Trading Task

Task 1

Technical Analysis

For trading instrument I have chosen BTC USDT which is well known pair for its liquidity and reliability in crypto world. This pair has good volume and can be easily traded for other crypto coins and exchanged for fiat.

All trading data was taken from Binance

I have analysed BTC USDT pair by building closing, opening price and simple moving average graphs over 30 day period from @January 25, 2024 to @February 23, 2024. This has helped to understand whether BTC USDT is experiencing rising or declining trend. In addition, standard deviation was plotted to understand suitable trading period and max price decline that user can experience during trading period.

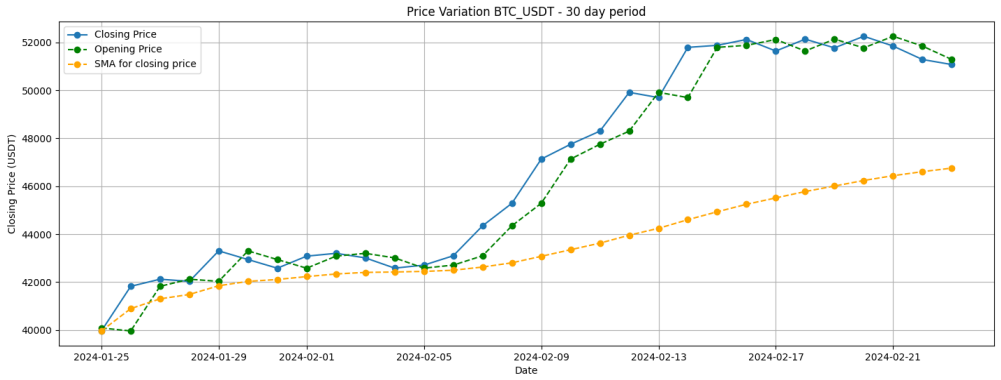

Figure 1.1 - shows closing, opening and simple moving average for BTC USDT from 01-25 to 02-23 for 2024

In addition, standard deviation graph for BTC USDT pair was plotted. For each day from @January 25, 2024 to @February 23, 2024 I have calculated standard deviation by analysing 1 hour candles

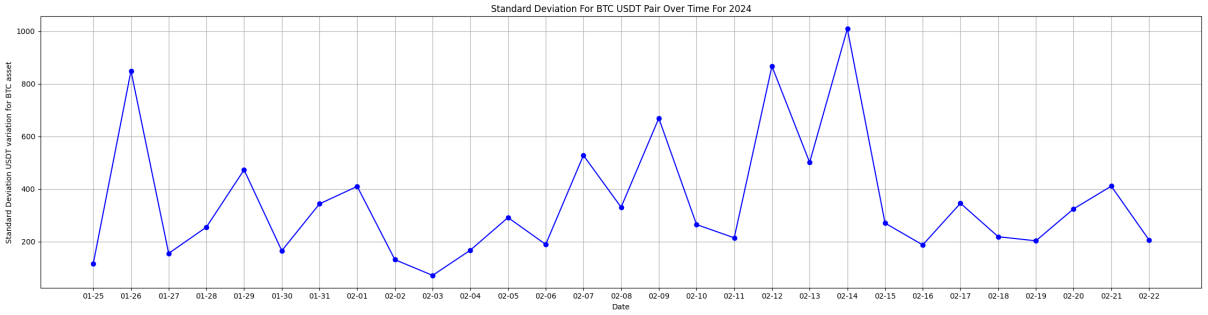

Figure 1.2 - shows standard deviation for BTC USDT from 2024-01-25 to 2024-02-22

From Figure 1.1 can be concluded that BTC USDT pair is experiencing uprising trend. Quite frequently closing price is higher then opening price. As well as, closing price gradually increases over 30 day period. Moreover, SMA (simple moving average - orange line) confirms this uprising trend by having a positive gradient.

