Quantitative Trading Strategy – BTC 15-Minute OHLCV Data

This project builds a rule-based **algorithmic trading model** using 15-minute interval OHLCV data for Bitcoin. The objective was to create a profitable trading strategy by combining **technical indicators** like RSI, EMA, SMA, ATR, Bollinger Bands, and volume filters to generate **buy/sell signals**.

Key components include:

- **Data cleaning & preprocessing** from Binance repository
- Technical indicator-based **signal generation**
- **Volatility and momentum filters** to reduce noise
- A **backtesting framework** with stop-loss logic
- Evaluation through metrics like **cumulative returns, Sharpe ratio, win rate, and drawdown**
- Equity curve & trade plots for visualization

The model was iteratively tuned and tested on subsequent months to ensure robustness and minimize overfitting. Final results show promising profitability under defined risk constraints.

Data preparation and data cleaning was done initially i.e removal of unnecessary variables , checking for missing values . then understanding variables and how data is presented .

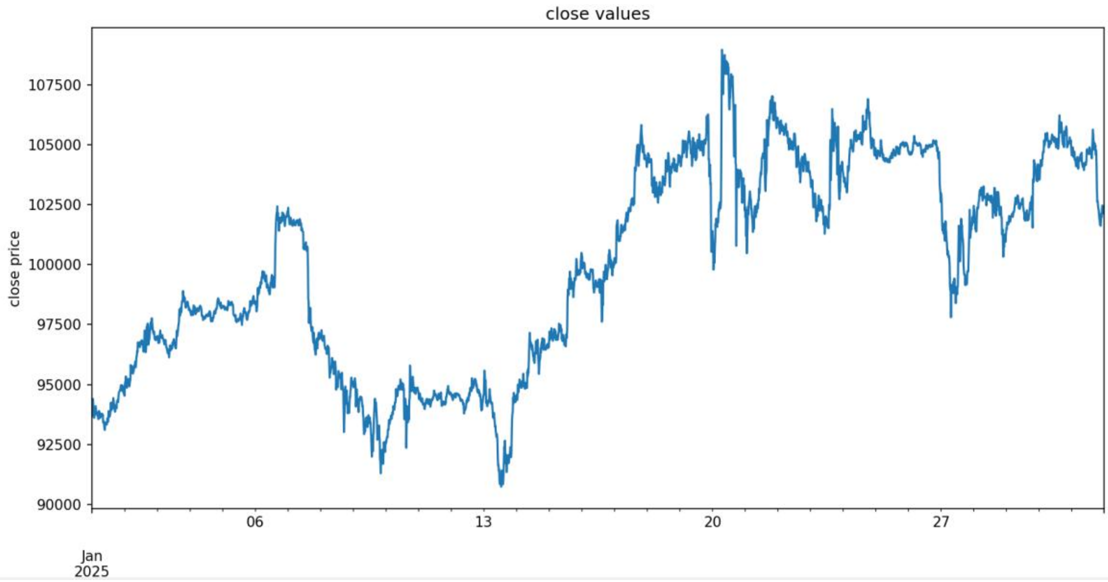
Open time used as index for analysis and data sorted by time then

```
# setting date time format , indexed open time for easier ploting
data['open time'] = pd.to_datetime(data['open time'])
data = data.sort_values('open time')
data.set_index('open time', inplace=True )
```

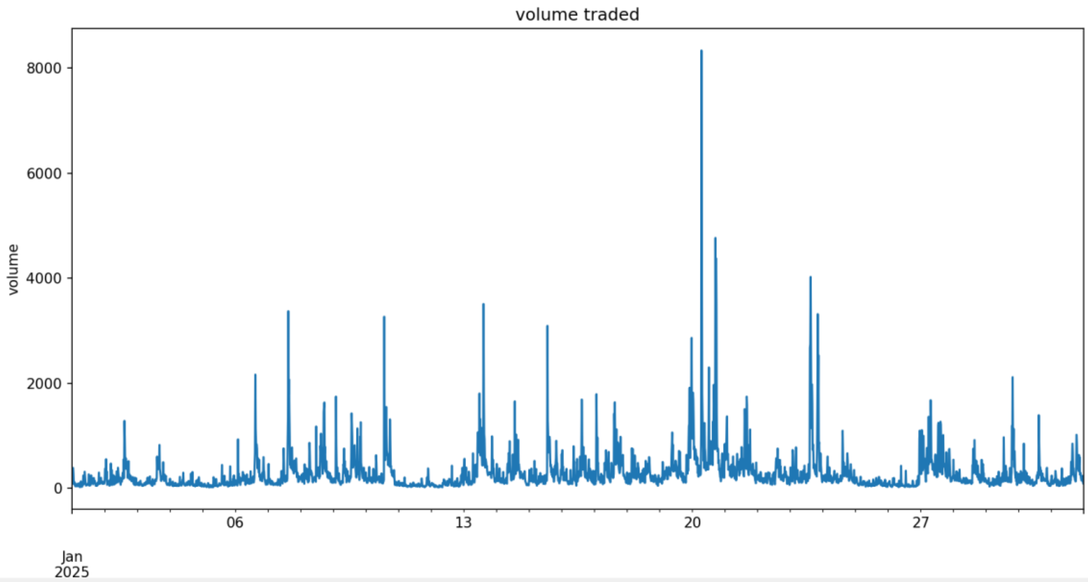
Starting with EDA

1.
```
print(data.head())
print(data.dtypes)
print(data.describe()) # basic data features
print(data.isnull().sum()) # check for missing values
```
Basic understanding of data here

2.
```python
import matplotlib.pyplot as plt
plt.figure(figsize=(30,10))
data['close'].plot()
plt.title('close values')
plt.xlabel('time')
plt.ylabel('close price')
plt.show()
```

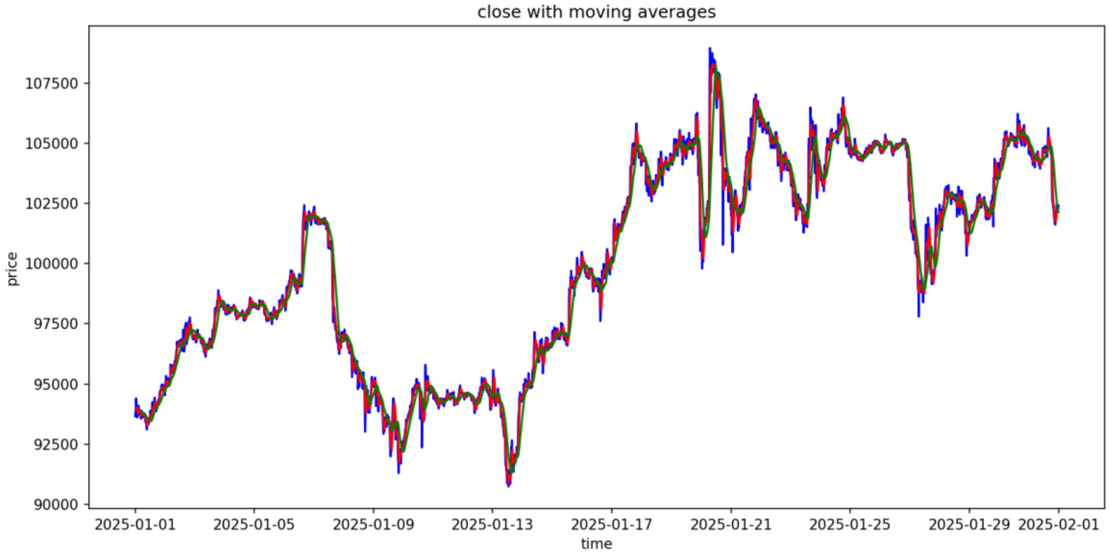


3.
```python
plt.figure(figsize=(30,10))
data['volume'].plot()
plt.title('volume traded')
plt.xlabel('time')
plt.ylabel('volume')
plt.show()
```



