine learning. The known statistical practices are mostly used by human traders to allow for data insight and are well vetted. The machine learning techniques are widely used in other contexts, with limited academic papers being available for this area.

Results -

Conclusions -

***Keywords* —** Algorithmic, Machine Learning, Statistics, R, Trading, Stocks

I INTRODUCTION

The stock market has been an early adopter of technology since its inception, with companies wanting to get an edge over their fellows and thus earning the most money. The first computer usage in the stock market was in the early 1970s with the New York Stock Exchange introducing the DOT system or the Designated Order Turnaround system, this allowed for bypassing of brokers and routed an order for specific securities to a specialist on the trading floor. Since this point the use of machines to allow for increase throughput and speed has been pandemic. From this point it was inevitable that computers would be used to aid in the decision making process of what to buy or sell and when. This was shown to be very effective and got significant traction in the financial market in 2001 with the showcase of IBMs MGD and Hewlett-Packard's

ZIP, these two algorithmic strategies were shown to consistently outperform their human counterparts. These were both based on academic papers from 1996 so the academic conception of algorithmic applications in financial markets has been present for several decades. Whilst in the current day over one billion shares are traded every day, this would not be possible without computerised assistance.

Unique ID	Deliverable	Description
DL1	Simulate the financial market	Have data for at least 10 companies for at least a year, with data for each minute where data is available.
DL2	Allow buying and selling of stocks	Have a functional buying and selling mechanism, with the data collected for each transaction processed.
DL3	Implement statistical methods	Implement as many statistical methods as are beneficial to allow for the insight into the data for each stock.
DL4	Implement a purely statistic strategy	Using just the statistical methods implemented in DL3, create a strategy that will buy and sell stocks to maximise profit made over the time frame given.
DL5	Create a hybrid strategy	Implement a machine learning trading strategy that uses the stock data as well as any statistical methods that are helpful to maximise profit made over the time frame given.
DL6	Implement tracking systems	Implement graphical and table outputs for the results of the computer logic and trading performance.
DL7	Create a testing criteria	Create a method with which to test the strategy so as to avoid over fitting.

Table 1: Deliverables

II RELATED WORK

III SOLUTION

Simulation

Logic

How the simulation was set up.

The logic behind the simulation was to allow trading at a fine grain level, this needed to be permitted by the data chosen to perform the algorithms on. A drawback was that 'tick data' or data for each trade executed was found to be too costly for the project so 'minute data' was used. Minute data provides an open, high, low, and close price for a stock in the given minute, this gave plenty of data points per day of trading. A simplification was created at this point, trading was only done on the price that opened the minute. This was done to reduce the complexity of calculating any inter-minute values. The running of the simulation was based on the progression through each date of available trading and for each day iterate through every minute of the trading day, starting at 09:30 until 16:00. Each trading day has 390 points at which trading is possible, if the data permits it. This is available for X days in the 18 months of data that was used.

Data

Talk through what data was used.

Forty Five different stocks were used in this paper, taking from each the maximum number of data points available between 01/03/2016 and 01/09/2017 giving an eighteen month window in which the first six months did not allow for trading to occur, only for data collection and training of models, which will be discussed in depth further in the paper, then the final twelve months are for testing each algorithmic approach. There were some limitations with the data that was used. Due to the volatility of the stock market and the fact that some days trading was halted for a myriad of reasons, there are not 390 data points per day, for eighteen months. Trading is not available on weekends, or on bank or national holidays, or if trading is halted for some other reason. This meant that the solution had to take into account holes in the data.

Talk through why that data was used.

The data was taken from the NASDAQ, the second largest stock market in the world. The number of stocks used was arbitrarily big enough to provide enough data to test any given algorithm, some of which were found to prefer an abundance of data whilst others were much more short sighted.

Set up

Why was the simulation set up that way?

