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* —**

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.

The aim of any algorithmic trading system is to 'beat the market' that is to buy low and sell high so gain the most capital over any given time frame, be that a day, a year, or a decade. This paper will look into the challenge faced by these algorithms and if, using any available tools, it is possible to outstrip any high street banks offering rates. This will be done through simulation of a stock market, using real stock data, and the simulation of buying and selling these stocks by an algorithm.

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

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 382 days in the 18 months of data that was used.

Data

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. These gaps are also not regular, any given day can be cut short or a whole day could be taken without a pattern. Gaps also exist in minutes that no trades were executed, this removes any values from the data and removes the possibility of trading within these minutes.

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 and volatile.

The data downloaded was in the form:

Ticker	Date	Time	Open	High	Low	Close
AA	01/03/16	09:30	9.1100000	9.1300000	9.1100000	9.1100000
...	...	...	...	...	...	...

Table 2: AA Data

This was then converted using an R script into the form:

Date	Time	AA	AAPL	ADBE	AIG	...
01/03/16	09:30	9.11	97.66	86.15	50.59	...
...	...	...	...	...	...	...

Table 3: All Data

Thus reducing computational overhead from the running of the simulation as all data manipulation had been done prior to the data being used. In the final form no other alterations had to be made to the data other than set data types.

Set up

The simulation was set up in such a way so as to minimise the impact that gaps in the data would cause. This meant using the data to provide the iteration behaviour. The data input in the form of Table 3 was cast for each row using a typecast function to allow correct manipulation of numerical values the dates were converted to numerical values for correct comparison, as were the times. The stock values were also numerical.

The simulation then uses all unique dates within the Dates column to allow for iteration through the table at the highest level, through each individual day. One the data had been subsampled for that day, then a list of unique minutes was created. Once the data had been further sub-sampled to the current minute, each of the stock columns were iterated through and tested for NA values. If an NA value is present for that stock then it is passed over. If the value is available however then two functions are used, 'shouldBuy' and 'shouldSell'. The first used is 'shouldBuy', this will take input of current date, current time, and current stock, to allow for a decision to be made on weather to buy an amount of that stock or not. These functions are based on manipulations of three other tables, 'Active', 'Sold', and 'Ledger'. They take the form:

Unique ID	Date Bought	Time Bought	Stock	Number of Shares	Cost per Share
...	...	...	...	...	...

Table 4: Active

Unique ID	Date Bought	Time Bought	Stock	Number of Shares	Cost per Share
...	...	...	...	...	...
Amount Bought	Date Sold	Time Sold	Price per Share	Amount Sold	
...	...	...	...	...	

Table 5: Sold

Date	Value	Stock Value	Capital Value
...	...	...	...

Table 6: Ledger

Each also comes with its own population function, 'Buy', 'Sell', 'Update'. The first two are called if any decision is made. If a stocks value is such that the 'shouldBuy' function returns true then the 'Buy' function is called, taking input of date, time, stock, and amount. This is then added as another row to the Active table, as this is now an active stock that can be sold. The amount of any given stock that should be purchased is calculated through another function 'getAmount', this will test all calculations done in 'shouldBuy' and calculate a confidence value that will then dictate, using the amount of liquid capital available, the amount of the stock that will be bought. The 'shouldSell' function is then queried every time an updated value is available for any given row in the 'Active' table. Using the new value a decision will be made if the stock should be sold, if this is found to be the case then the 'Sell' function is called. The function will remove the row of the 'Active' table that corresponds to this block of stock to be sold and add it to the 'Sold' table along with all the details of the transaction.

The final table is 'Ledger' which uses the function 'Update', this is called on a regular basis to allow for tracking of the algorithm throughout the simulation. All overall details of capital are stored in this table. The Date column is a concatenation of both Time and Date in any other table to allow for minute by minute updates of the simulation or if this is not required then daily updates are possible.

Functions

These are some of the more regularly utilised functions in the simulation and are the key underpinnings of the statistical and machine learning techniques that are used throughout the simulation.

Buy(Date, Time, Stock, Amount) - Adds a row to the 'Active' table.

Sell(RowData, Date, Time, Stock, StockPrice) - Adds a row to the 'Sold' table and removes the corresponding row of the 'Active' table.

