Algorithm Trading Bot

Python | Binance | AWS

Personal Project Demo
Johnny Chan

Agenda

1. Automated Account Book - Data Cleansing & Inner Join (Pandas)
    a. Account Overview
    b. Trading Journal - Transaction History

2. Binance Trading Bot - Analysis
    a. Concept Map
    b. Get Historical Data
    c. Technical Analysis
        i. Autocorrelation
        ii. Correlation
        iii. Risk and Return

3. Algorithm Trading Bot
    a. Bollinger Bands + RSI Strategy
        i. Optimization and Result
        ii. Back test with optimized paramsters
        iii. Visualization
        iv. Summary
        v. Live test connecting to Binance API (to be continued)

Automated Account Book

Account Overview

Table A: Asset

asset	free
BTC	0.0522
USDT	1303.3146
FIL	10.4840
GRT	253.3971

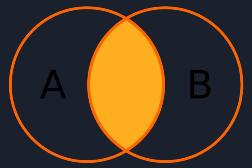


Table B: Real TimePrice

symbol	price
BTCUSDT	57247.12000000
FILUSDT	143.63000000
GRTUSDT	1.50600000

asset	free	price
BTC	0.0522	57477.50000000
USDT	1303.3146	1.0
FIL	10.4840	143.85000000
GRT	253.3971	1.49160000

	asset	free	price	USDT
0	BTC	0.0522	57350.930	2996.0040
1	USDT	1303.3146	1.000	1303.3146
2	FIL	10.4840	143.210	1501.4145
3	GRT	253.3971	1.488	377.0549
Total	NaN	1567.2479	57496.628	6177.7880

Logic:

1. Transform Table B symbol column to be primary key
   a. change column name to ‘asset’
   b. slice the symbol name to exclude “USDT”
2. Inner join
3. Times ‘free’ with ‘price’ to get the current asset value in USDT
4. Calculate Total

```
class summary():

    def spot(self):
        spot_asset = pd.DataFrame(client.get_account()['balances'])
        spot_asset.drop(columns = 'locked', inplace = True)
        spot_asset['free'] = spot_asset['free'].astype(float)
        spot_asset = spot_asset[spot_asset['free']>0]
        return spot_asset


    def Coin_Except_USDT(self):
        Coin_In_Wallet = self.spot()['asset']
        Coin_Except_USDT = Coin_In_Wallet[Coin_In_Wallet != "USDT"] + "USDT"
        Coin_Except_USDT = Coin_Except_USDT.tolist()
        return Coin_Except_USDT

    def spot_balance(self):
        coin = self.Coin_Except_USDT()
        ticker = pd.DataFrame(client.get_symbol_ticker())
        ticker = ticker[ticker.symbol.isin(coin)]
        ticker['symbol'] = ticker['symbol'].str[0:-4]
        ticker.loc[-1,'symbol'] = 'USDT'
        ticker.loc[-1,'price'] = 1.0
        ticker.rename(columns = {'symbol':'asset'}, inplace = True)
        spot_asset = pd.merge(self.spot(),ticker)
        spot_asset[['free','price']] = spot_asset[['free','price']].astype(float)
        spot_asset['USDT'] = spot_asset['free'] * spot_asset['price']
        spot_asset.loc['Total'] = spot_asset[['free','price','USDT']].sum()

        return spot_asset
```

Automated Account Book

Trading Journal - Transaction History

In terms of trading, keep tracking a trading journal is very important so that you can understand the cost and PNL anytime. By creating this program, I gained some solid Data Cleansing skills through Python and Pandas Module

symbol	time	price	Cost (USDT)	Quantity	commission	commissionAsset	isBuyer
BTCUSDT	2021-05-11 01:18:07.570	55420.7600	999.9568	0.0180	1.8040e-05	BTC	1.0
BTCUSDT	2021-05-08 17:24:18.912	58819.1800	-2010.3095	-0.0342	2.0123e+00	USDT	0.0
BTCUSDT	2021-05-04 00:37:57.474	56590.0000	299.9836	0.0053	5.3000e-06	BTC	1.0
BTCUSDT	2021-05-03 16:09:25.208	57495.0800	499.9772	0.0087	8.7000e-06	BTC	1.0
BTCUSDT	2021-05-03 16:07:35.893	57643.0400	499.9957	0.0087	8.6700e-06	BTC	1.0
BTCUSDT	2021-05-03 13:51:47.410	57508.7000	160.9669	0.0028	2.8000e-06	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	31.6235	0.0005	5.5000e-07	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	24.9538	0.0004	4.3000e-07	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	70.7791	0.0012	1.2300e-06	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	70.7791	0.0012	1.2300e-06	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	70.7791	0.0012	1.2300e-06	BTC	1.0
BTCUSDT	2021-05-03 13:51:04.276	57497.2300	70.7791	0.0012	1.2300e-06	BTC	1.0
BTCUSDT	2021-05-03 02:59:30.515	58044.3800	-1862.4051	-0.0321	1.8643e+00	USDT	0.0
BTCUSDT	2021-05-03 02:59:30.515	58044.3800	-287.2063	-0.0050	2.8749e-01	USDT	0.0
BTCUSDT	2021-05-02 10:50:45.422	56726.3700	154.4092	0.0027	2.7200e-06	BTC	1.0
BTCUSDT	2021-05-02 10:40:56.008	56686.5100	274.4761	0.0048	4.8400e-06	BTC	1.0
BTCUSDT	2021-04-29 13:55:54.511	53600.0000	432.6592	0.0081	8.0700e-06	BTC	1.0
BTCUSDT	2021-04-29 13:05:25.041	54366.1000	1730.6361	0.0318	3.1830e-05	BTC	1.0
BTCUSDT	2021-04-29 13:01:59.802	54364.3600	1153.7748	0.0212	2.1220e-05	BTC	1.0
BTCUSDT	2021-04-28 08:28:37.988	54457.1000	299.9497	0.0055	5.5100e-06	BTC	1.0
BTCUSDT	2021-04-04 12:19:58.933	57884.6600	-126.4093	-0.0022	1.2654e-01	USDT	0.0
BTCUSDT	2021-04-04 02:26:41.447	57403.0100	125.5978	0.0022	2.1900e-06	BTC	1.0
BTCUSDT	2021-04-02 15:39:36.232	59438.7800	-40.7342	-0.0007	4.0775e-02	USDT	0.0
BTCUSDT	2021-03-31 11:03:27.432	57687.6700	25.4403	0.0004	4.4000e-07	BTC	1.0
BTCUSDT	2021-03-31 05:08:30.755	58615.4200	14.4194	0.0002	2.5000e-07	BTC	1.0
NaN	NaT	51420.0227	2684.8718	0.0522	NaN	NaN	NaN

Average Buy Price & Cost

Key Points:

1. **Calculate quantity by subtracting commission**
2. **Transform Cost and Quantity to negative number where isBuyer equals to 0 (Sell Order)**
3. **Calculate total cost and total quantity**
4. **Calculate average price by dividing total cost with total quantity**

```
class summary():
    def transaction_history(self,symbol):
        trades = pd.DataFrame(client.get_my_trades(symbol = symbol))
        trades = trades.loc[:,['symbol','time','price','quoteQty','qty','commission','commissionAsset','isBuyer']]
        trades.rename(columns = {'quoteQty':'Cost (USDT)', 'qty':'Quantity'},inplace=True)
        trades[['Cost (USDT)','Quantity','price','commission']] = trades[['Cost (USDT)','Quantity','price','commission']].astype(float)

        trades.loc[trades.commissionAsset != 'USDT' , 'Quantity'] = trades['Quantity'] - trades['commission']
        trades.loc[trades.isBuyer == False, ['Cost (USDT)','Quantity',]] = 0-trades.loc[:,['Cost (USDT)','Quantity']]
        trades.loc[trades.commissionAsset == 'USDT' , 'Cost (USDT)'] = trades['Cost (USDT)'] + trades['commission']     ####
        trades['time'] = pd.to_datetime(trades['time'], unit = 'ms')
        trades.sort_values(by='time', ascending = False, inplace=True)

        trades.loc['Total'] = trades[['Cost (USDT)','Quantity']].sum()
        trades.loc['Total','price'] = trades.iloc[-1]['Cost (USDT)'] / trades.iloc[-1]['Quantity']                    #'price' = Average Cost
        return trades

    def spot(self):
        spot_asset = pd.DataFrame(client.get_account()['balances'])
        spot_asset.drop(columns = 'locked', inplace = True)
        spot_asset['free'] = spot_asset['free'].astype(float)
        spot_asset = spot_asset[spot_asset['free']>0]
        USDT_row = spot_asset[spot_asset.asset == "USDT"]
        spot_asset = spot_asset[spot_asset.asset != "USDT"]                                  #Put USDT to first row
        spot_asset = pd.concat([USDT_row,spot_asset])
        return spot_asset

    def Coin_Except_USDT(self):
        Coin_In_Wallet = self.spot()['asset']
        Coin_Except_USDT = Coin_In_Wallet[Coin_In_Wallet != "USDT"] + "USDT"
        Coin_Except_USDT = Coin_Except_USDT.tolist()                                         #['BTCUSDT', 'FILUSDT' , 'GRTUSDT']
        return Coin_Except_USDT

    def spot_balance(self):
        coin = self.Coin_Except_USDT()                                                       #['BTCUSDT', 'FILUSDT' , 'GRTUSDT']
        ticker = pd.DataFrame(client.get_symbol_ticker())
        ticker = ticker[ticker.symbol.isin(coin)]
        pair_only = ticker['symbol'].str[0:-4]
        ticker['symbol'] = pair_only                                                         #['BTC','FIL','GRT']
        ticker.loc[-1,'symbol'] = 'USDT'
        ticker.loc[-1,'price'] = 1.0
        ticker.rename(columns = {'symbol':'asset'}, inplace = True)
        spot_asset = pd.merge(self.spot(),ticker)                                            #Inner Join
        spot_asset[['free','price']] = spot_asset[['free','price']].astype(float)
        spot_asset.rename(columns = {'price':'Current Ex. Rate','free':'Quantity'},inplace=True)
        spot_asset['Current Value (USDT)'] = spot_asset['Quantity'] * spot_asset['Current Ex. Rate']

        spot_asset.insert(3,'Avg Buy Price',None)
        spot_asset.insert(5,'Cost (USDT)',None)
        return spot_asset
```

Automated Account Book

Account Overview

So now we have all the available coins in our wallet and average buy price, quantity from trading journal. Let's combine them together so we can see the performance easily.

