VIETNAM NATIONAL UNIVERSITY - HCMC
UNIVERSITY OF ECONOMICS AND LAW
FACULTY OF FINANCE AND BANKING

FINAL PROJECT

PERSONAL CUSTOMIZED TECHNICAL ANALYSIS CHART FOR CRYPTO, US STOCK, VN STOCK USING PYTHON

Lecturer:

Mr. Ngo Phu Thanh

Author:
Le Nguyen Phu Loc
K194141730

Ho Chi Minh city, January 2022

Table of Contents

A. Project Overview

1. Indicator

- Indicators are statistics used to measure current conditions as well as to forecast financial or economic trends.
- In the world of investing, indicators typically refer to technical chart patterns deriving from the price, volume, or open interest of a given security. Common technical indicators include moving averages, moving average convergence divergence (MACD), relative strength index (RSI), and on-balance-volume (OBV).
- In economics, indicators usually refer to pieces of economic data used to measure the overall health of the economy and predict its direction. They include the Consumer Price Index (CPI), Gross Domestic Product (GDP), and unemployment figures.

1.1 SMA

- Simple moving averages calculate the average of a range of prices by the number of periods within that range.
- A simple moving average is a technical indicator that can aid in determining if an asset price will continue or if it will reverse a bull or bear trend.
- A simple moving average can be enhanced as an exponential moving average (EMA) that is more heavily weighted on recent price action.

The formula for SMA is:

$$\text{SMA} = \frac{A_1 + A_2 + \ldots + A_n}{n}$$

where:

A_n = the price of an asset at period n

n = the number of total periods

1.2 SUPER TREND

- A Super Trend is a trend following indicator similar to moving averages. It is plotted on price and the current trend can simply be determined by its placement vis-a-vis price. It is a very simple indicator and is constructed with the help of just two parameters- period and multiplier.

- When we construct the Supertrend indicator strategy, the default parameters are 10 for Average True Range (ATR) and 3 for its multiplier. The average true range (ATR) plays a key role in 'Supertrend' as the indicator uses ATR to compute its value and it signals the degree of price volatility.

- **The supertrend indicator calculation is shown below:**
- Up = (high + low / 2 + multiplier x ATR
- Down = (high + low) / 2 – multiplier x ATR
- **Calculation of Average True Range:**
- [(Prior ATR x 13) + Current TR] / 14
- Here, 14 indicates a period. Hence, the ATR is derived by multiplying the previous ATR with 13. Add the latest TR and divide it by period.
- Thus, ATR plays an important role in the supertrend technical analysis indicator.

1.3 MACD

- Moving average convergence divergence (MACD) is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA.
- MACD triggers technical signals when it crosses above (to buy) or below (to sell) its signal line.
- The speed of crossovers is also taken as a signal of a market is overbought or oversold.
- MACD helps investors understand whether the bullish or bearish movement in the price is strengthening or weakening.

- MACD=12-Period EMA − 26-Period EMA
- MACD is calculated by subtracting the long-term EMA (26 periods) from the short-term EMA (12 periods). An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points.
- The exponential moving average is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average (SMA), which applies an equal weight to all observations in the period.

2.1 Crypto Currency

- A cryptocurrency is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced

by a disparate network of computers. A defining feature of cryptocurrencies is that they are generally not issued by any central authority, rendering them theoretically immune to government interference or manipulation.

2.2 Stock

- Stock markets are venues where buyers and sellers meet to exchange equity shares of public corporations.

- Stock markets are vital components of a free-market economy because they enable democratized access to trading and exchange of capital for investors of all kinds.

- They perform several functions in markets, including efficient price discovery and efficient dealing.

- In the United States, the stock market is regulated by the Securities and Exchange Commission (SEC) and local regulatory bodies.

B. Objective of Project

1. Project's Purpose

- I created this personal project to simulate the TradingView chart for personal use, as TradingView only enables you to utilize up to three indicators while using it for free. The indicators utilized are ones that I frequently use for swing traders and long-term investors.
- In terms of the crypto market, the indications, in my opinion, can only be employed with high cap coins. In other words, Binance listed coins will be crypto currencies that may use the indicator to examine the chart and interpret the indications to swing trade. Low cap coins, aka shitcoins, will not use the indication since the crypto market's price volatility is so high that lowcap coins can establish a new peak in a few seconds and then crash to zero in a few seconds. Not to mention the fraudulent projects that cause us to lose money, such as the Squid Game currency based on a Korean TV show, thus these indicators will only be utilized for large coins.
- Initially, I intended to create only a cryptocurrency chart, but I learned that it is feasible to expand the project to include both US and Vietnamese equities, which I typically use for swing trading. As a result, I updated the code and included two new functions to utilize the charts of the US stock market and stock vn.
- Users will have to design their own indicator calculation functions and charting functions if utilized for personal reasons. Here are several signs that I frequently use:
- SMA: I use it to examine if I should purchase or sell based on indicator lines. My own method, which I frequently employ, is to solely purchase using the rule of SMA9 > SMA20 > SMA50 > SMA200.
- Super Trend: I use it to forecast whether the trend will rise or fall.

- MACD: I also use it to forecast trends, but this indicator will be a deciding factor in whether I continue to monitor other indicators. If the blue MACD exceeds the red MACD, I will continue to monitor the above indicators before making a purchase decision.
- The unique feature is that you may utilize functions to create charts for any cryptocurrency or asset you choose.

C. Python Code Explanation

1. Install the library I need to use

```
# pip install git+https://github.com/vuthanhdatt/vnstock-data-python.git
# pip install python-binance
# pip install pandas_ta
# pip install plotly==5.5.0
# pip install yfinance
```

- The first library is from my friend project, Vu Thanh Dat, he created it to get historical data from VN stock market. I also try to use the VNQuant from Mr. Pham Dinh Khanh but the installation is very confused which I failed to install the library.
- Python-binance for getting crypto's historical data.
- Pandas-ta for indicator calculating
- Plotly for plotting chart, the special thing of this library is you could interact with the chart to zoom in, zoom out, export .jpg file of the plot, inspect the value on the charts,…
- Yfinance for getting US stock historical data.