Update(Date, TotalCapital) - Adds a row to the 'Ledger' table for re-tracking of the simulation.

shouldBuy(Time, Date, Stock) - Calculates using any method set if the current stock should be bought at the current price.

shouldSell(UniqueID, Date, Time) - Calculates if the given block of stocks should be sold at the current price.

getMax(List) - Get the maximum value of the given list.

getMin(List) - Get the minimum value of the given list.

getAverage(List) - Get the mean value of the given list.

getSD(List) - Get the standard deviation of the given list.

getXDate(Date, X) - Get the date X days ago.

getXClose(Date, Stock, X) - Get the close X days ago.

getXOpen(Date, Stock, X) - Get the open X days ago.

getXDataPoints(Date, Time, Stock, X) - Get the last X data points for the given stock.

getXDayDataPoints(Date, Stock, X) - Get all data for the day X days ago.

getXSincePrice(Date, Time, Stock, Price) - Get the number of data points between the current data point and the last data point that had the given price.

getTotalGainLoss(Date, Time, Stock, X) - Over the last X data points get the total gain and loss.

Statistical Methods

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 to comparisons between two unique stocks.

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) + ($EMA_{Yesterday}$ 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.

$SAR = SAR_{Yesterday} + AF_{Yesterday}(EP_{Yesterday} - SAR_{Yesterday})$

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.

$SAR = SAR_{Yesterday} - AF_{Yesterday}(EP_{Yesterday} - SAR_{Yesterday})$

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

Aroon - Developed by Tushar Chande, Aroon is indicative of the strength of the current trend. It was designed to be similar but uniquely different to standard momentum oscillators, which focus on price relative to time, whilst Aroon focuses on time relative to price. It has two components, Up and Down, both are shown as percentages. Up will maximise on an upward trend and Down will maximise on a downward trend.

Aroon Up = ((X - DaysSinceHigh(X))/X) x 100
Aroon Down = ((X - DaysSinceLow(X))/X) x 100
Default value of X = 25. //

Aroon Oscillator - A join of the two values of Aroon into a single value.

Aroon Oscillator = Aroon Up - Aroon Down

Average Directional Index (ADX) - This is part of a group of indicators developed by Welles Wilder. The group is made up of Average Directional Index, Minus Directional Index, and Plus Directional Index, and is called the Directional Movement System.

COMPLEX NEED TIME

Average True Range (ATR) - Developed by J. Welles Wilder as a measure for volatility, ATR has been used in a wide variety of applications outside of the financial world. The initial idea was based around a concept called True Range, calculated as such:

The greatest of:
High(X) - Low(X)
ABS(High(X) - PreviousClose)
ABS(Low(X) - PreviousClose)

This was then used in conjunction with the previous ATR to calculate the new ATR.

New ATR = ((Prev ATR x (X-1)) + TR) / X - Default X is 14

BandWidth - One of the two indicators derived from Bollinger Bands by John Bollinger, the other being %B. This is a single value that takes all Bollinger Bands as components.

Bandwidth = ((Upper Band - Lower Band) / Middle Band) x 100

%B Indicator - Another Bollinger Band derivative, %B indicator gives an indication as to the relationship of the current price and the Upper and Lower Bollinger Bands.

%B = (Current Price - Lower Band) / (Upper Band - Lower Band)

Chande Trend Meter (CTM) - Developed by Tushar Chande. This indicator assigns a numerical score to a stock based on several other indicators.

NEED TO RESEARCH CANT FIND AN EQUATION

Commodity Channel Index (CCI) - Developed by Donald Lambert. Used to show a comparison between the current price and the average price over a given timespan. Uses multiple other calculations as component parts, including Simple Moving Average, Typical Price, and Mean Deviation.

Typical Price = (High + Low + Close) / 3

Mean Deviation = SUM(ABS(Period Value - Period Average)) / X
Default X = 20. Find the sum of the deviation from the average value of the last 20 periods within each period.

CGI = (Typical Price - X period SMA of Typical Price) / (0.05 x Mean Deviation)// Default X = 20.

Coppock Curve - Developed by Edwin Coppock. Using a Weighted Moving Average as well as a period based Rate Of Change, this simple indicator as been used by many as a sell indicator as the value crosses the positive-negative boundary. It is calculated as the WMA of the ROC plus the ROC over a different period.

Coppock Curve = WeightedMovingAverage(X, RateOfChange(Y)) + RateOfChange(Z)
Default X = 10, Y = 14, Z = 11.

Correlation Coefficient - This is a measure of the similarities between two stocks, this is useful to identify trends that effect many stocks. This uses the variance of two stocks using an average price over a given time frame as well as the covariance between them.

