DESIGN DOCUMENT

Objective:

To get the trading position for the market.

Description:

The trading algorithm is based in the following formulation: -

$$\min\left\{\frac{1}{2}\left(\|\omega\|^2 + A\theta^2\right) + \alpha\sum_{t=1}^{T} a_t - \beta\sum_{t=1}^{T} b_t + \gamma\sum_{t=1}^{T}(c_t + d_t)\right\} \quad (1)$$

Subject to:-

$$r_t\left[\omega^T \emptyset(x^{t-1}) + \theta\right] + (a_t - b_t) = 0 \qquad \text{such that } a_t, b_t \geq 0$$

$$\delta\omega^T\left[\emptyset(x^t) - \emptyset(x^{t-1})\right] + (c_t - d_t) = 0 \qquad \text{such that } c_t, d_t \geq 0$$

The formulation named as "OMEGA RATIO FORMULATION" aims at minimizing the losses (contained in a_t), transaction costs (contained in c_t and d_t) and maximizing the profits (contained in b_t).

The Ø's represents the samples. Each sample is BarsBack dimensional i.e. each sample has past BarsBack features which can simply be the closing prices or some complex Indicator or a combination of both.

The T is the TradingWindowSize (T>=1).

α, β, γ are the weighing factors for the cumulative losses, profits and transaction costs respectively. The condition α>β ensures that minimizing losses is more important than maximizing profits. So we carefully implement this condition while selecting the parameters.

r_t is the return at time t given by r_t =price$_t$ -price$_{t-1}$

δ is the transaction cost incurred for the scrip being traded.

The above formulation is our primal formulation. After we apply KKT conditions and convert the above into dual formulation, it becomes:-

$$Min\frac{1}{2}\sum_{t=1}^{T}\sum_{u=1}^{T}\Big\{\lambda_t r_t\lambda_u r_u\phi(x^{t-1})^T\phi(x^{u-1}) + \lambda_t r_t g_u\delta[\phi(x^{t-1})^T\phi(x^u) - \phi(x^{t-1})^T\phi(x^{u-1})] + g_t g_u\delta^2[\phi(x^t)^T\phi(x^u) - \phi(x^t)^T\phi(x^{u-1}) - \phi(x^u)^T\phi(x^{t-1}) + \phi(x^{t-1})^T\phi(x^{u-1})] + \frac{1}{A}\lambda_t r_t\lambda_u r_u + \lambda_u r_u g_t\delta[\phi(x^{u-1})^T\phi(x^t) - \phi(x^{u-1})^T\phi(x^{t-1})]\Big\} \quad (2)$$

Constrained to the following conditions: -

$\beta \leq \lambda_t \leq \alpha$

$-\gamma \leq g_t \leq \gamma$

Here, $\Phi(x^t)^T\Phi(x^u) = K(x^t, x^u)\ or\ Kernel(x^t, x^u)$. The kernel function can be one of the many commonly used kernels such as rbf kernel, polynomial kernel etc.

Minimizing equation (2) gives set of λ's and g's which gives us ω and θ.

$\omega = \sum_{t=1}^{T}\lambda_t r_t\Phi(x^{t-1}) + \sum_{t=1}^{T}g_t\delta(\Phi(x^t) - \Phi(x^{t-1}))$

$\theta = \frac{1}{A}\sum_{t=1}^{T}\lambda_t r_t$

Ideally, the current position in the market is given by sgn($\omega^T\emptyset(x^t) + \theta$) .But to inculcate the confidence of trader into our strategy and to rule out the possibility of bad trades, we compute the current position as: -

If TempPosition[t]> threshold and TempPosition[t-1]>threshold and HighPrice[t]>HighPrice[t-1] and shortmovingaverage[t]>longmovingaverage[t], Then Current Position=1. {i.e If we are confident enough, enter the trade}

ElseIf TempPosition[t]> threshold and Position[t-1]==1 and shortmovingaverage[t]>longmovingaverage[t], Then Current Position=1 {Continue your position as long as trend is positive and Trader is confident}

ElseIf TempPosition[t]< -threshold and Temposition[t-1]<-threshold and HighPrice[t]<HighPrice[t-1] and shortmovingaverage[t]<longmovingaverage[t], Then Current Position=-1 . {i.e If we are confident enough, enter the trade}

ElseIf TempPosition[t]< -threshold and position[t-1]==-1 and shortmovingaverage[t]<longmovingaverage[t], Then Current Position=-1 {Continue your position as long as trend is negative and Trader is confident}

Else, Current Position=0

Where , $(\omega^T \emptyset(x^t) + \theta / || \omega ||)$ can be termed as TempPosition. Division by $|| \omega ||$ helps to normalize the output of trader.