- Next, I create my Binance api key which I will need to get the data from Binance.

```
apikey = 'rz1iJcFceAwjl1CpFbM3axkx0fRDb6vhFQ2xZkqma2xCRysgKpH6v8qkRlMJX9R0'
secret = 'AMI2gLDPCSe84a9AIN2M9eD3rXkvgwvAOgSANBRG8Zd9HEWtr2Hknfyw4XArrhbZ'
```

2. Import the library

```
from binance import Client, ThreadedWebsocketManager, ThreadedDepthCacheManager
import pandas as pd
import pandas_ta as ta
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import yfinance as yf
from datetime import date as dt
```

- Setting for Vu Thanh Dat's library:

```
from vnstock_data.all_exchange import VnStock
COOKIES={"vts_usr_lg":"ABCDEF","language": "en-
US","__RequestVerificationToken":"GhijKL"}
vndata = VnStock(COOKIES)
```

3. Get data and preprocess data

- Set the api key for Binance API:

```
client = Client(apikey, secret)
tickers= client.get_all_tickers()
ticker_df= pd.DataFrame(tickers)
ticker_df.set_index('symbol',inplace=True)
```

- Get all the ticker and add it to a dataframe:

```
client = Client(apikey, secret)
tickers= client.get_all_tickers()
ticker_df= pd.DataFrame(tickers)
# ticker_df.set_index('symbol',inplace=True)
ticker_df
```

✓ 0.4s

	symbol	price
0	ETHBTC	0.06871100
1	LTCBTC	0.00304400
2	BNBBTC	0.01042400
3	NEOBTC	0.00050500
4	QTUMETH	0.00227000
...	...	...
1905	ROSETRY	4.35500000
1906	SCRTUSDT	5.38500000
1907	API3BTC	0.00011504
1908	API3BUSD	4.03100000
1909	API3USDT	4.02400000

1910 rows × 2 columns

- Set the symbol as index of the dataframe for ticker call function by using the ticker:

```
client = Client(apikey, secret)
tickers= client.get_all_tickers()
ticker_df= pd.DataFrame(tickers)
ticker_df.set_index('symbol',inplace=True)
ticker_df
```

✓ 0.4s

symbol	price
ETHBTC	0.06857900
LTCBTC	0.00304400
BNBBTC	0.01042100
NEOBTC	0.00050500
QTUMETH	0.00227300
...	...
ROSETRY	4.35700000
SCRTUSDT	5.39700000
API3BTC	0.00011606
API3BUSD	4.09700000
API3USDT	4.07800000

1910 rows × 1 columns

- Now I could get the price from the symbol as I set:

```
client = Client(apikey, secret)
tickers= client.get_all_tickers()
ticker_df= pd.DataFrame(tickers)
ticker_df.set_index('symbol',inplace=True)
ticker_df.loc['BTCUSDT']
```

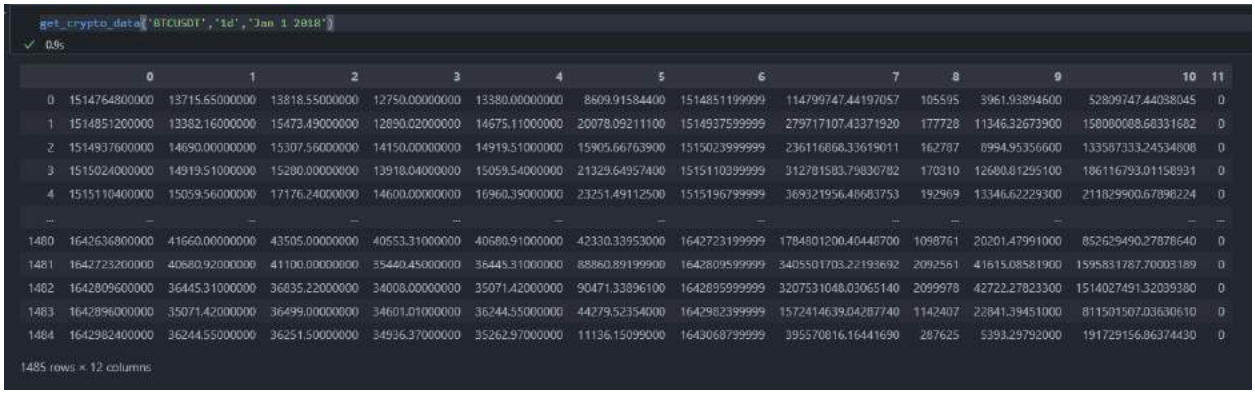
✓ 0.4s

```
price    35210.45000000
Name: BTCUSDT, dtype: object
```

- Then I define a function to get the historical data from Binance API, all I need to input the symbol, the interval of the data like 1 minute, 1 day, 1 month,… And last is the data start time:

```
def get_crypto_data(symbol, interval, starttime):
    df = pd.DataFrame(client.get_historical_klines(symbol, interval, starttime))
    return df
```

- The function will return a dataframe like this:



```
get_crypto_data('BTCUSDT','1d','Jan 1 2018')
```