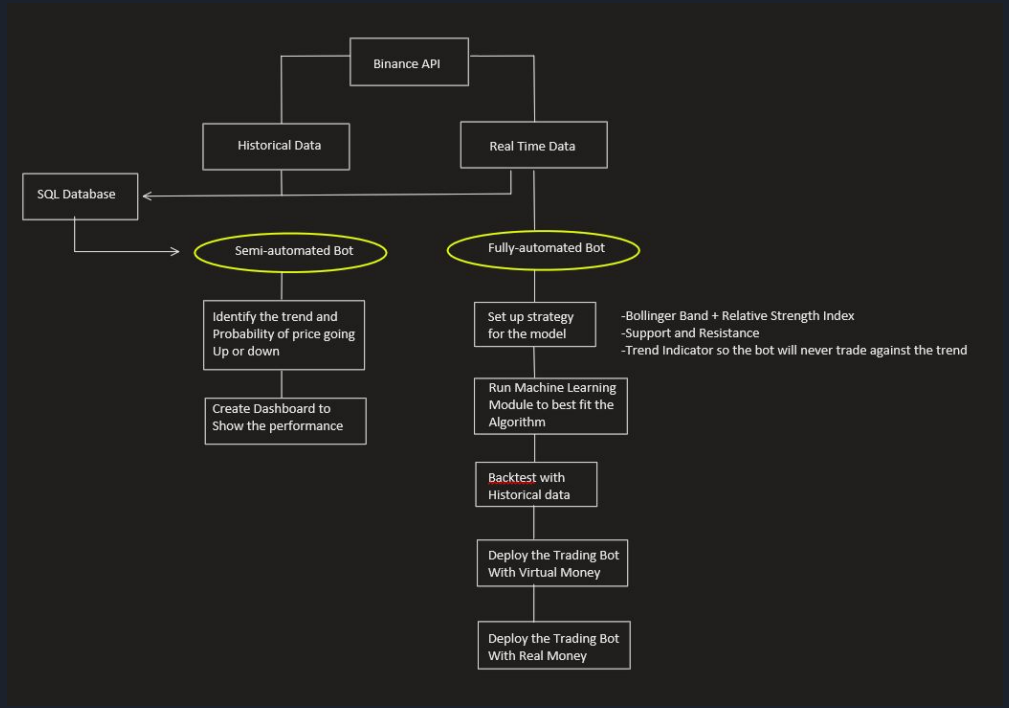
	asset	Quantity	Current Ex. Rate	Avg Buy Price	Current Value (USDT)	Cost (USDT)
0	USDT	1303.3146	1.0000	1.0	1303.3146	1303.3146
1	BTC	0.0522	56801.9100	51420.0227	2967.3233	2684.8718
2	FIL	10.4840	141.2700	143.1012	1481.0755	1499.9737
3	GRT	253.3971	1.4739	1.6021	373.4820	405.8927
Total	NaN	NaN	NaN	NaN	6125.1954	5894.0528
PNL	NaN	NaN	NaN	NaN	231.1426	NaN
PNL (%)	NaN	NaN	NaN	NaN	3.9216	NaN

Now we can compare current value, cost, PNL easily

Algorithm Trading Bot
Concept Map

The goal of this project is to create an Algorithm trading bot for crypto currency. Two types of model will be developed, fully-automated and semi-automated. They will be programmed by Python.

- Fully-automated trading bot will be handling all of the open and close orders in Futures market
- Semi-automated one will be created to provide insights such as pointing out the trend, how likely the price will go up or down etc., long and short call will need to be made by user



Algorithm Trading Bot - Get Historical Data

Code Review

```python
from configparser import ConfigParser
from binance.client import Client
import pandas as pd
import winsound

config = ConfigParser()
config.read(r'D:\Python\Binance_Python_Project\API_Key.ini')
client = Client(config['Account']['key'],config['Account']['secret'])

Pairs = ['BNBUSDT','BTCUSDT','ETHUSDT', 'FILUSDT','TRXUSDT','UNIUSDT','LTCUSDT','XRPUSDT']
Pair = 'BTCUSDT'
time_interval = '15Min_90Days'

#Extract Data from Binance API
def Data_Extractor(symbol,interval,period):
    for item in symbol:
        klines = client.futures_historical_klines(item, interval, period)
        Dataset = pd.DataFrame(klines)
        Dataset = Dataset.astype(float)
        Dataset = Dataset.iloc[:,:6]

    #Assign Column Name
        try:
            print(item, interval,' Data Collected!')
            Dataset.columns = ['Date','Open','High','Low','Close','Volume']
            Dataset['Date'] = pd.to_datetime(Dataset['Date'], unit='ms')
            Dataset.to_csv(rf'D:/Python/Binance_Python_Project/Future Data/{time_interval}/'+item+'.csv', index=False)
        except ValueError:
            print('No Data for',item)

Data_Extractor(Pairs, '15m', '90 days ago UTC')
winsound.Beep(500, 500)
```

Background:

For the purpose of backtesting, it is more efficient to store the data in the drive so that I don't need to call the API to get the data every time

Logics:

1. Connect Binance API
2. Input desired time interval and period
3. Collect historical klines data of the 8 pairs
4. Slice the data so we get clean datasets for each pair
5. Export to CSV

Algorithm Trading Bot - Get Historical Data

Result

What's inside (BTCUSDT):

- 1 Hour timeframe
- past 365 Days

	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	2021-02-05 04:00:00	37424.55	37574.92	37250	37311.66	5742.479
3	2021-02-05 05:00:00	37311.67	37405.18	37052.09	37106.66	5242.133
4	2021-02-05 06:00:00	37106.83	37739.37	37057.01	37422.39	10524.862
5	2021-02-05 07:00:00	37422.72	37780	37422.72	37708.46	7206.643
6	2021-02-05 08:00:00	37708.46	37867.1	37451	37772.46	8097.227
7	2021-02-05 09:00:00	37772.47	37840	37242.67	37311.35	8083.122
8	2021-02-05 10:00:00	37309.14	37735.41	37244.25	37451.54	7491.17
9	2021-02-05 11:00:00	37451.54	37781.07	37443.5	37745.41	5871.118
10	2021-02-05 12:00:00	37745.4	38219	37567.21	37916.26	13853.73
11	2021-02-05 13:00:00	37916.26	38411.82	37817.5	38333.6	12055.8
12	2021-02-05 14:00:00	38329.03	38369.31	38012	38138.52	9892.321
13	2021-02-05 15:00:00	38138.14	38380	37900	38006.64	11167.993
14	2021-02-05 16:00:00	38006.64	38199.77	37626	37855.58	14245.382
15	2021-02-05 17:00:00	37855.1	38004.64	37768.24	37938.35	5869.695
16	2021-02-05 18:00:00	37944.2	37981.5	37254.94	37453.9	8762.429
17	2021-02-05 19:00:00	37453.9	37700	37250	37638.22	10520.627
18	2021-02-05 20:00:00	37638.77	37812.74	37508.68	37760.09	5828.498
19	2021-02-05 21:00:00	37760.08	37969	37685.08	37851.92	4847.168
20	2021-02-05 22:00:00	37851.92	38015.72	37678.56	37823.28	5086.873
21	2021-02-05 23:00:00	37823.29	38393.93	37785	38362.65	6699.907
22	2021-02-06 00:00:00	38357.17	38937.78	38269	38779.44	18184.863
23	2021-02-06 01:00:00	38779.45	39585.48	38312.23	38848.77	31874.524
24	2021-02-06 02:00:00	38848.76	39550	38650	39424.27	15380.246
25	2021-02-06 03:00:00	39424.25	39800	39218.69	39441.31	12962.213

Algorithm Trading Bot

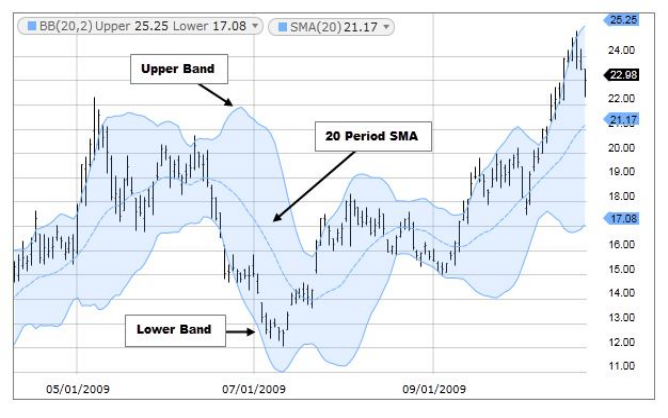
Strategy - Mean Reversion

Introduction

It is believed that price once goes away from the mean, it will eventually come back to the mean, the travel from deviation back to mean is the profit margin I'm looking for.