From Figure 1.2 shows that price for BTC varies from 71 - 1000 USDT from @January 25, 2024 to @February 23, 2024 respectively.

```
standard_deviation = [
116.83, 849.42, 155.57, 255.26, 473.17, 165.61,
343.77, 410.19, 131.59, 71.83, 167.67, 291.77,
189.58, 528.15, 331.02, 669.18, 265.12, 214.67,
867.06, 501.24, 1009.12, 270.56, 187.68, 346.15,
218.94, 203.21, 324.67, 411.15, 206.93]
```

Figure 1.3 - shows standard deviation data points for BTC USDT from 01-25 to 02-22 used to plot Figure 1.2

Hence, it can be seen that 1 day trading timeframe gives a chance to enter the start price successfully where later on BTC asset can be sold at higher price. In addition, maximum percentage stop loss is approximately 2.1% according to standard deviation data from Figure 1.2 (if calculating it at @February 14, 2024).

$$\text{Maximum Stop Loss Percentage} = \frac{\text{Standard Deviation}}{\text{Bitcoin Price}} = 100 \times \frac{1,009}{49,000} \approx 2.1$$

Following code was used to plot graph in Figure 1.1

```
from binance import Client
import pandas as pd
import matplotlib.pyplot as plt

API_KEY = "PASTE_YOUR_KEY"
API_SECRET = "PASTE_YOUR_SECRET"

def calculate_simple_moving_average():
    current_moving_average = []
    total_sum = 0
    ct = 0
    for el in df['close']:
        total_sum += el
        ct += 1
        curr_average = round(total_sum / ct, 2)
        current_moving_average.append(curr_average)
    return current_moving_average

client = Client(API_KEY, API_SECRET)

klines_1_day = client.get_historical_klines("BTCUSDT", Client.KLIN
                                            limit=30, end_str="202
klines_1_day = [el[:6] for el in klines_1_day]
df = pd.DataFrame(klines_1_day, columns=['timestamp', 'open', 'hig
df['open'] = pd.to_numeric(df['open'])
df['close'] = pd.to_numeric(df['close'])
df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
df.set_index('timestamp', inplace=True)
moving_average_arr = calculate_simple_moving_average()
df['moving_average'] = moving_average_arr

plt.figure(figsize=(25, 6))
plt.plot(df['close'], label='Closing Price', marker='o')
plt.plot(df['open'], label='Opening Price', marker='o', linestyle=
plt.plot(df['moving_average'], label=f'SMA for closing price', lin
plt.title(f'Price Variation BTC-USDT - 30 day period')
plt.xlabel('Date')
```

```
plt.ylabel('Closing Price (USDT)')
plt.legend()
plt.grid(True)
plt.show()
```

Following code was used to plot Standard Deviation Graph in Figure 1.2

```
from binance import Client
from datetime import datetime, timedelta
import matplotlib.pyplot as plt

CLOSE_PRICE_IDX = 4
START_DATE_STR = '2024-01-25'
END_DATE_STR = '2024-02-23'
API_KEY = "PASTE_YOUR_KEY"
API_SECRET = "PASTE_YOUR_SECRET"


def calculate_closing_price_std(current_closing_price):
    total = 0
    ct = 0
    for closing_price in current_closing_price:
        closing_price = float(closing_price)
        ct += 1
        total += closing_price
    closing_price_day_average = total / ct
    total_square_distance_from_mean = 0
    for el in current_closing_price:
        el = float(el)
        total_square_distance_from_mean += (el - closing_price_day
    return round((total_square_distance_from_mean / ct) ** 0.5, 2)


client = Client(API_KEY, API_SECRET)

start_date = datetime.strptime(START_DATE_STR, '%Y-%m-%d')
end_date = datetime.strptime(END_DATE_STR, '%Y-%m-%d')

expected_std_res = [116.83, 849.42, 155.57, 255.26, 473.17, 165.61
                    189.58, 528.15, 331.02, 669.18, 265.12, 214.67
                    346.15, 218.94, 203.21, 324.67, 411.15, 206.93
```

```
date_list = [(start_date + timedelta(days=x)).strftime('%m-%d') fo

standard_deviation_res = []
for i in range(1, len(date_list)):
    start_date = date_list[i - 1]
    end_date = date_list[i]
    klines_1_hour = client.get_historical_klines("BTCUSDT", Client
                                                 end_str=end_date)
    closing_price_arr = [el[CLOSE_PRICE_IDX] for el in klines_1_ho
    standard_deviation_res.append(calculate_closing_price_std(clos
date_list.pop()

plt.figure(figsize=(25, 6))
plt.plot(date_list, standard_deviation_res, marker='o', linestyle=
plt.title('Standard Deviation For BTC-USDT Pair Over Time For 2024
plt.xlabel('Date')
plt.ylabel('Standard Deviation USDT variation for BTC asset')
plt.tight_layout()
plt.grid(True)
plt.show()
```