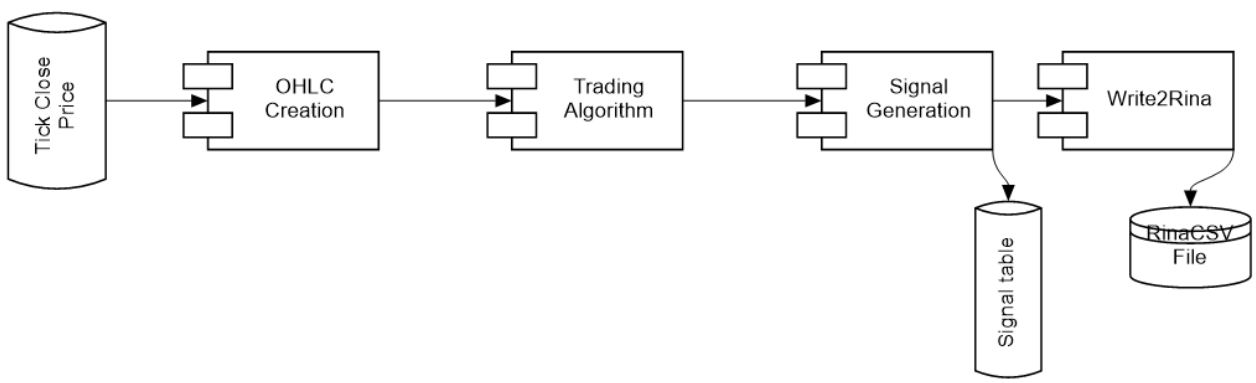
Thus alpha, beta, gamma, dell, A, threshold, kernelfunc, TradingWindowSize, BarsBack, BarTimeInterval are the parameters of algorithm.

So the algorithm requires past T closing prices to compute the current position. The output of the of the trading algorithm is ternary with 1, -1, 0 as the 3 possible states representing the long, short, and neutral position in the market respectively.

The preprocessing and post processing can be done can be done in an outside module. Preprocessing involves reading the data prices for each tick and preparing the OHLC bars for the time interval specified by BarTimeInterval. Sequence of states can be converted into Trading Signals post processing. We stand Long as long as the output is +1, Short as long as the output is -1 and Neutral whenever the output is 0. Static Trailing and Stop Loss can be applied after we take positions in the market.

After we get the position for current time instant or current bar, we repeat the procedure for next bar by shifting the entire T sized Trading Window to the right exactly by 1 bar.

Component Relationship Diagram:



Design Considerations:

Assumptions:

- The input stream is available for every tick in a continuous manner.
- The signal file is generated for every entry and exit of the trade.
- The RinaCSV file is generated at the end of the trading session.
- The ternary output of Trader is stored for reference purpose.

Constraints:

- The input to algorithm is rounded off to 2 places of decimal.
- The rounding off can be done using methods OHLCCreation before calling the algorithm.

Optimizer Used:

ISMO algorithm is being used to optimize the dual objective.

The update rule for λ_k and g_k is given by:-

$$\lambda_k^{new} = \lambda_k^{old} - \left(\frac{f_{old}(x_{k-1})}{r_k \left(K_{k-1,k-1} + \frac{1}{A} \right)} \right)$$

$$g_k^{new} = g_k^{old} - \left(\frac{f_{old}(x_k) - f_{old}(x_{k-1})}{\delta_k \left(K_{k,k} + K_{k-1,k-1} - 2K_{k,k-1} \right)} \right)$$

Where,

$$f_{old(x_k)} = \sum_{u=1}^{T} \lambda_u r_u \left(K_{k,u-1} + \frac{1}{A} \right) + \sum_{u=1}^{T} g_u \delta \left(K_{k,u} - K_{k,u-1} \right)$$