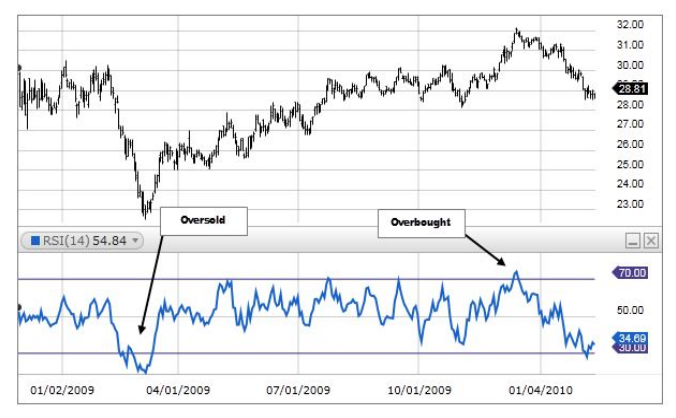
Bollinger Bands

As price touched lower band, buy order executed.
As price touched middle band, sell order executed



Relative Strength Index

As RSI below over sold (30) area, buy order executed.



Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Functions and Workflow

Key Function:

1. Binance API
    a. Get Historical Klines
    b. Get Future Balance
    c. Buy and Sell order
    d. Distribute buying quantity for each pairs
2. Websocket
    a. Stream real-time Klines
3. Mutli-Processing
    a. Stream multiple pairs simutaneously
4. SQL database
    a. Use SQLITE to store trade records
5. Amazon Web Service
    a. Cloud server to run the bot
6. SourceTree & Git
    a. Version Control UI
7. Development Environment
    a. Pycharm, Anaconda

Workflow

1. Get the lastest historical kline
2. Start websocket to get real time kline
3. Append real time kline to historical kline
4. Check buy or sell trigger
5. Run the program parallelly

Algorithm Trading Bot (5Min)

Strategy - Mean Reversion

Optimization

sd_up / sd_low / OverSold / Stop_Loss	Annual Return	DrawDown	Pair	Sharpe Ratio	Total Order
[0.0, 1.7, 27, 4.5, 14]	16.22105375	17.67462016	8	4.946835667	94.625
[0.0, 2.0, 30, 4.5, 17]	15.37842421	17.60665863	8	5.089649767	95.125
[0.0, 1.7, 27, 3.0, 14]	15.20879643	16.12797003	8	4.778107068	99.625
[0.0, 2.0, 27, 3.0, 14]	14.90546331	16.11569897	8	4.490834274	94.5

80% Training Set

```
BNBUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1149803.78
BTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1115246.44
ETHUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1123474.25
FILUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 928667.21
TRXUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 898143.15
UNIUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1273125.91
LTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1146549.08
XRPUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1097661.96
```

20% Test Set

	Pair	sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
UNIUSDT_5Min_90Days		[1.7, 27, 3.0]	102.427002	12.521335	3.549051	23	0.913043
BNBUSDT_5Min_90Days		[1.7, 27, 3.0]	13.608775	12.431184	7.557781	22	0.818182
LTCUSDT_5Min_90Days		[1.7, 27, 3.0]	12.834520	6.301673	5.841372	19	0.684211
ETHUSDT_5Min_90Days		[1.7, 27, 3.0]	8.361473	12.119756	4.894491	22	0.818182
BTCUSDT_5Min_90Days		[1.7, 27, 3.0]	7.128658	7.668061	3.313040	30	0.766667
XRPUSDT_5Min_90Days		[1.7, 27, 3.0]	4.989952	3.428336	17.986520	29	0.620690
FILUSDT_5Min_90Days		[1.7, 27, 3.0]	-0.758690	-3.483372	10.657184	23	0.565217
TRXUSDT_5Min_90Days		[1.7, 27, 3.0]	-0.873018	-5.208961	17.291569	34	0.647059

Algorithm Trading Bot (1Min)

Strategy - Mean Reversion

Optimization

sd_low / OverSold / Stop_Lo	Annual Retu	DrawDow	P	Sharpe Rat	Total Ord
[1.7, 27, 3.5, 14]	53.0187248	17.832475	8	4.271185514	513.875
[2.0, 27, 4.5, 14]	45.99861977	18.99342365	8	3.860280637	448.5
[2.0, 30, 4.5, 17]	45.24043299	19.29028127	8	3.939661344	475.5
[2.0, 27, 3.5, 14]	42.46312446	19.2892516	8	3.627637713	464.125
[2.0, 30, 4.5, 14]	42.41411577	22.30742646	8	3.525836092	727.875

80% Training Set

```
BNBUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1299317.66
BTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1094109.50
ETHUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1113661.78
FILUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1028476.29
TRXUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1115497.61
UNIUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1321790.35
LTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1109013.51
XRPUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1273994.64
```

20% Test Set

Pair	sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
UNIUSDT_1Min_90Days	[1.7, 27, 3.5]	211.616460	11.878482	4.432711	132	0.704545
BNBUSDT_1Min_90Days	[1.7, 27, 3.5]	151.943694	10.920088	4.524873	119	0.697479
XRPUSDT_1Min_90Days	[1.7, 27, 3.5]	103.791231	6.489545	8.979033	121	0.677686
TRXUSDT_1Min_90Days	[1.7, 27, 3.5]	7.163900	5.363968	7.599507	137	0.656934
ETHUSDT_1Min_90Days	[1.7, 27, 3.5]	6.909623	11.296453	3.994579	112	0.714286
LTCUSDT_1Min_90Days	[1.7, 27, 3.5]	6.298948	7.004432	8.453643	117	0.641026
BTCUSDT_1Min_90Days	[1.7, 27, 3.5]	4.628314	7.319218	3.039308	127	0.645669
FILUSDT_1Min_90Days	[1.7, 27, 3.5]	0.714978	2.171541	9.123104	115	0.652174

Algorithm Trading Bot (1Min, Z-Score + RSI)

Strategy - Mean Reversion

Optimization

sd_low / OverSold / Stop_Lo	Annual Retu	DrawDow	P	Sharpe Rat	Total Ord
[1.7, 27, 3.5, 14]	53.0187248	17.832475	8	4.271185514	513.875
[2.0, 27, 4.5, 14]	45.99861977	18.99342365	8	3.860280637	448.5
[2.0, 30, 4.5, 17]	45.24043299	19.29028127	8	3.939661344	475.5
[2.0, 27, 3.5, 14]	42.46312446	19.2892516	8	3.627637713	464.125
[2.0, 30, 4.5, 14]	42.41411577	22.30742646	8	3.525836092	727.875

80% Training Set

```
BNBUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1299317.66
BTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1094109.50
ETHUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1113661.78
FILUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1028476.29
TRXUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1115497.61
UNIUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1321790.35
LTCUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1109013.51
XRPUSDT
Start Portfolio Value: 1000000.00
Final Portfolio Value: 1273994.64
```

20% Test Set

Pair	sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
UNIUSDT_1Min_90Days	[1.7, 27, 3.5]	211.616460	11.878482	4.432711	132	0.704545
BNBUSDT_1Min_90Days	[1.7, 27, 3.5]	151.943694	10.920088	4.524873	119	0.697479
XRPUSDT_1Min_90Days	[1.7, 27, 3.5]	103.791231	6.489545	8.979033	121	0.677686
TRXUSDT_1Min_90Days	[1.7, 27, 3.5]	7.163900	5.363968	7.599507	137	0.656934
ETHUSDT_1Min_90Days	[1.7, 27, 3.5]	6.909623	11.296453	3.994579	112	0.714286
LTCUSDT_1Min_90Days	[1.7, 27, 3.5]	6.298948	7.004432	8.453643	117	0.641026
BTCUSDT_1Min_90Days	[1.7, 27, 3.5]	4.628314	7.319218	3.039308	127	0.645669
FILUSDT_1Min_90Days	[1.7, 27, 3.5]	0.714978	2.171541	9.123104	115	0.652174

Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Calculate Trading Quantity

By using PyCharm, this program allows parallel run so I used 90% of available and divided it into 3 shares equally for each pair to trade.

I created a function to calculate the maximum quantity I can buy for each pair by applying 3x leverage so that whenever I change a pair it will know how many quantity should it be looking at.

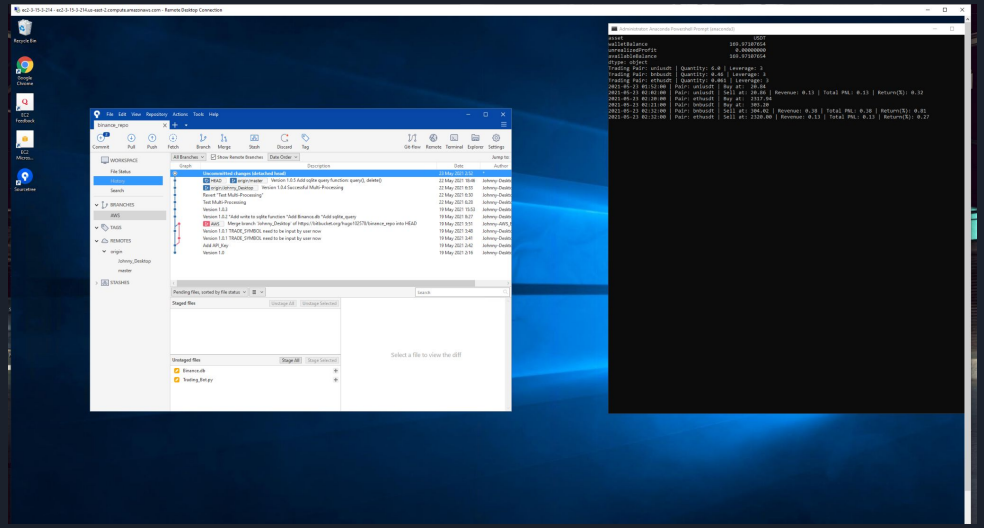
```python
def min_trade_amount(symbol):
    temp = pd.DataFrame(client.futures_exchange_info()['symbols'])
    temp = temp[temp['symbol'] == str.upper(symbol)]
    temp = temp['filters'].item()
    temp = pd.DataFrame(temp)
    temp = temp[temp['filterType']=='LOT_SIZE']
    temp = temp['stepSize'].item()
    return temp

def quantity(symbol, leverage, min_trade_amount):
    symbol = str.upper(symbol)
    walletBalance = float(future_balance()['walletBalance'])
    budget_per_pair = (walletBalance * 0.9) / 3
    price = float(client.get_symbol_ticker(symbol=symbol)['price'])
    quantity = budget_per_pair / (price / leverage)
    quantity = Decimal(quantity).quantize(Decimal(min_trade_amount), rounding=ROUND_DOWN)
    return quantity
```

```
updateTime          2021-05-23 01:44:09.240000+08:00
asset                                           USDT
walletBalance                           176.67041633
unrealizedProfit                          0.00000000
availableBalance                        176.67041633
dtype: object
Trading Pair: bnbusdt | Quantity: 0.49 | Leverage: 3
Trading Pair: uniusdt | Quantity: 7.0 | Leverage: 3
Trading Pair: ethusdt | Quantity: 0.065 | Leverage: 3
```

Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Amazon Web Service (AWS) and SourceTree Version Control



Regarding to version control, my own desktop would be the development environment. Whenever there's an update, through Sourcetree (Git UI) I can easily upload my new version on EC2 machine.

Sourcetree

As the program will be running 24/7, I decided to deploy the system on AWS cloud server and Elastic Compute Cloud (EC2) is a perfect fit for it.

amazon web services™ | EC2

Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Multi-Processing

In order to run the program parallelly for multiple pairs, Mulit-Processing module has been imported so the program will start multiple stream through websocket at same time

```
p1 = Process(target=main, args=('ETHUSDT',))
p2 = Process(target=main, args=('BNBUSDT',))
p3 = Process(target=main, args=('XRPUSDT',))

if __name__ == '__main__':
    p1.start()
    p2.start()
    p3.start()
    p1.join()
    p2.join()
    p3.join()
```

```
updateTime          2021-05-23 01:44:09.240000+08:00
asset                                           USDT
walletBalance                           176.67041633
unrealizedProfit                          0.00000000
availableBalance                        176.67041633
dtype: object
Trading Pair: bnbusdt | Quantity: 0.49 | Leverage: 3
Trading Pair: uniusdt | Quantity: 7.0 | Leverage: 3
Trading Pair: ethusdt | Quantity: 0.065 | Leverage: 3
2021-05-22 19:50:00 | Pair: bnbusdt | Buy at:  303.61
2021-05-22 19:51:00 | Pair: uniusdt | Buy at:  20.24
2021-05-22 19:52:00 | Pair: uniusdt | Sell at: 19.97 | Revenue: -1.90 | Total PNL: -1.90 | Return(%): -4.02
2021-05-22 19:54:00 | Pair: bnbusdt | Sell at: 299.81 | Revenue: -1.86 | Total PNL: -1.86 | Return(%): -3.75
2021-05-22 19:55:00 | Pair: ethusdt | Buy at:  2257.72
2021-05-22 19:55:00 | Pair: bnbusdt | Buy at:  298.70
2021-05-22 20:02:00 | Pair: bnbusdt | Sell at: 302.88 | Revenue: 2.01 | Total PNL: 0.14 | Return(%): 0.30
2021-05-22 20:02:00 | Pair: ethusdt | Sell at: 2284.20 | Revenue: 1.72 | Total PNL: 1.72 | Return(%): 3.52
2021-05-22 22:18:00 | Pair: uniusdt | Buy at:  20.56
2021-05-22 22:24:00 | Pair: uniusdt | Sell at: 20.82 | Revenue: 1.82 | Total PNL: -0.08 | Return(%): -0.16
2021-05-22 23:30:00 | Pair: ethusdt | Buy at:  2340.65
2021-05-22 23:36:00 | Pair: uniusdt | Buy at:  20.59
2021-05-22 23:36:00 | Pair: ethusdt | Sell at: 2316.57 | Revenue: -1.57 | Total PNL: 0.16 | Return(%): 0.31
2021-05-22 23:37:00 | Pair: ethusdt | Buy at:  2319.11
2021-05-22 23:37:00 | Pair: uniusdt | Sell at: 20.35 | Revenue: -1.72 | Total PNL: -1.80 | Return(%): -3.74
2021-05-22 23:55:00 | Pair: ethusdt | Sell at: 2294.01 | Revenue: -1.61 | Total PNL: -1.45 | Return(%): -2.93
2021-05-22 23:56:00 | Pair: ethusdt | Buy at:  2287.99
2021-05-22 23:56:00 | Pair: bnbusdt | Buy at:  298.68
2021-05-23 00:01:00 | Pair: bnbusdt | Sell at: 294.09 | Revenue: -2.25 | Total PNL: -2.10 | Return(%): -4.31
2021-05-23 00:11:00 | Pair: ethusdt | Sell at: 2281.84 | Revenue: -0.39 | Total PNL: -1.84 | Return(%): -3.83
```

Algorithm Trading Bot

SQLite

Monitoring performance is important so that we know how much profit or loss the program is generating.

Each time either a buy or sell order has been triggered, it will call the API from Binance to get the trade record and connect to SQLITE DB and store the data.

```
Started from: 2021-05-23
Today:        2021-05-30
---------------------------
Starting Balance: 2397.44
Total PNL: 96.73
Return: 4.03%
```

```
#Store to SQLITE
time.sleep(0.1)
trade_his = pd.DataFrame(client.futures_account_trades(symbol=TRADE_SYMBOL)).tail(1).values
trade_his = tuple(trade_his[0])
write_to_sqlite(conn, cur, trade_his)
```

	symbol	id	orderId	side	price	qty	realizedPnl	marginAsset	quoteQty	commission	commissionAsset	time	positionSide	buyer	maker
0	BNBUSDT	288377944	31456271161	BUY	265.46	0.49	0.00	USDT	130.08	0.05	USDT	1621757701292	LONG	1	0
1	UNIUSDT	107951560	12846779354	BUY	18.01	6.00	0.00	USDT	108.08	0.04	USDT	1621757820999	LONG	1	0
2	UNIUSDT	107955370	12846824745	SELL	18.60	6.00	3.50	USDT	111.58	0.04	USDT	1621757940348	LONG	0	0
3	BNBUSDT	288422083	31456560480	SELL	270.23	0.49	-3.96	USDT	132.41	0.05	USDT	1621758014293	LONG	0	0
4	ETHUSDT	629561756	8389765498224621924	BUY	2133.78	0.07	0.00	USDT	147.23	0.06	USDT	1621817040629	LONG	1	0
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
468	UNIUSDT	112555014	12996650194	SELL	24.63	7.00	-1.69	USDT	172.43	0.07	USDT	1622378940640	LONG	0	0
469	ETHUSDT	664247320	8389765498874967397	BUY	2394.19	0.33	0.00	USDT	792.48	0.32	USDT	1622378940831	LONG	1	0
470	ETHUSDT	664247320	8389765498874967397	BUY	2394.19	0.33	0.00	USDT	792.48	0.32	USDT	1622378940831	LONG	1	0
471	BNBUSDT	311550047	31814975843	SELL	326.66	2.40	-10.72	USDT	783.98	0.31	USDT	1622378940480	LONG	0	0
472	UNIUSDT	112555032	12996650429	BUY	24.66	3.00	0.00	USDT	73.98	0.03	USDT	1622378941429	LONG	0	1

[473 rows x 15 columns]