Algorithm

Since Standard Deviation from Figure 1.2 shows that maximum stop loss can not be larger than 2.1%, so I do not expect starting price to follow negative Arithmetic Sequence at any point in time.

$$a_{\text{entring\_price}} = a_{\text{starting\_price}} - (n-1) \cdot d$$

In my opinion, from the information that we currently have in Figure 1.1 and 1.2, price would rather follow this classic positive Arithmetic Sequence

$$a_{\text{entring\_price}} = a_{\text{starting\_price}} + (n-1) \cdot d$$

Hence, in my opinion making algorithm that expects price to decrease several times as described in the Task Part 1 is not suitable for BTC USDT pair for the current trading period.

To my mind, according to Figure 1.1 and 1.2 better approach is to buy BTC whenever our capital is larger then zero. Since, there is a high chance that BTC is going to increase in price. When sell BTC when it’s exit price is larger then 4% compared to entering price.

If price for BTC decreases by more then 2.1% (maximum stop loss) - this is anomaly according to our data. To avoid any further potential loses, BTC should be sold whenever this threshold is reached. As per trading timeframe, one day should be sufficient since price variation during one day from Figure 1.2 is large enough.

Below is the trading algorithm pseudo code. I have assumed that 1 USDT = 1 USA dollar

```
TOTAL_CAPITAL = 15_000$
CAPITAL_PORTION = 1000$
current_order_queue = queue()
MAXIMUM_STOP_LOSS_PERCENTAGE = 2.1%
THRESHOLD_PERCENTAGE_PROFIT = 4%
for open_close_price in 1_day_price_candles_btc_usdt:
    open_price = open_close_price[OPEN_PRICE]
    close_price = open_close_price[CLOSE_PRICE]
    timestamp = open_close_price[TIMESTAMP]

        if current_order_queue.notEmpty():
       oldest_order = current_order_queue.first()
       price_percentage_difference = calc_percent(close_price - ol
       if price_percentage_difference < 0 and  |price_percentage_d
          usdt_returned = oldest_order.sell()
          current_order_queue.removeFirst()
          TOTAL_CAPITAL += usdt_returned
       if price_percentage_difference > 0 and price_percentage_dif
          usdt_returned = oldest_order.sell()
          current_order_queue.removeFirst()
          TOTAL_CAPITAL += usdt_returned

    if capital_at_the_end > 0:
        one_dollar_btc_amount = 1 / open_price
        amount_of_btc_bought = CAPITAL_PORTION * one_dollar_btc_am
        buying_order = Order(open_price, timestamp, amount_of_btc_
        TOTAL_CAPITAL -= CAPITAL_PORTION

 exist for loop and convert btc asset held to usdt amount to calcu
 total_deposit for metrics output
 while current_order_queue.notEmpty():
       oldest_order = current_order_queue.removeFirst()
```

```
        TOTAL_CAPITAL += oldest_order.buying_price() * oldest_order
return TOTAL_CAPITAL
```

As per state data structure I have chosen queue. It allows to store orders state in FIFO way which is suitable for tracking stop loss percentage to sell not successful asset if needed. FIFO queue allows to remove not successful order at O(1) time complexity and add another order at O(1) time complexity. Memory Complexity is O(n) where n is the number of orders execute.

