Inter IIt Tech Meet 11.0

Submitted by:
Team ID 43

EFFICACY OF PRICE ACTION TRADING STRATEGIES IN THE CONTEXT OF THE INDIAN EQUITIES MARKET

INDEX

METHODOLODY

Two propositions are discussed in this study.

We begin by outlining the hypothesis before going over the basic procedure followed to implement and test the hypothesis.

Next, we discuss the code-design and tools, we employed to backtest our theory.

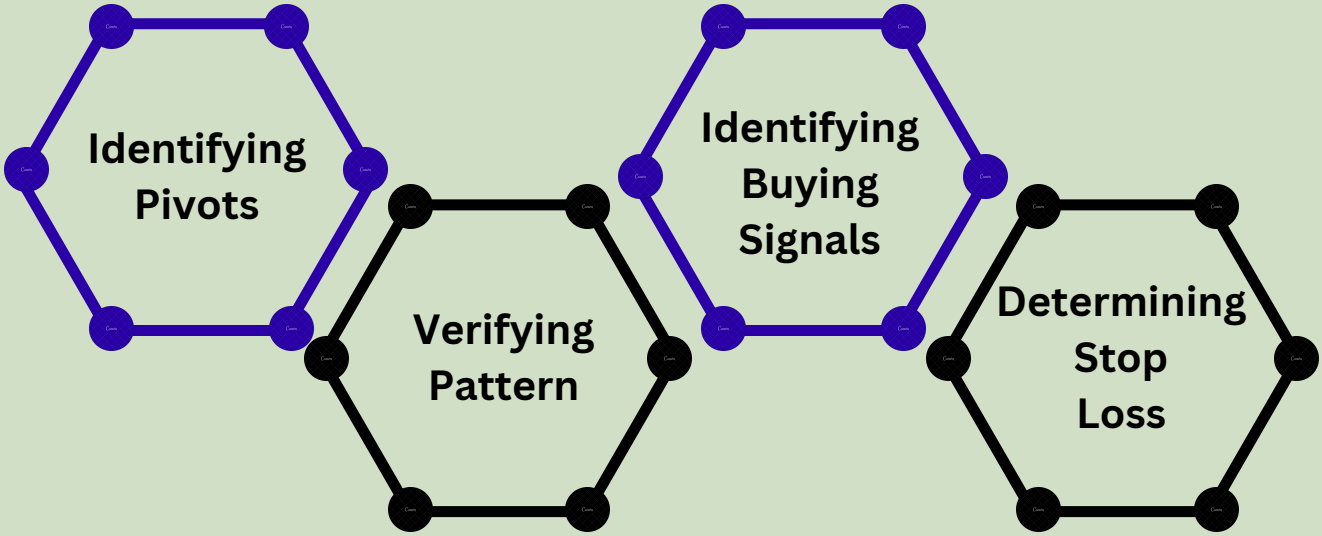
Finally analysis is offered, followed by the conclusion.

HYPOTHESIS

Hypothesis 1:

Hypothesis 2:
Indian Volatility Index (INDIA VIX) follows an opposite trend to NIFTY FUTURES. CandleStick Reversal patterns, combined with Support and Reversal levels when applied on India VIX, give an effective indication of opposite trend reversal in NIFTY FUTURES.

Method for detecting pattern



Verifying Pattern

We consider 5 alternating extremas E1, E2,...E5 where E1 is the most recently encountered extrema. Here we call E3 as the extrema of the head, and E1 and E5 as the extremas of shoulders. E1, E3, E5 are local minimas whereas E2, E4 are local maximas. We define the equation of neckline as the line joining E2 and E4. S1 , H, S2 are defined as the distance between E1, E3, E5 and the neckline respectively. We get the Equation of neckline as :

$$y = \left(\frac{E_2 - E_4}{X_2 - X_4}\right)x + c$$

We have characterised IHS by the following conditions:

