ALGO-TRADING-BOT

Introduction:-

The objective of our project is to implement an Algo trading bot. The advantage of algo trading over manual trading are related to speed, accuracy and reduced costs. Trading with algorithms has the advantage of scanning and executing on multiple indicators at a speed that no human could do. One of the biggest advantages of the algo trading is the ability to remove human emotion from the markets, as trades are constrained within a set of predefined criteria.

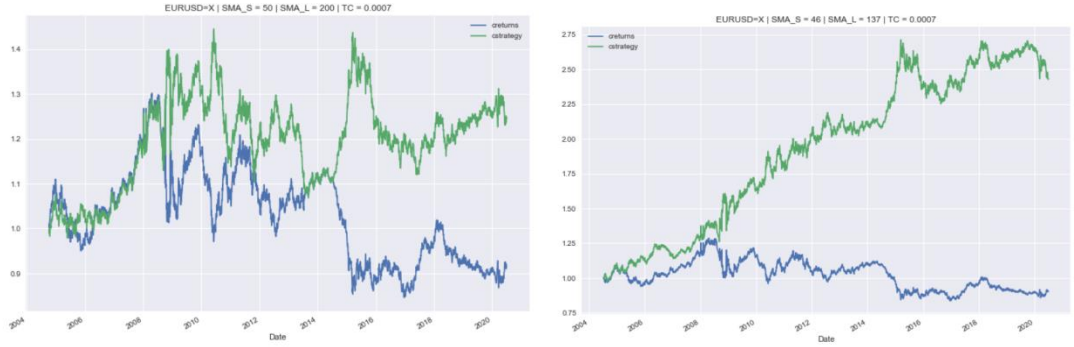
Implementation details:-

In this project, we made many popular indicators such as RSI, MACD, Bollinger Bands, Stochastic Osc, Pivot points etc. using the python and then we back tested them on the historical data of Nifty 50.

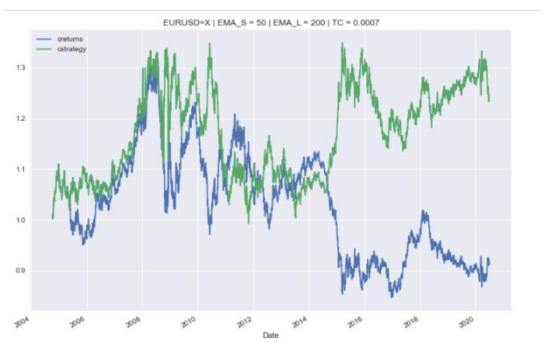
1. SMA :- In this file we implement a "Simple Moving Average Crossover" strategy. In this strategy we use two simple moving averages two find the entry and exit point for trading. Whenever the short SMA is upper then the longer period SMA, then it is a signed to go long and when longer period SMA is greater then we go short.
    - To implement this strategy, we made a class 'SMABacktester'. Firstly we initialize the class by giving the symbol of the asset in which we wanted to do trading, two periods one for short SMA and one for longer period SMA and then we give starting and ending date and the approximate trading cost.
    - In **get_data()** method, we take data from forex_pairs csv file and make a dataframe of the asset with date in which we wanted to do trade. Then we calculate logarithmic returns for each consecutive period for buy and hold(that is benchmark for any strategy). Also we make two columns for short period sma and long period sma. We calculate the sma by taking last n periods prices average.
    - If we wanted to change the period for sma_s and sma_l, to do this we make a **set_parameters()** method.
    - Then we make a **test_strategy()** method.
        - In this method firstly we define our strategy by numpy method "where". Whenever the position is 1 we go long, position is 0 we square off our positions and position is -1 then we go short.
        - To calculate the returns of this strategy for each consecutive periods, we make a strategy columns and we multiply the returns of buy and hold returns with our positions that we take in this period.
        - We also make a column trades to find out when we change our position i.e. when we take our trades, to find out it we find the absolute difference of each consecutive period position, whenever the position difference is two it means we changed our position i.e. take a trade.

- To find the strategy returns after the trading costs we subtract the multiply of trading cost and trade(0 or 2).
- Then we make two columns for cumulative returns for buy and hold strategy and sma crossover strategy.
- This complete dataframe now our results.
- Finally we find the absolute performance and outperformance/under performance of our strategy.

- In **plot_results()** method we plot line plots for buy and hold returns and our strategy returns to make a clear picture. To plot them it is necessary to find firstly creturns and cstrategy i.e. our results.
- To change the sma_s and sma_l in our test_strategy() method we make another method **update_and_run().** The output of this method is the negative of our results. We did it to find out the maximum of our result by finding the minimum of update_and_run.
- To find the best combination of shorter period and longer period moving average we make a method **optimize_parameters().** In this method we find global minimum of update_and_run, by finding this we find the maximum returns of our strategy. In this method we use brute method of scipy.optimize library.