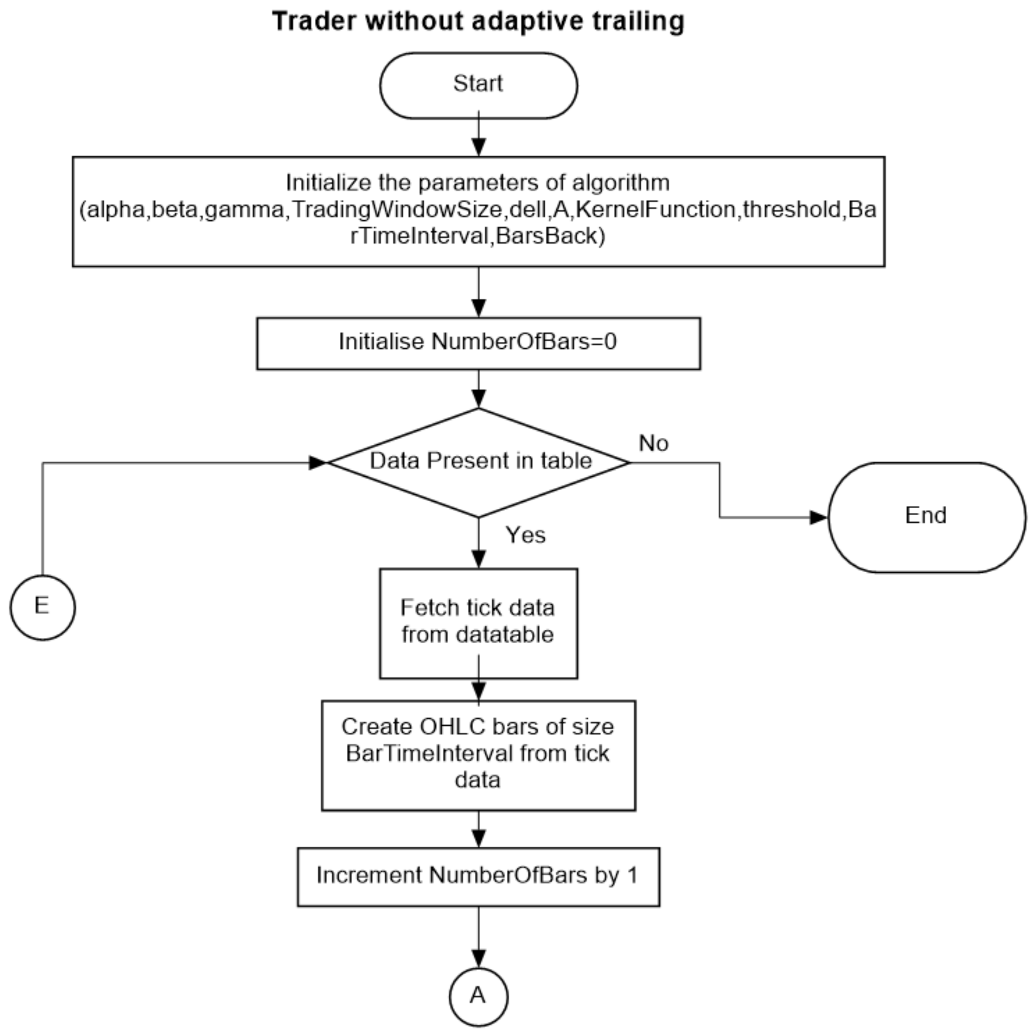
And, $K_{k,k} = Kernel(x_k, x_k)$