$$c = \left(\frac{E_4X_2 - E_2X_4}{X_2 - X_4}\right) \quad (1)$$

$$H = \left(\frac{E_4 - E_2}{X_2 - X_4}\right)X_3 + c - E_3 \quad (2)$$

$$S_1 = \left(\frac{E_4 - E_2}{X_2 - X_4}\right)X_1 + c - E_1 \quad (3)$$

$$S_2 = \left(\frac{E_4 - E_2}{X_2 - X_4}\right)X_5 + c - E_5 \quad (4)$$

$$H > 1.4*\left(\frac{S_1 + S_2}{2}\right) \quad (5)$$

$$H < 4*\left(\frac{S_1 + S_2}{2}\right) \quad (6)$$

$$E_1 > E_3 \quad (7)$$

$$E_3 > E_5 \quad (8)$$

DECIDING LOOKBACK PERIOD AND IDENTIFYING PIVOTS

For the detection of IHS, we have taken subsample or lookback from t to t+l+d-1 (and we fit kernel regressions), where t varies from 1 to T-l-d+1, and l and d are fixed parameters. We have set l = 94 and d = 6; hence each window consists of 100 trading minutes.

The motivation for taking such lookback is to narrow our focus to IHS patterns that are completed within the span of l+d trading minutes. If we try to look at a larger dataset in one go, various additional patterns will emerge, and it would be impossible to distinguish signal from noise.

The value of d = 6 is chosen since we require at least 6 candlesticks before and after a particular candlestick to detect an extremum and it also accounts for the lag that occurs between pattern detection and pattern completion, hence d is the number of treading minutes that must pass before the pattern is detected.

The lag d ensures that we have detected the breakout without using any forward bias.

We have also made sure that we aren't considering very old pivots through a rolling window concept by deleting the pivots before our decided lookback.

BUYING & SELLING CRITERIA

After the pattern has been identified, we enter a long position when the price of the stock crosses the neckline. This is the breakout point. The distance of the bottom of the head from the neckline is said to be the head height.

The stop-loss is triggered when the price falls down more than the head height after the breakout point. If the current price rises more than the head height above the breakout point, a trailing stop-loss is initiated.

CODE DESIGN

```python
# DETECTION OF PIVOTS
def pivotId(px, candle, num_before, num_after):
    if candle-num_before < 0 or candle+num_after >= len(px):
        return 0

    pivotIdLow=1
    pivotIdHigh=1
    for i in range(candle-num_before, candle+num_after):
        if(px[candle]>px[i]):
            pivotIdLow=0
        if(px[candle]<px[i]):
            pivotIdHigh=0
    if pivotIdLow:
        return 1
    elif pivotIdHigh:
        return 2
    else:
        return 0
```

Function to detect Maxima (and Minima) : Num_before = Num_After = N (Value of N taken to be _____).

```python
def logic(pivot,points, context, security, lookback1):
    count = 0  # TO FIND NO. OF PIVOTS
    flag =0  # TO CHECK ALTERNATING PIVOTS

    # STORING COORDINATES OF PIVOTS
    maxx = []
    maxy = []
    minx = []
    miny = []
```

Beginning of 'logic' function, which is the core function of the entire strategy, used for identifying relevant IHS patterns. X-coordinates are dates and Y-coordinates are prices of the maximas and minimas of the IHS pattern.

```python
    # DETECTION OF INVERSE HEAD & SHOULDERS
    for i in range(len(pivot)-1, -1, -1):
        if (pivot[i][1] == 1 and flag == 0):

            # CHECKING BEARISH TREND
            m = pivot[i][0] - (lookback1-1)
            if (m >=0):
                if (context.exponentialavg[security][m] - pivot[i][2] < 0 ):
                    return 0,0,0,0

            minx.append(pivot[i][0] - context.iterator[security])
            miny.append(pivot[i][2])
            count +=1
            flag = 1
```

Continuing the 'logic' function here, mainly identifying the bearish trend in this snippet of code.

