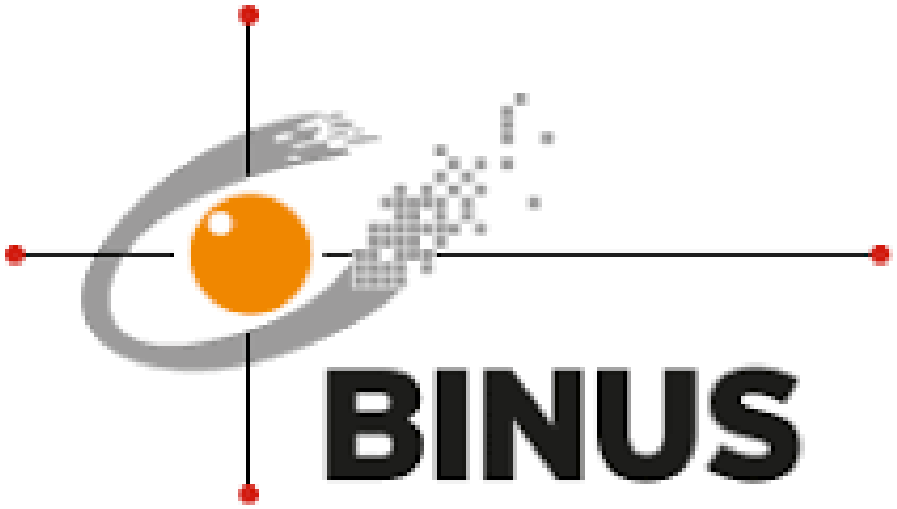
Algorithm Design and Analysis Final Report

Algorithmic Trading Bot and Indicator



BINUS
UNIVERSITY
INTERNATIONAL

Made By

Abdul Moiz - 2602186195

Nicholaus Santo - 2602174415

Ghai

1. Introduction / Background

Algorithmic trading or sometimes can also be called as automated trading, is a system in which a computer program is used to set and determine rules and options for which a trade will be placed based on the predetermined rules and options. In theory, it can also be considered as trading with the help of automatic presets and easy setups.

This project aims to give solutions to problematic trading issues such as time management, risk management, lack of strategy, and many other proposed problems that may occur when a party is entering the trading world. This project aims to help those who are just starting to get into trading to not make big mistakes and help the trader understand and be able to predict the market through an easier interface.

2. Problems

a. **Inconsistent Trading Plan**

Experienced traders get into a trade with a well-defined plan. They know their exact entry and exit points, the amount of capital to invest in the trade and the maximum loss they are willing to take. Beginner traders may not have a trading plan in place before they commence trading. Even if they have a plan, they may be more prone to stray from the defined plan than would season traders. Novice traders may reverse course altogether. For example, going short after initially buying securities because the share price is declining—only to end up getting whipsawed.

b. **Poor Risk Management**

Some investors find it difficult to handle the volatility, swings in the

market, or more risky trades. Other investors might require a reliable stream of interest income. These investors should avoid shares of more risky growth and startup companies and instead invest in the blue-chip stocks of well-established companies. Keep in mind that every investment return carries a risk. U.S. Treasury bonds, bills, and notes are the lowest risk investments that are readily available. From there, different investment kinds climb up the risk ladder and will also provide higher returns in order to make up for the increased risk taken. Look at the risk profile of an investment that provides very tempting returns to determine how much you could lose if things don't work out.

- **False Signals**

  A reduced stock price could result from deteriorating fundamentals, the resignation of the CEO, or increasing competition. The same factors also offer solid indications that the stock may not rise any time soon. A company's value may have decreased for basic causes recently. A low share price could be a deceptive buy signal, therefore it's crucial to constantly be skeptical. There is frequently a solid underlying explanation for a price fall. Before making an investment in a stock, we have to do our research and consider its outlook. The rule of thumb is that we want to put money into businesses that will continue to expand in the future.

3. Solutions and Methods

a. Solutions

i. Trading Bots

Trading Bots are automated algorithmic programs created to execute

trading strategies in the international markets for crypto assets using specialized trading methods based on predetermined criteria. They were created by outside developers and are available for purchase, subscription, or free download. These bots can operate around-the-clock on the market to make up for trader limitations and execute trades at their best (Seth, 2023).

Benefits of Trading Bots:

1. The volatile market can be traded continuously using trading bots. Trading bots fill the void left by human traders' inability to continuously watch the market without missing out on possibilities (Seth, 2023).
2. Trading bots have improved trading accuracy and market timing when the proper parameters are used. Finding the best moment to trade and the proper currency to earn earnings on the extremely volatile market is everything (Seth, 2023).
3. Automated trading bots have immediate access to market data and may quickly analyze it to determine prices (Seth, 2023).

ii. Compromised Trading Indicators

Trading indicators are essential for those that want to be successful in the business or to at least ease in when it comes to making profits. However, there are a multitude of ways to utilize each and every indicator that are present in the market. Some indicators are easier to use than others. There are also modified indicators that can be suited to each and every person.

By modifying those indicators, we can easily generalize those indicators making it easier for laymen to use (Seth, 2023).

Some of the more common indicators are:

1. Relative Strength Index
2. MACD
3. Moving Average
4. Exponential Moving Average
5. Bollinger Bands
6. Stochastic Oscillators

b. Methods

To create the said trading instruments, we have used some of the following Algorithm / Logics:

i. Greedy Algorithm

Any algorithm that employs the problem-solving heuristic of selecting the solution that is locally optimal at each stage is considered greedy. A greedy method frequently fails to find the best solution to a problem, but a greedy heuristic can quickly find answers that are close to the global best solution.

Such an algorithm would be the best probable solution to the problem due to its nature in always trying or attempting to find the next best solution to a problem. The market of trading is always considered to be volatile and inconsistent, with the greedy algorithm, there will always

be a chance of finding the best possible solution albeit not the solution that is entirely correct or wrong.

ii. Brute-Force Algorithm

The brute-force method speaks for itself, it will force a passable decision by trying all the possible answers. However, such an approach is very inefficient for trading and it can prove to be problematic as well. With every failed attempt, the amount of money will also be lost, making the brute-force method to be very inappropriate for trading.

We made a brute-force counterpart to test out the capabilities and see the result. The brute-force method is also one of the simplest logic that can be implemented into the trading instruments. We also made it because we wanted a comparison between the brute-force method against the greedy algorithm.

4. Measurement

To measure the results of our bot and indicators we will be using the Backtest python module, as it helps in showing the win rate, the amount of trades, the return percentages, alongside other parameters (Lûster, 2019). However, some of the main parameters that we will be paying attention to would be as follows:

a. Win Rate (Fail and Success)

By measuring the win rate, we can see if the strategy or the tactics are good enough to help new traders make profit and we can also see how the strategy or tactic fares against the other strategies and tactics by the win rate.

b. **Return Percentages (Initial Costs)**

The return percentages are important for measuring the total amount of money that has improved from the original amount of money that was put in.

c. **Time Frame**

The variety in the time frame is to determine which module is better at longer entries and short entries. The measurement of the time frame will be done through the 5 minute time frame, the 1 hour time frame, and the 1 day time frame.