The simulation was set up in an iterative way in order to minimise the disruptions that missing data would cause, whilst also being exhaustive and searching for data given a reference point is straightforward, so no data point is missed or an inaccurate search is done. A simple testing framework was set up during implementation, variable controlled which stocks were being iterated through at any given moment,what dates were available, when any trade execution could commence, and the amount of capital that was available to any given algorithm. Once these had been established then each date from the start date to the end date variables was found through merged table of all dates across all stocks as some stocks are available to trade when others are not, these are then iterated through. Within this loop is another very similar loop that performs the same action but for minutes rather than dates. All available times are found for the specific date, and iterated through. Finally, for each stock that is available, during testing not all stocks were used so as to help with the removal of bugs, instead a random number generator would choose a predefined number of stocks and iterate over these instead. A stock value is found for that date, at that time, for that stock. Once this has been done then then calculations can be made to decide if any action needs to be taken given this new information. Any buy or sell actions are handled by a single function each that will take in all current information, including the time, date, and stock price. These two functions shall be known as the 'shouldBuy' and 'shouldSell' functions throughout this paper. These functions will be given all available information and will then make a decision using the methods they have been assigned. Any action that is taken will be stored in tables set up before the simulation was started. There are three tables, initially only two were used but a third was introduced at a later date to increase the speed of execution of the simulation. These tables are 'Active', 'Sold', and 'Ledger'. Active is any bundle of stocks that have been bought by the algorithm, stored as a row containing; A unique identifying number to allow for easy retrieval of the row, the date on which the stock was bought, the time at which the stock was bought, the name of the stock, the number of shares that were bought during this execution, and the cost per share or stock price as these are the same. This table is iterated through every time a new stock price becomes available for a stock that is present in the table and the 'shouldSell' function is called, then any necessary calculations are done to decide if this bundle of stocks should be sold or not. If it is decided that this bundle should be sold then this row is removed from the 'Active' table and added to the 'Sold' table. This table contains the same headers as the 'Active' table with the addition of the total amount bought, meaning number of shares multiplied by price they were bought at, the date sold on, the time sold at, the price per share that they were sold at, and the total amount sold. The two totals, total bought and total sold, may seem to be superfluous by they are relied upon by the final table. This table is the most used, it has a row added each time a change is made within the simulation regardless of any actions taken. Each minute a new row is added to this table to keep track of all capital that is within the simulation, the reason for this table is to show at the end of the simulation all trades that were performed and the amount of capital that is liquid or in stocks at any given moment. During implementation minute by minute updates were not required and this only served to increase the execution time of the simulation by a significant amount, and thus the function that adds a new row to this table was only called at the end of each day iteration instead of at the end of each minute iteration. This table contains the heading; date - this was a concatenation of both time and date with a simplified name, value - this was the total value within the simulation, stock value - this is the total amount of capital that are currently within stocks, and capital value - this is the amount of liquid capital available.

ADD THREE TABLES TO SHOW HOW THEY LOOK

Functions

What functions are critical to the running of the simulation?

Statistical Methods

What statistical methods were used first?

Once a working simulation had been established and all functions had been shown to be working through hard coding of behaviour, statistical methods had to be implemented. These were separated into two groups, the first are called technical overlays, these are methods that provided insight into the data by providing numbers that are on the same scale as the price itself, whilst technical indicators are the other group and these give insight into the data through numbers that are in no way related to the stock price, and thus allow much more insightful comparisons to be made inter-stock as opposed to intra-stock, that is comparing two stocks is easier if technical indicators are used as opposed to using technical overlays which are very dependant on the stock price and do not translate as well between two unique stocks.

Basic Functions

What are some basic functions that need to be defined before they are used?

Technical Overlays

Bollinger Bands - Developed by John Bollinger, these are volatility bands placed above and bellow the current stock price and are based on the standard deviation. These are designed to give an indication of how the volatility of a given stock changes over time. For any given stock over a time frame X, the three bands are calculated as such:

Upper band $=$ Simple Moving Average(X) $+$ (Standard Deviation$(X) * 2$)
Middle band $=$ Simple Moving Average(X)
Upper band $=$ Simple Moving Average(X) $-$ (Standard Deviation$(X) * 2$)

The standard time period is 20 days.