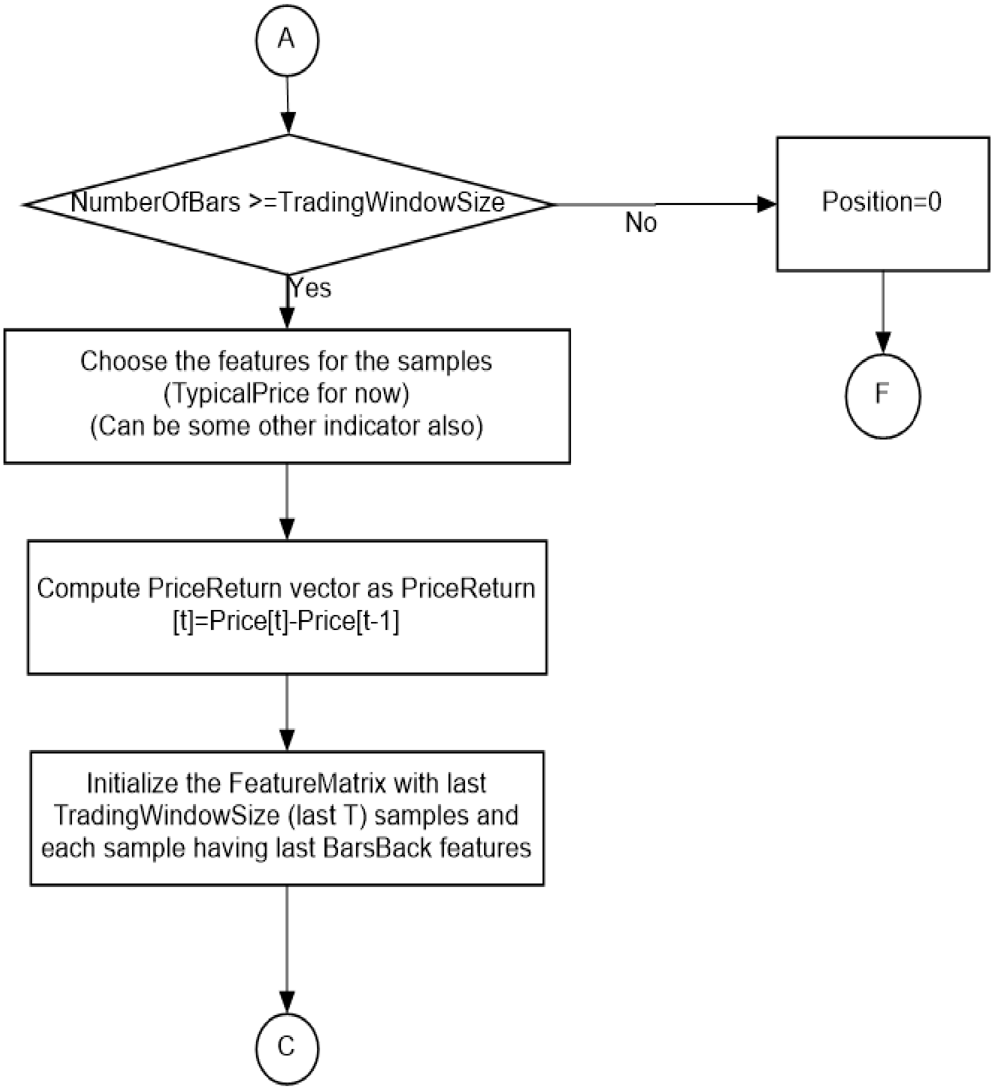
DataStructure Used:

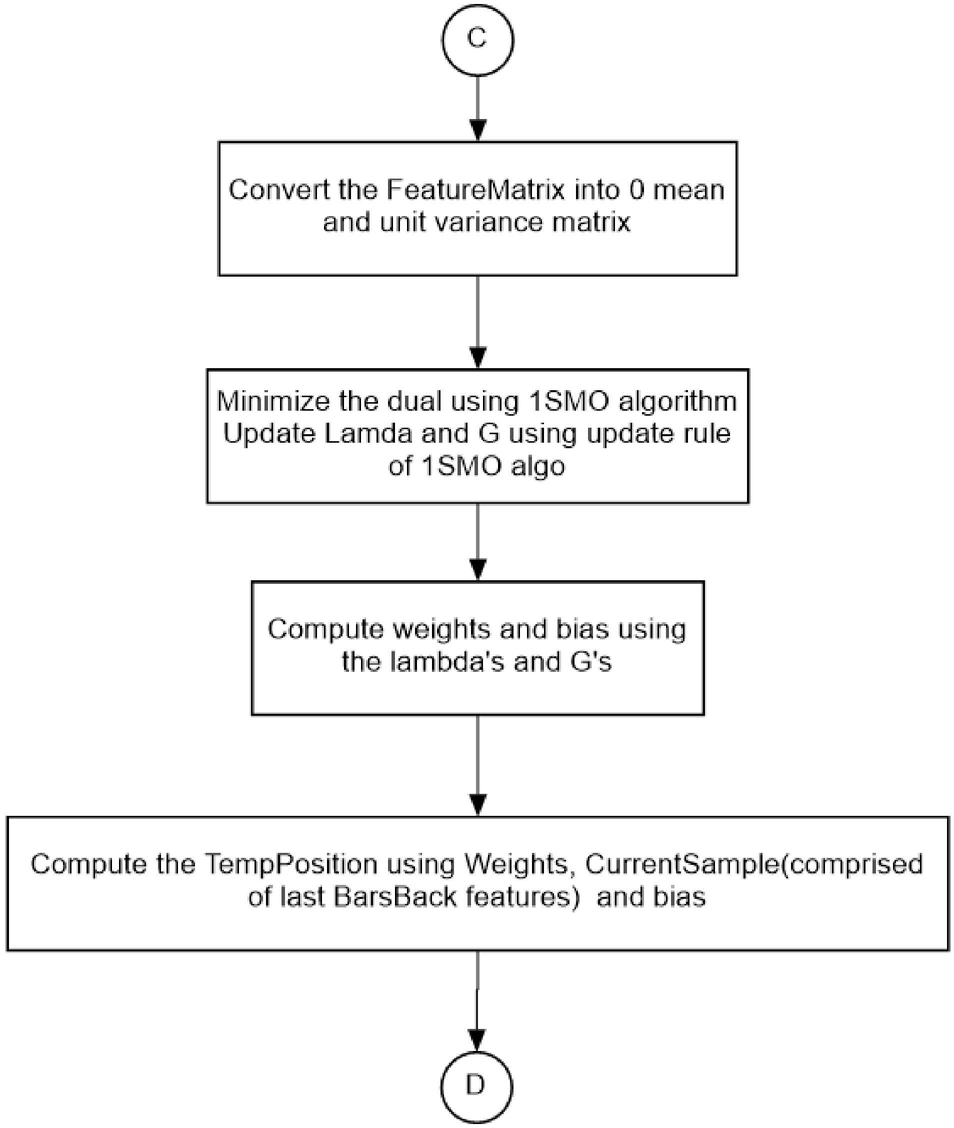
A class with the following UML diagram:-