✓ 0.9s

	0	1	2	3	4	5	6	7	8	9	10	11
0	1514764800000	13715.65000000	13818.55000000	12750.00000000	13380.00000000	8609.91584400	1514851199999	114799747.44197057	105595	3961.93894600	52809747.44038045	0
1	1514851200000	13382.16000000	15473.49000000	12890.02000000	14675.11000000	20078.09211100	1514937599999	279717107.43371920	177728	11346.32673900	158080088.68331682	0
2	1514937600000	14690.00000000	15307.56000000	14150.00000000	14919.51000000	15905.66763900	1515023999999	236116868.33619011	162787	8994.95356600	133587333.24534808	0
3	1515024000000	14919.51000000	15280.00000000	13918.04000000	15059.54000000	21329.64957400	1515110399999	312781583.79830782	170310	12680.81295100	186116793.01158931	0
4	1515110400000	15059.56000000	17176.24000000	14600.00000000	16960.39000000	23251.49112500	1515196799999	369321956.48683753	192969	13346.62229300	211829900.67898224	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1480	1642636800000	41660.00000000	43505.00000000	40553.31000000	40680.91000000	42330.33953000	1642723199999	1784801200.40448700	1098761	20201.47991000	852629490.27878640	0
1481	1642723200000	40680.92000000	41100.00000000	35440.45000000	36445.31000000	88860.89199900	1642809599999	3405501703.22193692	2092561	41615.08581900	1595831787.70003189	0
1482	1642809600000	36445.31000000	36835.22000000	34008.00000000	35071.42000000	90471.33896100	1642895999999	3207531048.03065140	2099978	42722.27823300	1514027491.32039380	0
1483	1642896000000	35071.42000000	36499.00000000	34601.01000000	36244.55000000	44279.52354000	1642982399999	1572414639.04287740	1142407	22841.39451000	811501507.03630610	0
1484	1642982400000	36244.55000000	36251.50000000	34936.37000000	35262.97000000	11136.15099000	1643068799999	395570816.16441690	287625	5393.29792000	191729156.86374430	0

1485 rows × 12 columns

- Then I preprocessed the data:

```
def get_crypto_data(symbol, interval, starttime):
    df = pd.DataFrame(client.get_historical_klines(symbol, interval, starttime))
    df.columns = ['Open Time','Open','High','Low','Close','Volume','Close
Time','Quote Asset Volume','Number of Trades','TB Base Volume','TB Quote
Volume','Ignore']
    df['Open Time'] = pd.to_datetime(df['Open Time']/1000, unit='s')
    df['Close Time'] = pd.to_datetime(df['Close Time']/1000, unit='s')
    numeric_columns= ['Open','High','Low','Close','Volume','Quote Asset Volume','TB
Base Volume','TB Quote Volume']
    df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, axis=1)
    df= df[['Close Time','Open','High','Low','Close','Volume']]
    df['Diff']= df['Close'] - df['Open']
    df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

  - I rename the columns to call it later.
  - I converte the time to datetime and correct it to the right time data type that I will use.
  - Create a list of dataframe need to convert to numeric and convert it.
  - Configure the dataframe to only columns I need.
  - Create a column calculating different between Close price and Open price for the volume bar color.
  - The final dataframe will look like this:

```python
def get_crypto_data(symbol, interval, starttime):
    df = pd.DataFrame(client.get_historical_klines(symbol, interval, starttime))
    df.columns = ['Open Time','Open','High','Low','Close','Volume','Close Time','Quote Asset Volume','Number of Trades','TB Base Volume','TB Quote Volume','Ignore']
    df['Open Time'] = pd.to_datetime(df['Open Time']/1000, unit='s')
    df['Close Time'] = pd.to_datetime(df['Close Time']/1000, unit='s')
    numeric_columns= ['Open','High','Low','Close','Volume','Quote Asset Volume','TB Base Volume','TB Quote Volume']
    df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, axis=1)
    df= df[['Close Time','Open','High','Low','Close','Volume']]
    df['Diff']= df['Close'] - df['Open']
    df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

✓ 0.3s

```python
get_crypto_data('BTCUSDT','1d','Jan 1 2018')
```

✓ 0.5s

	Close Time	Open	High	Low	Close	Volume	Diff	Color
0	2018-01-01 23:59:59.999000064	13715.65	13818.55	12750.00	13380.00	8609.915844	-335.65	red
1	2018-01-02 23:59:59.999000064	13382.16	15473.49	12890.02	14675.11	20078.092111	1292.95	gray
2	2018-01-03 23:59:59.999000064	14690.00	15307.56	14150.00	14919.51	15905.667639	229.51	gray
3	2018-01-04 23:59:59.999000064	14919.51	15280.00	13918.04	15059.54	21329.649574	140.03	gray
4	2018-01-05 23:59:59.999000064	15059.56	17176.24	14600.00	16960.39	23251.491125	1900.83	gray
...	...	...	...	...	...	...	...	...
1480	2022-01-20 23:59:59.999000064	41660.00	43505.00	40553.31	40680.91	42330.339530	-979.09	red
1481	2022-01-21 23:59:59.999000064	40680.92	41100.00	35440.45	36445.31	88860.891999	-4235.61	red
1482	2022-01-22 23:59:59.999000064	36445.31	36835.22	34008.00	35071.42	90471.338961	-1373.89	red
1483	2022-01-23 23:59:59.999000064	35071.42	36499.00	34601.01	36244.55	44279.523540	1173.13	gray
1484	2022-01-24 23:59:59.999000064	36244.55	36251.50	33029.54	33769.02	34385.467295	-2475.53	red

1485 rows × 8 columns

- For US stock and VN stock, I just need to change the data input and do all the other step like I did with crypto.
- The function will output a dataframe like below:

```python
def get_stock_data(symbol, interval, starttime):
    df=pd.DataFrame(yf.download(symbol,interval=interval,start=starttime,progress=False,auto_adjust=True))
    # df=pd.DataFrame(yf.download(symbol,interval=interval,start=starttime,progress=False,auto_adjust=True)).dropna()
    # df['Diff']= df['Close'] - df['Open']
    # df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    # df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

✓ 0.3s

