## Simple Backtesting Framework

The following assessment contains 6 parts.

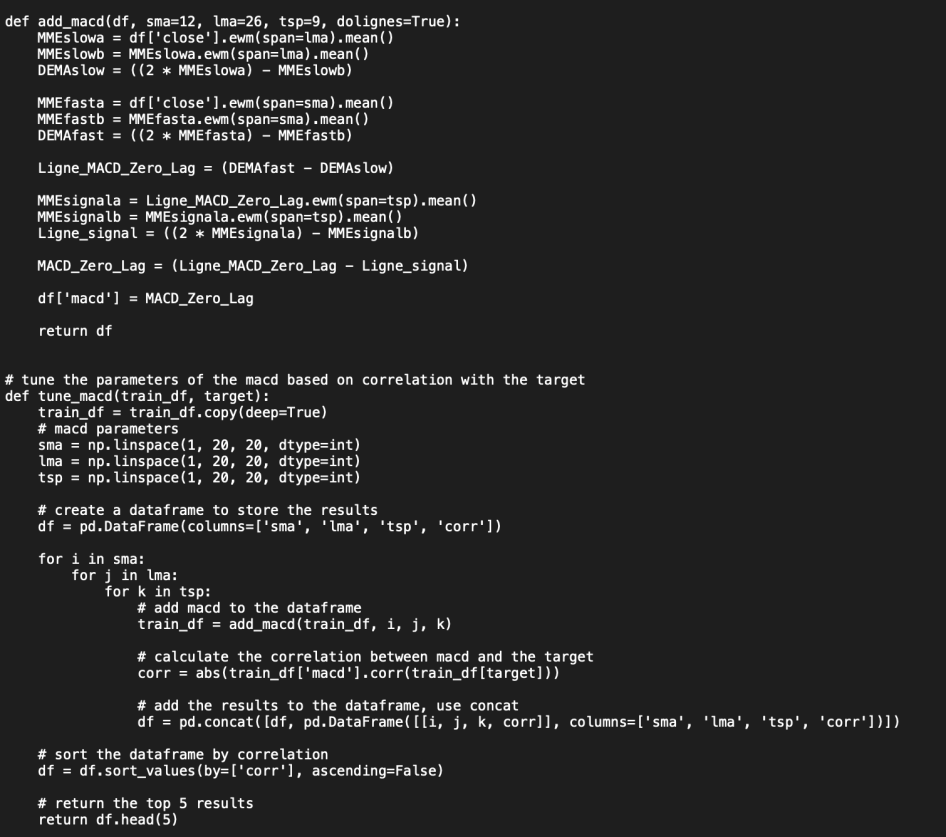
The goal will be implementing a simple trading backtesting framework using python backtrader library and python-binance library and give you a brief idea of how trading looks like.