Variance($Stock_1$) = Average($Price^2$, X) - (Average(Price, X)^2)
Variance($Stock_2$) = Average($Price^2$, X) - (Average(Price, X)^2)
Covariance($Stock_1$, $Stock_2$) = Average(Price($Stock_1$) x Price($Stock_2$), X) - (Average(Price($Stock_1$), X) x Average(Price($Stock_2$), X))
Correlation Coefficient = Covariance / SQRT(Variance($Stock_1$) x Variance($Stock_2$))

DecisionPoint Price Momentum Oscillator (PMO) - An oscillator that is calculated as a smoothed version of the rate of change using the exponential moving average as part of the smoothing process.

Smoothing Multiplier = (2 / Time period)
Custom Smoothing Function = Close - Smoothing Function(previous day) * Smoothing Multiplier + Smoothing Function(previous day)
PMO Line = 20-period Custom Smoothing of (10 * 35-period Custom Smoothing of (((Today's Price/Yesterday's Price) * 100) - 100))
PMO Signal Line = 10-period EMA of the PMO Line

Detrended Price Oscillator (DPO) - Used to identify cycle details. High, low, and cycle length can be calculated.

DPO = Price X/2 + 1 periods ago less the X-period simple moving average.

Mass Index - Volatility indicator used to show a trend reversal before it occurs. Developed by Donald Dorsey.

Single EMA = 9-period exponential moving average (EMA) of the high-low differential
Double EMA = 9-period EMA of the 9-period EMA of the high-low differential
EMA Ratio = Single EMA divided by Double EMA
Mass Index = 25-period sum of the EMA Ratio

MACD (Moving Average Convergence/Divergence Oscillator) - Developed by Gerald Appel. Is said to be one of the most effective momentum indicators as well as being very simplistic to perform.

MACD Line: (12-day EMA - 26-day EMA)
Signal Line: 9-day EMA of MACD Line
MACD Histogram: MACD Line - Signal Line

MACD Histogram - Developed by Thomas Aspray as a development on the MACD, to preemptively detect crossovers between the two lines in MACD.

MACD: (12-day EMA - 26-day EMA)
Signal Line: 9-day EMA of MACD
MACD Histogram: MACD - Signal Line

Percentage Price Oscillator (PPO) - A momentum oscillator that is related to MACD. Calculated as a percentage showing the relationship between two moving averages.

Percentage Price Oscillator (PPO): (12-day EMA - 26-day EMA)/26-day EMA x 100
Signal Line: 9-day EMA of PPO
PPO Histogram: PPO - Signal Line

Pring's Know Sure Thing (KST) - Developed by Martin Pring. Using the smoothed rate of change over four different length periods, this momentum oscillator gives a more well based indication of movement than a typical momentum oscillator.

RCMA1 = 10-Period SMA of 10-Period Rate-of-Change
RCMA2 = 10-Period SMA of 15-Period Rate-of-Change
RCMA3 = 10-Period SMA of 20-Period Rate-of-Change
RCMA4 = 15-Period SMA of 30-Period Rate-of-Change
KST = (RCMA1 x 1) + (RCMA2 x 2) + (RCMA3 x 3) + (RCMA4 x 4)
Signal Line = 9-period SMA of KST

Pring's Special K - Developed by Martin Pring, this momentum oscillator is a concatenation of three different velocities to provide more stable prediction of movement.

Special K = 10 Period Simple Moving Average of ROC(10) * 1
+ 10 Period Simple Moving Average of ROC(15) * 2
+ 10 Period Simple Moving Average of ROC(20) * 3
+ 15 Period Simple Moving Average of ROC(30) * 4
+ 50 Period Simple Moving Average of ROC(40) * 1
+ 65 Period Simple Moving Average of ROC(65) * 2
+ 75 Period Simple Moving Average of ROC(75) * 3
+100 Period Simple Moving Average of ROC(100)* 4
+130 Period Simple Moving Average of ROC(195)* 1
+130 Period Simple Moving Average of ROC(265)* 2
+130 Period Simple Moving Average of ROC(390)* 3
+195 Period Simple Moving Average of ROC(530)* 4

Rate of Change (ROC) and Momentum - A pure momentum oscillator used in many other indicators.

ROC = [(Close - Close n periods ago) / (Close n periods ago)] * 100

Relative Strength Index (RSI) - Developed by J. Welles Wilder. A range of zero to one hundred, RSI, a momentum oscillator, gives an indication of the speed and change of price movements.