5. The Code and Theories Behind It

a. **The Trading Bot (tradingbot.py)**

i. **Automated Trading with MetaTrader5 (MT5):**

Automated trading employs algorithms or scripts to execute predefined trading strategies without manual intervention. MetaTrader5 (MT5) provides an API allowing interaction with its platform programmatically. This script harnesses this API to automate trading decisions based on predefined conditions and market data (MetaQuotes Ltd., 2021).

ii. **Initialization and Account Login:**

The script starts by importing essential libraries, which are MetaTrader5 for MT5 interfacing and pandas for data manipulation and analysis. Other libraries are time.

```
import MetaTrader5 as MT
import pandas as pd
import time
from datetime import datetime
```

The initialization phase initializes the MT5 library via MT.initialize() and logs into a demo account using specific credentials - a login ID, password, and server details. This login process establishes a connection to the MT5 platform, enabling subsequent interactions.

```
MT.initialize()
login = 69120408
password = 'cq1ppuvf'
server = 'MetaQuotes-Demo'
MT.login(login, password, server)
account = MT.account_info()
```

iii. Defining Trading Functions:

The core functionalities of the script revolve around placing orders and managing positions. Three main functions are defined as:

buy_order() facilitates the placement of buy orders.

```
def buy_order(ticker, quantity):
```

sell_order() manages the placement of sell orders.

```
def sell_order(ticker, quantity):
```

close_order() orchestrates the closure of existing positions.

```
def close_order(position):
```

These functions interact with the MT5 API, specifying trade parameters like the currency pair ('USDJPY').

```
ticker = 'USDJPY'
quantity = 0.01
positions = MT.positions_get()
```

Order type (buy/sell), action, symbol, volume, type, price, type time and order filling type. They form the crux of trade execution within the platform.

```
"action": MT.TRADE_ACTION_DEAL,
"symbol": ticker,
"volume": quantity,
"type": MT.ORDER_TYPE_BUY,
"price": MT.symbol_info_tick("USDJPY").ask,
"type_time": MT.ORDER_TIME_GTC,
"type_filling": MT.ORDER_FILLING_IOC,
```

iv. Main Trading Loop and Data Analysis:

The script's primary operational segment, the main loop, iterates for a specified number of cycles. Within each iteration, it gathers one-minute OHLC (Open, High, Low, Close) data for the 'USD JPY' currency pair using MT.copy_rates_range(). This data, structured into a Pandas DataFrame, serves as the foundation for detailed data analysis, providing insights for trade decision-making based on historical market movements.

```
for i in range(100):
    ohlc = pd.DataFrame(MT.copy_rates_range('USDJPY',MT.TIMEFRAME_M1,datetime(2023,11,22),datetime.now()))
    ohlc['time']=pd.to_datetime(ohlc['time'],unit='s')
    print(ohlc)

    current_close = list(ohlc[-1:]['close'])[0]
    last_close = list(ohlc[-2:]['close'])[0]
    last_high = list(ohlc[-2:]['high'])[0]
    last_low = list(ohlc[-2:]['low'])[0]
```

v. Trading Strategy Implementation: Brute-Force

The decision-making process is guided by trading conditions established using the retrieved OHLC data. These conditions form the underlying strategy, leveraging technical indicators, price patterns, or market analysis

methodologies to identify potential trade entry or exit points. Conditions are meticulously crafted based on historical data to optimize trade decisions.

```
long_condition = current_close > last_high
short_condition = current_close < last_low
close_long = current_close < last_close
close_short = current_close > last_close
no_positions = len(MT.positions_get()) == 0
```

This logic is considered as a **brute-force** strategy due to how the bot will always enter a trade based on the closing of the previous singular candle, which means that the bot can only judge whether it's a buy or sell position only based on the two most recent candlestick. By using the said logic, the bot will always be in a trade for every single candlestick to try out whether the trade is a winning trade or a losing trade. In the case of a winning trade, the bot will hold and be in the trade until it has reached the targeted profit. If it is a losing trade, the bot will stop the trade and accept the loss.

vi. **Trade Execution and Position Management:**

Based on the predefined trading conditions, the code executes buy or sell orders using buy_order() and sell_order() functions. Simultaneously, it manages existing positions, ensuring alignment with the defined strategy or adaptation to evolving market conditions. The close_order() function closes positions as per the specified criteria, facilitating effective position

management.

```
close_long and already_buy:
 close_position()
 print('Buy Position Closed Without Entry')
close_short and already_sell:
 close_position()
 print('Sell Position Closed Without Entry')
```

vii. Error Handling and Delay Mechanism:

The script incorporates robust error handling mechanisms to manage exceptions that may arise during data retrieval or trade execution. This ensures stability and prevents abrupt halting of the code due to unforeseen errors.

```
try:
    already_sell = MT.positions_get()[0]._asdict()['type']==1
    already_buy = MT.positions_get()[0]._asdict()['type']==0
except:
    pass
```

b. Trading Indicators (Bollinger Bands and RSI)

i. Getting Data

To use an indicator, there needs to be data for the indicators to be based on and the data can be gathered the same way.

```
indices = pd.DataFrame(MT.copy_rates_range('USDJPY', MT.TIMEFRAME_D1,datetime(2013,12,29),datetime(2023,12,30)))
indices = indices[indices.high!=indices.low]
indices.reset_index(drop=True)
indices['time']=pd.to_datetime(indices['time'],unit='s')
```

```
dfRT = pd.DataFrame()
dfRT['Time']= indices['time']
dfRT['Open']= indices['open']
dfRT['High']= indices['high']
dfRT['Low']= indices['low']
dfRT['Close'] = indices['close']
dfRT['Volume'] = indices['tick_volume']
```

ii. Loading Indicators Based On The Data:

After the data is loaded, the indicators are then placed according to the settings that are optimal for the strategy towards the data that we have just gathered and processed.

```
dfRT['EMA']=ta.ema(indices.close, length=200)
dfRT['EMA2']=ta.ema(indices.close, length=150)
dfRT['RSI']=ta.rsi(indices.close, length=12)
my_bbands = ta.bbands(indices.close, length=14, std=2.0)
#my_bbands[0:100]
dfRT=dfRT.join(my_bbands)
dfRT.dropna(inplace=True)
dfRT.reset_index(inplace=True, drop=True)
```

iii. Loading The Signals From The Indicators:

The signals are based on certain conditions from the indicators, which is why this indicator is a **greedy algorithm approach**. The algorithm creates a signal only when conditions, such as when the RSI value is below the value of 75, the EMA level is overlapping to create a sell or buy indication, and if the price range is touching the bollinger band, eliminating the probability of the trade being a bad trade and predicting where the market will go based on the previous value of the prices with

the indicators (Fernando, 2023).

```
def addemasignal(df):
    emasignal = [0]*len(df)
    for i in range(0, len(df)):
        if df.EMA2[i]>df.EMA[i]:
            emasignal[i]=2
        elif df.EMA2[i]<df.EMA[i]:
            emasignal[i]=1
    df['EMASignal'] = emasignal
addemasignal(dfRT)

def addorderslimit(df, percent):
    ordersignal=[0]*len(df)
    for i in range(1, len(df)):
        if  df.Close[i]<=df['BBL_14_2.0'][i] and df.EMASignal[i]==2:
            ordersignal[i]=df.Close[i]-df.Close[i]*percent
        elif df.Close[i]>=df['BBU_14_2.0'][i] and df.EMASignal[i]==1:
            ordersignal[i]=df.Close[i]+df.Close[i]*percent
    df['ordersignal']=ordersignal

addorderslimit(dfRT, 0.000)

def pointposbreak(x):
    if x['ordersignal']!=0:
        return x['ordersignal']
    else:
        return np.nan
dfRT['pointposbreak'] = dfRT.apply(lambda row: pointposbreak(row), axis=1)
```