```python
get_stock_data('TSLA','1d','2015-1-1')
```

✓ 1.4s

Date	Open	High	Low	Close	Volume
2014-12-31	44.618000	45.136002	44.450001	44.481998	11487500
2015-01-02	44.574001	44.650002	42.652000	43.862000	23822000
2015-01-05	42.910000	43.299999	41.431999	42.018002	26842500
2015-01-06	42.012001	42.840000	40.841999	42.256001	31309500
2015-01-07	42.669998	42.956001	41.956001	42.189999	14842000
...	...	...	...	...	...
2022-01-14	1019.880005	1052.000000	1013.380005	1049.609985	24246600
2022-01-18	1026.609985	1070.790039	1016.059998	1030.510010	22247800
2022-01-19	1041.709961	1054.670044	995.000000	995.650024	25147500
2022-01-20	1009.729980	1041.660034	994.000000	996.270020	23496200
2022-01-21	996.340027	1004.549988	940.500000	943.900024	34126500

1778 rows × 5 columns

--

- Next I use a simple function to inspect the all dataframe if there is any NaN:

- Then I drop the NaN in all dataframe and create a Diff column for volume bar color

```
def get_stock_data(symbol, interval, starttime):
          df=pd.DataFrame(yf.download(symbol,interval=interval,start=starttime,progr
ess=False,auto_adjust=True)).dropna()
          df['Diff']= df['Close'] - df['Open']
          df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
          df.loc[df['Diff'] <= 0, 'Color'] = 'red'
   return df
```

- The final function will ouput a dataframe like this:

```python
def get_stock_data(symbol, interval, starttime):
    df=pd.DataFrame(yf.download(symbol,interval=interval,start=starttime,progress=False,auto_adjust=True)).dropna()
    df['Diff']= df['Close'] - df['Open']
    df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

✓ 0.4s

```python
get_stock_data('TSLA','1D','2020-1-1')
```

✓ 4.6s

Date	Open	High	Low	Close	Volume	Diff	Color
2019-12-31	81.000000	84.258003	80.416000	83.666000	51428500	2.666000	gray
2020-01-02	84.900002	86.139999	84.342003	86.052002	47660500	1.152000	gray
2020-01-03	88.099998	90.800003	87.384003	88.601997	88892500	0.501999	gray
2020-01-06	88.094002	90.311996	88.000000	90.307999	50665000	2.213997	gray
2020-01-07	92.279999	94.325996	90.671997	93.811996	89410500	1.531998	gray
...	...	...	...	...	...	...	...
2022-01-14	1019.880005	1052.000000	1013.380005	1049.609985	24246600	29.729980	gray
2022-01-18	1026.609985	1070.790039	1016.059998	1030.510010	22247800	3.900024	gray
2022-01-19	1041.709961	1054.670044	995.000000	995.650024	25147500	-46.059937	red
2022-01-20	1009.729980	1041.660034	994.000000	996.270020	23496200	-13.459961	red
2022-01-21	996.340027	1004.549988	940.500000	943.900024	34126500	-52.440002	red

520 rows × 7 columns

- For VN stock, the function need endtime input so I use an another function to get the endtime always up to date and don’t need to input the endtime.

```python
def get_vnstock_data(symbol, starttime, interval):
    df= vndata.price(symbol, starttime, dt.today())
    # df= vndata.price(symbol, starttime, dt.today()).dropna()
    # df= df.reindex(index=df.index[::-1])
    # if interval == '1W':
    #     df= df.resample('1W').mean()
    # elif interval == '1M':
    #     df= df.resample('1M').mean()
    # df['Diff']= df['Close'] - df['Open']
    # df['Adj Open']= (df['Open']*df['Adj Close'])/df['Close']
    # df['Adj High']= (df['High']*df['Adj Close'])/df['Close']
    # df['Adj Low']= (df['Low']*df['Adj Close'])/df['Close']
    # df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    # df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

✓ 0.3s

```python
get_vnstock_data('VCB', '2020-1-1','1D')
```

✓ 0.2s

Date	High	Low	Open	Close	Volume	Adj Close	Average	High-Low
2022-01-24	93500	88200	89400	93000	2728300	93000	90779	5300
2022-01-21	90000	86100	86900	89200	2611300	89200	88257	3900
2022-01-20	87500	83100	85200	87100	2794300	87100	85664	4400
2022-01-19	87500	85000	87500	86800	1414800	86800	86327	2500
2022-01-18	87600	84000	85500	87600	1735600	87600	86353	3600
...	...	...	...	...	...	...	...	...
2020-01-08	87900	86200	86900	87000	842280	66800	86973	1700
2020-01-07	87900	85400	87000	87800	1122600	67500	86537	2500
2020-01-06	89500	87500	89200	87500	880110	67200	88540	2000
2020-01-03	91800	89900	91500	89900	536130	69100	90806	1900
2020-01-02	91400	89700	90200	90800	386290	69800	90686	1700

517 rows × 8 columns

- As we can see, the date columns is reverse so I reindex it to the right order then I create a condition to calculate the data for different interval and drop the NaN like I did with the US Stock. VN stock use the adjust price so I calculate all the price as adjusted price for later use and create a color column for the volume bar.

- The final function will ouput a dataframe like this:

```python
def get_vnstock_data(symbol, starttime, interval):
    df= vndata.price(symbol, starttime, dt.today())
    # df= vndata.price(symbol, starttime, dt.today()).dropna()
    df= df.reindex(index=df.index[::-1])
    if interval == '1W':
        df= df.resample('1W').mean()
    elif interval == '1M':
        df= df.resample('1M').mean()
    df['Diff']= df['Close'] - df['Open']
    df['Adj Open']= (df['Open']*df['Adj Close'])/df['Close']
    df['Adj High']= (df['High']*df['Adj Close'])/df['Close']
    df['Adj Low']= (df['Low']*df['Adj Close'])/df['Close']
    df.loc[df['Diff'] >= 0, 'Color'] = 'gray'
    df.loc[df['Diff'] <= 0, 'Color'] = 'red'
    return df
```

✓ 0.4s