Algorithm Trading Bot

SQLite

```
updateTime          2021-05-24 07:18:27.772000+08:00
asset                                           USDT
walletBalance                           170.70576673
unrealizedProfit                          0.00000000
availableBalance                        170.70576673
dtype: object
Trading Pair: ethusdt | Quantity: 0.069 | Leverage: 3
Trading Pair: uniusdt | Quantity: 8.0 | Leverage: 3
Trading Pair: bnbusdt | Quantity: 0.55 | Leverage: 3
2021-05-24 00:43:00 | Pair: ethusdt | Buy at:  2133.92 | Quantity: 0.069
2021-05-24 00:49:00 | Pair: ethusdt | Sell at: 2159.95 | Revenue: 1.80 | Total PNL: 1.80 | Return(%): 3.66
2021-05-24 01:11:00 | Pair: uniusdt | Buy at:  16.57 | Quantity: 8.0
2021-05-24 01:22:00 | Pair: uniusdt | Sell at: 16.76 | Revenue: 1.54 | Total PNL: 1.54 | Return(%): 3.49
2021-05-24 03:46:00 | Pair: ethusdt | Buy at:  2143.13 | Quantity: 0.067
2021-05-24 03:49:00 | Pair: bnbusdt | Buy at:  273.77 | Quantity: 0.55
2021-05-24 04:00:00 | Pair: bnbusdt | Sell at: 274.76 | Revenue: 0.54 | Total PNL: 0.54 | Return(%): 1.08
2021-05-24 04:01:00 | Pair: ethusdt | Sell at: 2150.84 | Revenue: 0.52 | Total PNL: 2.31 | Return(%): 4.83
2021-05-24 04:40:00 | Pair: bnbusdt | Buy at:  270.11 | Quantity: 0.53
2021-05-24 04:41:00 | Pair: ethusdt | Buy at:  2105.07 | Quantity: 0.068
2021-05-24 04:42:00 | Pair: bnbusdt | Sell at: 266.62 | Revenue: -1.85 | Total PNL: -1.31 | Return(%): -2.74
2021-05-24 04:42:00 | Pair: uniusdt | Buy at:  17.02 | Quantity: 8.0
2021-05-24 04:43:00 | Pair: ethusdt | Sell at: 2084.72 | Revenue: -1.38 | Total PNL: 0.93 | Return(%): 1.95
2021-05-24 04:43:00 | Pair: bnbusdt | Buy at:  265.00 | Quantity: 0.54
2021-05-24 04:44:00 | Pair: ethusdt | Buy at:  2083.11 | Quantity: 0.069
2021-05-24 04:47:00 | Pair: uniusdt | Sell at: 17.32 | Revenue: 2.40 | Total PNL: 3.94 | Return(%): 8.69
2021-05-24 04:52:00 | Pair: ethusdt | Sell at: 2114.53 | Revenue: 2.17 | Total PNL: 3.10 | Return(%): 6.46
2021-05-24 04:55:00 | Pair: bnbusdt | Sell at: 269.33 | Revenue: 2.34 | Total PNL: 1.03 | Return(%): 2.17
2021-05-24 05:31:00 | Pair: bnbusdt | Buy at:  271.41 | Quantity: 0.55
2021-05-24 05:31:00 | Pair: ethusdt | Buy at:  2116.46 | Quantity: 0.07
2021-05-24 05:31:00 | Pair: uniusdt | Buy at:  17.36 | Quantity: 8.0
2021-05-24 05:34:00 | Pair: bnbusdt | Sell at: 268.50 | Revenue: -1.60 | Total PNL: -0.57 | Return(%): -1.14
2021-05-24 05:35:00 | Pair: bnbusdt | Buy at:  268.99 | Quantity: 0.55
2021-05-24 05:36:00 | Pair: ethusdt | Sell at: 2096.91 | Revenue: -1.37 | Total PNL: 1.73 | Return(%): 3.50
2021-05-24 05:36:00 | Pair: uniusdt | Sell at: 17.16 | Revenue: -1.65 | Total PNL: 2.30 | Return(%): 4.96
2021-05-24 05:41:00 | Pair: bnbusdt | Sell at: 272.34 | Revenue: 1.84 | Total PNL: 1.27 | Return(%): 2.59
2021-05-24 08:32:00 | Pair: ethusdt | Buy at:  2246.56 | Quantity: 0.069
2021-05-24 08:32:00 | Pair: bnbusdt | Buy at:  287.89 | Quantity: 0.54
2021-05-24 08:41:00 | Pair: ethusdt | Sell at: 2269.78 | Revenue: 1.60 | Total PNL: 3.33 | Return(%): 6.45
2021-05-24 08:41:00 | Pair: bnbusdt | Sell at: 291.81 | Revenue: 2.12 | Total PNL: 3.39 | Return(%): 6.55
2021-05-24 13:17:00 | Pair: ethusdt | Buy at:  2365.81 | Quantity: 0.066
2021-05-24 13:18:00 | Pair: bnbusdt | Buy at:  303.32 | Quantity: 0.51
2021-05-24 13:19:00 | Pair: uniusdt | Buy at:  20.41 | Quantity: 8.0
2021-05-24 13:29:00 | Pair: bnbusdt | Sell at: 308.12 | Revenue: 2.45 | Total PNL: 5.84 | Return(%): 11.33
2021-05-24 13:29:00 | Pair: ethusdt | Sell at: 2382.43 | Revenue: 1.10 | Total PNL: 4.43 | Return(%): 8.51
2021-05-24 13:30:00 | Pair: uniusdt | Sell at: 20.76 | Revenue: 2.76 | Total PNL: 5.06 | Return(%): 9.29
2021-05-24 15:50:00 | Pair: bnbusdt | Buy at:  316.38 | Quantity: 0.49
2021-05-24 15:59:00 | Pair: bnbusdt | Sell at: 318.56 | Revenue: 1.07 | Total PNL: 6.91 | Return(%): 13.37
2021-05-24 16:19:00 | Pair: uniusdt | Buy at:  21.46 | Quantity: 7.0
2021-05-24 16:30:00 | Pair: uniusdt | Sell at: 21.89 | Revenue: 3.05 | Total PNL: 8.11 | Return(%): 16.19
```

End

Algorithm Trading Bot - Technical Analysis

Risk & Return (Sharpe Ratio) Analysis - Result

	A	B	C	D	E	F	G
1		BTC	ETH	BNB	XRP	LTC	DOGE
2	Daily Return	0.61%	0.83%	1.19%	0.82%	0.64%	2.48%
3	Annual Return	11.70%	15.94%	22.82%	15.69%	12.32%	47.31%
4	Standart Deviation	3.57%	4.71%	6.86%	7.91%	5.10%	23.19%
5	Sharpe Ratio	3.2826	3.3866	3.3284	1.9842	2.4134	2.0398

From the table above, we can see that DOGE has extremely high annual return of 47.31%. However, the abnormal return always has a downside: Buyers of DOGE takes 23.19% of risk with Sharpe Ratio 2.0398.

Among the six pairs I would rather pick the one with less return but higher Sharpe Ratio so that my portfolio would be more stable. BNB seems a good pick even it doesn't have the highest Sharpe Ratio, it does take the balance between favourable annual return of 22.82% and acceptable risk of 6.86%.

Algorithm Trading Bot - Technical Analysis

Correlation Analysis Result

	A	B	C	D	E	F	G
1		BTC Close	ETH Close	BNB Close	XRP Close	LTC Close	DOGE Close
2	BTC Close	1.0000	0.9840	0.8569	0.6900	0.9814	0.6515
3	ETH Close	0.9840	1.0000	0.8608	0.7306	0.9812	0.7048
4	BNB Close	0.8569	0.8608	1.0000	0.8489	0.8627	0.7975
5	XRP Close	0.6900	0.7306	0.8489	1.0000	0.7526	0.7624
6	LTC Close	0.9814	0.9812	0.8627	0.7526	1.0000	0.7181
7	DOGE Close	0.6515	0.7048	0.7975	0.7624	0.7181	1.0000

Here comes a correlation matrix shows clearly how each pairs correlate with each others.

Basically ETH, BNB, LTC and BNB have very high correlation so they move in the same way most of the time.

Although XRP and DOGE have lower correlation but they have positive index as well so they do not fulfill the requirement of hedging risk which required negative index.

Algorithm Trading Bot - Technical Analysis

Correlation and R&R Analysis Code Review

```python
import pandas as pd
import numpy as np
from math import sqrt

#Create a combined list for specific crypto currency
pair = ['BTC','ETH','BNB','XRP','LTC','DOGE']
Daily_Return = np.array([])
Annual_Return = np.array([])
Sharpe_Ratio = np.array([])
Stand_Deviation = np.array([])
Combined = pd.DataFrame()
Correlation = pd.DataFrame()

for item in pair:
    Dataset = pd.read_csv(r'D:/Python//Binance_Python_Project/Data/'+item+'USDT.csv', parse_dates=True, index_col=0)

    Close = Dataset.loc[:,'Close']
    Close = Close.rename(item+' Close')

    Return = Close.pct_change(-1)
    Return = Return.rename(item+' % Return')

    Combined = pd.concat([Combined,Close,Return],axis=1)

#Create Close Price table for Correlation Analysis
    Correlation = pd.concat([Correlation,Close],axis=1)

#Analyze Risk and Return
    Daily_Return = np.append(Daily_Return, Return.mean())
    Annual_Return = np.append(Annual_Return, Return.mean() * sqrt(365))
    Stand_Deviation = np.append(Stand_Deviation, Return.std())
    Sharpe_Ratio = np.append(Sharpe_Ratio, Return.mean() / Return.std() * sqrt(365))

array = np.array([Daily_Return, Annual_Return, Stand_Deviation, Sharpe_Ratio])
Risk_And_Return = pd.DataFrame(array, index=['Daily Return', 'Annual Return','Standart Deviation','Sharpe Ratio'], columns = pair)

#Analyze Correlation
Correlation = Correlation.corr()

#Export Result to Excel Table
writer = pd.ExcelWriter(r'D:/Python//Binance_Python_Project/Analysis/Crpyto_Analysis_Result.xlsx', engine = 'xlsxwriter')
Risk_And_Return.to_excel(writer, sheet_name = 'Risk And Return')
Correlation.to_excel(writer, sheet_name = 'Correlation')
writer.save()

#Correlation.to_csv(r'C:/Python/Analysis/Correlation.csv')
Combined.to_csv(r'D:/Python//Binance_Python_Project/Analysis/Comparison_Pair.csv')
```

In order to improve efficiency, I put together two analysis into one program

Logics:

1. Select the target pairs and create a DataFrame to store the new dataset (See next page)
2. Store the close price of each pair and calculate Daily Return
3. Base on Daily return, generate Annual return, Standard Deviation and Sharpe Ratio
4. Concatenate all results into a DataFrame
5. Export to Excel