iv. Visualization

The signals from the indicators will be visualized to help the trader when they are going to enter a trade.

```
dfpl = dfRT.copy()
fig = go.Figure(data=[go.Candlestick(x=dfpl.index,
                open=dfpl['Open'],
                high=dfpl['High'],
                low=dfpl['Low'],
                close=dfpl['Close']),
                go.Scatter(x=dfpl.index, y=dfpl.EMA, line=dict(color='orange', width=2), name="EMA"),
                go.Scatter(x=dfpl.index, y=dfpl.EMA2, line=dict(color='yellow', width=2), name="EMA2"),
                go.Scatter(x=dfpl.index, y=dfpl['BBL_14_2.0'], line=dict(color='blue', width=1), name="BBL"),
                go.Scatter(x=dfpl.index, y=dfpl['BBU_14_2.0'], line=dict(color='blue', width=1), name="BBU")])

fig.add_scatter(x=dfpl.index, y=dfpl['pointposbreak'], mode="markers",
                marker=dict(size=6, color="Black"),
                name="Signal")
fig.update_xaxes(rangeslider_visible=False)
fig.update_layout(autosize=False, width=600, height=600,margin=dict(l=50,r=50,b=100,t=100,pad=4), paper_bgcolor="white")
fig.show()
```

6. The Flow

The project was done in steps that fits our likings and the following is how we completed the project:

a. **Creating the code**

   At first, we created the brute-force trading bot to make sure that the bot runs and it can be implemented with other strategies. After the bot runs smoothly, we created the indicators that can also be used as strategies for the bot. We found the strategies by looking at some examples on the internet and we implemented the strategy that fits our understanding.

b. **Benchmarking**

   After the codes are done, we test the codes' consistency and how it performs with different variables until we are satisfied.

c. **Testing on a Demo Account**

   Before we tried to use a real account, we used a demo account to make sure that the indicators and the bot worked in a safer environment.

d. **Optimizing**

   Based on the performance with the demo account, we optimized the bot and the indicators to be able to perform the way we wanted it to perform.

e. **Final Testing**

   After we optimized everything, we did a final test on the bot and indicators using real accounts and see how it performs in the real environment.

f. **Finalization**

   After the final test, we finalized the code and deemed it finished.

7. Result

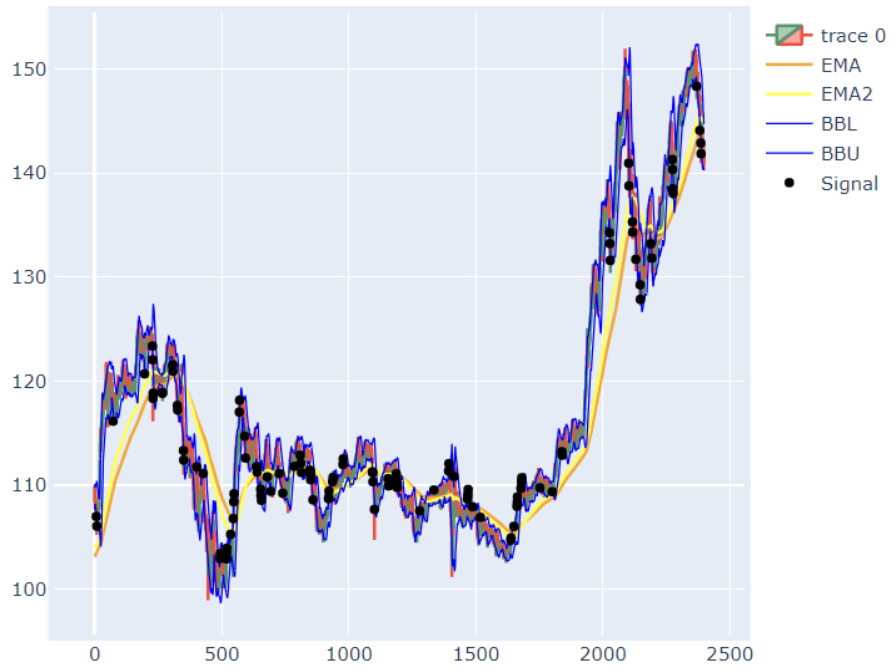
a. Brute-Force Trading Bot

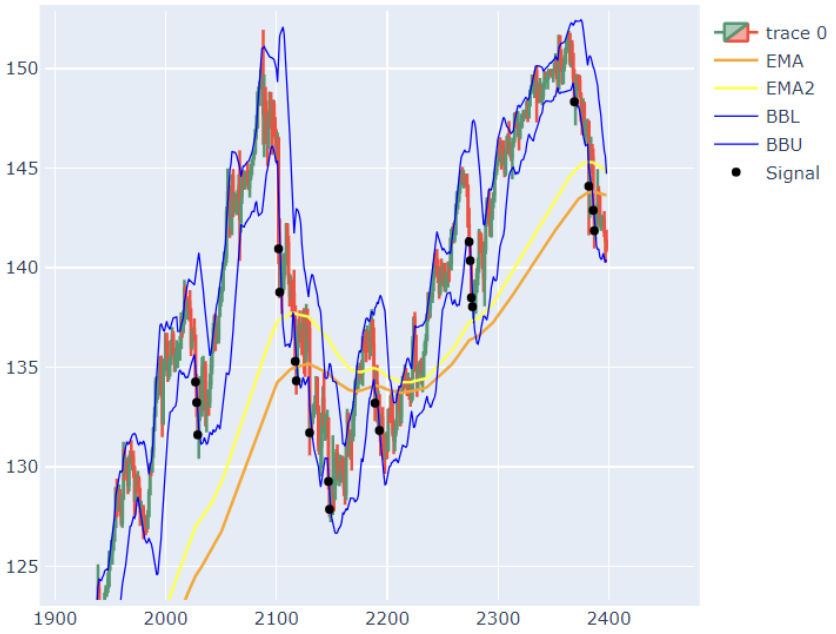
Time	Price	Profit	Change
2023.11.16 08:44:27	151.333	-0.12	-0.01 %
2023.11.16 08:44:41	151.333	0.03	0.00 %
2023.11.21 07:51:31	147.731	-4.33	-0.43 %
2023.11.21 07:51:33	147.731	-4.33	-0.43 %
2023.11.21 07:51:39	147.730	-4.10	-0.41 %
2023.11.21 07:51:44	147.737	4.21	0.42 %
2023.11.21 07:51:44	147.729	2.03	0.20 %
2023.11.21 07:51:45	147.729	2.09	0.21 %
2023.11.21 07:51:46	147.729	1.68	0.17 %
2023.11.21 07:51:47	147.729	1.57	0.16 %
2023.11.21 07:51:48	147.729	-4.26	-0.42 %
2023.11.21 07:51:49	147.741	-3.95	-0.39 %
2023.11.21 07:51:50	147.740	-3.89	-0.39 %

b. Bollinger Band (Signal, Performance, Result)