4. Then we used moving averages and exponential moving average and plotted the same

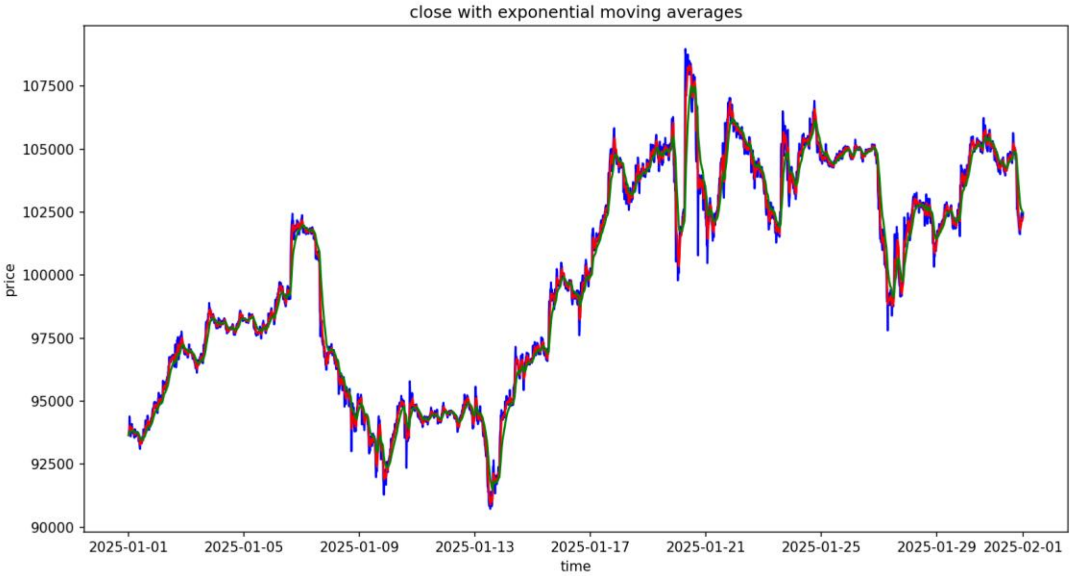
```python
plt.figure(figsize=(15,7))
plt.plot(data['close'], label='close price', color='blue')
plt.plot(data['SMA_5'], label='ma5', color='red')
plt.plot(data['SMA_20'], label='ma20', color='green')
plt.title('close with moving averages') #is short MA5 is more than long MA20 then buy else sell
plt.xlabel('time')
plt.ylabel('price')
plt.legend
plt.show()
```



Significant difference between red and green can be used a signal to make decision. If SMA 5 (red) is significantly more than SMA 20 (green) then it is a good signal to buy. If SMA 5 (red) is significantly less than SMA 20 (green) then it is a good signal to sell.

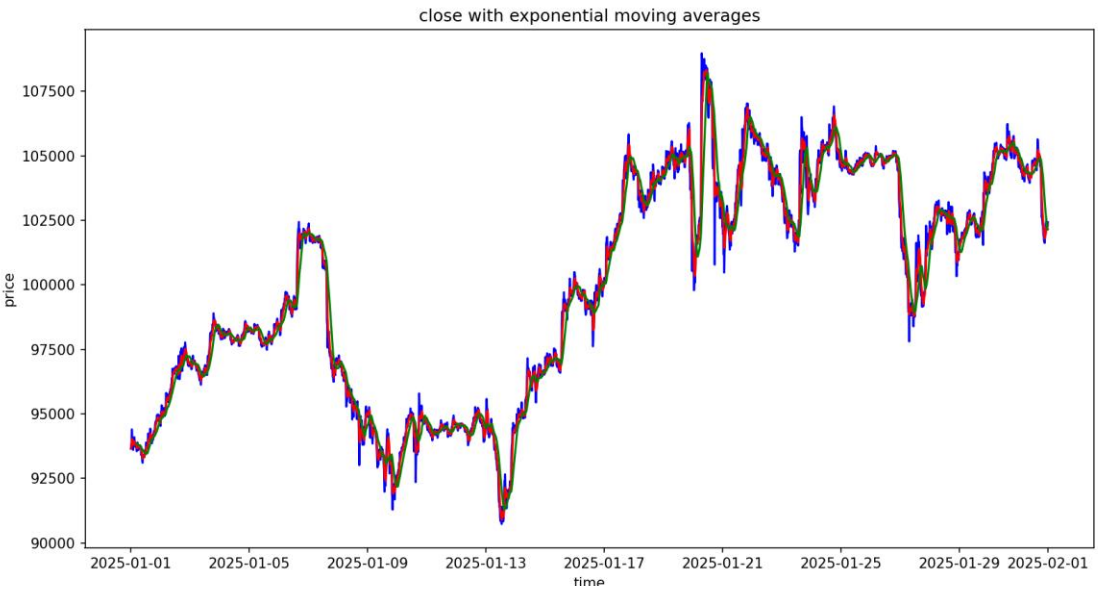
Similar is the case with EMA the difference between SMA and EMA is that EMA assigns more weight to most recent values whereas SMA assigns equal weights to all the values.

```python
plt.figure(figsize=(15,7))
plt.plot(data['close'], label='close price', color='blue')
plt.plot(data['EMA_5'], label='ema5', color='red')
plt.plot(data['EMA_20'], label='ema20', color='green')
plt.title('close with exponential moving averages') #is short MA5 is more than long MA20 then
plt.xlabel('time')
plt.ylabel('price')
plt.legend
plt.show()
```



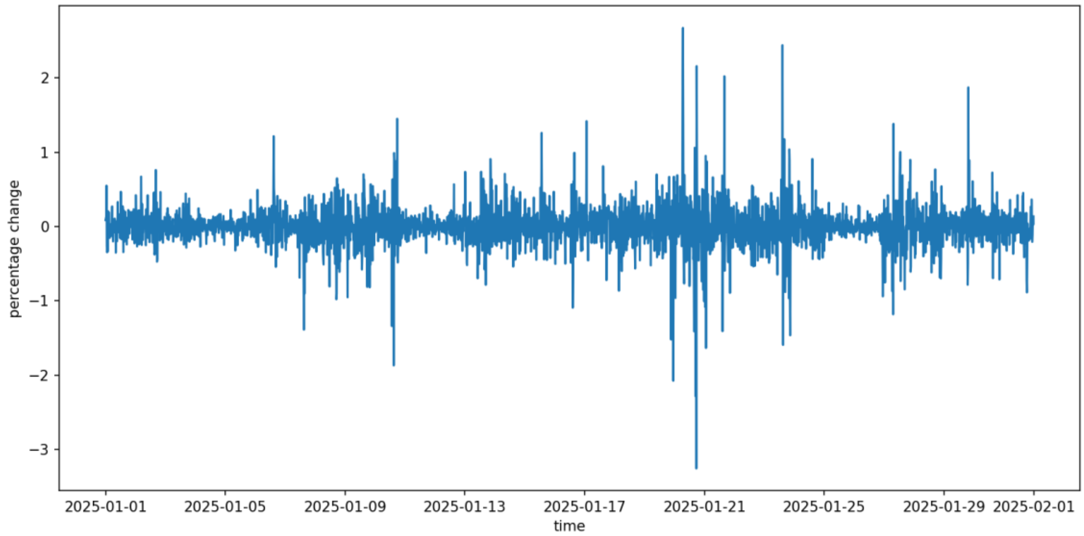
Now we compare EMA 5 with SMA 20 for better signal.

```
plt.figure(figsize=(15,7))
plt.plot(data['close'], label='close price', color='blue')
plt.plot(data['EMA_5'], label='ma5', color='red')
plt.plot(data['SMA_20'], label='ema20', color='green')
plt.title('close with exponential moving averages') #is short MA5 is more than long MA20 then buy else sell
plt.xlabel('time')
plt.ylabel('price')
plt.legend
plt.show()

#If Close > SMA and rising → bullish signal.
#If SMA rising steadily → long-term uptrend.
#If EMA crosses above SMA → buy signal.
#If EMA crosses below SMA → sell signal.
#If Close far above SMA/EMA → price may revert.
#EMA reacts faster → catches short-term moves
#SMA gives smoother long-term trend.
# we use for short EMA and for long SMA and use this as signal as mentioned above
```



