# Bollinger Bands Forex Strategy

The Bollinger Bands strategy is applied to a financial dataset using a combination of parameters such as the moving average period, standard deviation multiplier, rolling window for calculating the highest high, and the risk-reward ratio. The strategy aims to generate buy or sell signals when the price crosses above or below the bands, along with predefined stop-loss and take-profit levels. The code also performs optimization to find the parameter combination that yields the highest final capital, and an equity curve is plotted to visualize the performance of the strategy over time.

## Importing Libraries:

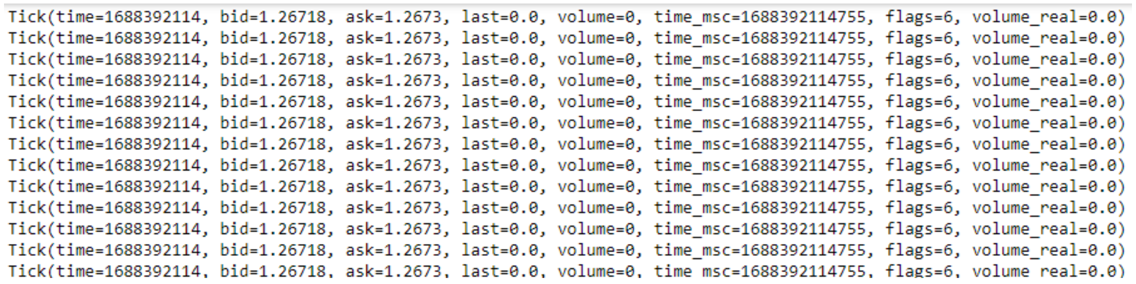
- The code begins by importing the necessary libraries such as MetaTrader5 (for accessing MetaTrader 5 terminal data), pandas (for data manipulation), datetime (for working with date and time), matplotlib.pyplot (for plotting charts), numpy (for numerical computations), time (for introducing delays), and yfinance (for downloading financial data from Yahoo Finance).

## Connecting to MetaTrader 5 Terminal:

- The code establishes a connection to the MetaTrader 5 terminal using the initialize() and login() functions from the MetaTrader5 library. It requires specifying the login ID, server name, and password.
- The account information is retrieved and displayed using the account_info() function.

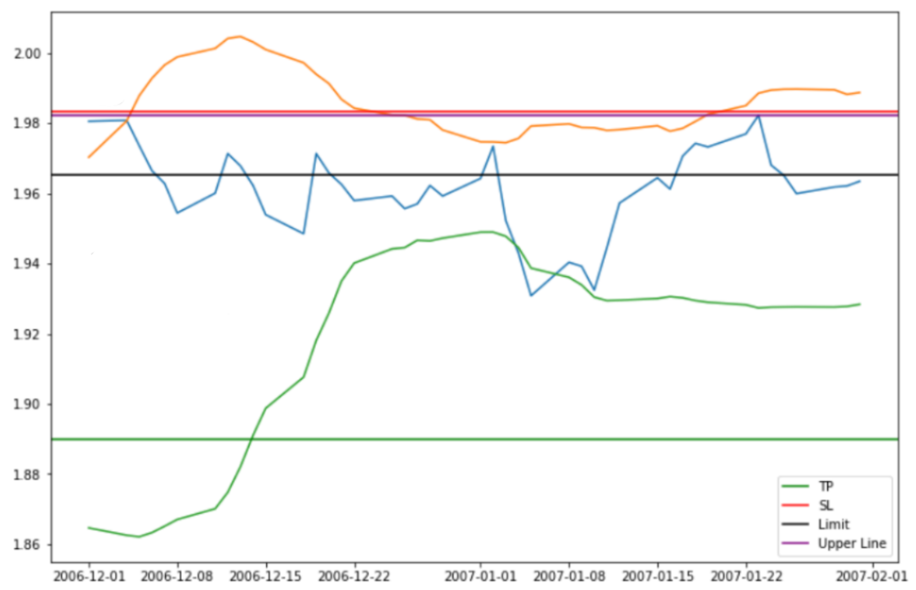
## Data Retrieval:

- The code retrieves data from various sources such as the MetaTrader 5 terminal, Yahoo Finance, and a CSV file.
- It demonstrates examples of accessing real-time tick data using symbol_info_tick() from the MetaTrader5 library.
- The downloaded data is stored in a Pandas DataFrame for further processing and analysis.


```
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
Tick(time=1688392114, bid=1.26718, ask=1.2673, last=0.0, volume=0, time_msc=1688392114755, flags=6, volume_real=0.0)
```

## Bollinger Bands Strategy:

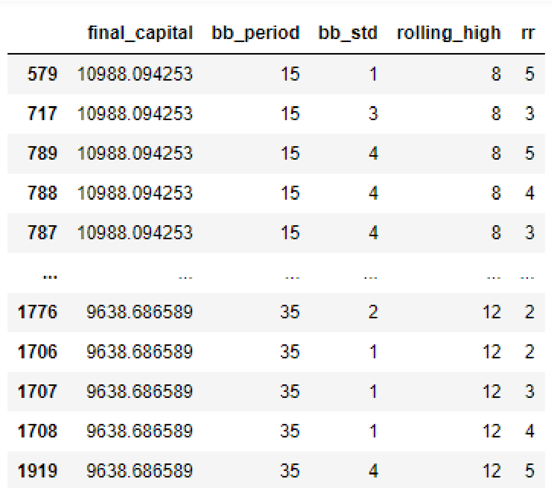
- The code defines a Bollinger Bands (BB) strategy function named bb_fx(). This function takes the following inputs:
    - df: DataFrame containing price data (Open, High, Low, Close, Volume)
    - bb_period: Period for calculating the moving average and standard deviation
    - bb_std: Number of standard deviations for determining the upper and lower Bollinger Bands
    - rolling_high: Rolling window for calculating the highest high
    - rr: Risk-reward ratio
    - capital: Initial capital
    - risk_per_trade: Risk per trade as a fraction of capital
- The strategy implements the following rules:
    - Calculates the Bollinger Bands using the moving average and standard deviation.
    - Identifies potential trade entry points when the price crosses above the upper Bollinger Band or below the lower Bollinger Band.
    - Sets stop-loss (SL) and take-profit (TP) levels based on the highest high within a rolling window.
    - Executes trades based on the predefined conditions and calculates the resulting capital and profit.
    - Records the trade details in an equity curve DataFrame.



- The function returns the final capital after executing the strategy on the given data.

## Optimization:

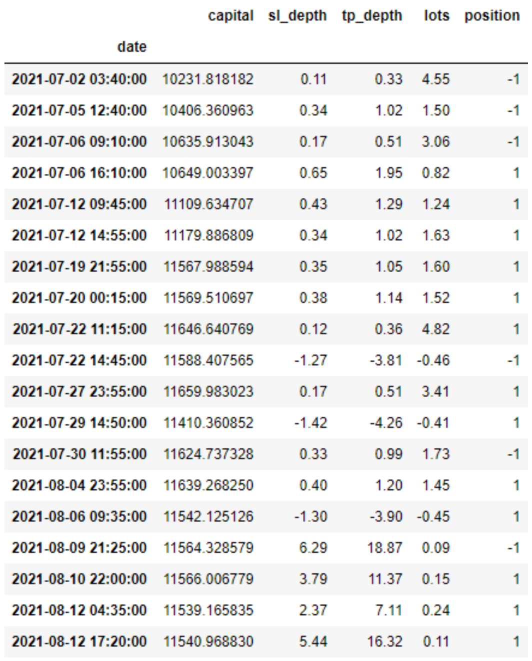
- The code performs optimization of the Bollinger Bands strategy parameters by iterating through various combinations of bb_period, bb_std, rolling_high, and rr values.
- It calls the bb_fx() function for each combination and records the final capital achieved.
- The results are stored in a DataFrame named optimize, which includes the parameter values and the corresponding final capital.
- The DataFrame is sorted in descending order based on the final capital, showing the optimized parameter combination with the highest capital.