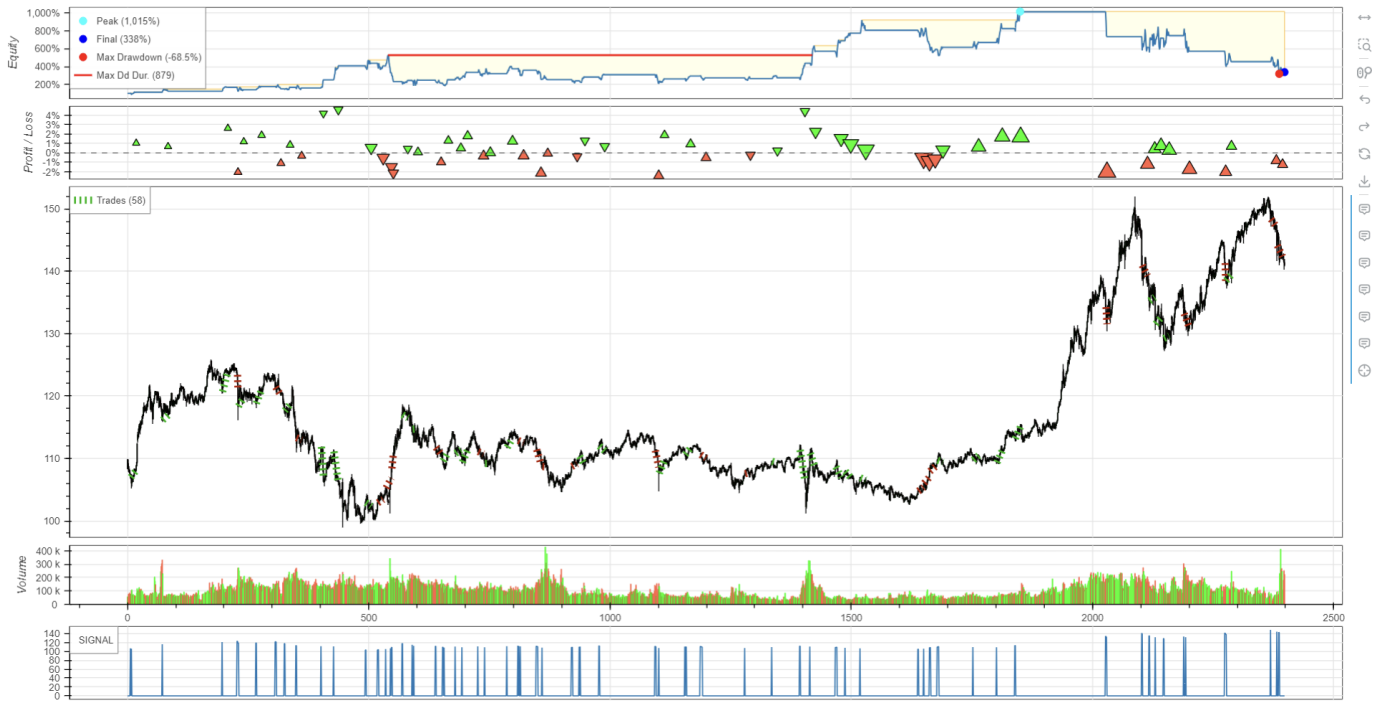
i. Daily Time Frame

1. Signal


trace 0
EMA
EMA2
BBL
BBU
Signal
150
140
130
120
110
100
0
500
1000
1500
2000
2500


trace 0
EMA
EMA2
BBL
BBU
Signal
150
145
140
135
130
125
1900
2000
2100
2200
2300
2400