Task 2

Part1 - Algorithm, Trading Statistics

From Task 1 it was stated that suitable trading period is 1 day.

Algorithm was tested on Binance Exchange for BTC USDT pair on historical data from @January 25, 2024 to @February 23, 2024.

Below is the actual algorithm implementation

```
from binance import Client
from datetime import datetime
from Order import Order
import matplotlib.pyplot as plt
import pandas as pd
from collections import deque

START_DATE_STR = '2024-01-25'
END_DATE_STR = '2024-02-24'
API_KEY = "PASTE_API_KEY"
API_SECRET = "PASTE_API_SECRET"
TIME_STAMP_IDX = 0
OPEN_PRICE_IDX = 1
CLOSE_PRICE_IDX = 4

INITIAL_CAPITAL_USDT = 15_000
MAXIMUM_PERCENTAGE_DECLINE = 2.1
THRESHOLD_PERCENTAGE_PROFIT = 4
TIMESTAMP_KEY = 'TIMESTAMP'
OPEN_PRICE_KEY = 'OPEN'
CLOSE_PRICE_KEY = 'CLOSE'
```

```
client = Client(API_KEY, API_SECRET)

klines_1_day = client.get_historical_klines("BTCUSDT", Client.KLIN
                                            start_str=START_DATE_S


def plot_net_gain_graph(net_gain_array, timestamp_arr):
    df = pd.DataFrame()
    df['net_gain'] = net_gain_array
    df['timestamp'] = timestamp_arr
    df.set_index('timestamp', inplace=True)
    plt.figure(figsize=(20, 6))
    plt.xticks(rotation=45)
    plt.plot(df['net_gain'], label='User Net Gain in USA dollar', 
    plt.title(f'Net Gain in USA Dollar - 30 day period, Year 2024'
    plt.xlabel('Date')
    plt.ylabel('Net Gain In Dollar')
    plt.legend()
    plt.grid(True)
    plt.show()


def execute_oldest_order(capital_at_the_end, capital_portion, clos
                         tail_order):
    capital_portion_with_gain = tail_order.amount_of_asset_bought 
    tail_order.execute(close_price, current_timestamp)
    net_gain += (tail_order.amount_of_asset_bought * close_price -
    orders_queue.popleft()
    capital_at_the_end += capital_portion_with_gain
    return capital_at_the_end, net_gain


def trading_session(price_stats, num_prices, initial_capital, stop_
    orders_queue = deque()
    net_gain = 0
    capital_per_trade = 1000
    capital_at_the_end = initial_capital
    num_profitable_trades = 0
    num_losing_trades = 0
    net_gain_array = []
    timestamp_arr = []
    maximum_stop_loss_percentage = float('inf')
```

```
    for idx in range(num_prices):
        current_timestamp = price_stats[TIMESTAMP_KEY][idx]
        open_price = price_stats[OPEN_PRICE_KEY][idx]

        close_price = price_stats[CLOSE_PRICE_KEY][idx]

        if len(orders_queue) > 0:
            tail_order = orders_queue[0]
            order_percentage_difference = (close_price - tail_orde
            maximum_stop_loss_percentage = min(maximum_stop_loss_p
            if order_percentage_difference < 0 and abs(order_perce
                capital_at_the_end, net_gain = execute_oldest_orde



                num_losing_trades += 1
            elif order_percentage_difference > 0 and order_percenta
                capital_at_the_end, net_gain = execute_oldest_orde



                num_profitable_trades += 1
        if capital_at_the_end > 0:
            one_usdt_asset_amount = 1 / open_price
            orders_queue.append(Order(open_price, current_timestam
            capital_at_the_end -= capital_per_trade
        net_gain_array.append(net_gain)
        timestamp_arr.append(current_timestamp)
    capital_at_the_end += sum([order.buying_price * order.amount_o
    percentage_profit = round((net_gain * 100 / initial_capital), 
    maximum_stop_loss_percentage_response = maximum_stop_loss_perc
    if maximum_stop_loss_percentage == float('inf'):
        maximum_stop_loss_percentage_response = 'N/A'
    return (f'✅Finished!\n'
            f'Initial capital = {round(initial_capital, 2)}$\n'
            f'Net gain = {round(net_gain, 2)}$\n'
            f'Capital at the end = {round(capital_at_the_end, 2)}$
            f'Capital percentage profit = {percentage_profit}%\n'
            f'Number of profitable trades = {num_profitable_trades
            f'Number of losing trades = {num_losing_trades}'
            f'Maximum Stop Loss Percentage Reached = {maximum_stop_



price_info = {
```

```
    OPEN_PRICE_KEY: [],
    CLOSE_PRICE_KEY: [],
    TIMESTAMP_KEY: []
}
for kline in klines_1_day:
    timestamp = kline[TIME_STAMP_IDX] / 1000
    utc_date = datetime.utcfromtimestamp(timestamp).strftime('%m-%
    price_info[OPEN_PRICE_KEY].append(float(kline[OPEN_PRICE_IDX])
    price_info[CLOSE_PRICE_KEY].append(float(kline[CLOSE_PRICE_IDX
    price_info[TIMESTAMP_KEY].append(utc_date)

financial_information, capital_gain_arr = trading_session(price_st
                                                          num_pric
                                                          initial_
                                                          stop_los


plot_net_gain_graph(capital_gain_arr, price_info[TIMESTAMP_KEY])
```

