Strategy Logic and Explanation

The stock I chose is BAJFINANCE.

Strategy has three parts:

1) Recognizing double top and bottom and generating signals
2) Coding Bollinger Bands and generating signals
3) Finding the common signals in 1 and 2

Finding the Local Extremas using scipy

In [3]:
```
arr = data.Close.to_numpy()
bottoms = (argrelextrema(arr, np.less, order=3)[0])
tops = (argrelextrema(arr, np.greater, order=3)[0])
ext_index = np.hstack((bottoms, tops))
ext_index = np.sort(ext_index)
df = data.Close[ext_index]
df = pd.DataFrame(df)
print(df)
```

This cell is used to find the local points of minima and maxima in the Closing prices of stocks, for which I have used the agrelextrema function from scipy. ext_index stores the index number of all such extremas. df is a dataframe that stores the closing prices of only the points that form extrema.

Finding where the Double bottom and top is formed

In [4]:
```
def find_pattern(extrema):

    double_tops = []
    double_bottoms = []

    # period range is 5 units
    for i in range(5, len(extrema)):
        period = extrema.iloc[i-5:i]

        if period.index[-1] - period.index[0] > 50:
            continue

        a, b, c, d, e = period.iloc[0:5].Close
#         x1 = abs(b - d)/min(b,d)
#         x2 = (c-b)/b
        if a<b and a<d and c<b and c<d and e<b and e<d and b>d:
               double_tops.append((period.index[0], period.index[1], period.index[2], period.index[-2], period.index[-1]))

        if a>b and a>d and c>b and c>d and e>b and e>d and b<d:
                double_bottoms.append((period.index[0], period.index[1], period.index[2], period.index[-2], period.index[-1]))

    return double_tops, double_bottoms
```

This cell serves the purpose of identifying the points of extrema that satisfy the condition of the pattern. I iterate over 5 points at a point and check if they satisfy the required condition of either double bottom or top. If so, I append the index of the fifth point to generate buy and sell breakout points.
I used

a) Technical Analysis for Algorithmic Pattern Recognition by Prodromos E. Tsinaslanidis and Achilleas D. Zapranis
b) https://www.youtube.com/watch?v=Db-fbXMJvts

as reference for this.

Bollinger Bands:

In [6]:
```
def Bollinger(data, n1 = 20, multiplier = 2):
    MA = data.Close.rolling(window=n1).mean()
    SD = data.Close.rolling(window=n1).std()
    data['bbmb'] = MA
    data['bbub'] = MA + (multiplier * SD)
    data['bblb'] = MA - (multiplier * SD)
```

This is my code for obtaining Bollinger bands which uses two parameters n1 and multiplier respectively for rolling window of moving average and factor for upper and lower bands.
These parameters are usually taken to be 20, 2 but we vary them ahead to obtain best returns.

In [8]:
```
def Trade_BOLLINGER(data):
    buy, sell = [], []
    flag = 0
    for i in range(2, len(data)):
        if flag == 1 and data.Close.iloc[i] > data.bbub.iloc[i] and data.Close.iloc[i - 1] < data.bbub.iloc[i - 1]:
            sell.append(i)
            flag = 0
        elif flag == 0 and data.Close.iloc[i] < data.bblb.iloc[i] and data.Close.iloc[i - 1] > data.bblb.iloc[i - 1]:
            buy.append(i)
            flag = 1
    if len(sell) != 0 and len(buy) != 0:
        if sell[0] < buy[0]:
            sell.pop(0)
        if buy[-1] > sell[-1]:
            buy.pop(-1)
    return (buy, sell)
```

This cell checks if the closing price crosses the upper or lower bands of Bollinger and generates sell and buy signals respectively.