```python
get_vnstock_data('VCB','2015-1-1','1d')
```

✓ 0.6s

Date	High	Low	Open	Close	Volume	Adj Close	Average	High-Low	Diff	Adj Open	Adj High	Adj Low	Color
2015-01-05	32100	31400	31700	31900	310010	16700	31799.0	700	200	16595.297806	16804.702194	16438.244514	gray
2015-01-06	34000	31900	31900	34000	1684600	17800	33382.0	2100	2100	16700.588235	17800.000000	16700.588235	gray
2015-01-07	34600	33100	34100	33600	1564340	17600	34028.0	1500	-500	17861.904762	18123.809524	17338.095238	red
2015-01-08	35000	33600	33700	34600	1004240	18100	34611.0	1400	900	17629.190751	18309.248555	17576.878613	gray
2015-01-09	37000	34900	34900	36900	2533981	19300	36541.0	2100	2000	18253.929539	19352.303523	18253.929539	gray
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-01-18	87600	84000	85500	87600	1735600	87600	86353.0	3600	2100	85500.000000	87600.000000	84000.000000	gray
2022-01-19	87500	85000	87500	86800	1414800	86800	86327.0	2500	-700	87500.000000	87500.000000	85000.000000	red
2022-01-20	87500	83100	85200	87100	2794300	87100	85664.0	4400	1900	85200.000000	87500.000000	83100.000000	gray
2022-01-21	90000	86100	86900	89200	2611300	89200	88257.0	3900	2300	86900.000000	90000.000000	86100.000000	gray
2022-01-24	93500	88200	89400	93000	2728300	93000	90779.0	5300	3600	89400.000000	93500.000000	88200.000000	gray

1764 rows × 13 columns

4. Calculating Indicator

- Next to create a function to calculate the indicator:

```
def get_crypto_indi(symbol, interval, starttime):
    df= get_crypto_data(symbol, interval, starttime)
    dsma9 = ta.sma(df["Close"], length=9)
    return dsma9
```

✓ 0.3s

```
get_crypto_indi('BTCUSDT','1d','Jan 1 2018')
```

✓ 0.9s

```
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN
             ...
1480    42508.148889
1481    41679.554444
1482    40847.477778
1483    40090.210000
1484    39200.040000
Name: SMA_9, Length: 1485, dtype: float64
```

- To calculate we need to get the data and target the data in the dataframe we need to calculate, in this case, the input data is the close price for crypto and US Stock. The input data for VN Stock will be the adjusted price.
- To calculate different indicators, we will need to change the length of SMA and input the right data for the indicators.
- At the end I create a column in the MACD's dataframe for the MACD histogram color, which indicate by the distance between the MACD and its signal line.

```
def get_crypto_indi(symbol, interval, starttime):
        df= get_crypto_data(symbol, interval, starttime)
        dsma9 = ta.sma(df["Close"], length=9)
        dsma20 = ta.sma(df["Close"], length=20)dsma50 = ta.sma(df["Close"],
length=50)
        dsma200 = ta.sma(df["Close"], length=200)
        dspt=
ta.supertrend(high=df['High'],low=df['Low'],close=df['Close'],period=10,multiplie
r=3)
        dmacd= ta.macd(df["Close"], length=9)
        dmacd.loc[dmacd['MACDh_12_26_9'] >= 0, 'Color'] = 'gray'
```

```
    dmacd.loc[dmacd['MACDh_12_26_9'] <= 0, 'Color'] = 'red'
    return dsma9,dsma20,dsma50,dsma200,dspt,dmacd
```

- The function will return the indicator and to indicate the indicator I just need to call it by its position.

```
get_crypto_indi('BTCUSDT','1d','Jan 1 2018')[0]
✓ 1.1s
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN
            ...
1480    42508.148889
1481    41679.554444
1482    40847.477778
1483    40090.210000
1484    39038.500000
Name: SMA_9, Length: 1485, dtype: float64
```

```
get_crypto_indi('BTCUSDT','1d','Jan 1 2018')[1]
✓ 2.1s
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN
            ...
1480    43302.8170
1481    42738.9500
1482    42128.2120
1483    41618.1345
1484    41006.5340
Name: SMA_20, Length: 1485, dtype: float64
```

- Function for calculating indicator of US Stock and VN Stock:

```
def get_stock_indi(symbol, interval, starttime):
    df= get_stock_data(symbol, interval, starttime)
    dsma9 = ta.sma(df["Close"], length=9)
    dsma20 = ta.sma(df["Close"], length=20)
    dsma50 = ta.sma(df["Close"], length=50)
    dsma200 = ta.sma(df["Close"], length=200)
```

```
        dspt=
ta.supertrend(high=df['High'],low=df['Low'],close=df['Close'],period=10,multiplie
r=3)
        dmacd= ta.macd(df["Close"], length=9)
        dmacd.loc[dmacd['MACDh_12_26_9'] >= 0, 'Color'] = 'gray'
        dmacd.loc[dmacd['MACDh_12_26_9'] <= 0, 'Color'] = 'red'
        return dsma9,dsma20,dsma50,dsma200,dspt,dmacd
def get_vnstock_indi(symbol, starttime, interval):
        df= get_vnstock_data(symbol, starttime, interval)
        dsma9 = ta.sma(df["Adj Close"], length=9)
        dsma20 = ta.sma(df["Adj Close"], length=20)
        dsma50 = ta.sma(df["Adj Close"], length=50)
        dsma200 = ta.sma(df["Adj Close"], length=200)
        dspt= ta.supertrend(high=df['Adj High'],low=df['Adj Low'],close=df['Adj
Close'],period=10,multiplier=3)
        dmacd= ta.macd(df["Adj Close"], length=9)
        dmacd.loc[dmacd['MACDh_12_26_9'] >= 0, 'Color'] = 'gray'
        dmacd.loc[dmacd['MACDh_12_26_9'] <= 0, 'Color'] = 'red'
        return dsma9,dsma20,dsma50,dsma200,dspt,dmacd
```

5. Plot chart functions

- To recreate the TradingView chart, I need to make a subplot with 3 rows but with the same axis for all 3 rows, set the rows spacing, set the rows width,set the chart theme, set the chart layout, add title to rows, plot the candle stick, add volume bar without legend:

```
def plot_crypto_chart(symbol, interval, starttime):
        df=get_crypto_data(symbol,interval,starttime)
# Plot candlesticks chart
        df_C = make_subplots(rows=3, cols=1, shared_xaxes=True,
vertical_spacing=0.03,row_width=[0.7,0.2,0.7],subplot_titles=(symbol + ' '+ interval
+ ' with SMA, SuperTrend', 'Volume',symbol + ' ' + interval +' with MACD'))
```

```
          df_C.add_trace(go.Candlestick(x = df['Close Time'], open= df['Open'],
high= df['High'], low= df['Low'], close= df['Close']),row = 1, col = 1)
          df_C.add_trace(go.Bar(x = df['Close Time'], y=
df['Volume'],showlegend=False,marker={'color':df['Color']}),row = 2, col = 1)
# Update layout
for i in range(3,0,-1):
          df_C.update_xaxes(row=i, col=1, rangeslider_visible=False)         #
Remove Range Slider
          df_C.update_layout(template='plotly_dark')
          df_C.update_layout(bargap=0.2)
          df_C.update_layout(height=1000, width=1500)
          df_C.update_traces(name='Price', selector=dict(type='candlestick'))
          return df_C.show()
```

- The chart will look like this before update the layout and after update the layout:

- Then I add the indicator and setting them to my personal color, adding name for the indicator, turn off the legends of the bars like volume, MACD histogram.

```
# Adding SMA
   df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[0], line = dict(color= '#ffffff'), name= 'SMA9'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[1], line = dict(color= '#fff176'), name= 'SMA20'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[2], line = dict(color= '#fbc02d'), name= 'SMA50'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[3], line = dict(color= '#f57f17'), name= 'SMA200'),row=1,
col=1)
    # Adding Super Trend
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[4]['SUPERTs_7_3.0'], line = dict(color= '#f34f50'), name= 'Sell
Super Trend'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[4]['SUPERTl_7_3.0'], line = dict(color= '#4caf50'), name= 'Buy
Super Trend'),row=1, col=1)
    #MACD
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[5]['MACD_12_26_9'], line = dict(color= '#1f77b4'), name=
'MACD'),row=3, col=1)
    df_C.add_trace(go.Bar(x = df['Close Time'], y= get_crypto_indi(symbol, interval,
starttime)[5]['MACDh_12_26_9'],showlegend=False,marker={'color':get_crypto_indi(symb
ol, interval, starttime)[5]['Color']}),row = 3, col = 1)
    df_C.add_trace(go.Scatter(x=df['Close Time'], y= get_crypto_indi(symbol,
interval, starttime)[5]['MACDs_12_26_9'], line = dict(color= '#d62728'), name= 'MACD
Signal'),row=3, col=1)
```

- The final chart will look like this:

- There are many things I can do with this chart like zoom in, zoom out, check the value,…

- Now I can use my chart instead of TradingView for a better trading process. I did the same for US Stock and VN Stock.
- Code for US Stock and VN Stock:

```
def plot_stock_chart(symbol, interval, starttime):
      df = get_stock_data(symbol,interval,starttime)
# Plot candlesticks chart
      df_C = make_subplots(rows=3, cols=1, shared_xaxes=True,
vertical_spacing=0.03,row_width=[0.7,0.2,0.7],subplot_titles=(symbol + ' '+
interval + ' with SMA, SuperTrend', 'Volume',symbol + ' ' + interval +' with
MACD'))
      df_C.add_trace(go.Candlestick(x = df.index, open= df['Open'], high=
df['High'], low= df['Low'], close= df['Close']),row = 1, col = 1)
      df_C.add_trace(go.Bar(x = df.index, y=
df['Volume'],showlegend=False,marker={'color':df['Color']}),row = 2, col = 1)
# Adding SMA
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[0], line = dict(color= '#ffffff'), name= 'SMA9'),row=1, col=1)
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[1], line = dict(color= '#fff176'), name= 'SMA20'),row=1, col=1)
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[2], line = dict(color= '#fbc02d'), name= 'SMA50'),row=1, col=1)
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[3], line = dict(color= '#f57f17'), name= 'SMA200'),row=1, col=1)
# Adding Super Trend
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[4]['SUPERTs_7_3.0'], line = dict(color= '#f34f50'), name= 'Sell Super
Trend'),row=1, col=1)
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[4]['SUPERTl_7_3.0'], line = dict(color= '#4caf50'), name= 'Buy Super
Trend'),row=1, col=1)
#MACD
```

```
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[5]['MACD_12_26_9'], line = dict(color= '#1f77b4'), name=
'MACD'),row=3, col=1)
      df_C.add_trace(go.Bar(x = df.index, y= get_stock_indi(symbol, interval,
starttime)[5]['MACDh_12_26_9'],showlegend=False,marker={'color':get_stock_indi(sy
mbol, interval, starttime)[5]['Color']}),row = 3, col = 1)
      df_C.add_trace(go.Scatter(x=df.index, y= get_stock_indi(symbol, interval,
starttime)[5]['MACDs_12_26_9'], line = dict(color= '#d62728'), name= 'MACD
Signal'),row=3, col=1)
# Update layout
      for i in range(3,0,-1):
      df_C.update_xaxes(row=i, col=1, rangeslider_visible=False)         #
Remove Range Slider
      df_C.update_layout(template='plotly_dark')
      df_C.update_layout(height=1000, width=1500)
      df_C.update_traces(name='Price', selector=dict(type='candlestick'))
      return df_C.show()
```