2. Performance



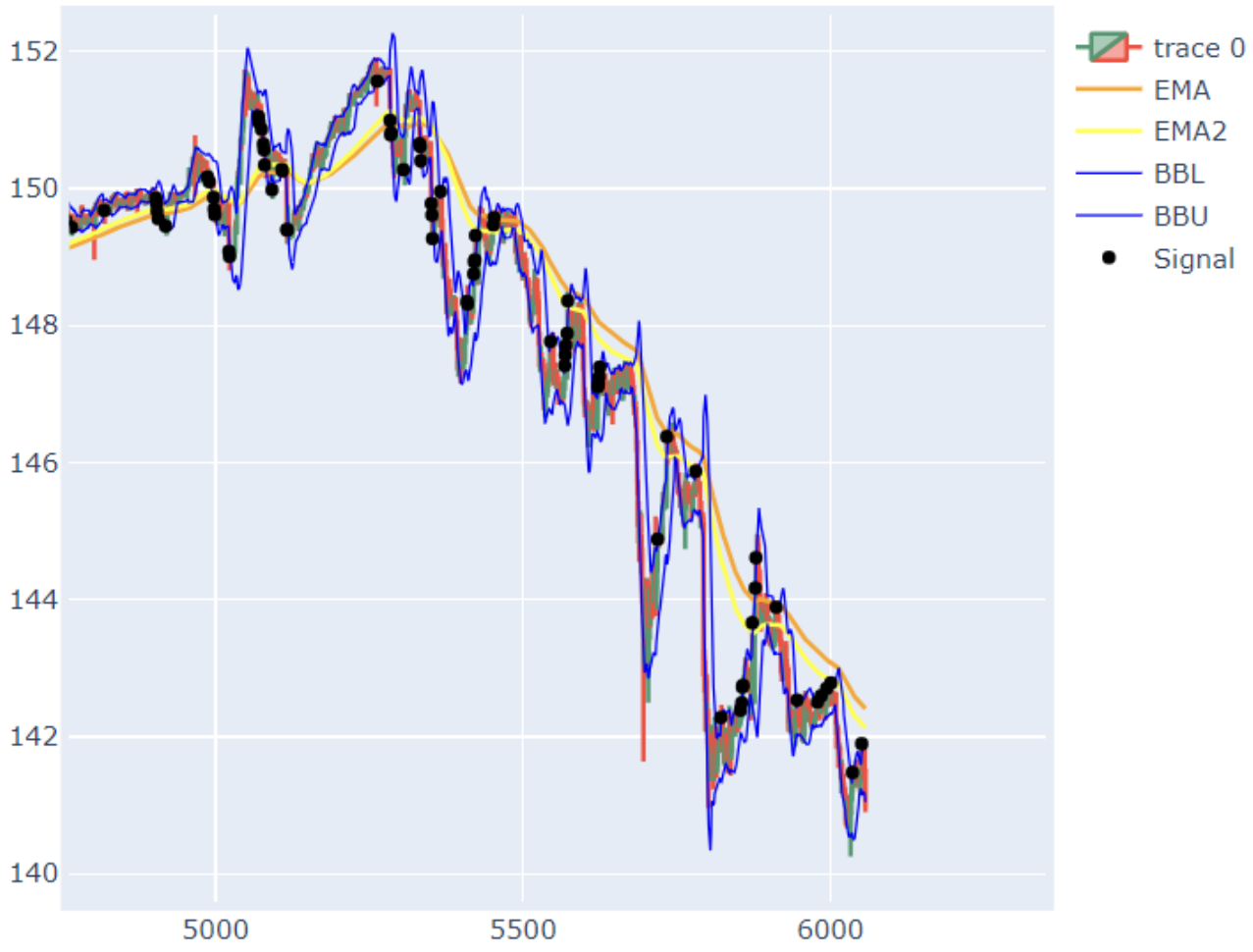
---

3. Result

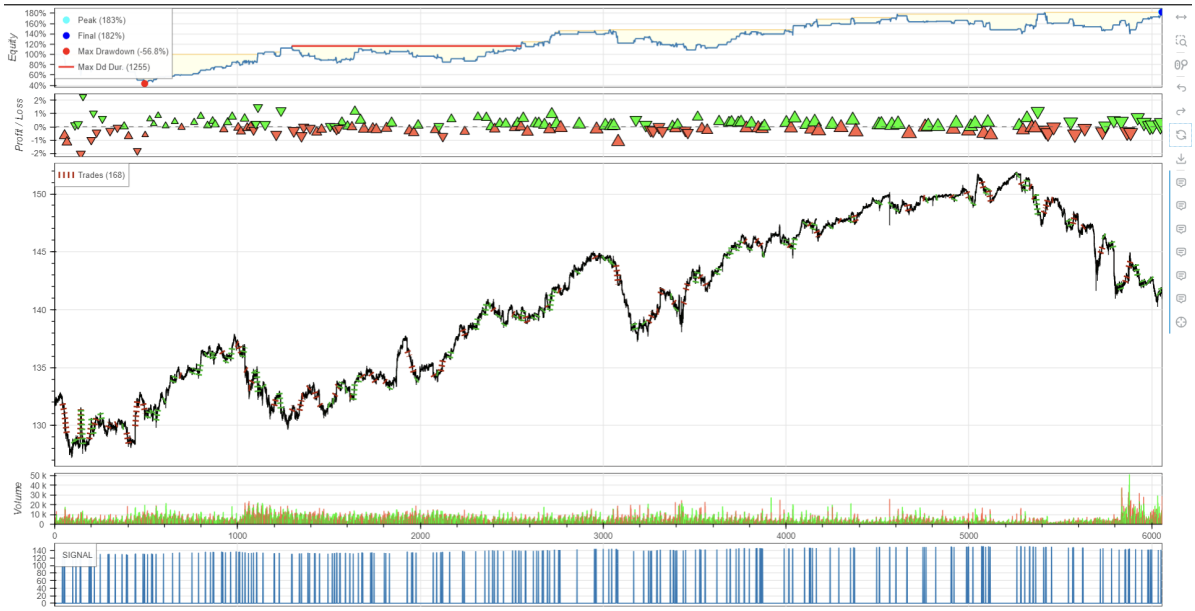
Start	0.0
End	2398.0
Duration	2398.0
Exposure Time [%]	26.802835
Equity Final [$]	3377.20248
Equity Peak [$]	10145.6683
Return [%]	237.720248
Buy & Hold Return [%]	28.45624
Return (Ann.) [%]	0.0
Volatility (Ann.) [%]	NaN
Sharpe Ratio	NaN
Sortino Ratio	NaN
Calmar Ratio	0.0
Max. Drawdown [%]	-68.5433
Avg. Drawdown [%]	-17.406276
Max. Drawdown Duration	880.0
Avg. Drawdown Duration	109.190476
# Trades	58.0
Win Rate [%]	58.62069
Best Trade [%]	4.588925
Worst Trade [%]	-2.417426
Avg. Trade [%]	0.294981
Max. Trade Duration	11.0
Avg. Trade Duration	10.086207
Profit Factor	1.680844
Expectancy [%]	0.30673
SQN	0.372376
_strategy	MyStrat
_equity_curve	Equity...
_trades	Size EntryB...