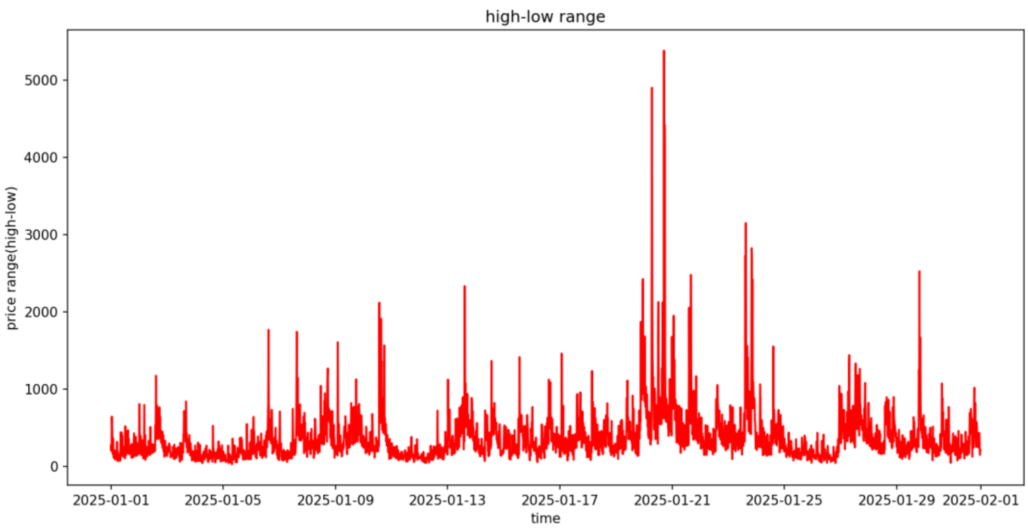
5. Now we explore the % change in volume to get a hint of momentum in market and what market sentiment is going on

```
data['perc_change'] = ((data['close'] - data['open'])/data['open']) * 100
plt.figure(figsize=(15,6))
plt.plot(data['perc_change'])
plt.xlabel('time')
plt.ylabel('percentage change')
plt.show()

#here change>0% means candle is bullish and change<0% means bearish close to 0 is low momentum
#basically large +change implies strong upward trend and buying pressure and opp for -ve change
```



6. Now we plot high low ranges to understand volatility

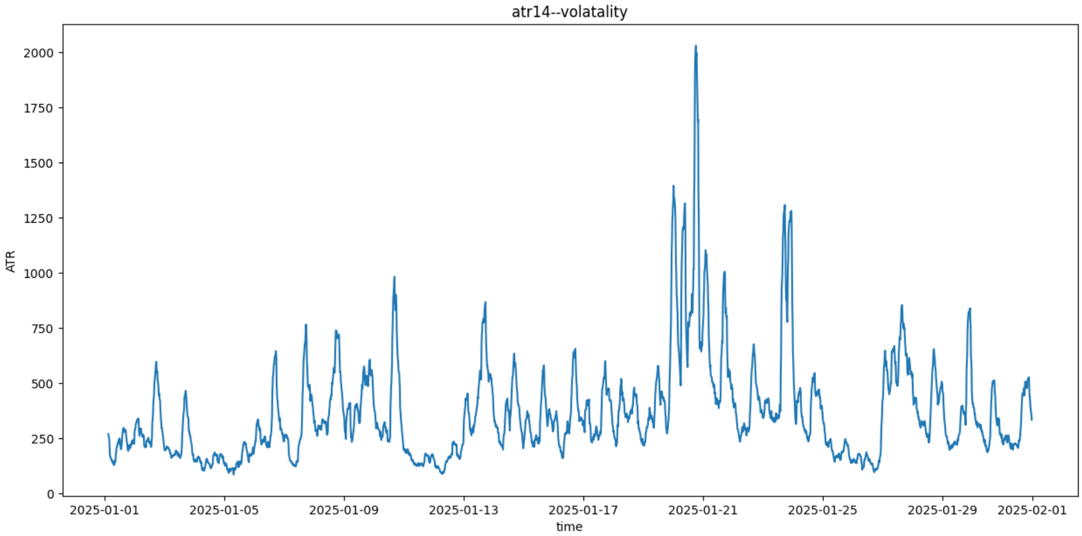
```
data['hl_range'] = data['high'] - data['low']
plt.figure(figsize=(15,7))
plt.plot(data['hl_range'], color='red')
plt.title('high-low range')
plt.xlabel('time')
plt.ylabel('price range(high-low)')
plt.show()
#large range --> high volatility small range -->low volatility small series then large--> potential breakout
```



Feature engineering

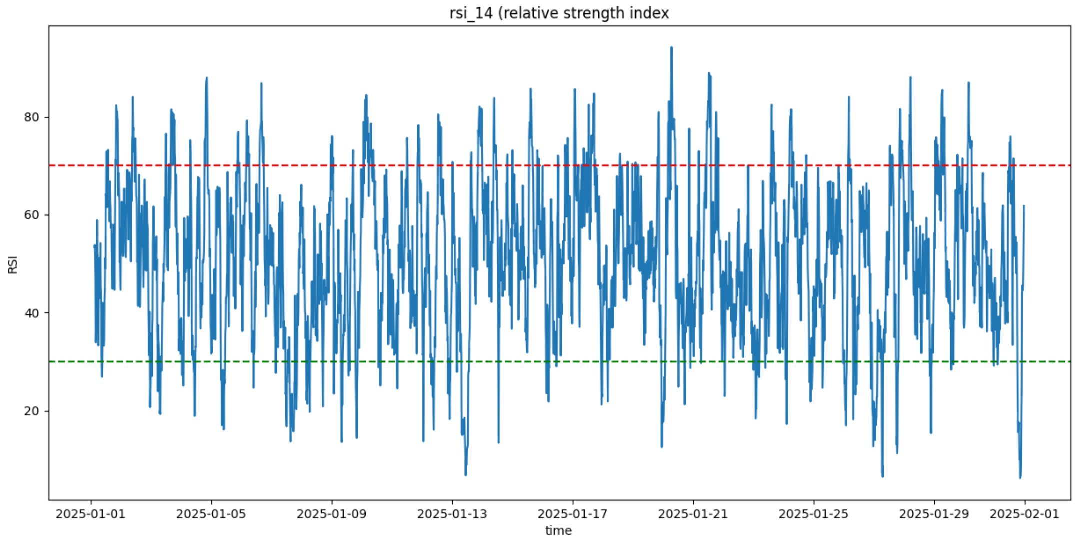
1. Average true range (ATR)

```python
#true range is how much price moved including any gaps 3 moments normal intraday movement , high - prev close, low - prev close
#then we take rolling mean of 14,20 whatever because we want not just volatility of one candle as it could have more noise
# rolling average if too smal then more noise and if too large then can miss changes

data['previous close'] =data['close'].shift(1)
data['tr1']= data['high']- data['low']
data['tr2']= data['high']- data['previous close']
data['tr3']= data['low']-data['previous close']
#true range
data['true_range']=data[['tr1', 'tr2', 'tr3']].max(axis=1)
#atr
data['atr14'] = data['true_range'].rolling(window=14).mean()


plt.figure(figsize=(15,7))
plt.plot(data['atr14'])
plt.title('atr14--volatality')
plt.ylabel('ATR')
plt.xlabel('time')
plt.show()
#When ATR is rising → volatility increasing → bigger price moves.
#When ATR is falling → volatility decreasing → smaller price moves → possible breakout coming.
```




```python
#RSI--> relative strength index how strong recent gains are vs recent losses--"is price moving up too fast or down too fast
#rsi>70 asset may be overbought --> price moved up too fast(may fall)
#rsi<30 asset may be oversold --> price fell too fast(may bounce up)
#can be used for entry exit choice
#calculation price change in two closes this is 'delta' then we define gain where delta >0 and loss where delta <0
#take average rolling mean of this as avg gain and loss and dividing these two we get relative strength

delta = data['close'].diff()
gain = delta.where(delta>0,0)
loss = -delta.where(delta<0,0)
avg_gain = gain.rolling(window=14).mean()
avg_loss = loss.rolling(window=14).mean()
rs= avg_gain / avg_loss

data['RSI_14']= 100 - (100 / (1+rs))

print (gain.head(20))
print(avg_gain.head(20))
print(rs.head(20))
plt.figure(figsize=(15,7))
plt.plot(data['RSI_14'])
plt.title('rsi_14 (relative strength index')
plt.xlabel('time')
plt.ylabel('RSI')
plt.axhline(70, color='red',  linestyle='--')
plt.axhline(30, color='green', linestyle='--')
plt.show()
```