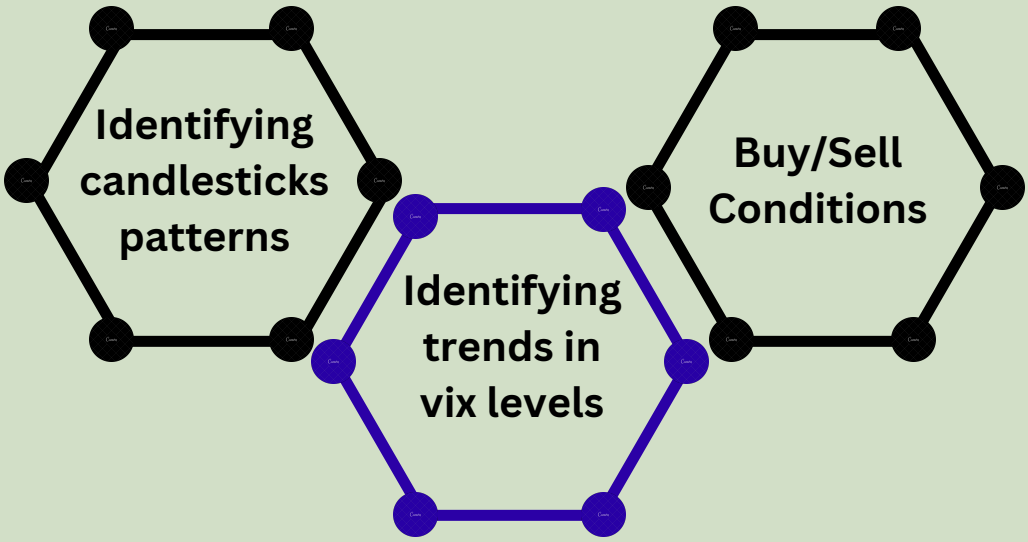
```
        # FIVE ALTERNATING PIVOTS FOR IHS
        if (count ==5):
            if (miny[1] < miny[0] and miny[1] < miny[2]):
                avgmin = (miny[0] + miny[2])/2
                avgmax = (maxy[0] + maxy[1])/2
                slope = (maxy[0] - maxy[1]) / (maxx[0] - maxx[1])
                c = maxy[0] - (slope * maxx[0])
                y = (slope * (lookback1-1)) + c     # BREAKOUT VALUE
                y2 = (slope * minx[1]) + c - miny[1]    # HEAD HEIGHT
                y3 = (slope * minx[0]) + c - miny[0]    # SHOULDER HEIGHT
                y4= (slope*minx[2]) +c - miny[2]
                shoavg = (y3 + y4)/2
                if (y2 < 1.4 * shoavg and y2 > 4*shoavg):   # HEAD TO SHOULDER RELATION
                    return 0,0,0,0

                return y, y2, y3, slope
            else:
                return 0,0,0,0
```

Constraints are first placed on the X and Y coordinates of the pivots and then a Neck Line is constructed(slope) to determine the buy signal.(When price crosses the neck line)

```
        elif (pivot[i][1] == 2 and flag == 1):
            m = pivot[i][0] - (lookback1-1)
            if (m >=0):
                if (context.exponentialavg[security][m] - pivot[i][2] < 0 ):
                    return 0,0,0,0

            maxx.append(pivot[i][0] - context.iterator[security])
            maxy.append(pivot[i][2])
            count +=1
            flag =0

        elif (pivot[i][1] == 2 or pivot[i][1] == 1):
            return 0,0,0,0

    return 0,0,0,0
```

Older pivots are removed and IHS patterns appearing in bullish trends are ignored for the purpose of our hypothesis.

VIX-Nifty Futures : Strategy



Identifying Candlesticks

- Coding Candlestick Patterns :
- Marubozu,Engulfing Patterns,Piercing Patternss etc

Identifying Trends in Vix Levels

- Candlestick patterns are used to detect bullish(or bearish) trends in the INDIA-VIX levels.