RSI = 100 - (100 / 1 + RS)
RS = Average Gain / Average Loss

StockCharts Technical Rank (SCTR)

MIGHT BE USEFUL BUT CHECK FIRST

Slope - A very simple idea. The main concept is to calculate the line of best fit over a given

time frame to show to trend over that time frame. This is a very simple tool used to give general trends.

Stochastic Oscillator (Fast, Slow, and Full) - Developed by George C. Lane. A momentum indicator that uses the close data along with the range between the high and low values over a given period to show the current momentum. Lane states; this "follows the speed or the momentum of price. As a rule, the momentum changes direction before price."

%K = (Close - Lowest Low over X)/(Highest High over X - Lowest Low over X) * 100.
%D = Simple Moving Average of %K over Y Periods.
Default value of X is 14. Default value of Y is 3.

StochRSI - Developed together by Tushar Chande and Stanley Kroll. Using Relative Strength Index or RSI, StochRSI is a measure of RSI relative to the max range of RSI over a set period. This indicator has a range of 0 to 1 with 0 indicating the lowest point over the period with 1 indicating the highest point over the period.

StochRSI = (RSI - Lowest Low RSI over X) / (Highest High RSI over X - Lowest Low RSI over X)
Default X value is 14.

TRIX - Developed by Jack Hutson. A triple smoothed Exponential Moving Average is used to calculate the percentage change over the last period.

Single Smoothed EMA = EMA of Close over X periods.
Double Smoothed EMA = EMA of Single Smoothed EMA over X periods.
Triple Smoothed EMA = EMA of Double Smoothed EMA over X periods.
TRIX = Single period percentage change in Triple Smoothed EMA.

True Strength Index - Developed by William Blau. Using two double smoothed price changes this is a momentum oscillator with the benefit of being relatively resistant to noise. Made up of two double smoothed price changes and the TSI calculation, this is a relatively simple indicator.

Double Smoother Price Change.
PC = Current Price minus Prior Price
Single Smoothed PC = EMA of PC over X periods.
Double Smoothed PC = EMA over Y periods of Single Smoothed PC.
Default X value is 25. Default Y value is 13.

Double Smoothed Absolute Price Change.
Absolute PC = ABS(Current Price minus Prior Price)
Single Smoothed PC = EMA of APC over X periods.
Double Smoothed PC = EMA over Y periods of Single Smoothed APC.
Default X value is 25. Default Y value is 13.

True Strength Value.
TSI = 100 x (Double Smoothed Price Change / Double Smoothed Absolute Price Change)

Ulcer Index - Developed by Peter Martin and Byron McCann. This volatility indicator was originally designed to measure downside risk in mutual funds, although it has now been re-purposed.

Percent-Drawdown = ((Close - Max Close over X periods)/Max Close over X periods) x 100
Squared Average = (Sum of Percent-Drawdown over X periods Squared)/X
Ulcer Index = Square Root of Squared Average
Default X value is 14.

Ultimate Oscillator - Developed by Larry Williams. This is a triple time frame based momentum oscillator. The use of multiple time frames is limit the effect that noise can have on a typical momentum oscillator. There are several steps to the Ultimate Oscillator, all of which rely on Buying Pressure, BP, and True Range, TR.

BP = Close - Min(Low, Prior Close)
TR = Max(High, Prior Close) - Min(Low, Prior Close)

Average X = (BP Sum over X periods) / (TR Sum over X periods)
Average Y = (BP Sum over Y periods) / (TR Sum over Y periods)
Average Z = (BP Sum over Z periods) / (TR Sum over Z periods)

Ultimate Oscillator = 100 x ((4 x Average X)+(2 x Average Y)+Average Z)/(4+2+1)
Default values are X = 7, Y = 14, Z = 28.

Vortex Indicator - Developed by Etienne Botes and Douglas Siepman, based on the work of Welles Wilder and Viktor Schauberger. Using the relationship between two oscillators a base, one capturing positive trend movement and the other capturing negative, the vortex indicator is adept as showing the bias of the data.

+VM = ABS(Current High - Prior Low)
-VM = ABS(Prior Low - Current High)

+VMX = Sum of +VM over X periods
-VMX = Sum of -VM over X periods

TRX = Sum of True Range over X periods

+VIX = +VMX/TRX
-VIX = -VMX/TRX

Default X value is 14.

The crossovers of these two values is then used to identify the start and end of a trend and the direction of said trend.