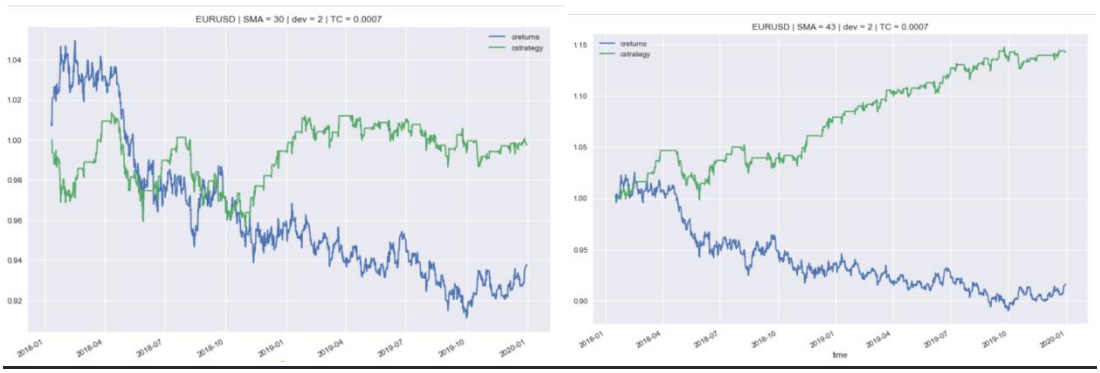


2. EMA:- This is exactly same as SMA. The only difference in SMA and EMA that in EMA we give more weighatage to recent prices over previous prices and in SMA we give same weighatage to all the prices in that period.
   - Because we give more weighatage to recent price in EMA, so EMA is more sensitive to price changes then SMA. So, many traders prefer EMA then SMA.
   - To find the EMA we use **ewm()** method. We passed periods for which we wanted to find EMA as span and min_periods attributes.
   - Rest all same in EMA and SMA.

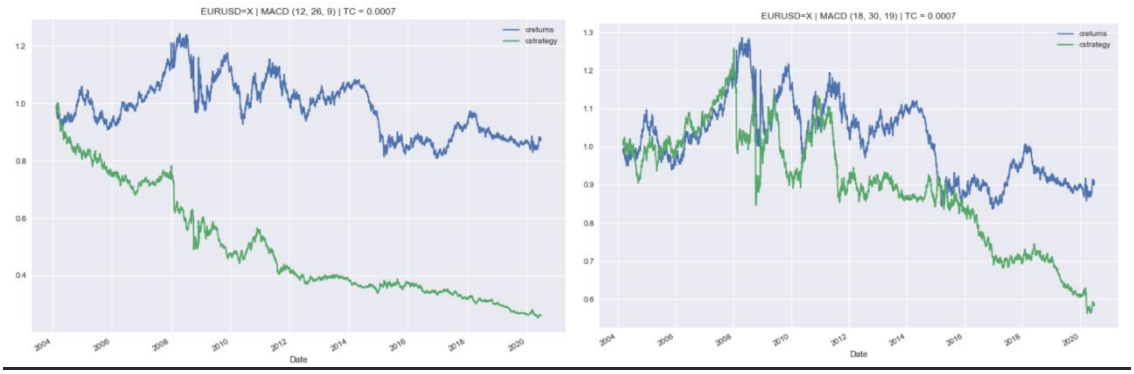


3. <u>SMA + EMA:-</u> In this strategy we use sma and ema of same time period to find out the positions. Whenever the EMA crosses SMA we change our position. When EMA is higher then the SMA, we go long and when EMA is shorter then the SMA we go short, because EMA is more sensitive to the recent prices then the older prices so it will react fast. The rest of the format is same as SMA and EMA strategy.

4. <u>Bollinger Bands:-</u> Bollinger band indicator / strategy is based on one phenomenon that "Ultimately the prices of any asset comes back to its mean no matter how much fluctuation it has."
   - In this Indicator we have a channel consisting of three line, one is the mean of prices that is SMA or EMA of a specified period and other two outer lines are the standard deviation of price from its mean.
   - According to our previous indicator SMA we make a class called BBBacktester.
   - In get_data() method we add three columns for the three lines of Bollinger bands channel.
   - Our trading strategy using this indicator is that when the price is lower than its lower channel(sma – standard deviation*deviation), we go long and when ever the price is higher than its upper channel(sma + standard deviation*deviation), we go short. And when the price crosses the sma line we go neutral i.e. we square off our position.
   - The rest BBBacktester class is same as the SMA class.