Taking an opposite position in Nifty

- If a bullish trend is detected in India Vix, then a bearish position is taken in the Nifty Futures and vice versa.

```python
from blueshift.finance import commission, slippage
from blueshift.api import(  symbol,
                            order_target_percent,
                            order_target,
                            order_target_value,
                            set_commission,
                            set_slippage,
                            schedule_function,
                            date_rules,
                            time_rules,
                            order,
                         )
import numpy as np
from scipy.signal import savgol_filter
from blueshift.library.technicals.indicators import bollinger_band, ema

def initialize(context):
    """
        A function to define things to do at the start of the strategy
    """
    # universe selection
    context.securities = [symbol('INDIAVIX'), symbol('NIFTY-I')]

    # define strategy parameters
    context.params = {'indicator_lookback':375,
                      'indicator_freq':'1m',
                      'buy_signal_threshold':0.5,
                      'sell_signal_threshold':-0.5,
                      'SMA_period_short':15,
                      'SMA_period_long':60,
                      'BBands_period':300,
                      'trade_freq':10,
                      'leverage':1}

    # variables to track signals and target portfolio
    context.signals = dict((security,0) for security in context.securities)
    context.target_position = dict((security,0) for security in context.securities)
    context.k = 0
    context.qt = 0

    context.exponentialavg = dict((security,[]) for security in context.securities)
    context.iterator = dict((security, 0) for security in context.securities)

    # set trading cost and slippage to zero
    set_commission(commission.PerShare(cost=0.0, min_trade_cost=0.0))
    set_slippage(slippage.FixedSlippage(0.00))

    freq = int(context.params['trade_freq'])
    schedule_function(run_strategy, date_rules.every_day(),
                      time_rules.every_nth_minute(freq))

    schedule_function(stop_trading, date_rules.every_day(),
                      time_rules.market_close(minutes=30))

    context.trade = True
```

```
def before_trading_start(context, data):
    context.trade = True

def stop_trading(context, data):
    context.trade = False

def run_strategy(context, data):
    """
        A function to define core strategy steps
    """
    if not context.trade:
        return

    generate_signals(context, data)
    generate_target_position(context, data)

def logic(pivot, points, context, security):
    count = 0
    flag = 0
    maxx = []
    maxy = []
    minx = []
    miny = []

    for i in range(len(pivot)-1, -1, -1):
        if (pivot[i] == 1 and flag == 0):
            m = i + context.iterator[security] - 375
            if (m >0):
                if (context.exponentialavg[m] - points[i] > 0.001 * points[i] ):
                    return 0,0
            minx.append(i)
            miny.append(points[i])
            count += 1
            flag = 1
            if (count ==5):
                if (miny[1] < miny[0] and miny[1] < miny[2]):
                    avgmin = (miny[0] + miny[2])/2
                    avgmax = (maxy[0] + maxy[1])/2
                    if (abs(miny[0] - avgmin) <= (0.04 * avgmin) and abs(miny[2] - avgmin) <= (0.04 *
avgmin) and abs(maxy[0] - avgmax) <= (0.04 * avgmax) and abs(maxy[1] - avgmax) <= (0.04 * avgmax)):
                        slope = (maxy[0] - maxy[1]) / (maxx[0] - maxx[1])
                        c = maxy[0] - (slope * maxx[0])
                        y = (slope * 375) + c
                        y2 = (slope * minx[1]) + c - miny[1]
                        return y, y2
        elif (pivot[i] == 2 and flag == 1):
            m = i + context.iterator[security] - 375
            if (m >0):
                if (context.exponentialavg[security][m] - points[i] > 0.01 * points[i] ):
                    return 0,0
            maxx.append(i)
            maxy.append(points[i])
            count +=1
            flag =0
        elif (pivot[i] == 2 or pivot[i] == 1):
            return 0,0

    return 0,0
```

```
def vix_val(px, params, context, security):

    pivot = []
    month_diff = len(px) // 30      #df shape is (249, 8)
    if month_diff == 0:
        month_diff = 1
    smooth = int(2*month_diff + 3)
    points = savgol_filter(px, smooth , 3)
    ind = ema(px , 150)
    context.exponentialavg[security].append(ind)
    context.iterator[security] = context.iterator[security] + 1
    for i in range(6, len(points)-6):
        pividlow=1
        pividhigh=1
        for j in range(i-6, i+7):
            pividlow=0
        if(points[i]<points[j]):
            pividhigh=0
        if pividlow and pividhigh:
            pivot.append(3)
        elif pividlow:
            pivot.append(1)
        elif pividhigh:
            pivot.append(2)
        else:
            pivot.append(0)
    (y, y2) = logic(pivot, points, context, security)
    y2 = y + y2
    if ((y - px[-1]) < 0 and (px[-1] - y) < 0.005 * px[-1]):
        return 1
    elif ((px[-1] - y2) < 0.005 * px[-1] or (y2 - px[-1]) < 0.005 * px[-1]):
        return -1
    else:
        return 0

def vix_bbands(px, params):
    upper, mid, lower = bollinger_band(px,params['BBands_period'])
    if upper - lower == 0:
        return 0

    ind2 = ema(px, params['SMA_period_short'])
    ind3 = ema(px, params['SMA_period_long'])
    last_px = px[-1]
    dist_to_upper = 100*(upper - last_px)/(upper - lower)

    if dist_to_upper > 95:
        return -1
    elif dist_to_upper < 5:
        return 1