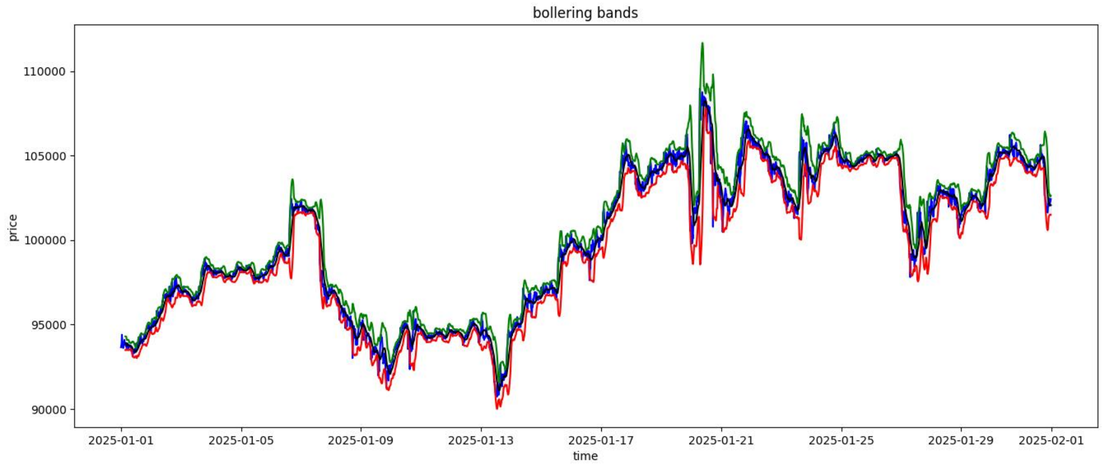
2.

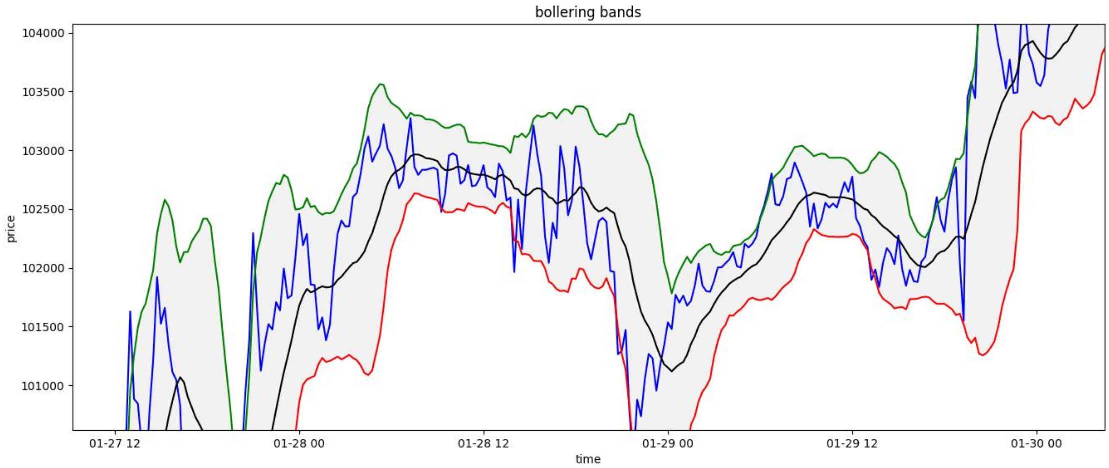


Here RSI below 30(green line) signals that price fell very fast and lot of sellers have already sold i.e a oversell situation. Price may bounce back here buyers may step in. at points where RSI is above 70 (red line) signals that price has risen very fast and has been overbought and now price may pull back . is RSI between 30 to 70 market is in normal zone above 50 buyers are strong and below 50 sellers are strong.

3.

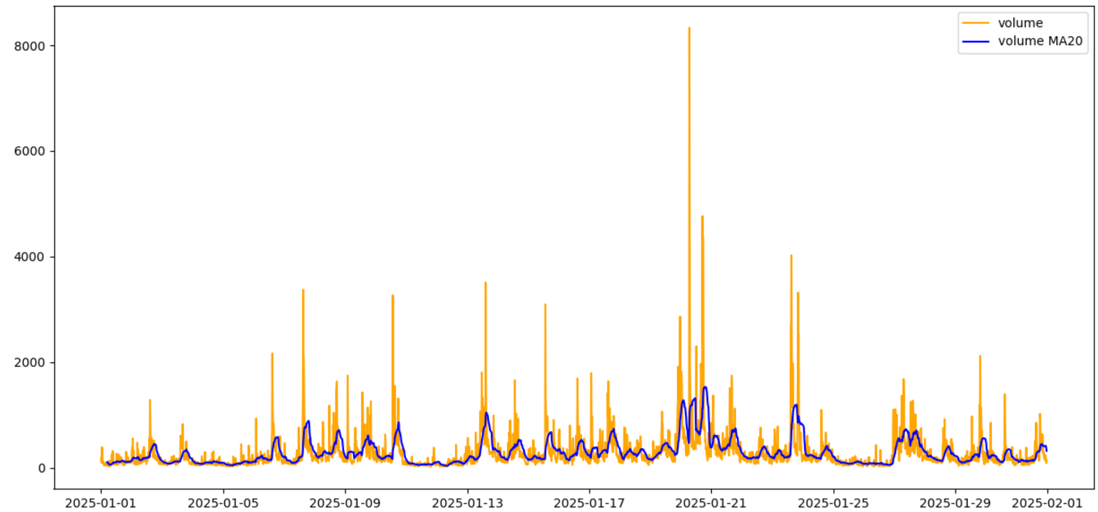
```
#bollinger bands --  price moves around a mean(avg) now when it moves way far from mean it means volatility has increased
#core idea is of normal distribution, when data is distributed normally 68%values lie in +- one std dev , 95% in +- 2 std devs
#and 99.7% in 3 std devs so in bollinger bands we use 2 std devs to as we assume that under normal circumstances 95% of prices
# should stay within the bands so we take a middle band which could be SMA 20 (usual because 20 day trading month) and then
# upper band 2 std devs from middle and lower -2 std devs from middle
# narrow band imply that std dev is low --> stable price --> low volatility
# wide bands --> high std dev --> breakout or trend (volatile)
#price hitting upper band --> relatively high price and may break out or will revert
# price hitting lower band --> relatively low pricee and may rise or break down further

data['middle_band'] = data['close'].rolling(window=15).mean()
data['std_dev']= data['close'].rolling(window=15).std()
data['upper_band'] = data['middle_band'] + (2 * data['std_dev'])
data['lower_band'] = data['middle_band'] - (2 * data['std_dev'])

plt.figure(figsize=(15,7))
plt.plot(data['close'],label='close price', color= 'blue')
plt.plot(data['middle_band'], label='middle band (sma 15)', color= 'black')
plt.plot(data['lower_band'], label='lower band', color='red')
plt.plot(data['upper_band'], label='upper band', color='green')
plt.fill_between(data.index, data['lower_band'], data['upper_band'], color='grey', alpha=0.1)
plt.title('bollering bands')
plt.xlabel('time')
plt.ylabel('price')
plt.show()

# interpretation is done with rsi combined RSI<30 and price near lower band --> high buy probability setup
# RSI > 70 price is near upper band --> high probability sell setup
```





4.