5. SMA + Bollinger Bands:- Till now we made separate two indicator file for SMA and Bollinger Band. In this file we make a combination of both two. For this we first import SMA and BollingerBands python file.
   - In this strategy we go long when both SMA and Bollinger Bands strategy give long signal and go short when both give short signal. If both give opposite signals then we go neutral.
   - Then we make a **optimal_strategy()** function and find out the result(cstrategy).
   - To maximize the results we use minimize method and give negative of our results.
   - Minimize method will find the local minima for our results.
   - To find out the local minima we firstly give starting parameters and their boundaries.
6. MACD:- In MACD indicator we use three EMA. One of the three called signal line of MACD. Whenever the difference between two EMA is greater then the signal EMA we take long position and vice versa.
   - To make this indicator we first add three column in data, one is EMA_S for short period EMA, one for long period EMA and other one for signal EMA using ewm() method.
   - We make another column named MACD for the difference of EMA_S and EMA_L.
   - In **test_starategy()** method we take long position when MACD is greater then MACD_signal we take long position and vice versa.
   - The rest code will be same as the previous indicators.



7. RSI:- The RSI is a Momentum Indicator used in technical analysis that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or

other asset. The RSI is displayed as an oscillator (a line graph that moves between two extremes) and can have a reading from 0 to 100.

- To calculate RSI, we make firstly a column “U” to find out the positive price difference in the consecutive raws and also make a column name “D” to find out the absolute negative price difference.
- Then we make a column “MA_U” to find the average up in prices and “MA_D” to find out average down in price for a specific period.
- Then we calculate RSI using the Rsi formula average up /(average up + average down) * 100
- In test_strategy() method we use two limits one is upper limit (generally 70) one is down limit(generally 30).
- Whenever the rsi is greater than the upper limit we go short and whenever the rsi is less then the lower limit we go long.
- The rest code is same.

8. RSI + MACD:- This is the most common combination used by the traders. As we combine two indicators SMA and Bollinger Bands, we can also combine RSI and MACD.
   - To combine them we make a optimal_strategy() method. Whenever the position for both indicators is one we go long and whenever the position for both indicators is -1 we go short. If position for one indicator is opposite the second indicator we go neutral.
   - To find the optimal parameters we use the minimize method by specifying starting parameters and bounderies.

```
In [5]: optimal_strategy((14, 70, 30, 12, 26, 9))
Out[5]: -0.5459285070880692

In [6]: bnds = ((10, 20), (60, 80), (20, 40), (5, 20), (20, 50), (5, 20))

In [7]: start_par = (14, 70, 30, 12, 26, 9)

In [8]: opts = minimize(optimal_strategy, start_par, method = "Powell" , bounds = bnds)

In [9]: opts
Out[9]:    direc: array([[ 1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                0.        ],
              [ 0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
                0.        ],
              [ 0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
                0.        ],
              [ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
                1.        ],
              [ 0.        ,  0.        ,  0.        ,  0.        ,  1.        ,
                0.        ],
              [ 0.09675752, -0.39877323, -1.35269101, -1.21411122, -0.67820993,
               -1.42468948]])
               fun: -1.3368046714133892
           message: 'Optimization terminated successfully.'
              nfev: 566
               nit: 4
            status: 0
           success: True
                 x: array([13.82309847, 72.19222662, 38.1494794 , 17.94404425, 35.4369478 ,
                16.40935294])

In [10]: optimal_strategy(opts.x)
Out[10]: -1.3368046714133892
```

9. <u>Stochastic Osc:-</u> A stochastic oscillator is a momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time. The sensitivity of the oscillator to market movements is reducible by adjusting that time period or by taking a moving average of the result. It is used to generate overbought and oversold trading signals, utilizing a 0–100 bounded range of values.

   - Stochastic Osc made up by two lines one is called fast stochastic osc or %K line and second is called slow stochastic osc or %D line.
   - We first make a column in our data by named “roll_low” to find minimum value in a specified period and a column “roll_high” to find maximum value.
   - Then we calculate %K by following formula –

     %K = 100 * (recent close price – roll_low) / (roll_high – roll_low)
   - %D is the moving average of the %K for a given period(generally 3 period).
   - In **test_strategy()** method whenever %K is greater then %D we go long and vice versa.
   - The rest code is same as pervious indicators.

Conclusion:-

- In this project we make an algo trading bot , which will automate our trading process. We can add any strategy to our trader.py file without changing any code apart from strategy specific method and attributes(we give strategy specific attributes and method to separate space).
- By using this we can eliminate our emotion from trading and we can execute our trades faster then we can think. We can also visualize our profit and loss more clearly on every trade.
- We can save a lot of time by using this because now we need not to stay in front of laptop screen to see charts and make decision because now our bot make decision according to our strategy.
- Thus, this is a very useful project in any trader's life.