Algorithm Trading Bot - Technical Analysis

Dataset created at backend

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		BTC Close	BTC % Return	ETH Close	ETH % Return	BNB Close	BNB % Return	XRP Close	XRP % Return	LTC Close	LTC % Return	DOGE Close	DOGE % Return
2	2021-04-22 00:00:00	55118.17	0.0247369144	2571.25	0.0908716791	572.4129	0.052940524	1.32371	0.0253607752	279.28	0.0837828398	0.2883459	-0.0571542362
3	2021-04-21 00:00:00	53787.63	-0.0467411608	2357.06	0.011600709	543.6327	-0.0728742495	1.29097	-0.0678984267	257.69	-0.0114700015	0.3058251	-0.036226465
4	2021-04-20 00:00:00	56425	0.0142336025	2330.03	0.0781585474	586.3635	0.163345318	1.38501	0.057703616	260.68	-0.0026780932	0.3173205	-0.2239712108
5	2021-04-19 00:00:00	55633.14	-0.0092051631	2161.12	-0.0333327369	504.0322	0.0469334806	1.30945	-0.0699730818	261.38	-0.0438249927	0.408903	0.2685853841
6	2021-04-18 00:00:00	56150.01	-0.064270366	2235.64	-0.0353641698	481.4367	-0.0646013172	1.40797	-0.0851159224	273.36	-0.09140464	0.3223299	0.1419884699
7	2021-04-17 00:00:00	60006.66	-0.0216539387	2317.6	-0.0433695055	514.6861	0.012614842	1.53896	-0.0058719034	300.86	-0.0245436566	0.2822532	-0.2276418181
8	2021-04-16 00:00:00	61334.8	-0.0288977292	2422.67	-0.0363438927	508.2743	-0.0620210215	1.54805	-0.119330303	308.43	0.0798235479	0.3654434	1.0039679776
9	2021-04-15 00:00:00	63159.98	0.0031837912	2514.04	0.0334785826	541.8824	-0.0147918742	1.75781	-0.0418983147	285.63	0.0242774152	0.1823599	0.5017338794
10	2021-04-14 00:00:00	62959.53	-0.0096810067	2432.6	0.0580247826	550.0182	-0.0029475521	1.83468	0.0214741859	278.86	0.042116671	0.1214329	0.2863382132
11	2021-04-13 00:00:00	63575	0.0620614768	2299.19	0.0755488401	551.6442	-0.0755244085	1.79611	0.2231498948	267.59	0.0947958432	0.094402	0.3346000455
12	2021-04-12 00:00:00	59860	-0.0023737372	2137.69	-0.0063541202	596.7104	0.1345868014	1.46843	0.0867760032	244.42	-0.0312710554	0.0707343	-0.048374815
13	2021-04-11 00:00:00	60002.43	0.0039033528	2151.36	0.0083759474	525.9275	0.1159140458	1.35118	-0.0157488345	252.31	-0.0157213076	0.07433	0.1665701457
14	2021-04-10 00:00:00	59769.13	0.027975902	2133.49	0.0322323126	471.2975	0.0381146258	1.3728	0.3495738343	256.34	0.1592800289	0.0637167	0.034666911
15	2021-04-09 00:00:00	58142.54	0.0011195382	2066.87	-0.0065322092	453.9937	0.0842810973	1.01721	-0.0367238326	221.12	-0.0220689045	0.0615878	-0.0031723526
16	2021-04-08 00:00:00	58077.52	0.0379613768	2080.46	0.0595832888	418.7048	0.1148996495	1.05599	0.1535579297	226.11	0.0322300845	0.0617838	0.0511578528
17	2021-04-07 00:00:00	55953.45	-0.0351381202	1963.47	-0.0704191345	375.5538	-0.0689475135	0.91542	-0.1652426069	219.05	-0.0758943638	0.0587769	-0.0873818215
18	2021-04-06 00:00:00	57991.15	-0.019259939	2112.21	0.0022919454	403.3648	0.0960216702	1.09663	0.1974034766	237.04	0.0702546505	0.0644047	0.073425979
19	2021-04-05 00:00:00	59129.99	0.0159441229	2107.38	0.0152672123	368.0263	0.0549346632	0.91584	0.4422677165	221.48	0.0952969685	0.0599992	0.0453860401
20	2021-04-04 00:00:00	58202.01	0.0201582979	2075.69	0.0334425348	348.8617	0.0835906976	0.635	0.0980270097	202.21	0.0357527019	0.0573943	0.0339210215
21	2021-04-03 00:00:00	57051.94	-0.0321979589	2008.52	-0.0586636297	321.9497	-0.0483116421	0.57831	-0.0472809344	195.23	-0.0767521044	0.0555113	-0.03600944
22	2021-04-02 00:00:00	58950.01	0.0039095415	2133.69	0.0841645283	338.2932	0.010436408	0.60701	0.0633068825	211.46	0.0423937691	0.0575849	-0.0734678444
23	2021-04-01 00:00:00	58720.44	-0.0003423529	1968.05	0.0253624887	334.7991	0.1072409387	0.57087	0.0018426872	202.86	0.031316726	0.062151	0.1558697338
24	2021-03-31 00:00:00	58740.55	-0.0001024741	1919.37	0.0428751508	302.3724	-0.0286665883	0.56982	0.013734211	196.7	0.0049044651	0.0537699	-0.004266643
25	2021-03-30 00:00:00	58746.57	0.0192780592	1840.46	0.0130563537	311.2962	0.1307563301	0.5621	-0.0058014079	195.74	0.0103752645	0.0540003	-0.0022854926
26	2021-03-29 00:00:00	57635.47	0.0333079767	1816.74	0.0768993663	275.2991	0.0256040485	0.56538	0.0364056313	193.73	0.0515089014	0.054124	0.0085417847
27	2021-03-28 00:00:00	55777.63	-0.0007078471	1687.01	-0.0149594483	268.4263	-0.0029607382	0.54552	-0.0036528346	184.24	0.000977942	0.0536656	-0.0141523991
28	2021-03-27 00:00:00	55817.14	0.0143851252	1712.63	0.0081173036	269.2234	0.0549072236	0.54752	-0.0284791596	184.06	0.0033251567	0.054436	0.0093638167

Algorithm Trading Bot - Technical Analysis

Background

There are few factors make me interested:

1. What is the correlation between major Crypto? Are they always moving towards same directions or opposite ?
2. Which pair would have the best performance? Sharpe Ratio is extremely important in a portfolio as it not only takes return as consideration, but also risk

The reasons I'd like to know these two factors:

1. By finding out negative correlation pairs, then can be used to hedge the risk so as one price drops another one could offset the loss

Algorithm Trading Bot

Strategy - Mean Reversion

Optimization

Optimization

- As 5 min time frame has the highest sharpe ratio, we will apply this time frame afterwards
- Train the data by first 70% of data (training set)

3 x 3 x 3 x9 = 324 combinations

```
cerebro.optstrategy(Strategy.BBRSI,
                    sd_up = [-0.7,0.0,0.3],
                    sd_low = [1.7, 2.0, 2.3],
                    OVER_SOLD = range(30,40,3),
                    Stop_Loss = [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.96, 0.97, 0.98]
                    )
```

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.7]	933%	6.86	10.53	580
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.75]	933%	6.86	10.53	580
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.8]	933%	6.86	10.53	580
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.85]	933%	6.86	10.53	580
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.9]	933%	6.86	10.53	580

Result:

- Top 5 parameters had exactly the same results, the only differences are stop losses, anything higher than 0.9 would lower the return and sharpe ratio
- In order to prevent tight stop loss keep stopping out for the trades, I will use 0.8