Figure 2.1 - shows algorithm code that was tested on BTC USDT historical data

After algorithm execution following statistics is shown

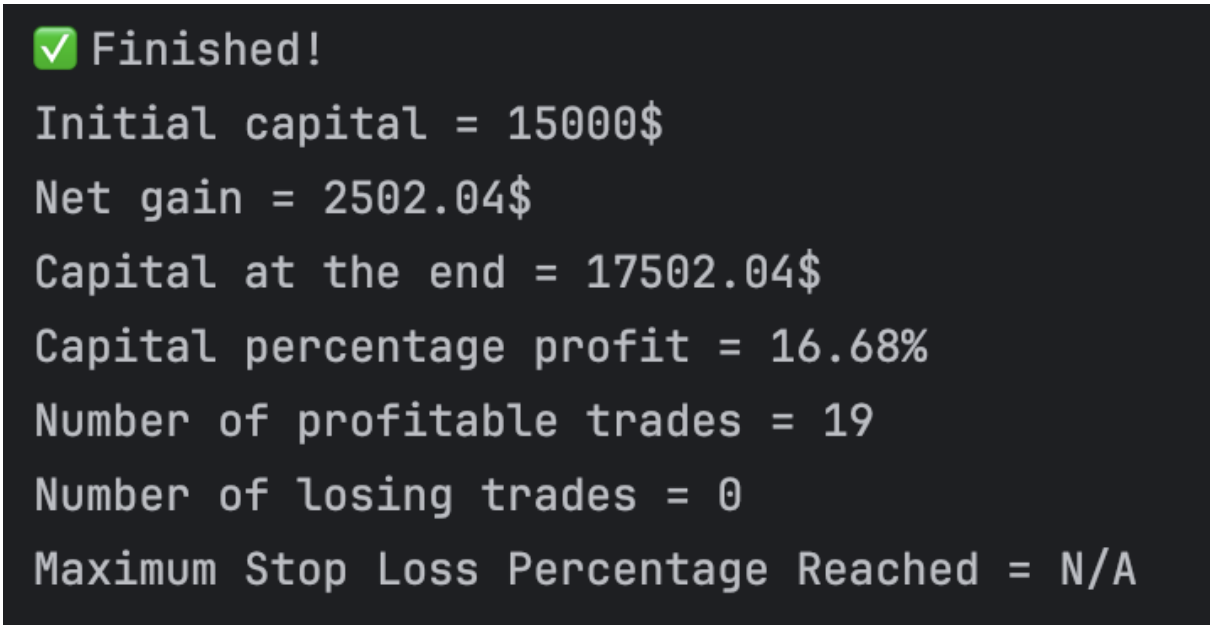

Figure 2.2 - Shows user 30 day trading statistics from 2024-01-25 to 2024-02-24

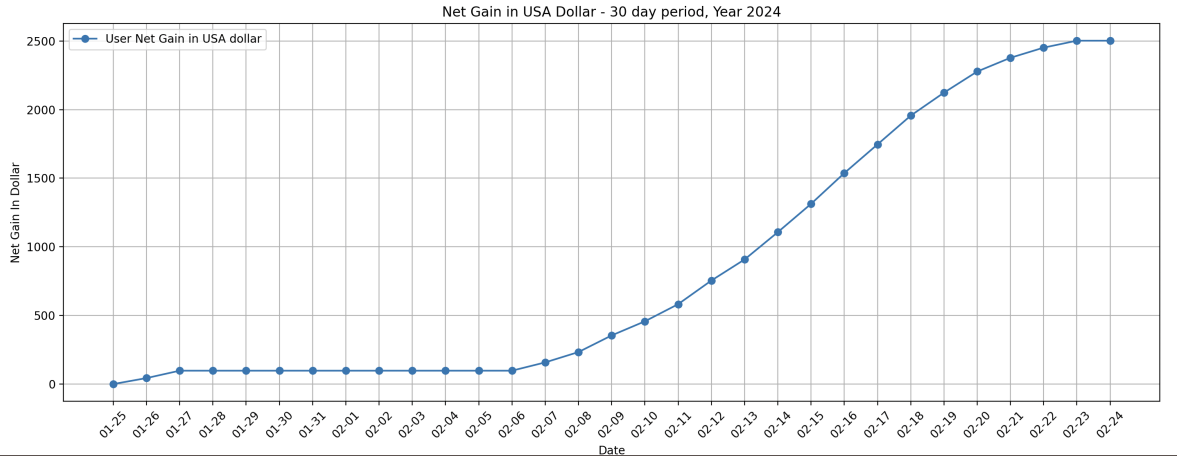

Figure 2.3 - User Capital Net Gain in Dollars versus Trading Date Period from 2024-01-25 to 2024-02-24

According to Figure 2.3 algorithm was most profitable from 2024-02-07 to 2024-02-23, coinciding with the period 2024-02-05 to 2024-02-17 from Figure 1.1 when opening and closing price for BTC USDT asset was increasing as well.

Part2 - Telegram Trading Bot

Current Bot has following functionality

- Trade on Binance historical data based on customer input such as initial trading capital in USD, maximum stop loss percentage, start date and end date. After customer has given those inputs, bot will fetch 1 day candles from Binance for BTC USDT pair, then will present trading statistics when using algorithm from Figure 2.1
- Customer can upload custom historical trading historical data in CSV format and test algorithm from Figure 2.1 on given data. In this case start date would be the first row in the date column and end date would be the last row value in the date column

Stack: AWS web services were used to host bot backend, Docker as building tool, Python

Telegram Bot Demo: Telegram Bot Demo

GitHub link - Trading Bot

Telegram Bot Link - Telegram Bot Link

Example of historical trading data in CSV

binance_1day_candle_btc_usdt.csv

binance_1hour_candle_btc_usdt.csv