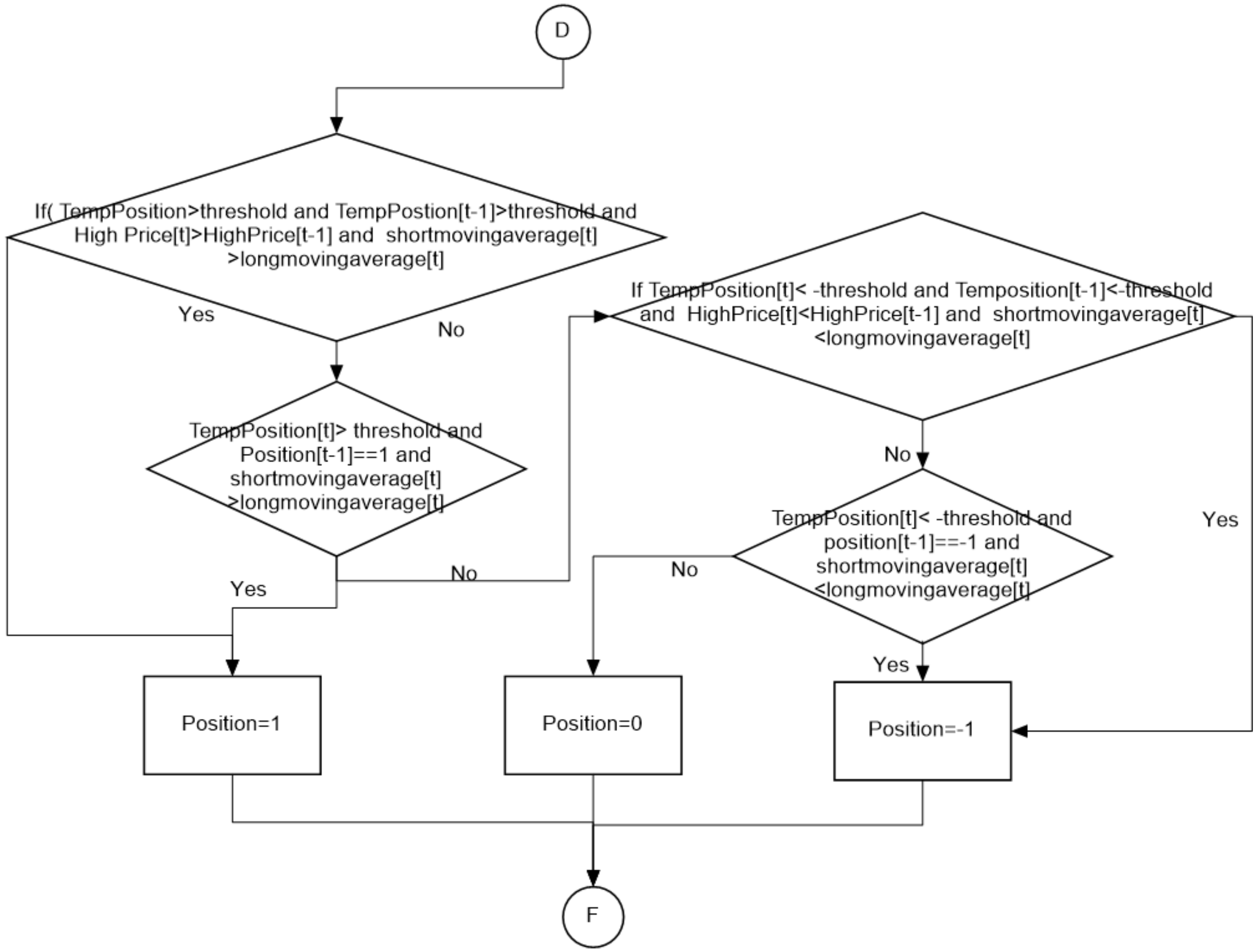
Trader
Private: m_str_dbName m_str_DataTableName m_str_SignalTableName m_str_RinaFileName m_str_UserName m_str_Password m_f_alpha m_f_beta m_f_gamma m_f_dell m_i_TradingWindowSize m_f_threshold m_i_BarsBack m_f_A m_i_BarTimeInterval m_i_TotalTicks m_i_t m_str_SessionCloseTime m_f_OHLCMatrix m_i_Position m_f_TempPosition m_f_FeatureMatrix m_f_Return m_i_ShareQuantity m_i_PositionInMarket m_i_RinaInternalFlag m_i_GenerateRina m_str_RinaFileName   m_KernelFunc() m_login() m_MovingAverage() m_OneSMO() m_CreateTable() m_CreateRinaFile()
Public: m_OHLCCreation() m_TradingAlgorithm() m_SignalGeneration() m_Write2Rina()

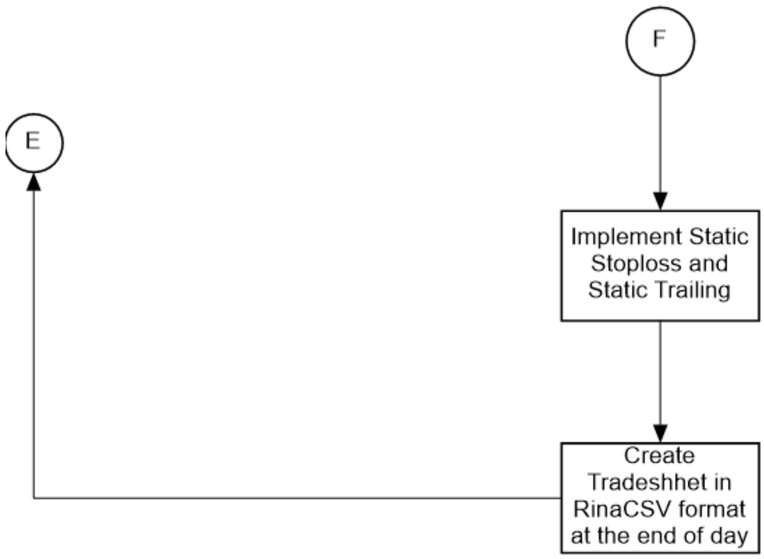
Flowchart for TradingAlgorithm:




A
NumberOfBars >=TradingWindowSize
No
Position=0
F
Yes
Choose the features for the samples
(TypicalPrice for now)
(Can be some other indicator also)
Compute PriceReturn vector as PriceReturn
[t]=Price[t]-Price[t-1]
Initialize the FeatureMatrix with last
TradingWindowSize (last T) samples and
each sample having last BarsBack features
C


C
Convert the FeatureMatrix into 0 mean and unit variance matrix
Minimize the dual using 1SMO algorithm Update Lamda and G using update rule of 1SMO algo
Compute weights and bias using the lambda's and G's
Compute the TempPosition using Weights, CurrentSample(comprised of last BarsBack features) and bias
D


D
If(TempPosition>threshold and TempPostion[t-1]>threshold and High Price[t]>HighPrice[t-1] and shortmovingaverage[t] >longmovingaverage[t]
Yes
No
TempPosition[t]> threshold and Position[t-1]==1 and shortmovingaverage[t] >longmovingaverage[t]
Yes
No
If TempPosition[t]< -threshold and Temposition[t-1]<-threshold and HighPrice[t]<HighPrice[t-1] and shortmovingaverage[t] <longmovingaverage[t]
No
Yes
TempPosition[t]< -threshold and position[t-1]==-1 and shortmovingaverage[t] <longmovingaverage[t]
No
Yes
Position=1
Position=0
Position=-1
F



PSEUDOCODE

Pseudocode for CreateOHLCMatrix:-

Aim: To compute the OHLC matrix.

Input: TickOHLCMatrix, NumberOfTicks

Output: OHLCMatrix

Date=TickOHLCMatrix[NumberOfTicks][0]

Time=TickOHLCMatrix[NumberOfTicks][1]

Open=Rounded to 2 decimal places(TickOHLCMatrix[0][2])

High=Maximum (Rounded to 2 decimal places (TickOHLCMatrix[0:NumberOfTicks][3]))

Low= Minimum(Rounded to 2 decimal places(TickOHLCMatrix[0:NumberOfTicks][4]))

Close=Rounded to 2 decimal places(TickOHLCMatrix[NumberOfTicks][5])

Append [Date, Time, Open , High, Low, Close] to m_f_OHLCMatrix

Increment m_i_t by 1.

Pseudocode for Trading Algorithm:-

Aim: To compute the position for time instant 't'

Input: m_f_OHLCMatrix, m_i_t

Output: m_i_Position[m_i_t]

Initialise Parameters (m_i_A,m_f_Alpha,m_f_betaa,m_f_gamma,m_f_dell,m_i_TradingWindowSize,m_i_BarsBack,m_f_Threshold,m_i_BartimeInterval)

Initialise l_f_FeatureVector=MovingAverage(TypicalPrice)

If m_i_t==m_i_TradingWindowSize

    Initialise m_f_TempPosition=zeros(1,m_i_TradingWindowSize-1)

    Initialise m_f_Profit=zeros(1,m_i_TradingWindowSize-1)

    Initialise m_f_CumulativeProfit=zeros(1,m_i_TradingWindowSize-1)

    Initialise m_f_FeatureMatrix=zeros(m_i_BarsBack,m_i_TradingWindowSize)

    For temp=1:TradingWindowSize-1

    Intialise m_f_Returns=ClosePrice[t]-ClosePrice[t-1]

    Initialise m_f_Weights=zeros(m_i_BarsBack,m_i_TradingWindowSize-1)

Append 0.0 to m_f_TempPosition,m_f_Profit,m_f_CumulativeProfit

Append ClosePrice[m_i_t]-ClosePrice[m_i_t-1] to m_f_Returns

Append horizontally l_f_FeatureVector[m_i_t-m_i_BarsBack:m_i_t] to m_f_Featurematrix.

l_f_PhiUsed=m_f_FeatureMatrix[:][m_i_t-m_i_TradingWindowSize:m_i_TradingWindowSize]

Convert l_f_PhiUsed to 0 mean and unit variance matrix.

l_f_l,l_f_g=OneSMO(l_f_PhiUsed)

Compute m_f_Weights[:][m_i_t] using l_f_l and l_f_g

Compute l_f_Theta using l_f_l and l_f_g.