Chandelier Exit - Developed by Charles Le Beau, this is designed to help stay within a trend and not to exit early. In the case of an uptrend the Chandelier Exit will typically be below the stock price and the inverse is true in the case of a downtrend. To calculate it over a time period X:

Long = High(X) - (Average True Range(X) * 3)
Short = Low(X) + (Average True Range(X) * 3)

The standard time period is 22 days.

Ichimoku Cloud - A multifaceted indicator developed by Goichi Hosoda, a Japanese journalist. This is an average based trend identifying indicator based on the standard candlestick charts. This indicator is used as a basis in a number of other theories including Target Price

Theory. There are five plots within Ichimoku Cloud.

Using time period X.
Tenkan-sen or Conversion line = (High(X) + Low(X)) / 2 - Default X is 9
Kijun-sen or Base line = (High(X) + Low(X)) / 2 - Default X is 26
Senkou Span A or Leading span A = (Conversion Line + Base Line) / 2
Senkou Span B or Leading span B = (High(X) + Low(X)) / 2 - Default X is 52
Chikou Span or Lagging span = CloseXPeriodsAgo(X) - Default X is 26

Kaufman's Adaptive Moving Average (KAMA) - Created by Perry Kaufman, this indicator is designed to remove market noise during volatile periods. It takes three parameters, X, Y, and Z. X is the number of periods that is used by the first step of the calculation, known as the efficiency ratio. This will be shown later. The second is the number of periods for the first and fastest exponential moving average or EMA. Third is the number of periods for the second and slowest EMA. The defaults for these values are (10, 2, 30).

Efficiency Ratio = Change/Volatility
Change = Absolute Value(Close(Now) - CloseXPeriodsAgo(X)) - Default X is 10
Volatility = Sum(Absolute Value(Close - CloseXPeridsAgo(X))) - Default X is 1, this sum is done 10 times, for the last 10 changes in price.

The next stage of KAMA is a smoothing constant is calculated.

Smoothing Constant = ((Efficiency Ratio x (fastest SC - slowest SC)) + slowest SC)^2

Final stage is the use of the previous KAMA value to calculate the next value.

New KAMA = Previous KAMA + (Smoothing Constant x (Current Price - Previous KAMA))

WORK ON THIS ONE

Keltner Channels - Very similar to Bollinger Bands but instead of using standard deviation average true range is used. Created by Chester Keltner, this indicator is made up of three lines in a similar way to Bollinger Bands.

Upper Channel Line = Exponential Moving Average(X) + (2 x Average True Range(Y))
- Default X = 20, Y = 10
Middle Line = Exponential Moving Average(X)
- Default X = 20
Lower Channel Line = Exponential Moving Average(X) - (2 x Average True Range(Y))
- Default X = 20, Y = 10

Moving Averages - These can come in multiple forms with multiple names. The catch all term for this type of smoothing average is a moving average. The most simple is known as a Simple Moving Average. This takes the average of the last X data points. There is no weighting or extra steps. A more complex version is the Exponential Moving Average, this uses weighting to give the more recent values more significance in the calculation. The initial value of EMA is the same as the SMA for the same period as EMA requires an initial value.

EMA_{Today} = (Current Stock Price x K) + ($\text{EMA}_{Yesterday}$ x (1 - K))
K = 2 / (N + 1)
N = Number of periods over which the EMA is applied

Moving Average Envelopes - Based on a Moving Average, this is a percentage based envelope that provide parallel bands above and below the Moving Average. Gives an indication of trends in the data as well as an indicator for stocks that are overbought and oversold when the trend is flat.

Upper Envelope = MovingAverage(X) + (MovingAverage(X) x Y)
Lower Envelope = MovingAverage(X) - (MovingAverage(X) x Y)
Typical values X = 20 and Y = 0.025

Parabolic SAR - Developed by Welles Wilder. SAR stands for 'stop and reverse', this was called a Parabolic Time/Price System. This indicator follows the stock price as the trend is formed, and will then 'stop and reverse' when the trend ends, to follow the new trend. This is one of the more complex indicators.