    elif dist_to_upper > 40 and dist_to_upper < 60 and ind2-ind3 < 0:
        return -1
    elif dist_to_upper > 40 and dist_to_upper < 60 and ind2-ind3 > 0:
        return 1
    else:
        return 0

def vix_bullish(px, params):
    close = px.close.values[-1]
    prev_close = px.close.values[-2]
    open = px.open.values[-1]
    prev_open = px.open.values[-2]

    if ((close >= prev_open > prev_close) and (close > open) and (prev_close >= open) and (close - open
> prev_open - prev_close)):
        return 1
    if ((open >= prev_close > prev_open) and (open > close) and (prev_open >= close) and (open - close
> prev_close - prev_open)):
        return -1
    return 0
```

```python
def vix_bullish(px, params):
    close = px.close.values[-1]
    prev_close = px.close.values[-2]
    open = px.open.values[-1]
    prev_open = px.open.values[-2]

    if ((close >= prev_open > prev_close) and (close > open) and (prev_close >= open) and (close - open
> prev_open - prev_close)):
        return 1
    if ((open >= prev_close > prev_open) and (open > close) and (prev_open >= close) and (open - close
> prev_close - prev_open)):
        return -1
    return 0

def vix_harami(px, params):
    close = px.close.values[-1]
    prev_close = px.close.values[-2]
    open = px.open.values[-1]
    prev_open = px.open.values[-2]

    if ((prev_open > prev_close) and (prev_close <= open < close <= prev_open) and ((close - open) <
(prev_open - prev_close))):
        return 1
    if ((prev_close > prev_open) and (prev_open <= close < open <= prev_close) and ((open - close) <
(prev_close - prev_open))):
        return -1
    return 0

def vix_marubozu(px, params):
    close = px.close.values
    high = px.high.values
    low = px.low.values
    open = px.open.values

    if(len(close)<1):
        return 0

    if (close[-1] > open[-1] and high[-1] == close[-1] and low[-1] == open[-1]):
        return 1
    elif (close[-1] < open[-1] and high[-1] == open[-1] and low[-1] == close[-1]):
        return -1
    else:
        return 0

def generate_target_position(context, data):
    """
        A function to define target portfolio
    """
    num_secs = len(context.securities) - 1

    security = context.securities[1]
    overall = context.portfolio.portfolio_value
    curr_price = data.current(context.securities[1], 'close')
    idc = context.account.net_exposure
    na_cash = overall - idc

    # Next 2 lines are the method to get_curr_time
    px = data.history(context.securities[0], ['open', 'high'], 2, '1m')
    curr_time = px.index[-1]

    if (context.k==0) and (context.signals[security] > context.params['buy_signal_threshold']):
        context.qt = ((overall // curr_price) // 50)*50
        context.k = 1
        order_target(security, context.qt)


    elif (context.k==0) and (context.signals[security] < context.params['sell_signal_threshold']):
        context.qt = -1*(((overall // curr_price) // 50)*50)
        context.k = -1
        order_target(security, context.qt)


    elif (context.k>0) and (context.signals[security] < context.params['sell_signal_threshold']):
        context.qt = 0
        context.k = 0
        order_target(security, context.qt)


    elif (context.k<0) and (context.signals[security] > context.params['buy_signal_threshold']):
        context.qt = 0
        context.k = 0
        order_target(security, context.qt)
```

```python
def generate_signals(context, data):
    """
        A function to define define the signal generation
    """
    try:
        # price_data = data.history(context.securities[1:], ['open','high','low','close'],
        #     context.params['indicator_lookback'], context.params['indicator_freq'])
        price_data_vix = data.history(context.securities[0], ['open','high','low','close'],
            context.params['indicator_lookback'], context.params['indicator_freq'])
    except:
        return

    for security in context.securities[1:]:
        context.signals[security] = signal_function(context, price_data_vix, context.params, security)

def signal_function(context, px_vix, params, security):
    """
        The main trading logic goes here, called by generate_signals above
    """
    vix = px_vix

    # Specifying Stoploss Condition
    curr_time = vix.index[-1].time()
    hour = curr_time.hour
    minute = curr_time.minute
    stoploss = 0.18

    if (hour < 10) or (hour == 10 and minute <= 30):
        stoploss = 0.08
    elif ((hour == 10 and minute > 30) or hour > 10) and (hour < 13 or (hour == 13 and minute <= 30)):
        stoploss = 0.06
    elif (hour == 14 and minute > 30) or (hour > 13):
        stoploss = 0.05

    # Checking
    positions = context.portfolio.positions

    quant = 0
    pricee = 0
    pnl = 0
    position = -1
    for key in positions.keys():
        if positions[key].position_side == 0: # LONG
            quant += positions[key].buy_quantity
            pricee = positions[key].buy_price
            pnl += positions[key].unrealized_pnl
            position = 0

        if positions[key].position_side == 1: # SHORT
            quant += positions[key].sell_quantity
            pricee = positions[key].sell_price
            pnl += positions[key].unrealized_pnl
            position = 1


    # Checking for Stoploss
    if (quant != 0 and pricee != 0) and (pnl/(quant*pricee)) <= (-1*stoploss):
        if position == 0: # LONG
            return -1
        if position == 1: # SHORT
            return 1

    take_profit = 0.10
    # Applying Take Profit
    if (quant != 0 and pricee != 0) and (pnl/(quant*pricee)) >= take_profit:
        if position == 0: # LONG
            return -1
        if position == 1: # SHORT
            return 1

    return -vix_bullish(vix, params)
```