ii. Hourly Time Frame

1. Signal



2. Performance

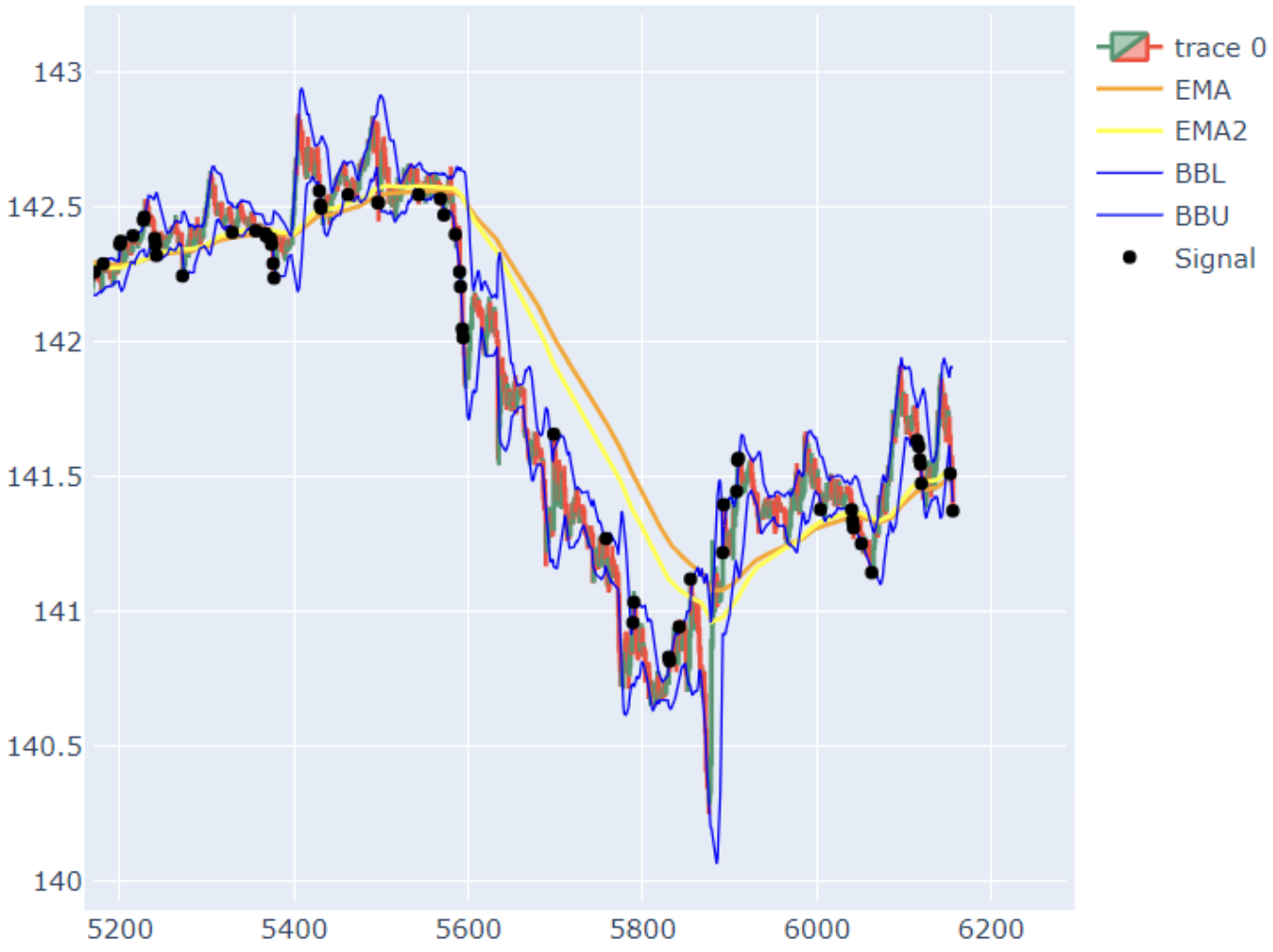


3. Result

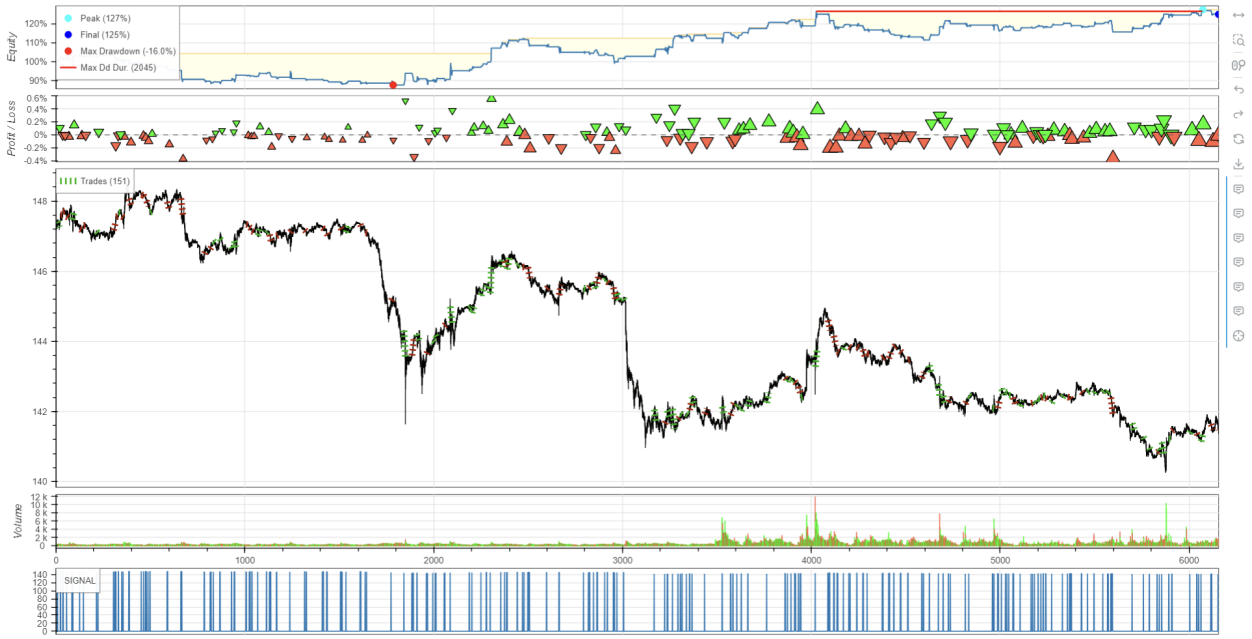
Start	0.0
End	6057.0
Duration	6057.0
Exposure Time [%]	32.667547
Equity Final [$]	1823.43798
Equity Peak [$]	1827.27898
Return [%]	82.343798
Buy & Hold Return [%]	7.000675
Return (Ann.) [%]	0.0
Volatility (Ann.) [%]	NaN
Sharpe Ratio	NaN
Sortino Ratio	NaN
Calmar Ratio	0.0
Max. Drawdown [%]	-56.816382
Avg. Drawdown [%]	-7.748795
Max. Drawdown Duration	1255.0
Avg. Drawdown Duration	187.870968
# Trades	168.0
Win Rate [%]	53.571429
Best Trade [%]	2.265875
Worst Trade [%]	-2.030871
Avg. Trade [%]	0.04256
Max. Trade Duration	11.0
Avg. Trade Duration	10.779762
Profit Factor	1.285503
Expectancy [%]	0.04382
SQN	0.924921
_strategy	MyStrat
_equity_curve	Equity...
_trades	Size Entry...

iii. 5-Minute Time Frame

1. Signal



2. Performance



---

3. Result

Start	0.0
End	6156.0
Duration	6156.0
Exposure Time [%]	29.0726
Equity Final [$]	1249.175
Equity Peak [$]	1273.677
Return [%]	24.9175
Buy & Hold Return [%]	-3.993779
Return (Ann.) [%]	0.0
Volatility (Ann.) [%]	NaN
Sharpe Ratio	NaN
Sortino Ratio	NaN
Calmar Ratio	0.0
Max. Drawdown [%]	-15.9699
Avg. Drawdown [%]	-2.302716
Max. Drawdown Duration	2046.0
Avg. Drawdown Duration	189.322581
# Trades	151.0
Win Rate [%]	47.682119
Best Trade [%]	0.557187
Worst Trade [%]	-0.368727
Avg. Trade [%]	0.012207
Max. Trade Duration	11.0
Avg. Trade Duration	10.854305
Profit Factor	1.287312
Expectancy [%]	0.012305
SQN	1.014754
_strategy	MyStrat
_equity_curve	Equity ...
_trades	Size Entry...

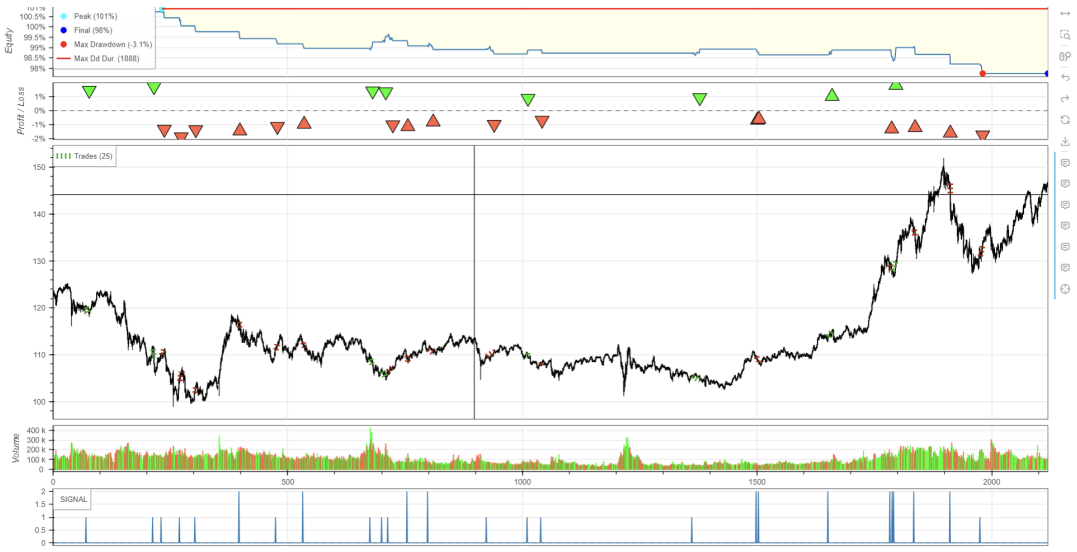
c. RSI Indicator (Signal, Performance, Result)