Algorithm Trading Bot

Strategy - Mean Reversion

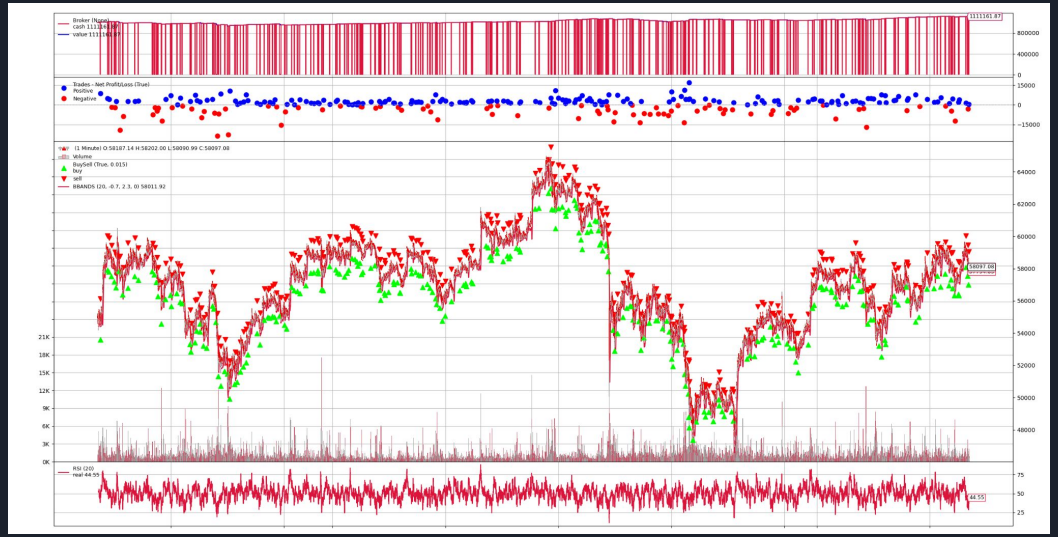
Optimization

Forward testing

- Test the optimized paramters on Test Set (30% of data)

```
if __name__ == '__main__':
    for symbol in Pair:

        cerebro = bt.Cerebro()
        cerebro.broker.setcommission(commission=0.0002)
        Dataset = pd.read_csv(rf'D:/Python/Binance_Python_Project/Future_Data/{time_interval}/{symbol}.csv',parse_dates=True, index_col=0)
        train, test = train_test_split(Dataset, test_size=0.3, shuffle=False)
        data = bt.feeds.PandasData(dataname = train,
                                   timeframe = bt.TimeFrame.TFrame("Minutes"),
                                   compression = 5,
                                   openinterest=-1)
        cerebro.adddata(data)
        cerebro.optstrategy(Strategy.BBRSI,
                            sd_up = [-0.7,0.0,0.3],
                            sd_low = [1.7, 2.0, 2.3],
                            OVER_SOLD = range(30,40,3),
                            Stop_Loss = [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.96, 0.97, 0.98]
                            )
        cerebro.broker.set_cash(1000000)
        cerebro.addsizer(bt.sizers.PercentSizer, percents = 99)

        cerebro.addanalyzer(bt.analyzers.SharpeRatio, timeframe=bt.TimeFrame.Days, annualize=True, riskfreerate=0, factor = 365, _name='sharpe')
        cerebro.addanalyzer(bt.analyzers.DrawDown, _name='dd')
        cerebro.addanalyzer(bt.analyzers.Returns, timeframe=bt.TimeFrame.Days, tann = 365, _name='returns')
        cerebro.addanalyzer(bt.analyzers.TradeAnalyzer, _name='trade')

        Result = cerebro.run()

        Result_List =[(
            [Temp[0].params.sd_up,Temp[0].params.sd_low,Temp[0].params.OVER_SOLD,Temp[0].params.Stop_Loss],
            Temp[0].analyzers.returns.get_analysis()['rnorm'],
            Temp[0].analyzers.sharpe.get_analysis()['sharperatio'],
            Temp[0].analyzers.dd.get_analysis()['max']['drawdown'],
            Temp[0].analyzers.trade.get_analysis()['total']['total'],
            ) for Temp in Result]

        Result_List = pd.DataFrame(Result_List, columns = ['sd_up / sd_low / OverSold / Stop_Loss','Annual Return','Sharpe Ratio','DrawDown','Total Order'])
        # Result_List = Result_List[Result_List['DrawDown']<10]    #Filtering
        Result_List = Result_List.dropna()
        Result_List.insert(0, 'Pair', symbol+ '_'+time_interval)
        Opt_List = Opt_List.append(Result_List)
        print(symbol,' Optimized! No. of Result: ',len(Result_List.index))

    winsound.Beep(500,1000)
    Opt_List.sort_values(by=['Annual Return','DrawDown'], ascending=[False,True], inplace=True)

    with pd.ExcelWriter(rf'D:/Python/Binance_Python_Project/Opt_Result/BBRSI_{Pair}_Training_Set_{time_interval}_Opt_Summary.xlsx') as writer:
        Pivot_Table().to_excel(writer, sheet_name = 'Pivot Table')
        Opt_List.to_excel(writer, sheet_name = 'Data', index = False)
```

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
BTCUSDT_5Min_180Days	[-0.7, 2.3, 30, 0.8]	101%	3.19	9.16	265	67%



Algorithm Trading Bot

Strategy - Mean Reversion

Run default settings with different timeframe

```
cerebro.addstrategy(Strategy.BBRSI,
                    sd_up = 0.0,
                    sd_low = 2.0,
                    OVER_SOLD = 30,
                    BB_Period = 20,
                    RSI_Period = 20,
                    Stop_Loss = 0.7)
```

As 5 minutes time-frame has highest return, I will use this as the time interval

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
LTCUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	2861%	6.30	25.41	964.00	69%
UNIUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	1471%	4.39	28.37	960.00	66%
BNBUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	1009%	5.12	31.12	966.00	69%
FILUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	643%	3.94	26.39	924.00	67%
ETHUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	355%	3.45	20.70	995.00	68%
BTCUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	286%	3.75	14.50	1,007.00	69%
XRPUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	184%	1.64	58.23	945.00	66%
TRXUSDT_5Min_180Days	[0.0, 2.0, 30, 0.7]	7%	0.43	43.96	945.00	66%
Average		**852%**	**3.63**	**31.09**	**963.25**	**68%**

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
ETHUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	1553%	5.49	25.69	2,579.00	68%
FILUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	725%	3.21	27.67	2,309.00	65%
UNIUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	344%	2.58	30.16	2,394.00	65%
BTCUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	310%	3.90	14.84	2,530.00	68%
LTCUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	275%	2.55	34.36	2,378.00	66%
BNBUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	275%	2.25	24.98	2,381.00	66%
XRPUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	127%	1.38	38.57	2,373.00	66%
TRXUSDT_1Min_90Days	[0.0, 2.0, 30, 0.7]	-72%	(1.28)	46.24	2,319.00	65%
Average		**442%**	**2.51**	**30.31**	**2,407.88**	**66%**

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
LTCUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	650%	3.85	27.84	679.00	75%
UNIUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	483%	2.66	43.75	405.00	67%
BNBUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	280%	3.13	22.92	657.00	69%
XRPUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	278%	2.15	54.61	623.00	71%
ETHUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	157%	2.26	31.03	636.00	72%
FILUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	154%	2.05	28.71	319.00	67%
BTCUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	57%	1.55	19.06	640.00	71%
TRXUSDT_15Min_365Days	[0.0, 2.0, 30, 0.7]	16%	0.55	40.43	622.00	70%
Average		**259%**	**2.27**	**33.54**	**572.63**	**70%**

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
LTCUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	343%	2.93	28.13	340.00	76%
UNIUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	264%	2.31	33.09	196.00	67%
BNBUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	225%	2.88	25.67	322.00	73%
BTCUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	189%	3.33	16.89	331.00	74%
XRPUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	163%	1.81	43.54	321.00	73%
ETHUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	124%	2.02	27.85	312.00	71%
TRXUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	99%	1.61	43.12	304.00	69%
FILUSDT_30Min_365Days	[0.0, 2.0, 30, 0.7]	33%	0.81	34.74	158.00	61%
Average		**180%**	**2.21**	**31.63**	**285.50**	**71%**

Pair	sd_up / sd_low / OverSold / Stop_Loss	Annual Return	Sharpe Ratio	DrawDown	Total Order	Win Rate
LTCUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	216%	2.45	30.48	164.00	73%
BNBUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	87%	1.76	23.58	154.00	69%
XRPUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	45%	1.02	40.08	157.00	71%
BTCUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	41%	1.22	20.43	157.00	70%
TRXUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	35%	0.83	40.48	149.00	70%
ETHUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	32%	0.88	26.79	149.00	64%
UNIUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	23%	0.64	46.66	92.00	64%
FILUSDT_1Hour_365Days	[0.0, 2.0, 30, 0.7]	-43%	(0.70)	37.35	73.00	63%
Average		**54%**	**1.01**	**33.23**	**136.88**	**68%**

Data Cleaner
Background

This program was made to clean the dataset from company internal frontend system.

Ultimately we need to produce Dashboard by Power BI yet the original format was messy which made it impossible to load into BI tools.

This report needs to be produced quarterly so writing a program for it would save a lot of time.

Main issues were:

- Column headers cannot match the corresponding values
- Messy 2-Dimension headers
- Multiple row headers
- Totals need to be removed

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Insurer Performance (by insurer) Summary															
2	Reporting Period: Effective date1 from 01/01/2021 to 31/03/2021															
3	Remarks: 1. Quotation first submitted date is used for quotation with unknown period of insurance. 2. This report is generated on 19/04/2021 and data is up to the previous day. 3. Co-insurance policy is recorded to each co-insurer's account for their respective share with number of policy count repeatedly to the co-insurers.															
4																
5		Class of Insurance				No. of Quotations Submitted			No. of Placed Policies				No. of Placed (%)		Placed Premium	
6	Currency:															
7			HKD													
8	Insurer:	Allied World Assurance Company, Ltd														
9		Commercial Business Package					34			17		50.00 %				670,589.24
10		Employees' Compensation					30			16		53.33 %				6,516,191.20
11					Total:											
12							64			33		51.56 %				7,186,780.44
13	Insurer:	AXA General Insurance Hong Kong Limited														
14		Commercial Business Package					39			21		53.85 %				675,841.21
15		Employees' Compensation					66			47		71.21 %				2,226,246.26
16		Office Package Insurance					18			16		88.89 %				791,033.07
17					Total:											
18							123			84		68.29 %				3,693,120.54
19	Insurer:	Chubb Insurance Hong Kong Limited														
20		Commercial Business Package					11			0		0.00 %				0
21		Employees' Compensation					25			10		40.00 %				454,822.41
22					Total:											
23							36			10		27.78 %				454,822.41
24	Insurer:	MSIG Insurance (Hong Kong) Limited														
25		Commercial Business Package					87			59		67.82 %				1,804,275.33
26		Employees' Compensation					98			65		66.33 %				3,024,660.65
27					Total:											
28							185			124		67.03 %				4,828,935.98
29	Insurer:	QBE Hongkong & Shanghai Insurance Ltd														
30		Commercial Business Package					9			1		11.11 %				16,955.79
31		Employees' Compensation					15			2		13.33 %				1,283,288.14
32					Total:											
33							24			3		12.50 %				1,300,243.93
34				HKD	Total:		432			254						17,463,903.30
35																