```
def plot_vnstock_chart(symbol, starttime, interval):
    df = get_vnstock_data(symbol,starttime, interval)
    # Plot candlesticks chart
    df_C = make_subplots(rows=3, cols=1, shared_xaxes=True,
vertical_spacing=0.03,row_width=[0.7,0.2,0.7],subplot_titles=(symbol+ ' ' +
interval + ' with SMA, SuperTrend', 'Volume', symbol + ' ' + interval +' with
MACD'))
    df_C.add_trace(go.Candlestick(x = df.index, open= df['Adj Open'], high=
df['Adj High'], low= df['Adj Low'], close= df['Adj Close']),row = 1, col = 1)
    df_C.add_trace(go.Bar(x = df.index, y=
df['Volume'],showlegend=False,marker={'color':df['Color']}),row = 2, col = 1)

    # Adding SMA
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[0], line = dict(color= '#ffffff'), name= 'SMA9'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[1], line = dict(color= '#fff176'), name= 'SMA20'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[2], line = dict(color= '#fbc02d'), name= 'SMA50'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[3], line = dict(color= '#f57f17'), name= 'SMA200'),row=1, col=1)

    # Adding Super Trend
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[4]['SUPERTs_7_3.0'], line = dict(color= '#f34f50'), name= 'Sell Super
Trend'),row=1, col=1)
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[4]['SUPERTl_7_3.0'], line = dict(color= '#4caf50'), name= 'Buy Super
Trend'),row=1, col=1)
```

```
    #MACD
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[5]['MACD_12_26_9'], line = dict(color= '#1f77b4'), name= 'MACD'),row=3,
col=1)
    df_C.add_trace(go.Bar(x = df.index, y= get_vnstock_indi(symbol, starttime,
interval)[5]['MACDh_12_26_9'],showlegend=False,marker={'color':get_vnstock_indi(s
ymbol, starttime, interval)[5]['Color']}),row = 3, col = 1)
    df_C.add_trace(go.Scatter(x=df.index, y= get_vnstock_indi(symbol, starttime,
interval)[5]['MACDs_12_26_9'], line = dict(color= '#d62728'), name= 'MACD
Signal'),row=3, col=1)