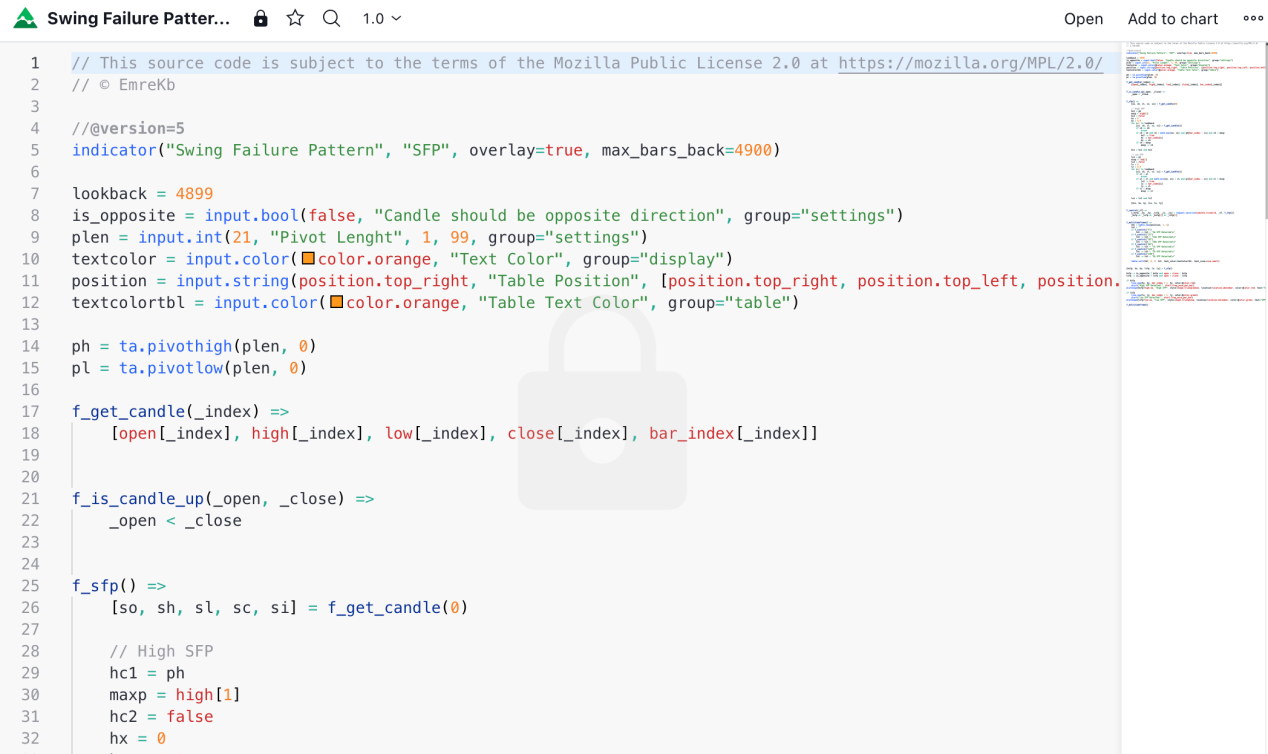
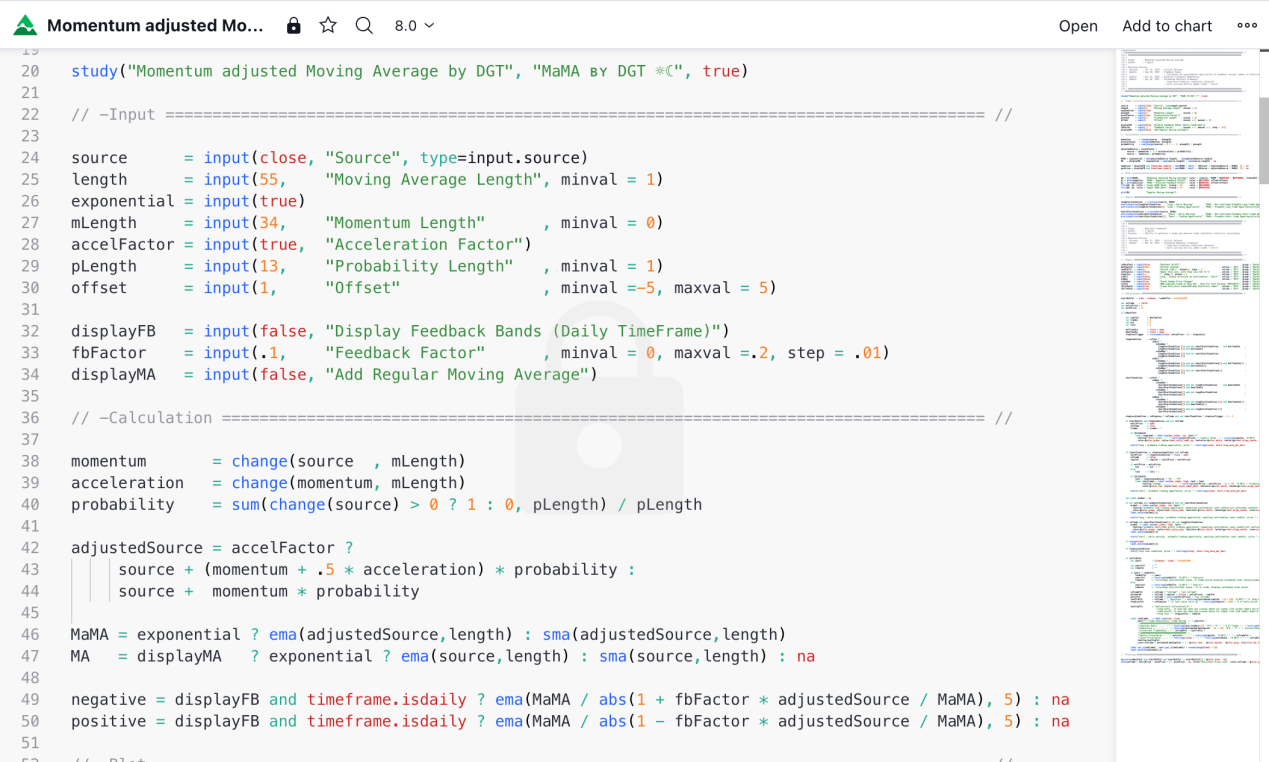
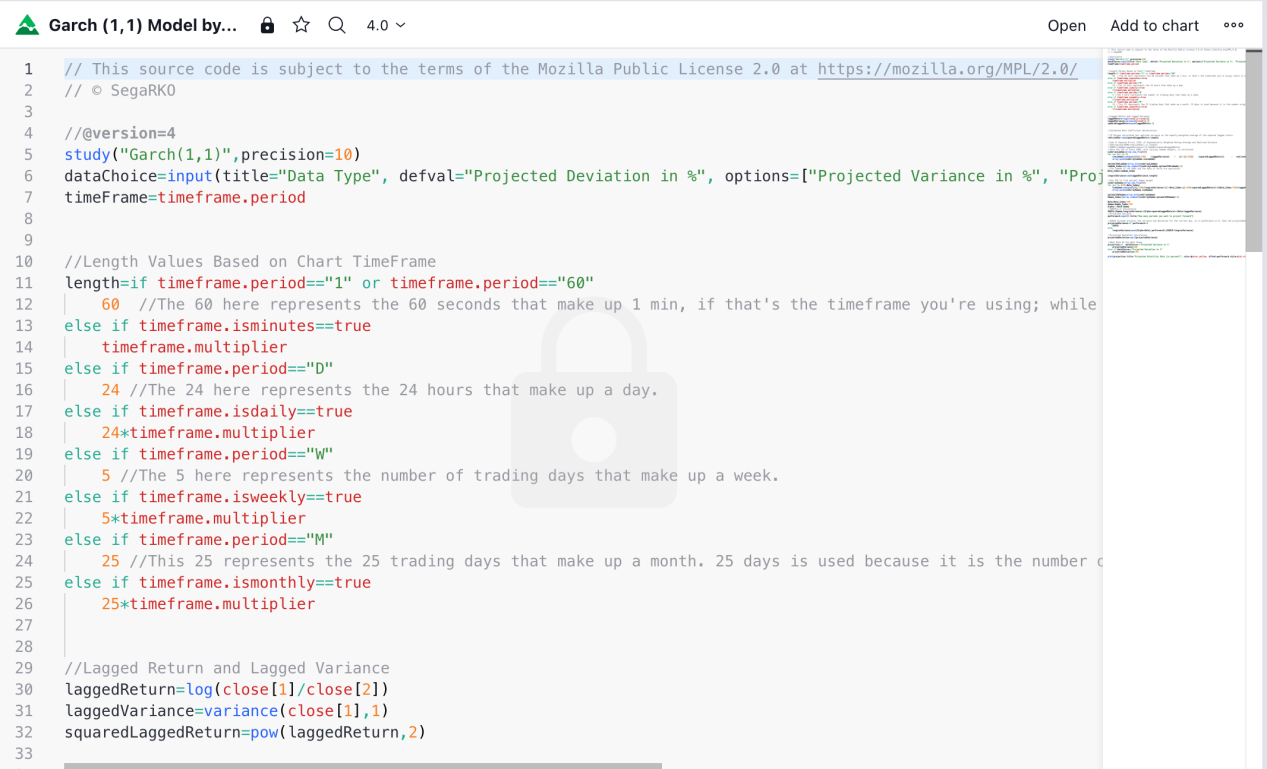
The following are the link to both library