i. Daily Time Frame

1. Signal

---

2. Performance



---

3. **Result**

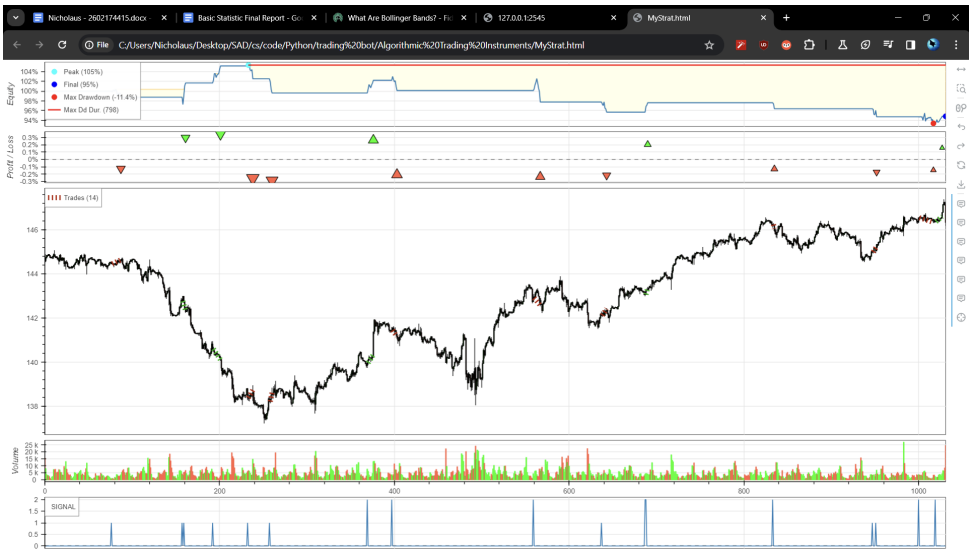
```
Start                                     0.0
End                                    2120.0
Duration                               2120.0
Exposure Time [%]                    6.883545
Equity Final [$]                   977.520493
Equity Peak [$]                   1008.726313
Return [%]                           -2.247951
Buy & Hold Return [%]                19.088556
Return (Ann.) [%]                          0.0
Volatility (Ann.) [%]                      NaN
Sharpe Ratio                               NaN
Sortino Ratio                              NaN
Calmar Ratio                               0.0
Max. Drawdown [%]                    -3.093586
Avg. Drawdown [%]                    -0.807341
Max. Drawdown Duration                  1888.0
Avg. Drawdown Duration                   474.0
# Trades                                  25.0
Win Rate [%]                               32.0
Best Trade [%]                        1.800077
Worst Trade [%]                      -1.867705
Avg. Trade [%]                       -0.381291
Max. Trade Duration                        16.0
Avg. Trade Duration                        4.84
Profit Factor                         0.528592
Expectancy [%]                       -0.374018
SQN                                  -1.553858
_strategy                              MyStrat
_equity_curve                         Equit...
_trades                      Size  EntryB...
```

ii. Hourly Time Frame

1. Signal

---

2. Performance



---

3. **Result**

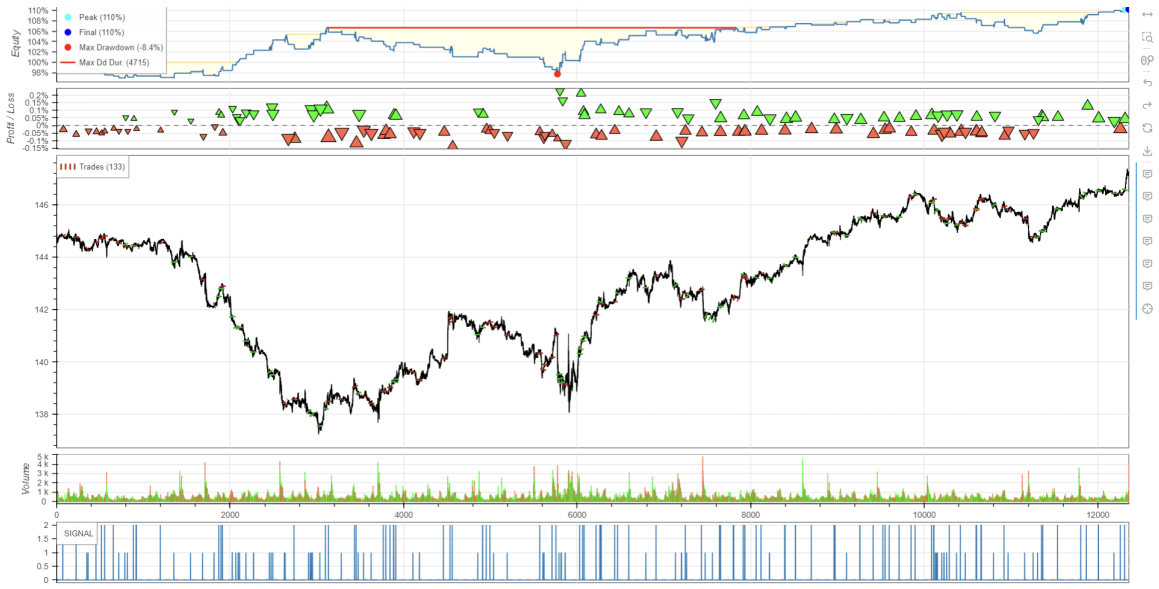
```
Start                                     0.0
End                                    1031.0
Duration                               1031.0
Exposure Time [%]                    9.205426
Equity Final [$]                   948.401384
Equity Peak [$]                   1053.482399
Return [%]                           -5.159862
Buy & Hold Return [%]                1.194988
Return (Ann.) [%]                         0.0
Volatility (Ann.) [%]                     NaN
Sharpe Ratio                              NaN
Sortino Ratio                             NaN
Calmar Ratio                              0.0
Max. Drawdown [%]                   -11.354333
Avg. Drawdown [%]                    -3.117863
Max. Drawdown Duration                   798.0
Avg. Drawdown Duration                   177.4
# Trades                                  14.0
Win Rate [%]                        35.714286
Best Trade [%]                       0.346159
Worst Trade [%]                     -0.285171
Avg. Trade [%]                      -0.035886
Max. Trade Duration                      16.0
Avg. Trade Duration                  5.785714
Profit Factor                        0.717492
Expectancy [%]                      -0.035637
SQN                                 -0.598848
_strategy                            MyStrat
_equity_curve                        Equit...
_trades                    Size  EntryB...
```

iii. 5-Minute Time Frame

1. Signal

---

2. Performance



---

3. Result

Start	0.0
End	12358.0
Duration	12358.0
Exposure Time [%]	10.203091
Equity Final [$]	1102.457969
Equity Peak [$]	1102.457969
Return [%]	10.245797
Buy & Hold Return [%]	1.17734
Return (Ann.) [%]	0.0
Volatility (Ann.) [%]	NaN
Sharpe Ratio	NaN
Sortino Ratio	NaN
Calmar Ratio	0.0
Max. Drawdown [%]	-8.383937
Avg. Drawdown [%]	-0.905651
Max. Drawdown Duration	4695.0
Avg. Drawdown Duration	333.34375
# Trades	133.0
Win Rate [%]	45.864662
Best Trade [%]	0.225986
Worst Trade [%]	-0.136889
Avg. Trade [%]	0.007619
Max. Trade Duration	119.0
Avg. Trade Duration	8.481203
Profit Factor	1.28252
Expectancy [%]	0.007643
SQN	1.234244
_strategy	MyStrat
_equity_curve	Equi...
_trades	Size Entry...

8. Discussion and Analysis

a. The Bot

Based on our result, the bot made a trade for almost every time a new candle was created and it resulted in the profit being a sum-zero gain or a negative gain. In the table above, the amount of loss overcame the amount of positive

profit due to the amount of trade that was taken being more faulty than good. Most of the trade that was taken was also closed out by the bot and not because of the stop-loss or the take-profit, meaning that the bot was very indecisive with the solutions that the bot had found according to the logic that was given.