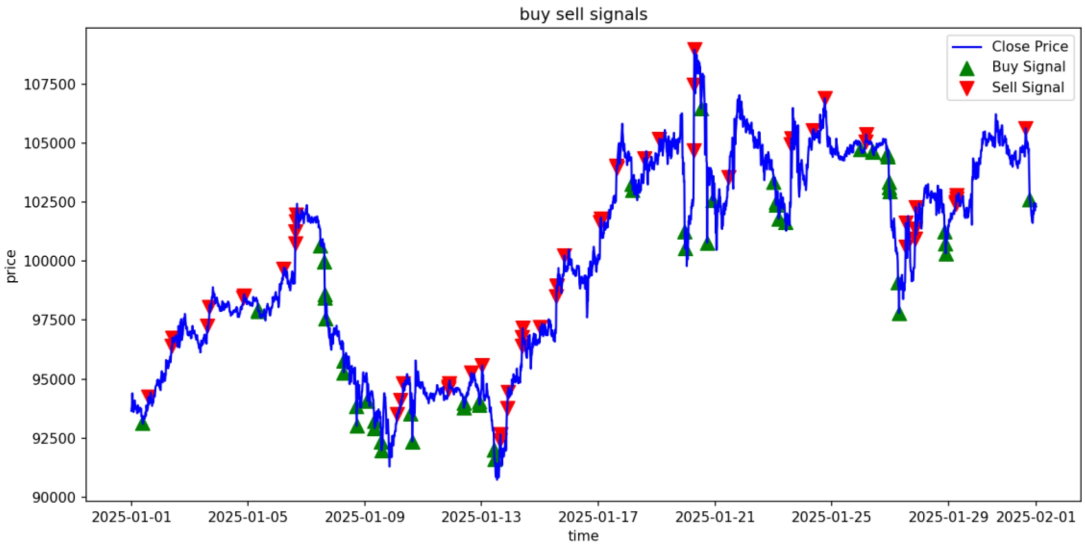
```
#volume moving average --> more than normal -> confirms breakout
#lower than normal -> signals weak moves

data['volume_ma20']= data ['volume'].rolling(window=20).mean()

plt.figure(figsize=(15,7))
plt.plot(data['volume'],label='volume', color='orange')
plt.plot(data['volume_ma20'],label='volume MA20', color='blue')
plt.title= ('volume and molume MA20')
plt.ylabel=('volume')
plt.xlabel=('time')
plt.legend()
plt.show()
```



Orange line → actual volume (spiky, irregular).
Blue line → 20-period moving average of volume → smooths it → shows trend.
When **orange line > blue line** → volume is **above average** → high participation → confirms moves.
When **orange line < blue line** → low participation → weak moves → often false breakouts.
If **price breaks upper Bollinger Band + volume spike → confirmed breakout → enter trade**.
If **price moves without volume** → be cautious → move may not sustain.
If **volume MA rising** → market is waking up → good for trend trading.
If **volume MA falling** → market is becoming quiet → prepare for consolidation or wait.

Pattern and signal detection

1.
```
#pattern and signal detection
# Buy Signal → RSI < 30 and Close < Lower Band
data['buy_signal'] = ((data['RSI_14']<30) & (data['close'] < data['lower_band'])).astype(int)
data['sell_signal'] = ((data['RSI_14']>70) & (data['close'] > data['upper_band'])).astype(int)
#1 in buy is buy 0 dont buy 1 in sell is sell and 0 dont sell
plt.figure(figsize=(15,7))
plt.plot(data['close'], label='Close Price', color='blue')
#base plot^
# Plot buy signals
plt.scatter(data.index, data['close'].where(data['buy_signal'] == 1), label='Buy Signal', marker='^', color='green', s=100)

# Plot sell signals
plt.scatter(data.index, data['close'].where(data['sell_signal'] == 1), label='Sell Signal', marker='v', color='red', s=100)
plt.title('buy sell signals')
plt.xlabel('time')
plt.ylabel('price')
plt.legend()
plt.show()
```

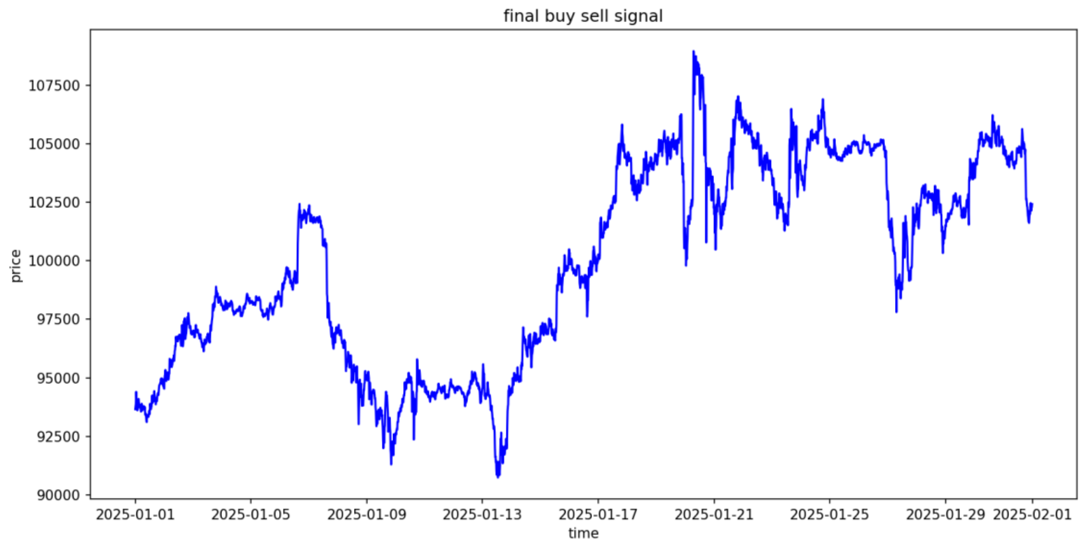


A **Buy signal** is triggered when **RSI < 30** (indicating strong selling pressure) *and* price closes **below the lower Bollinger Band** (showing it is far below its recent average).
A **Sell signal** is triggered when **RSI > 70** (strong buying pressure) *and* price closes **above the upper Bollinger Band** (far above its recent average).
We see **green triangles** (Buy signals) where the price is low and momentum is oversold. **Red triangles** (Sell signals) appear where the price is high and momentum is overbought.

2. Buy signal:-
   RSI < 30
   AND Close < Lower Band
   AND ATR > ATR_MA → market is moving
   AND EMA5 > SMA20 → small uptrend building
   AND Volume > Volume MA → smart money participating
   Sell signal:-
   RSI > 70
   AND Close > Upper Band
   AND ATR > ATR_MA
   AND EMA5 < SMA20 → downtrend building
   AND Volume > Volume MA

```python
#creating filters
data['atr_MA_50'] = data['atr14'].rolling(window=50).mean()
data['EMA_5'] = data['close'].ewm(span=5,adjust=False).mean()
data['SMA_20'] = data['close'].rolling(window=20).mean()

data['final_buy_signal'] = (
    (data['RSI_14']<30)&
    (data['close']<data['lower_band'])&
    (data['atr14']>data['atr_MA_50'])&
    (data['EMA_5']>data['SMA_20'])&
    (data['volume']>data['volume_ma20'])
).astype(int)

data['final_sell_signal'] = (
    (data['RSI_14']>70)&
    (data['close']>data['upper_band'])&
    (data['atr14']>data['atr_MA_50'])&
    (data['EMA_5']<data['SMA_20'])&
    (data['volume']>data['volume_ma20'])
).astype(int)
```

```python
plt.figure(figsize=(15,7))
plt.plot(data['close'], label='close price', color='blue')
plt.scatter(data.index,data['close'].where(data['final_buy_signal']==1), label='buy signal', marker='^',color='green',s=100)
plt.scatter(data.index,data['close'].where(data['final_sell_signal']==1), label='sell signal', marker='v',color='red',s=100)
plt.title('final buy sell signal')
plt.xlabel('time')
plt.ylabel('price')
plt.show()
```