WHAT IS VIX ?

- VIX (CBOE Volatility Index) is a real-time index that represents the market's expectations for relative strength of near-term price changes of the NIFTY Index.
- Investors use the VIX to measure the level of risk, fear, or stress in the market when making investment decisions
- Generally when the VIX rises, the stocks fall and vice-versa. The INDIAVIX displays a negative correlation with the NIFTY index as well.
- **NOTE** - VIX is about expected volatility and not about actual volatility

A Comparison between Charts of INDIAVIX and NIFTY

CODE DESIGN

```
def vix_engulfing(px, params):
    # px contains the OHLC data of INDIAVIX
    # params consists the extra parameters initialized in the start of the code
    close= px.close.values[-1]
    prev_close= px.close.values[-2]
    open= px.open.values[-1]
    prev_open= px.open.values[-2]

    # We are initializing signal_rcvd with '0' as if no trade pattern is observed
      then we wouldn't send any signal (0)
    signal_rcvd= 0

    # When we observe a bullish engulfing pattern on INDIAVIX it returns a
      buy_signal (1)
    if ((close >= prev_open > prev_close) and (close > open) and (prev_close >=
        open) and (close - open > prev_open - prev_close)):
        signal_rcvd= 1

    # When we observe a bearish engulfing pattern on INDIAVIX it returns
      sell_signal (-1)
    if ((open >= prev_close > prev_open) and (open > close) and (prev_open >=
        close) and (open - close > prev_close - prev_open)):
        signal_rcvd= -1

    # It returns the opposite signal on the {security} we are buying to the signal
      it generated.
    # Basically if INDIAVIX gives a buy_signal we need to sell the security and
      vice-versa

    return -(signal_rcvd)
```

Several other candlestick patterns may be chosen in place of bullish and bearish engulfing patterns to detect trends in vix-levels

The CBOE Volatility Index (VIX) is a real-time index that represents the market's expectations for the relative strength of near-term price changes of the NIfty-50 Index. Investors use the VIX to measure the level of risk, fear, or stress in the market when making investment decisions.The VIX generally rises when stocks fall, and declines when stocks rise.

VIX calculation involves inferring its value as implied by options prices.

India VIX :: computation methodology

India VIX uses the computation methodology of CBOE, with suitable amendments to adapt to the NIFTY options order book.

The formula used in the India VIX calculation is:

$$\sigma^2 = \frac{2}{T}\sum \frac{\Delta K_i}{K_i^2} e^{RT} Q(K_i) - \frac{1}{T}\left[\frac{F}{K_0} - 1\right]^2$$

where:

σ = India VIX/100 ™ - India VIX= σx 100

T =Time to expiration

K_i = Strike price of i th out-of-the-money option; a call if $K_i > F$ and a put if $K_i < F$

ΔK_i = Interval between strike prices- half the distance between the strike on either side of K_i:

$$\Delta K_i = \frac{K_{i+1} - K_{i-1}}{2}$$

R = Risk-free interest rate to expiration

$Q(K_i)$ = Midpoint of the bid ask quote for each option contract with strike K_i

F = Forward index taken as the latest available price of NIFTY future contract of corresponding expiry

Ko = First strike below the forward index level, F.