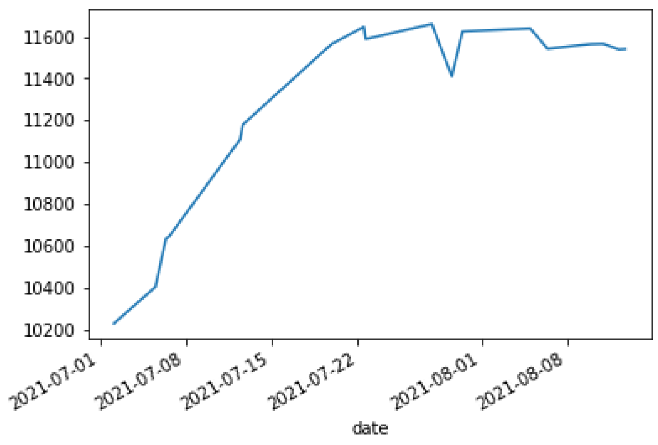
| | final_capital | bb_period | bb_std | rolling_high | rr |
|---|---|---|---|---|---|
| **579** | 10988.094253 | 15 | 1 | 8 | 5 |
| **717** | 10988.094253 | 15 | 3 | 8 | 3 |
| **789** | 10988.094253 | 15 | 4 | 8 | 5 |
| **788** | 10988.094253 | 15 | 4 | 8 | 4 |
| **787** | 10988.094253 | 15 | 4 | 8 | 3 |
| **...** | ... | ... | ... | ... | ... |
| **1776** | 9638.686589 | 35 | 2 | 12 | 2 |
| **1706** | 9638.686589 | 35 | 1 | 12 | 2 |
| **1707** | 9638.686589 | 35 | 1 | 12 | 3 |
| **1708** | 9638.686589 | 35 | 1 | 12 | 4 |
| **1919** | 9638.686589 | 35 | 4 | 12 | 5 |

## Execution of the Strategy:

- The code executes the Bollinger Bands strategy with the optimized parameter values obtained from the previous step.
- It generates an equity curve by recording the capital at each trade.
- The equity curve is plotted using matplotlib.pyplot.
- Trades & Equity Curve:



| date | capital | sl_depth | tp_depth | lots | position |
|---|---|---|---|---|---|
| **2021-07-02 03:40:00** | 10231.818182 | 0.11 | 0.33 | 4.55 | -1 |
| **2021-07-05 12:40:00** | 10406.360963 | 0.34 | 1.02 | 1.50 | -1 |
| **2021-07-06 09:10:00** | 10635.913043 | 0.17 | 0.51 | 3.06 | -1 |
| **2021-07-06 16:10:00** | 10649.003397 | 0.65 | 1.95 | 0.82 | 1 |
| **2021-07-12 09:45:00** | 11109.634707 | 0.43 | 1.29 | 1.24 | 1 |
| **2021-07-12 14:55:00** | 11179.886809 | 0.34 | 1.02 | 1.63 | 1 |
| **2021-07-19 21:55:00** | 11567.988594 | 0.35 | 1.05 | 1.60 | 1 |
| **2021-07-20 00:15:00** | 11569.510697 | 0.38 | 1.14 | 1.52 | 1 |
| **2021-07-22 11:15:00** | 11646.640769 | 0.12 | 0.36 | 4.82 | 1 |
| **2021-07-22 14:45:00** | 11588.407565 | -1.27 | -3.81 | -0.46 | -1 |
| **2021-07-27 23:55:00** | 11659.983023 | 0.17 | 0.51 | 3.41 | 1 |
| **2021-07-29 14:50:00** | 11410.360852 | -1.42 | -4.26 | -0.41 | 1 |
| **2021-07-30 11:55:00** | 11624.737328 | 0.33 | 0.99 | 1.73 | -1 |
| **2021-08-04 23:55:00** | 11639.268250 | 0.40 | 1.20 | 1.45 | 1 |
| **2021-08-06 09:35:00** | 11542.125126 | -1.30 | -3.90 | -0.45 | 1 |
| **2021-08-09 21:25:00** | 11564.328579 | 6.29 | 18.87 | 0.09 | -1 |
| **2021-08-10 22:00:00** | 11566.006779 | 3.79 | 11.37 | 0.15 | 1 |
| **2021-08-12 04:35:00** | 11539.165835 | 2.37 | 7.11 | 0.24 | 1 |
| **2021-08-12 17:20:00** | 11540.968830 | 5.44 | 16.32 | 0.11 | 1 |


date