https://github.com/mementum/backtrader

https://python-binance.readthedocs.io/en/latest/

1. Data collecting and data preprocessing for ETH and ADA
   1. Kline data source
      1. <https://data.binance.vision/?prefix=data/spot/monthly/klines/ETHUSDT/1h/>
      2. <https://data.binance.vision/?prefix=data/spot/monthly/klines/ADAUSDT/1h/>
      3. Trade data source
         1. <https://data.binance.vision/?prefix=data/spot/monthly/aggTrades/ETHUSDT/>
         2. <https://data.binance.vision/?prefix=data/spot/monthly/aggTrades/ADAUSDT/>
         3. For trade data, merge them into kline data that use number of trade at a certain timestamp(trades\_at\_current\_ts) and buy/sell ratio at a certain timestamp(buy\_sell\_ratio\_at\_current\_ts).
            1. You will need to calculate the above by yourselves.
   2. Write a script to perform downloading one hour kline bar and trade data for 20180101-20230101, merge them together into a single csv.
2. Alpha implementing
   1. Go to tradingview.com
   2. Checkout the pinescript of the following community script(The script in below are all public so you should be able to see the source code without any problem) and convert them into a function/class(separate them in a different alpha file) that you will be passing in a dataframe and then get an output the corresponding result by appending an additional column in the original dataframe. (NOTES: please verify the signal you implement is performing the same as the website. Provide a sceenshoot of the past 6 month alpha prediction output when you compare with tradingview)
      1. For each alpha file
         1. Define a class, take MACD as example
            1. You will take the original dataframe as input, and other parameter the alpha use and output the dataframe with additional column with the alpha.
            2. Also you will need to tune the parameter before you train the model. Use a better idea of tuning parameter rather use the logic in the below graph.
            3. Here is the example code for MACD:



* + 1. Swing Failure Pattern by EmreKb
       1. 
    2. Momentum adjusted Moving Average by DGT
       1. 
    3. Garch (1,1) Model by SegaRKO
       1. 