The brute-force method seems to be very ineffective and it has a negative impact towards the goal that we were trying to achieve. Compared to the performances of the other strategies, the brute-force method that was implemented into the bot is so far behind, the brute-force method has almost a zero percent return rate. However, the brute-force method was only used as a testing and simple strategy. The ineffectiveness and the failure of the strategy was as expected.

After several testing and benchmarks, the time complexity of the bot has been found to be O(n) and since we ran the bot to measure up to 100 candlesticks, we concluded that the time complexity of the bot is O(100) and since the time complexity is O(100) the space complexity is also the same measure.

b. Bollinger Bands

The bollinger bands have shown good results overall, it has always shown a positive return rate with a passable win rate. The bollinger band was originally installed as an indicator, but it was implemented into the bot and it was backtested and automated with the help of the Backtesting python module (Lûster, 2019). The extra line of codes for the backtesting are as follows:

```
def SIGNAL():
    return dfpl.ordersignal

class MyStrat(Strategy):
    initsize = 0.9
    mysize = initsize
    def init(self):
        super().init()
        self.signal = self.I(SIGNAL)

    def next(self):
        super().next()
        TPSLRatio = 2
        perc = 0.02

        if len(self.trades)>0:
            if self.data.index[-1]-self.trades[-1].entry_time>=10:
                self.trades[-1].close()
            if self.trades[-1].is_long and self.data.RSI[-1]>=75:
                self.trades[-1].close()
            elif self.trades[-1].is_short and self.data.RSI[-1]<=25:
                self.trades[-1].close()

        if self.signal!=0 and len(self.trades)==0 and self.data.EMASignal==2:
            sl1 = min(self.data.Low[-1],self.data.Low[-2])*(1-perc)
            tp1 = self.data.Close[-1]+(self.data.Close[-1] - sl1)*TPSLRatio
            self.buy(sl=sl1, tp=tp1, size=self.mysize)

        elif self.signal!=0 and len(self.trades)==0 and self.data.EMASignal==1:
            sl1 = sl1 = max(self.data.High[-1],self.data.High[-2])*(1+perc)
            tp1 = self.data.Close[-1]-(sl1 - self.data.Close[-1])*TPSLRatio
            self.sell(sl=sl1, tp=tp1, size=self.mysize)

bt = Backtest(dfpl, MyStrat, cash=1000, margin=1/15, commission=.000)
stat = bt.run()
bt.plot()
print(stat)
print(dfpl)
```

The result for the backtesting shows that the bollinger bands are very optimized for high returns (Hayes, 2020). In almost every time frame the bands were tested, it has a very high return rate. However, it is clear that most of the bigger time frames, such as the daily time frame, were more effective and produced bigger results. The win rate and the return rate on the daily time frame is way bigger compared to the win rate and return rate on the hourly and 5-minute

time frame.

The time complexity of the bollinger bands indicator is O(n) and after running it for a couple of times, the average has been found to be 3 seconds to initialize the indicator, but it took around 10 seconds to initialize the indicator when the data wasn't loaded. The space complexity is O(n).

c. RSI Indicator

The RSI indicator was also helped by the Backtesting python module and the code for the backtesting strategy is as follows:

```
def SIGNAL():
    return dfpl.TotSignal

class MyStrat(Strategy):
    initsize = 0.02
    mysize = initsize
    def init(self):
        super().init()
        self.signal1 = self.I(SIGNAL)

    def next(self):
        super().next()
        slatr = 1.3*self.data.ATR[-1]
        TPSLRatio = 1.3

        if self.signal1==2 and len(self.trades)==0:
            sl1 = self.data.Close[-1] - slatr
            tp1 = self.data.Close[-1] + slatr*TPSLRatio
            self.buy(sl=sl1, tp=tp1, size=self.mysize)

        elif self.signal1==1 and len(self.trades)==0:
            sl1 = self.data.Close[-1] + slatr
            tp1 = self.data.Close[-1] - slatr*TPSLRatio
            self.sell(sl=sl1, tp=tp1, size=self.mysize)

bt = Backtest(dfpl, MyStrat, cash=1000, margin=1/500, commission=.00)
stat = bt.run()
print(stat)
bt.plot()
```

The result of this indicator seems to be more tailored towards the smaller

time frame, meaning that this indicator is better for trying to get quick profits and wins. However, the win rate and the return rate of this indicator is not as good and as effective as the Bollinger Bands indicator.

The time complexity for this indicator model is also O(n) and the precise time on the first initialization is 18 seconds and the average time to load the whole indicator signal with the data loaded is 7 seconds. The space complexity is O(n),

9. Conclusion and Recommendation

a. Conclusion

The trading bot worked very well and it is very stable, even when the variables are changed and modified. The bot is very helpful when it comes to automating real trades in the real environment. The brute-forced that is implemented in the current version of the bot has proven to be ineffective and faulty for trading, hence it should not be used. However, the two greedy approaches that are implemented in the indicators have worked very well and as expected, the results are always informative and it can be used to help traders with entering the market at the right time.

The result of the two strategies that are implemented with the greedy approaches has different results, albeit always positive. The RSI indicator has shown results that are fitted better for smaller time frames, such as the 5-Minute, the 15-Minute, the 1-Minute, and many other minutely based time frames. On the other hand, the Bollinger Bands are best used on bigger time frames, such as the Daily, Weekly, and Monthly time frames. In conclusion, if the traders are looking for quick trading signals with smaller positive returns, they can use the RSI

indicator and if the traders are looking for bigger returns with the cost of more time, they can use the Bollinger Bands.

b. Recommendation

Recommendation for future research and projects that are similar, one might want to look into other types of indicators and strategies as there are an abundance of strategies that can be implied into trading the forex market. A model where the instruments are easier to control with better interface can also be looked upon and researched inside of future projects. Another suggestion would be to make a model that is suited better for the Crypto market instead of the forex market.

References

Fernando, J. (2023, March 31). *Relative Strength Index – RSI*. Investopedia. https://www.investopedia.com/terms/r/rsi.asp

Hayes, A. (2020, May 7). *Bollinger Band®*. Investopedia. https://www.investopedia.com/terms/b/bollingerbands.asp

Johnson, K. (2019, February 26). *pandas-ta 0.3.14b : An easy to use Python 3 Pandas Extension with 130+ Technical Analysis Indicators. Can be called from a Pandas DataFrame or standalone like TA-Lib. Correlation tested with TA-Lib.* PyPI. https://pypi.org/project/pandas-ta/

Lûster, Z. (2019, January 17). *Backtesting: Backtest trading strategies in Python*. PyPI. https://pypi.org/project/Backtesting/

McKinney, W. (2018). *Python Data Analysis Library — pandas: Python Data Analysis Library*. Pydata.org. https://pandas.pydata.org/

MetaQuotes Ltd. (2021, October 23). *MetaTrader5: API Connector to MetaTrader 5 Terminal*. PyPI. https://pypi.org/project/MetaTrader5/

Seth, S. (2023, March 21). *Basics of algorithmic trading: Concepts and examples*. Investopedia. https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp

Links

https://github.com/SAD-Nich/TradingBot-ADA-FinalProject.git

Link to GitHub Repo with the bot and indicators, alongside this report.