Compute m_f_TempPosition[m_i_t] using m_f_Weights, l_f_PhiUsed and l_f_Theta

```
If (m_f_TempPosition[m_i_t]>m_f_Threshold) and (m_f_TempPosition[m_i_t-1]>m_f_Threshold) and (l_f_BarHigh[m_i_t]>l_f_BarHigh[m_i_t-1]) and (l_f_ShortMA[m_i_t]>l_f_LongMA[m_i_t])

    m_i_Position[m_i_t]=1

Else If  (m_f_TempPosition[m_i_t]>m_f_Threshold) and m_i_Position[m_i_t-1]==1 and (l_f_ShortMA[m_i_t]>l_f_LongMA[m_i_t]):

    m_i_Position[m_i_t]=1

Else IF  (m_f_TempPosition[m_i_t]<-m_f_Threshold) and
(l_i_TempPosition[m_i_t-1]<-m_f_Threshold) and
(l_f_BarHigh[m_i_t]<HighPrice[m_i_t-1]) and
(l_f_ShortMA[m_i_t]<l_f_LongMA[m_i_t]):

    m_i_Position[t]=-1

Else If  (m_f_TempPosition[m_i_t]<-m_f_Threshold) and m_i_Position[m_i_t-1]==-1 and (l_f_ShortMA[m_i_t]<l_f_LongMA[m_i_t]):

    m_i_Position[m_i_t]=-1

Else:

    M_i_Position[m_i_t]=0
```

<u>Pseudocode for OneSMO</u>:

Aim: To get optimum lamda and G

Input: l_f_PhiUsed

Output: $l\_f\_l^{new}$,$l\_f\_g^{new}$

Initialize l_f_l=1000*ones(1,TradingWindowSize)

Initialize l_f_g=1000*(1,TradingWindowsize)

Initialise l_i_MaxIterations

Initialise l_i_epsilon

Initialize l_i_iterate=0

While (l_i_iterate<l_i_MaxIterations) OR ($|\lambda^{new} – \lambda^{old}|$<l_i_epsilon and $|g^{new} – g^{old}|$<epsilon)

For k= 1 to m_i_TradingWindowSize

Compute λ_k^{new} using 1SMO update rule.

For k= 1 to m_i_TradingWindowSize

Compute g_k^{new} using 1SMO update rule.

l_i_iterate=l_i_iterate+1

Return $l\_f\_l^{new}$ and $l\_f\_g^{new}$

Pseudcode For SignalGneration:

Aim: To generate the Signals from Positions.

Input: Positions

If Position[m_i_t-1]==0 and Position[m_i_t]==1

Generate a Long Entry Signal

If Position[m_i_t-1]==0 and Position[m_i_t]==-1

Generate a Short Entry Signal

If Position[m_i_t-1]==-1 and Position[m_i_t]==1

Generate a Short Exit Signal

Generate a Long Entry Signal

```
If Position[m_i_t-1]==1 and Position[m_i_t]=-1
        Generate a Long Exit Signal
        Generate a Short Entry Signal
If Position[m_i_t-1]==1 and Position[m_i_t]==0
        Generate a Long Exit Signal
If Position[m_i_t-1]=-1 and Position[m_i_t]==0
        Generate a Short Exit Signal
```

Pseudocode for Write2Rina:-

Aim: To generate RinaCSV file from Signals at the end of the session.

Input: Signals for the day (SignalsRead)

```
l_i_Datasize=length(SignalsRead,1)
l_i_Counter=0
While l_i_Counter<l_i_DataSize
        If m_i_RinaInternalFlag==1
                Write Trade Entry line into the file.
                Increment l_i_Counter  by 1.
                Set m_i_RinaInternalFlag=2
        If m_i_RinaInternalFlag==2
                Write a Trade Exit Line into the file
                Increment l_i_Counter by 1.
                Set m_i_RinaInternalFlag=1
                Increment m_i_TradeNum by 1.
End of while
```