1. Modeling
   1. Assume you have a 10 core machine, use a parallel method to use as many resource as possible while training you model.
      1. Plot the time you use before using the parallel method and after apply this technique.
   2. Use the following model to train the above alpha(feel free to use whatever library), also implement ways to reduce overfitting.
      1. Note that if the model using classification/regression, you will need to use different y as noted in point e
         1. Lasso regression
         2. Transformer
         3. Random forest
         4. LSTM
         5. GRU
         6. CNN
   3. Cross validation
   4. Do not use price as y. use the following as y.
      1. use the diff ratio of price from previous bar as y when you are doing regression
      2. Use the chance of going up or down as y when you are doing classification.
   5. Also train the data using different mark out
      1. previous 5 bar and current bar diff
      2. previous 10 bar and current bar diff
      3. previous 20 bar and current bar diff
      4. previous 40 bar and current bar diff
      5. previous 60 bar and current bar diff
      6. previous 80 bar and current bar diff
      7. previous 100 bar and current bar diff
2. Back testing
   1. Use Backtrader library and feed the data you form in step 1 in
   2. You will be able to do both long a position or short a position
      1. Meaning you will be implementing a margin method. As long as you have enough USDT, you will be able to use up to 3x leverage on both opening long position or short position.
      2. When you get a buy signal when you are in a short position. You should decide whether you flip the position or flat it.
         1. Meaning when your position is -10 ETH
            1. And you get a strong enough buy signal. You should flip the position to 10 ETH.
            2. If you get a not so strong buy signal. You should just flat the position to 0 ETH.
   3. Use the model you trained in the step 3 and perform backtesting
      1. Use 10000 USDT fund as initial fund
         1. If you lose more than what you have as initial fund, you will need to stop trading and don’t need to continue looping the historical data.
         2. Assume only 5000 USDT can be used per trade.
3. Paper trading
   1. The api you will be using will be in [https://python-binance.readthedocs.io/en/latest/market\_data.html#](https://python-binance.readthedocs.io/en/latest/market_data.html)
   2. You do not need to submit real time order, instead of sending a real order, print the trade out instead. You only need to simulate the trading for a day.
   3. The endpoint you may need are, change the input parameter to fit your need.
      1. trades = client.get\_historical\_trades(symbol='BNBBTC')
      2. klines = client.get\_historical\_klines("BNBBTC", Client.KLINE\_INTERVAL\_1MINUTE, "1 day ago UTC")

d) implement the code for trading using the model you trained above.

6. Analytics

* 1. Analyze and calculate the Sharpe ratio, Annualized yearly return, max Drawdown, number of trades, Profit & Loss per trade, Total Profit & Loss for all the cases above.
     1. Use a table to explain
  2. Will it be better if you use only one of the above signal or use all of them
     1. Compare the the a) stats for this.
     2. Use a table to explain

c) what can you do to lower the Drawdown while maintaining the annualized return

d) If training the model is too slow when you are doing it, what can you do to improve it?Implement the idea also.

e) Bonus: What can you do to make the project better?

1. Folder structure(zip the folder before sending it)
   1. data/2022/binance\_ETH\_USDT\_2018\_2023.csv(use the following format for all the years)
      1. /2021/binance\_ADA\_USDT\_2018\_2023.csv, etc
   2. src/data\_gen.py
      1. Implement data\_loader to download the files in the above
         1. def data\_loader()
   3. src/backtest.py
      1. This file will be used for backtester related code.
      2. The following two methods need to be implemented
         1. class BaseStrategy(bt.Strategy)
         2. def backtest(strategy, kline\_data, start\_date, end\_date, predictions)
      3. src/model\_train.py
         1. It will call the backtester framework you implement in the backtest.py
         2. The following method need to be implemented
            1. def backtest\_models(tested\_strategy=BaseStrategy)
   4. src/model/model\_lasso.py, model\_CNN.py, etc.....
   5. src/alpha/alpha\_SWING.py, etc ......
   6. src/alpha/alpha\_verification/alpha\_SWING\_screenshot.png, etc.....
   7. src/run\_strategy.py
      1. The following method need to be implemented.
         1. def run\_strategy(symbol, interval, model\_type)