In the case of a rising SAR:
EP or extreme point is a variable that is equal to the highest value of the current uptrend.
AF or acceleration factor is a variable that starts at 0.02 and is increment by 0.02 each time the EP is changed, meaning that it is incremented each time a new high is reached. The maximum value of AF is 0.2.

SAR = $\text{SAR}_{Yesterday}$ + $\text{AF}_{Yesterday}$($\text{EP}_{Yesterday}$ - $\text{SAR}_{Yesterday}$)

In the case of a falling SAR:
This uses the same variable names but inverse behaviour, the EP is equal to the lowest point in the current downtrend. AF is the same but is incremented when EP reaches a new low.

SAR = $\text{SAR}_{Yesterday}$ - $\text{AF}_{Yesterday}$($\text{EP}_{Yesterday}$ - $\text{SAR}_{Yesterday}$)

These are used to indicate a trend and once a price falls to the other side of the value calculated in the current trend, that trend is over and SAR will flip to the opposite trend.

Pivot Points - An overlay used to indicate directional movement and then shows these in potential support and resistance levels. These are predictive indicators and they exist in multiple forms, the most well known are the standard, Denmark, and Fibonacci versions. These are calculated using the previous days high, low, and close values and are then not recalculated throughout the trading day. A calculation has multiple components, the pivot point, multiple supports, and multiple resistances.

Standard Pivot Points.
Pivot Point = (High + Low + Close)/3
Support One = (PP x 2) - High
Support Two = PP - (High - Low)
Resistance One = (PP x 2) - Low
Resistance Two = PP + (High - Low)

Denmark Pivot Points. These are the most complex calculations as they have conditional statements in them and do not use the same calculation methods as the other two.
Pivot Point = X / 4
Support One = (X / 2) - High
Resistance One = (X / 2) - Low

Where X is calculated:
If Close $<$ Open: X = High + (2 x Low) + Close
If Close $>$ Open: X = (2 x High) + Low + Close
If Close = Open: X = High + Low + (2 x Close)

Fibonacci Pivot Points.
Pivot Point = (High + Low + Close)/3
Support One = PP - (0.382 x (High - Low))
Support Two = PP - (0.618 x (High - Low))
Support Three = PP - (1 x (High - Low))
Resistance One = PP + (0.382 x (High - Low))
Resistance Two = PP + (0.618 x (High - Low))
Resistance Three = PP + (1 x (High - Low))

Price Channels - Using three calculations to show an upper, lower, and middle bound, used to indicate the start of an upwards or downward trend and act accordingly.

Upper = High(X)
Center = (High(X) + Low(X)) / 2
Lower = Low(X)
The default X value is 20.

Technical Indicators

Accumulation/Distribution Line

Aroon

Aroon Oscillator

Average Directional Index (ADX)

Average True Range (ATR)

BandWidth

%B Indicator

Chaikin Money Flow (CMF)

Chaikin Oscillator

Chande Trend Meter (CTM)

Commodity Channel Index (CCI)

Coppock Curve

Correlation Coefficient

DecisionPoint Price Momentum Oscillator (PMO)

Detrended Price Oscillator (DPO)

Ease of Movement (EMV)

Force Index

Mass Index

MACD (Moving Average Convergence/Divergence Oscillator)

MACD Histogram

Money Flow Index (MFI)

Negative Volume Index (NVI)

On Balance Volume (OBV)

Percentage Price Oscillator (PPO)

Percentage Volume Oscillator (PVO)

Price Relative / Relative Strength

Pring's Know Sure Thing (KST)

Pring's Special K

Rate of Change (ROC) and Momentum

Relative Strength Index (RSI)

RRG Relative Strength

StockCharts Technical Rank (SCTR)

Slope

Standard Deviation (Volatility)

Stochastic Oscillator (Fast, Slow, and Full)

StochRSI

TRIX

True Strength Index

Ulcer Index

Ultimate Oscillator

Vortex Indicator

Williams %R

Machine Learning

IV RESULTS

V EVALUATION

VI CONCLUSIONS