Some of these symbols are further explained below.

STOCK SELECTION & ROBUSTNESS CHECK

VIX indicates the market sentiment as a whole. Hence we work on NIFTY Futures as the target for the strategy.

BUYING & SELLING CRITERIA

We enter long or short positions based on engulfing trends in VIX. Defining candlesticks: White candlestick- Open > Close; Black Candlestick- Open < Close

Bullish engulfing: If a white candlestick closes higher than the previous candlestick's opening after opening lower than the previous close, given that the previous close was less than its open (i.e black candlestick), we say a bullish engulfing pattern has been detected. The amplitude of the candlestick (magnitude of open - close) is strictly greater than the one before. In other words, the body of the white candlestick completely engulfs the body of the preceding black candlestick.
Identification:

- Previous close < Previous open
- Close > Previous Open
- Open < Previous Close

Engulfing condition (magnitude of open - close strictly greater than the one before) is also satisfied by above three conditions.

Bearish Engulfing: As the name suggests, bearish engulfing is the antipode to bullish engulfing. A black candlestick that completely engulfs the preceding white candlestick indicates a bearish engulfing. Identification:

- Previous close > Previous open
- Close < Previous Open
- Open > Previous Close

Engulfing condition (magnitude of open - close strictly greater than the one before) is again satisfied by above three conditions.

Bullish engulfing indicates an uptrend in VIX, hence we enter a short position on NIFTY Futures. Since a bearish engulfing indicates downtrend, we enter a long position.

Transaction	Cost
Brokerage	Rs 20 or 0.03% per order, whichever is lower
STT/CTT	0.025% on sell side
Transaction Charges	0.00345% of total turnover
GST	18%
SEBI Charges	Rs10/Crore + GST
Stamp Charges	0.003% or Rs300/Crore on buy side

Transaction	Cost
Brokerage	Rs 20 or 0.03% per order, whichever is lower
STT/CTT	0.01% on sell side
Transaction Charges	0.002% of total turnover
GST	18%
SEBI Charges	Rs10/Crore + GST
Stamp Charges	0.002% or Rs200/Crore on buy side

Assumptions on Brokerage and Transation Costs

Type of Stock	Equity Buy	Equity Sell
Price	500000	500000
Quantity	20	20
Brokerage	0	0
STT	10000	10000
Transaction Charges	345	345
GST	62.1	62.1
SEBI	11.8	11.8
Stamp Charge	1500	0
Net Charge	11918.9	10418.9
% Charge	0.119189	0.104189

Type of Stock	Future Buy	Future Sell
Price	500000	500000
Quantity	20	20
Brokerage	20	20
STT	0	0
Transaction Charges	200	200
GST	39.6	39.6
SEBI	11.8	11.8
Stamp Charge	200	0
Net Charge	471.4	271.4
% Charge	0.004714	0.002714

*The charges mentioned above are calculated considering a portfolio value of 1 Cr.

Slippage:

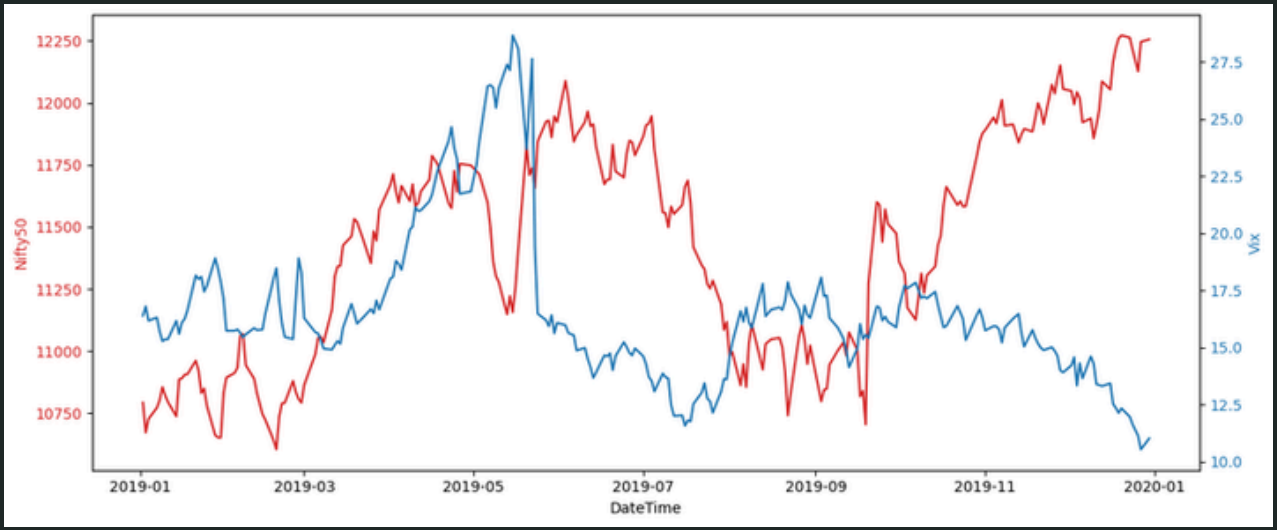
The average slippage of the transactions is assumed to be **0.00046%** based on general transactions.