No signals, we loosen the filters here

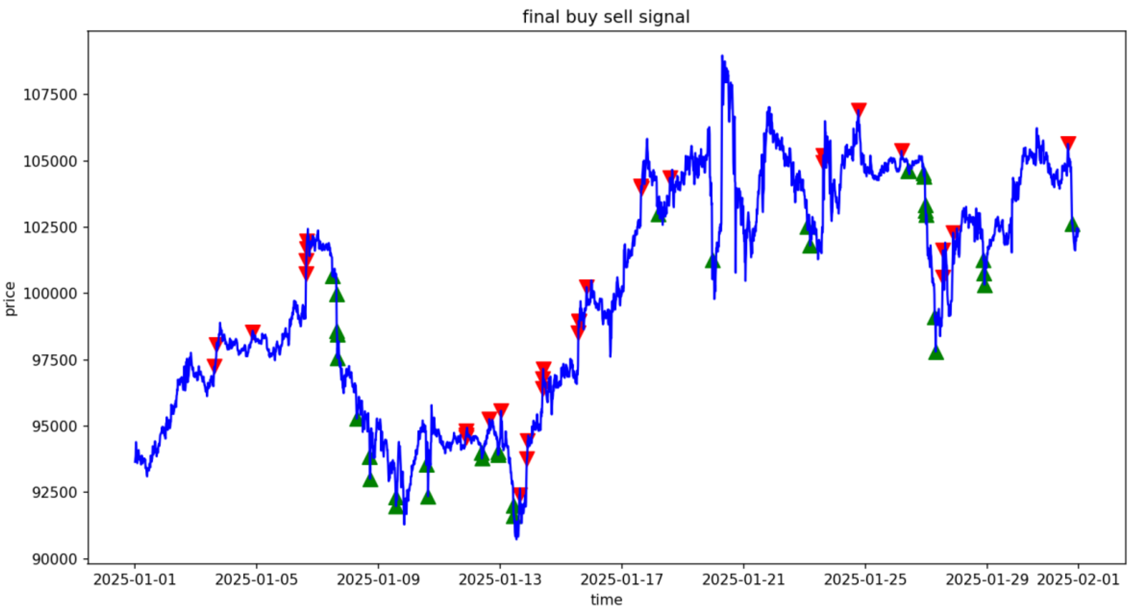
Changes done

1. Considering only 90% of ATR_MA

2. 2% tolerance in EMA SMA

3. 5% tolerance in volume

```python
data['final_buy_signal'] = (
    (data['RSI_14']<30)&
    (data['close']<data['lower_band'])&
    (data['atr14']>0.9 * data['atr_MA_50'])&
    (data['EMA_5']>0.98 * data['SMA_20'])&
    (data['volume']>0.95 * data['volume_ma20'])
).astype(int)

data['final_sell_signal'] = (
    (data['RSI_14']>70)&
    (data['close']>data['upper_band'])&
    (data['atr14']>0.9 * data['atr_MA_50'])&
    (data['EMA_5']<1.02 * data['SMA_20'])&
    (data['volume']>0.95 * data['volume_ma20'])
).astype(int)
```

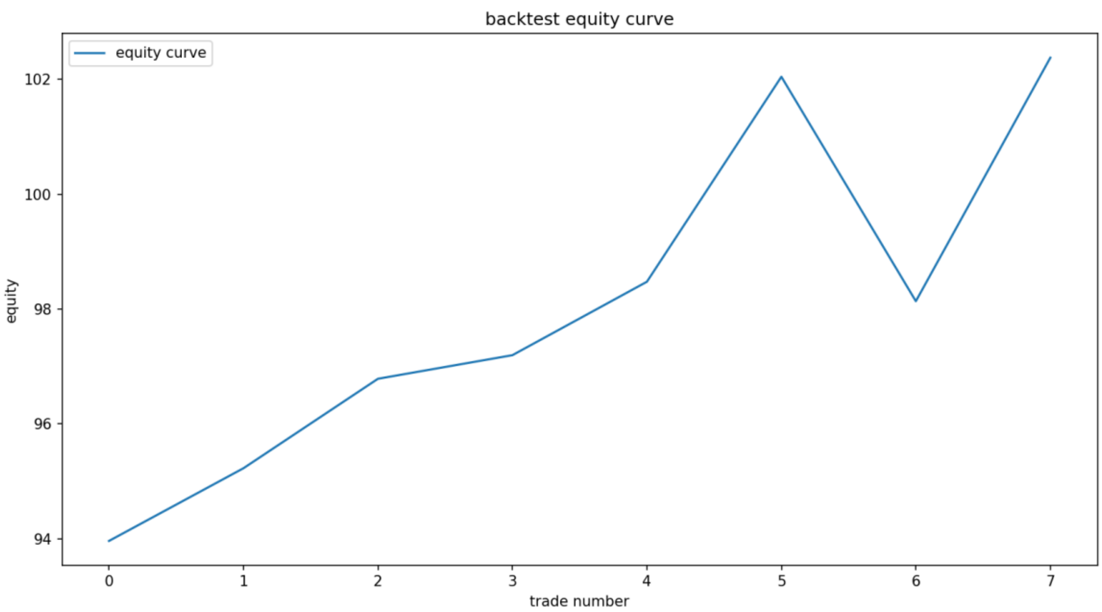


Back testing

First test

```python
#backtesting
#initializing variables
position = 0 # 0 = no position, 1 = long
entry_price = 0
pnl=[]
position_value=[]

#loop through data
for index, row in data.iterrows():
    if position ==0 and row['final_buy_signal']==1:
        position=1
        entry_price=row['close']

    elif position==1 and row['final_sell_signal']==1:
        position=0
        exit_price=row['close']
        trade_return= (exit_price - entry_price) / entry_price
        pnl.append((trade_return))
        position_value.append(exit_price)
```

```python
if pnl:
    cumulative_return = (1 + pd.Series(pnl)).prod() - 1
else:
    cumulative_return = 0

print(f"number of trades: {len(pnl)}")
print(f"cumulative return: {cumulative_return:.2%}")

equity_curve = (pd.Series(pnl).add(1).cumprod())*100

plt.figure(figsize=(15,7))
plt.plot(equity_curve,label='equity curve')
plt.title('backtest equity curve')
plt.xlabel('trade number')
plt.ylabel('equity')
plt.legend()
plt.show()
```