Data Cleaner

Result

Before

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Insurer Performance (by insurer) Summary															
2	Reporting Period: Effective date f from 01/01/2021 to 31/03/2021															
3	Remarks: 1. Quotation first submitted date is used for quotation with unknown period of insurance. 2. This report is generated on 19/04/2021 and data is up to the previous day. 3. Co-insurance policy is recorded to each co-insurer's account for their respective share with number of policy count repeatedly to the co-insurers.															
4																
5		Class of Insurance				No. of Quotations Submitted			No. of Placed Policies				No. of Placed (%)		Placed Premium	
6	Currency:															
7		HKD														
8	Insurer:	Allied World Assurance Company, Ltd														
9		Commercial Business Package				34			17			50.00 %			670,589.24	
10		Employees' Compensation				30			16			53.33 %			6,516,191.20	
11					Total:											
12						64			33			51.56 %			7,186,780.44	
13	Insurer:	AXA General Insurance Hong Kong Limited														
14		Commercial Business Package				39			21			53.85 %			675,841.21	
15		Employees' Compensation				66			47			71.21 %			2,226,246.26	
16		Office Package Insurance				18			16			88.89 %			791,033.07	
17					Total:											
18						123			84			68.29 %			3,693,120.54	
19	Insurer:	Chubb Insurance Hong Kong Limited														
20		Commercial Business Package				11			0			0.00 %			0	
21		Employees' Compensation				25			10			40.00 %			454,822.41	
22					Total:											
23						36			10			27.78 %			454,822.41	
24	Insurer:	MSIG Insurance (Hong Kong) Limited														
25		Commercial Business Package				87			59			67.82 %			1,804,275.33	
26		Employees' Compensation				98			65			66.33 %			3,024,660.65	
27					Total:											
28						185			124			67.03 %			4,828,935.98	
29	Insurer:	QBE Hongkong & Shanghai Insurance Ltd														
30		Commercial Business Package				9			1			11.11 %			16,955.79	
31		Employees' Compensation				15			2			13.33 %			1,283,288.14	
32					Total:											
33						24			3			12.50 %			1,300,243.93	
34				HKD	Total:	432			254						17,463,903.30	
35																

After

	A	B	C	D	E	F	G
1	Insurer	Class of Insurance	No. of Quotations Submitted	No. of Placed Policies	No. of Placed (%)	Placed Premium	
2	Allied World Assurance Company, Ltd	Commercial Business Package	34	17	50.00%	670,589.24	
3	Allied World Assurance Company, Ltd	Employees' Compensation	30	16	53.33%	6,516,191.20	
4	AXA General Insurance Hong Kong Limited	Commercial Business Package	39	21	53.85%	675,841.21	
5	AXA General Insurance Hong Kong Limited	Employees' Compensation	66	47	71.21%	2,226,246.26	
6	AXA General Insurance Hong Kong Limited	Office Package Insurance	18	16	88.89%	791,033.07	
7	Chubb Insurance Hong Kong Limited	Commercial Business Package	11	0	0.00%	0	
8	Chubb Insurance Hong Kong Limited	Employees' Compensation	25	10	40.00%	454,822.41	
9	MSIG Insurance (Hong Kong) Limited	Commercial Business Package	87	59	67.82%	1,804,275.33	
10	MSIG Insurance (Hong Kong) Limited	Employees' Compensation	98	65	66.33%	3,024,660.65	
11	QBE Hongkong & Shanghai Insurance Ltd	Commercial Business Package	9	1	11.11%	16,955.79	
12	QBE Hongkong & Shanghai Insurance Ltd	Employees' Compensation	15	2	13.33%	1,283,288.14	
13							

Data Cleaner

Code Review

```python
import pandas as pd
import numpy as np
Dataset = pd.DataFrame()
Dataset = pd.read_csv('InsPerfByInsurerSummRpt.csv')

#Store headers
headers = Dataset.iloc[3,:]
headers = headers.dropna(axis=0)

#Store Insurer Name
insurers = []
for index,row in Dataset.iterrows():
    if row[0] == 'Insurer:':
        insurers.append(row[1])

#Slice the data
Dataset = Dataset.iloc[6:,1:]
First_col=Dataset.iloc[:,0]
Second_col=Dataset.iloc[:,5]
Third_col=Dataset.iloc[:,8]
Fourth_col=Dataset.iloc[:,10]
Fifth_col=Dataset.iloc[:,14]

#Concat every columns
Final = pd.concat([First_col, Second_col,Third_col,Fourth_col,Fifth_col], axis=1)
Final.columns = [headers]
Final = Final.dropna(axis=0)

#Assign Insurer to first column
Final.insert(0, 'Insurer', np.nan)
insurer_order=-1
for index, row in Final.iterrows():
    if row[1] =='Commercial Business Package' and insurer_order < 5:
        insurer_order += 1
        Final.loc[index,'Insurer'] = insurers[insurer_order]
    else:
        Final.loc[index,'Insurer'] = insurers[insurer_order]

Final.to_csv('Final.csv', index=False)
```

Logic:

1. Store the headers
2. Store the insurer names
3. Slice the data column by column
4. Concatenate the filtered columns into one DataFrame
5. Assign headers back to corresponding values
6. Assign insurers back to corresponding values
7. Export to CSV

Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Optimization Result

5Min_90Days

sd_up / sd_low / OverSold / RSI_Period	Annual Return	DrawDown	Pair	Sharpe Ratio	Total Order
[0.2, 2.4, 40, 14]	104%	7.46	8	3.41	74.4
[-0.6, 2.0, 43, 14]	92%	7.75	7	3.57	132.9
[0.0, 2.4, 40, 14]	85%	7.16	7	3.04	73.9

15Min_90Days

sd_up / sd_low / OverSold / RSI_Period	Annual Return	DrawDown	Pair	Sharpe Ratio	Total Order
[-1.4, 2.0, 43, 20]	143%	6.10	7	5.15	109.6
[-1.0, 2.0, 46, 20]	106%	6.28	7	4.43	54.0
[-1.4, 2.0, 40, 14]	103%	6.25	7	4.24	97.1

30Min_90Days

sd_up / sd_low / OverSold / RSI_Period	Annual Return	DrawDown	Pair	Sharpe Ratio	Total Order
[-1.4, 2.4, 37, 20]	114%	6.83	8	4.27	41.1
[0.6, 2.4, 40, 14]	81%	6.16	7	3.58	12.9
[0.2, 2.4, 40, 14]	76%	5.84	7	3.79	12.9

1Hour_90Days

sd_up / sd_low / OverSold / RSI_Period	Annual Return	DrawDown	Pair	Sharpe Ratio	Total Order
[-1.4, 2.0, 40, 14]	136%	6.56	7	2.52	27.1
[-1.4, 1.2, 46, 14]	133%	7.15	7	3.05	43.9
[-1.4, 1.6, 43, 14]	103%	6.55	7	3.37	36.7

Algorithm Trading Bot

Strategy - Bollinger Bands + RSI

Optimization Result

5Min_90Days						
Parameters [sd_up, sd_low, over_sold]	Pair	Avg APR	Avg Compound Return	Avg Sharpe Ratio	Avg Max DrawDown	Avg No. of Order
[0.2, 2.0, 46]	8	50.0%	9.2%	2.64	6.88	48.50
[-1.4, 2.0, 43]	8	48.4%	9.5%	3.26	4.95	138.00
[-1.8, 2.0, 43]	8	32.0%	6.6%	2.73	4.27	139.38
[-0.2, 2.0, 46]	8	30.1%	5.8%	1.94	7.18	48.63
[-0.6, 2.0, 46]	8	29.8%	6.1%	2.24	6.43	48.63
[-1.4, 2.0, 46]	8	21.4%	4.6%	2.90	3.38	49.50
[-1.0, 2.0, 46]	8	17.0%	3.7%	1.63	5.58	49.13
[-1.8, 2.0, 46]	8	13.1%	2.9%	2.22	3.04	49.63
[-2.2, 2.0, 43]	8	7.8%	1.5%	0.78	5.13	140.25
[-2.2, 2.0, 46]	8	6.0%	1.2%	1.34	2.97	49.75
总计	**80**	**25.6%**	**5.1%**	**2.17**	**4.98**	**76.14**

30Min_90Days						
Parameters [sd_up, sd_low, over_sold]	Pair	Avg APR	Avg Compound Return	Avg Sharpe Ratio	Avg Max DrawDown	Avg No. of Order
[-1.8, 1.6, 46]	8	33.0%	6.7%	2.29	4.84	34.25
[-2.2, 1.6, 46]	8	26.8%	5.7%	1.97	4.51	34.50
[-1.4, 2.0, 46]	8	24.3%	5.3%	2.91	2.34	10.88
[-1.8, 2.0, 46]	8	19.5%	4.3%	2.78	1.74	11.13
[-2.2, 2.0, 46]	8	19.1%	4.2%	2.83	1.36	11.25
总计	**40**	**24.5%**	**5.2%**	**2.56**	**2.96**	**20.40**

1Hour_90Days						
Parameters [sd_up, sd_low, over_sold]	Pair	Avg APR	Avg Compound Return	Avg Sharpe Ratio	Avg Max DrawDown	Avg No. of Order
[-1.4, 2.0, 43]	8	60%	11%	3.16	5.08	15.88
[-1.8, 2.0, 43]	8	49%	8%	2.46	4.53	16.13
[-1.4, 1.6, 46]	8	25%	5%	1.79	5.99	17.88
[-1.4, 2.4, 37]	8	23%	4%	1.67	5.66	11.38
[-1.4, 2.0, 46]	8	20%	4%	2.08	3.82	7.38
[-1.8, 2.4, 37]	8	18%	4%	1.89	4.07	11.63
[-1.8, 1.6, 46]	8	16%	3%	1.35	5.88	18.13
[-1.8, 2.4, 40]	8	12%	2%	1.65	2.46	6.00
[-1.8, 2.0, 46]	8	12%	2%	1.34	3.66	7.50
[-1.4, 2.4, 40]	8	10%	2%	1.04	4.05	6.00
[-1.8, 2.4, 43]	8	4%	1%	1.46	1.99	3.38
[-1.4, 2.4, 43]	8	3%	0%	0.91	3.54	3.38
[-1.4, 2.4, 46]	8	1%	0%	1.01	2.25	1.75
[-1.8, 2.4, 46]	8	-1%	0%	0.40	1.72	1.75
总计	**112**	**18%**	**3%**	**1.60**	**3.91**	**9.15**