APPENDIX

- Algorithm for Head and Shoulder
- Searching for Head and Shoulders Bottom Patterns under Directional Changes
- Predictive power of Head and Shoulders price pattern
- Search for patterns in compressed time series
- Feature based control chart pattern recognition using CART Analysis
- Exact discovery of timeseries motifs
- Sector Rotation
- Hypothesis testing in trading

HYPOTHESIS TESTING

We have two time series, one for Nifty Futures and another one for INDIAVIX. Our hypothesis is that they move opposite to each other. We've plotted the data for *Nifty50 Index and INDIAVIX* on a double scale graph:



A) ADF TEST FOR STATIONARITY -

- The ADF Test is one of the most popular statistical tests to check whether the time series is *stationary* or not.
- **Null Hypothesis:** If failed to be rejected, it suggests the time series is not stationarity.
- **Alternative Hypothesis:** The null hypothesis is rejected, it suggests the time series is stationary.

CODE SNIPPET

```
# ADF TEST
from statsmodels.tsa.stattools import adfuller
def adf_test(df):
    result = adfuller(df.values)
    print('ADF Statistics: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

TEST RESULTS

```
ADF Statistics: -1.502859
p-value: 0.532097
Critical values:
        1%: -3.458
        5%: -2.874
        10%: -2.573


ADF Statistics: -2.294671
p-value: 0.173692
Critical values:
        1%: -3.459
        5%: -2.874
        10%: -2.573
```

- Given are the results for the ADF Test of Nifty Futures and Vix respectively.
- As the p-values are greater than 0.05, this shows that **both** the time series are **not stationary.**

B) DIFFERENCE METHOD -

- Differencing is a popular and widely used data transform for making time series data stationary.

CODE SNIPPET

```
# DIFFERENCE METHOD
df_train_transformed = data.diff().dropna()
```

Now, we will check the our transformed dataset for Stationarity using ADF Test again on this new dataset.

TEST RESULTS

```
ADF Statistics: -14.471608
p-value: 0.000000
Critical values:
        1%: -3.458
        5%: -2.874
        10%: -2.573


ADF Statistics: -4.881453
p-value: 0.000038
Critical values:
        1%: -3.459
        5%: -2.874
        10%: -2.573
```

- As the p-values of both the time series are very much less than 0.05, both the time series have **now become stationary**.

C) DURBIN WATSON TEST -

- The Durbin Watson Test measures autocorrelation in residuals from regression analysis.
- For a large dataset, d is approximately equal to 2*(1-p), where p is the sample autocorrection of the residuals, and **d = 2, therefore, indicates no autocorrelation**.

```
# DURBIN WATSON METHOD
from statsmodels.stats.stattools import
durbin_watson

out = durbin_watson(results.resid)
for col, val in zip(data.columns, out):
    print(col, ':', round(val, 2))
```

TEST RESULTS

```
0 : 2.01
1 : 2.0
```

Here, for both of the time series (Nifty as well as Vix), the d comes out to be equivalent to 2. Hence **they are not autocorrelated**, respectively.

D) GRANGERS CAUSALITY TEST -

- The Granger causality test is a statistical hypothesis test for determining whether one time series is useful in forecasting another.

CODE SNIPPET

```
# GRANGERS CAUSALITY TEST
maxlag = 15
test = 'ssr_chi2test'

def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):
    df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
    for c in df.columns:
        for r in df.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)]
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df.loc[r, c] = min_p_value

    return df
```