Williams %R - Developed by Larry Williams and based on the Stochastic Oscillator developed by George C. Lane. This is the inverse of the Fast SO as the FSO reflects the relationship between the Close and the Lowest Low over a given period, whilst this reflects the relationship between the Close and the Highest High. This momentum indicator has the same benefits and drawbacks as the Stochastic Oscillator.

%R = (Highest High over X - Close)/(Highest High over X - Lowest Low over X) * -100
Default value of X is 14.

Testing

Once each of these methods had been implemented they were each tested. With each being subtly different, the exact testing methods used for each one differed slightly. An example; with Moving Averages two were calculated with the difference being the length over which they were calculated. Then the relationship between these two was observed and if the shorter term average was significantly higher than the longer term average then an upward trend was being observed, when the short term average was significantly lower than the long term average then the opposite was true. Once each has been tested individually they can also be used in conjunction with other techniques, this has also been shown in the results section.

Machine Learning

Machine Learning is a very broad topic with much of contemporary computer science based on it or consisting of research done around it. The main technique that is being applied is that of a Support Vector Machine or SVM. This maximises the distance between two clusters, and is based in statistical learning theory. Using a kernel mapping, that is mapping a vector into a higher dimensional space, allows for linear separation to be performed, even on non linear datasets if the correct kernel mapping is chosen. Linear separation is the key to this machine learning technique, it maximises the distance between the known elements of each of the two classes using was is basically a constraint satisfaction problem. This 'margin' between the two classes is our confidence in the separation, if the margin is small then there is very little distance within the higher dimensional space between the two classes meaning a smaller confidence of success, if the margin is large then there is a higher confidence in the successful determination of a given points class.

Set Up

The set up of data and input revolved around two functions. These are 'getPeaks' and 'getTroughs'.

They take input of date, time, stock, and X, X is the number of data points that will be used to make a decision about the buying or selling potential of the given data point. After X/2 data

points have passed then a decision will be made using X/2 data points either side. 'getPeaks' will take this input, sub-sample the data and decide if this point was optimal to sell at, meaning it is a peak. 'getTroughs' will do the opposite, it will decide if this is a point at which it would have been optimal to buy. This data is then stored in tables 'SVMBuyData' and 'SVMSellData'. These are in the form:

Date	Time	Stock Value	Should Buy
01/03/16	09:30	9.11	False
...	...	...	...

Table 7: SVMBuyData

and

Date	Time	Stock Value	Should Sell
01/03/16	09:30	9.11	False
...	...	...	...

Table 8: SVMSellData

These tables are then used as input to the SVM learning function that is found as part of the e1071 CRAN library, which contains all set up and query functions for the SVM.

The functions 'getPeaks' and 'getTroughs' function in a similar way to 'shouldBuy' and 'shouldSell', they are very tune-able and are able to be changed to very different techniques without having an effect on the main functionality of the simulation. An example of a technique that could be used within these functions is to differentiate. This technique is designed around the idea that data can be modelled as an equation and when the gradient of the line is 0, this is a stationary point, these will either be peaks or troughs. Then the second derivative is used in conjunction with the data either side of the current point to decide if this stationary point is a peak or trough. This is an example of one of the many techniques used. Other techniques used are;

Rolling average - Take the highest points over variable length rolling average.
Spikes - Take any point as a peak if both neighbours are lower that that value and the inverse for troughs.
Median - Take the top and bottom X% of values.

Query

The query function within the e1071 library is very simple, given an SVM that has learnt its separator a function call will return the group that the given point belongs to.

This system is managed within the 'shouldBuy' and 'shouldSell' functions that were discussed within the statistical methods section. These have been modified so as to retrain and query the 'shouldBuySVM' and the 'shouldSellSVM' each time that they are called. Firstly one of these functions will be called within the simulation, for simplicity, 'shouldBuy' will be used

as an example. 'shouldBuy' is called, within this function is the call to create and train the SVM associated with this function, namely 'shouldBuySVM', this is trained on the dataframe that is shown in table 7. The data within this dataframe has been continuously updated throughout the simulation so as to avoid bulk computation when the SVM is used. This is done by continual calls to the 'getPeaks' function on regular intervals. Once the SVM has been trained then it is queried with the latest price for the given stock. This will return a class and then a decision will be made based on the class that has been returned.

IV RESULTS

- Testing Criteria for each of the stat methods
- decision criteria for each of the peak trough methods
- conjunction test criteria for each of the good stat methods
- best stat method result possible
- best ml method result possible

V EVALUATION

VI CONCLUSIONS