Function to find where both BB and Chart Pattern generate same signal

In [11]:
```
def Trade_Signal(data, dt, db):
    buy, sell = [], []
    for i in range(len(dt)):
        if data.Close[dt[i][1]] > data.bbub[dt[i][1]] and data.Close[dt[i][3]] < data.bbub[dt[i][3]]:
            sell.append(dt[i])
    for i in range(len(db)):
        if data.Close[db[i][1]] < data.bblb[db[i][1]] and data.Close[db[i][3]] > data.bblb[db[i][3]]:
            buy.append(db[i])
    return (buy, sell)
```

This checks if the points where chart pattern is formed satisfies the condition of Bollinger bands signals or not.
Reference:
https://www.tradingsetupsreview.com/double-top-and-bottom-trading-with-bollinger-bands/

Calculating the returns when we trade on the final signals generated

In [12]:
```
def get_pattern_return(data, dtdb, x, capital):
    entry_price = data.Close[dtdb[0][-1]]
    entry_i = dtdb[0][-1]
    stop_price = data.Close[dtdb[0][1]]

    if x:
        tp_price = entry_price + (data.Close[dtdb[0][0]] - data.Close[dtdb[0][1]])
    else:
        tp_price = entry_price - (data.Close[dtdb[0][1]] - data.Close[dtdb[0][0] ])

    exit_price = -1
    for i in range(dtdb[0][-1] - dtdb[0][0]):
        if entry_i + i >= len(data):
            return np.nan

        exit_price = data.Close[entry_i + i]
        if  x and (exit_price > tp_price or exit_price < stop_price):
            break

        if not x and (exit_price < tp_price or exit_price > stop_price):
            break

    if x: # long
        return capital *(1 + (exit_price - entry_price) / entry_price)
    else: # short
        return capital *(1 - (exit_price - entry_price) / entry_price)
```

The above cell is our final trade executing cell.
dtdb is a list of tuples which contain index of five points that form a chart pattern. Hence, dtdb[0] refers to the first tuple in the list. The first tuple is changed everytime from the cell where we call the function.
Entry price is the last point of our pattern, exit price is the closing pric of consequent cells and stop loss is defined to be the highest/lowest price of our pattern.
Take profit price is also defined at a certain distance from the entry point.
We iterate over a window that is same as the window length of our pattern and check if we can obtain our desired tp_price or our we exit at stop loss. If none is satisfied, we exit at the closing price of the last point in the defined window after entry point.

Reference: https://www.youtube.com/watch?v=6iFqjd5BOHw

```
returns = []
Bollinger(data,19,1.5)
buy, sell = Trade_Signal(data, dt, db)
capital = 50
for i in range(len(buy)):
    capital = get_pattern_return(data, buy[i:], 1, capital)

capital1 = 50
for i in range(len(sell)):
     capital1 = get_pattern_return(data, sell[i:], 0, capital)

returns.append(capital + capital1 - 100)

print("Maximum Returns by trading on signals generated by BB and Double Bottom/Top Chart Pattern =", max(returns))
Roll_Max = data['Adj Close'].cummax()
Daily_Drawdown = data['Adj Close']/Roll_Max - 1.0
Max_Daily_Drawdown = Daily_Drawdown.min()
print("MAX Drawdown =",Max_Daily_Drawdown)
daily_return = data['Adj Close'].pct_change().dropna()
sharpe = ((daily_return.mean())/daily_return.std())*np.sqrt(252)
print("Sharpe Ratio =", sharpe)
```

This is the last cell in our code that looks to optimise the BB parameters for BAJFINANCE in the given time frame. I have allotted equal amount of money initially for trading in double bottom and top respectively.

The combination of two different indicators reduces the number of trading opportunities and we might miss out on some. The code can be improved further in the conditions of stop loss or take profit price to improve the strategy. We could trade together with one capital on both double bottom and top, which my code fails to incorporate now.

Metrics Results:

1) Cumulative returns: 7.072976332537834
2) Sharpe ratio: 1.2599649469280538
3) Max Drawdown: -0.6243781276628886