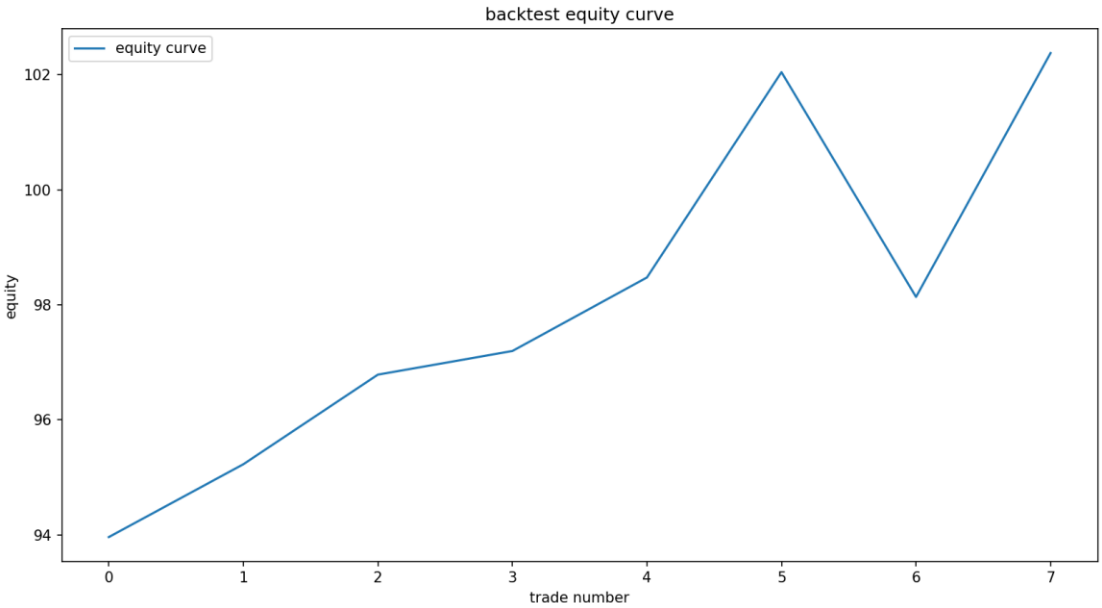
Number of trades = 8

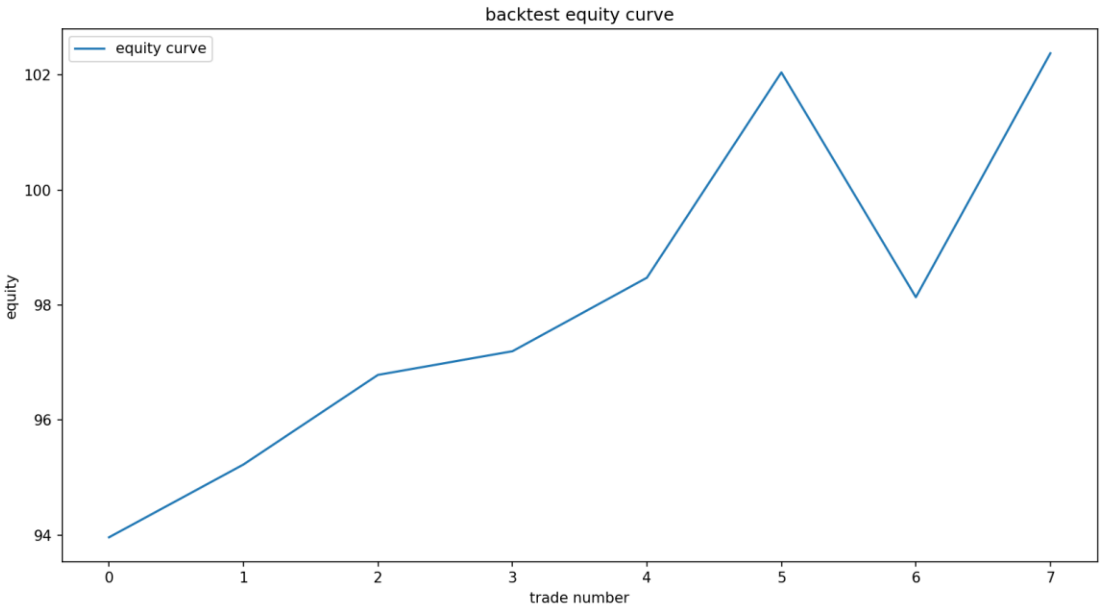
Good number not too high not too low means filters are in right zone

Cumulative profit=2.38%

Positive money making model without any tuning

Equity curve is also stedily upward sloping no extreme zigzag movement final value is higher than starting point (one dip only → normal) no huge drawdown model is not extremely volatile(no extreme risk)





```python
#win rate
returns = np.array(pnl)
transaction_cost= 0.001
win_rate = np.sum(returns>0)/len(returns)
avg_return =  np.mean(returns)
max_gain = np.max(returns)
max_loss = np.min(returns)
sharpe_ratio = np.mean(returns)/np.std(returns)
returns_after_cost = returns - transaction_cost
cum_return_after_cost= np.prod(1+returns_after_cost)-1

print(f"total trades: {len(returns)}")
print(f"win rate: {win_rate:.2%}")
print(f"average return per trade:{avg_return:.2%}")
print(f"max gain: {max_gain: .2%}")
print(f"max loss: {max_loss: .2%}")
print(f"sharpe ratio : {sharpe_ratio:.2f}")
print(f"cumulative return after cost: {cum_return_after_cost: .2%}")
```

```
number of trades: 8
cumulative return: 2.38%
total trades: 8
win rate: 75.00%
average return per trade:0.35%
max gain:  4.32%
max loss: -6.04%
sharpe ratio : 0.11
cumulative return after cost:  1.56%

Process finished with exit code 0
```

Conclusion :

What we did

1. Data indicators:-
    a. RSI (14) --- to detect overbought and oversold conditions
    b. Bollinger bands --- to detect volatility extremes
    c. ATR --- measures volatility magnitude
    d. EMA 5 / SMA 20 --- to catch trend direction
    e. Volume MA ---- volume confirmation for smart money activity

2. Signal logic
    a. Final buy signal
    b. Final sell signal
       Binary triggers (0 or 1)
3. Backtesting
    a. Check row-by-row
    b. Buy if not holding + buy signal
    c. Sell if holding + sell signal
    d. Calculate return on each trade
    e. Track those returns
       Gives:
        I. Trade count
        II. Individual % returns
        III. Final equity curve
4. Metrics generated
    a. Number of trades : 8
    b. Cumulative return 2.38%
    c. Win rate 75.00% (6 out of 8 trades profitable)
    d. Average return per trade 0.35% (includes both wins and losses)
    e. Max gain – 4.32%
    f. Max loss - -6.04%
    g. Sharpe ratio 0.11
    h. Cumulative return with 0.1% transaction cost – 1.56%

February

```
cumulative return: -8.14%
total trades: 7
win rate: 71.43%
average return per trade:-1.05%
max gain:  4.33%
max loss: -13.23%
sharpe ratio : -0.20
cumulative return after cost: -8.79%

Process finished with exit code 0
```

March

```
number of trades: 9
cumulative return: 6.69%
total trades: 9
win rate: 77.78%
average return per trade:0.76%
max gain:  4.96%
max loss: -4.69%
sharpe ratio : 0.27
cumulative return after cost:  5.74%

Process finished with exit code 0
```

April

```
number of trades: 7
cumulative return: 8.57%
total trades: 7
win rate: 85.71%
average return per trade:1.19%
max gain:  3.59%
max loss: -2.08%
sharpe ratio : 0.73
cumulative return after cost:  7.82%

Process finished with exit code 0
```

Stop loss addition (3%)

```
entry_price = 0
pnl=[]
position_value=[]
buy_marker = []
sell_marker=[]
stop_loss_pct = 0.03
#loop through data
for index, row in data.iterrows():
    if position == 0 and row['final_buy_signal']==1 and row['recent_returns'] > -0.04:
        position = 1
        entry_price=row['close']
        entry_index = index #for future ploting
        buy_marker.append((index,row['close'])) #stores buy
    elif position == 1:
        price_change = (row['close']- entry_price)/entry_price
        #while in position
        if price_change < -stop_loss_pct:
            position = 0
            pnl.append(price_change) #record loss
            position_value.append(row['close'])
            sell_marker.append((index, row['close']))

```

```python
elif row['final_sell_signal'] == 1:
    position = 0
    pnl.append(price_change)
    position_value.append(row['close'])
    sell_marker.append((index,row['close']))
```

No buying in large down trend

```python
#creating filters
data['atr_MA_50'] = data['atr14'].rolling(window=50).mean()
data['EMA_5'] = data['close'].ewm(span=5,adjust=False).mean()
data['SMA_20'] = data['close'].rolling(window=20).mean()
data['recent_returns'] = data['close'].pct_change(periods=20)#percentage cange in 20 candles
data['final_buy_signal'] = (
    (data['RSI_14']<30)&
    (data['close']<data['lower_band'])&
    (data['atr14']>0.9 * data['atr_MA_50'])&
    (data['EMA_5']>0.98 * data['SMA_20'])&
    (data['volume']>0.95 * data['volume_ma20'])&
    (data['recent_returns']>-0.02) #new condition
).astype(int)
```

After stop loss results

January

```
number of trades: 11
cumulative return: 7.70%
total trades: 11
win rate: 72.73%
average return per trade:0.71%
max gain:  4.32%
max loss: -3.10%
sharpe ratio : 0.28
cumulative return after cost:  6.53%

Process finished with exit code 0
```

February

```
number of trades: 13
cumulative return: -1.50%
total trades: 13
win rate: 53.85%
average return per trade:-0.06%
max gain:  6.15%
max loss: -3.65%
sharpe ratio : -0.02
cumulative return after cost: -2.77%

Process finished with exit code 0
```

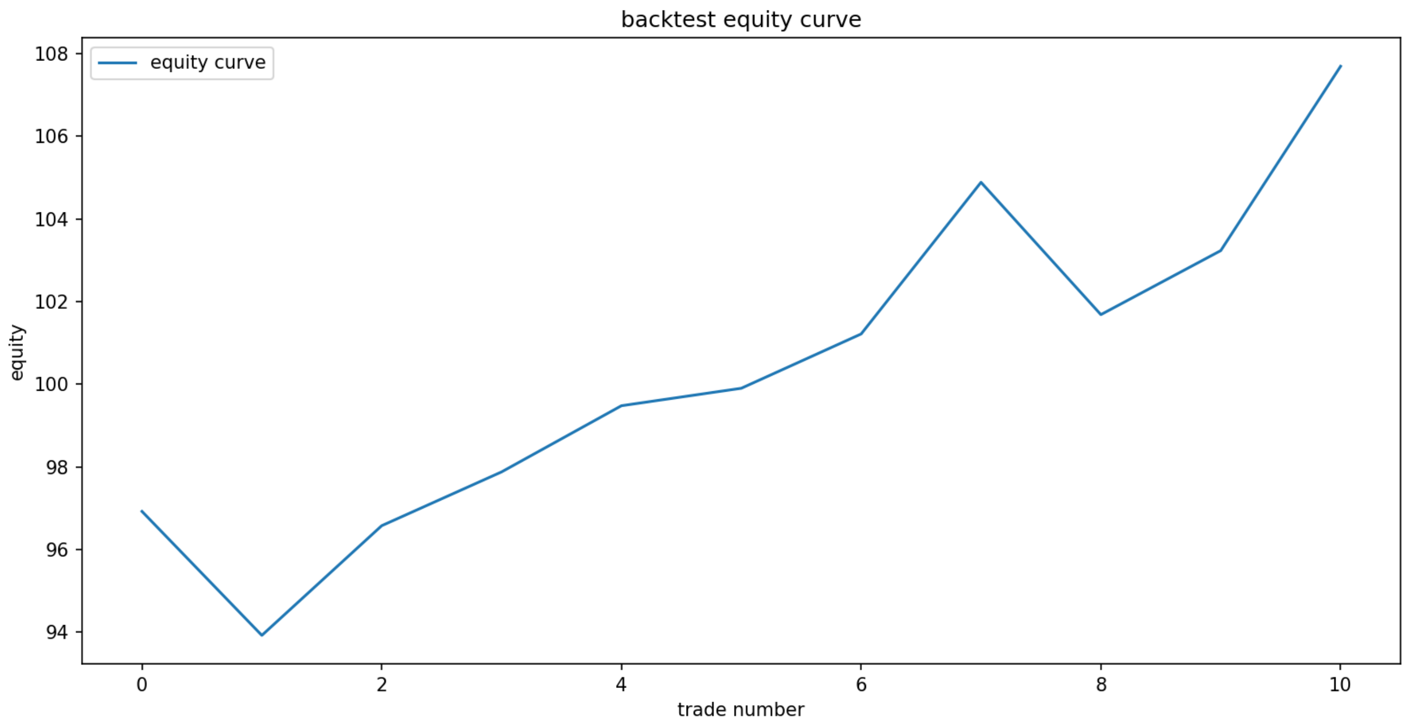
March

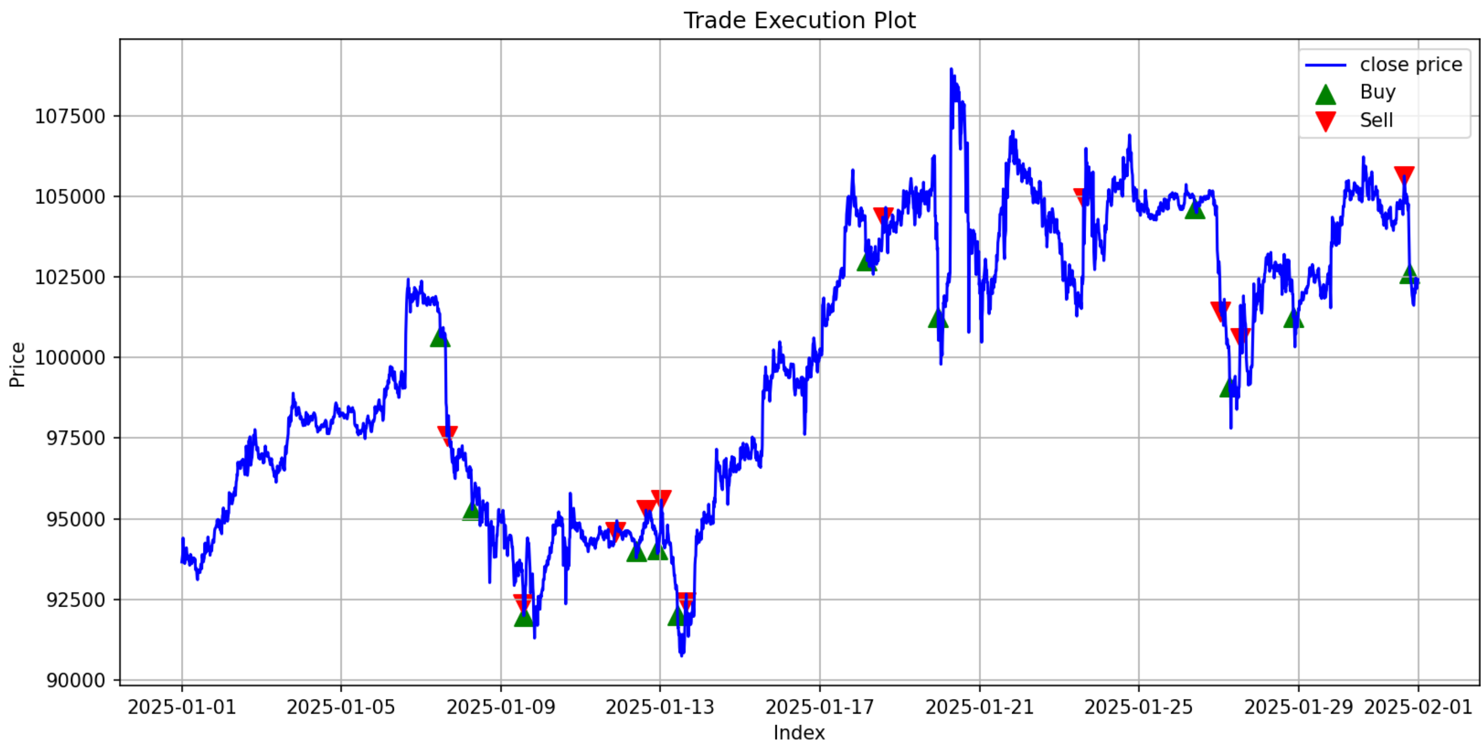
```
number of trades: 13
cumulative return: 12.37%
total trades: 13
win rate: 61.54%
average return per trade:0.97%
max gain:  7.91%
max loss: -4.11%
sharpe ratio : 0.27
cumulative return after cost:  10.93%

Process finished with exit code 0
```

April

```
number of trades: 9
cumulative return: 10.45%
total trades: 9
win rate: 77.78%
average return per trade:1.16%
max gain:  7.31%
max loss: -3.86%
sharpe ratio : 0.36
cumulative return after cost:  9.47%

Process finished with exit code 0
```

Trade plot


backtest equity curve
equity curve
108
106
104
102
100
98
96
94
equity
0
2
4
6
8
10
trade number


Trade Execution Plot
close price
Buy
Sell
107500
105000
102500
100000
97500
95000
92500
90000
Price
2025-01-01
2025-01-05
2025-01-09
2025-01-13
2025-01-17
2025-01-21
2025-01-25
2025-01-29
2025-02-01
Index