    # Update layout
    for i in range(3,0,-1):
        df_C.update_xaxes(row=i, col=1, rangeslider_visible=False)         #
Remove Range Slider
    df_C.update_layout(template='plotly_dark')
    # df_C.update_layout(bargap=1)
    df_C.update_layout(height=1000, width=1500)
    df_C.update_traces(name='Price', selector=dict(type='candlestick'))
    return df_C.show()
```

- Then I create a final function for an easier function run:

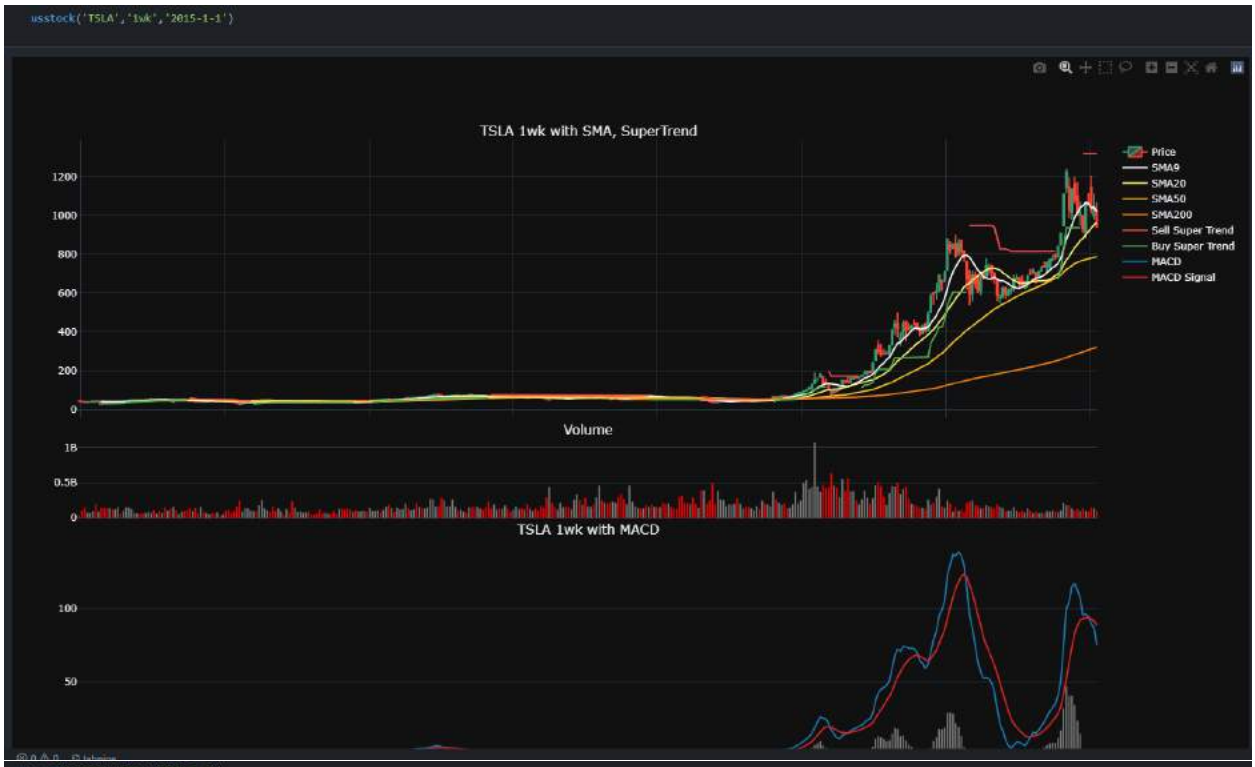
```
def crypto(symbol, interval, starttime):
   get_crypto_data(symbol, interval, starttime)
   get_crypto_indi(symbol, interval, starttime)
   return plot_crypto_chart(symbol, interval, starttime)
def usstock(symbol, interval, starttime):
    get_stock_data(symbol, interval, starttime)
    get_stock_indi(symbol, interval, starttime)
    return plot_stock_chart(symbol, interval, starttime)
    def vnstock(symbol, starttime, interval):
          get_vnstock_data(symbol, starttime, interval)
          get_vnstock_indi(symbol, starttime, interval)
          return plot_vnstock_chart(symbol, starttime, interval)
```

- The final function will ouput like this:

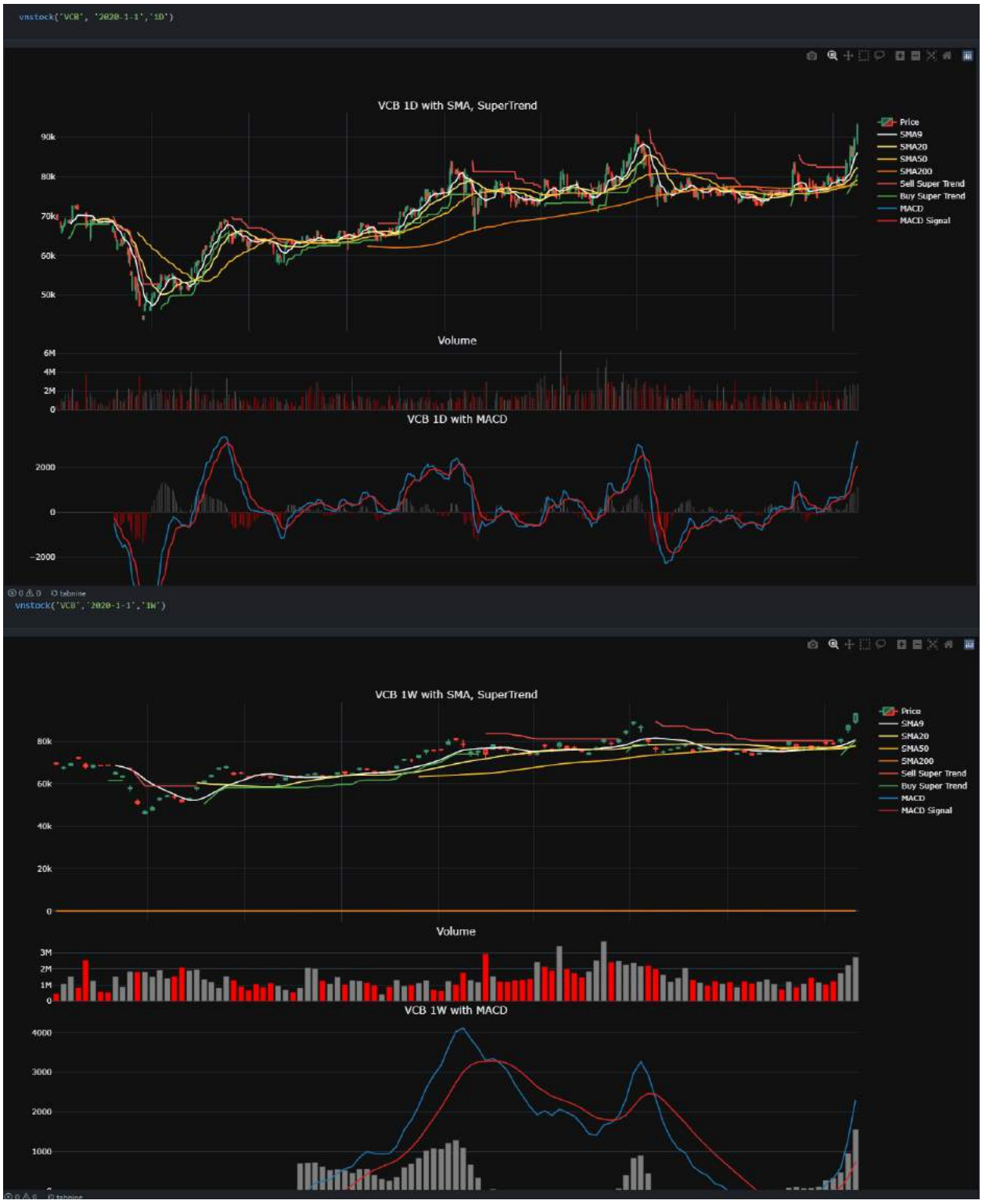

crypto('BTCUSDT','1d','Jan 1 2018')
BTCUSDT 1d with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
Volume
BTCUSDT 1d with MACD
crypto('BTCUSDT','1w','Jan 1 2018')
BTCUSDT 1w with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
Volume
BTCUSDT 1w with MACD


crypto('BTCUSDT','1M','Jan 1 2018')
BTCUSDT 1M with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
Jan 31, 2021
open: 28.92363k
high: 41.95k
low: 28.13k
close: 33.09298k
Volume
BTCUSDT 1M with MACD


usstock('TSLA','1d','2015-1-1')
TSLA 1d with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
Volume
TSLA 1d with MACD


usstock('TSLA','1wk','2015-1-1')
TSLA 1wk with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
1200
1000
800
600
400
200
0
Volume
1B
0.5B
0
TSLA 1wk with MACD
100
50


usstock('TSLA','1mo','2015-1-1')
TSLA 1mo with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
1200
1000
800
600
400
200
0
Volume
2B
1.5B
1B
0.5B
0
TSLA 1mo with MACD
200
150
100
50


vnstock('VCB', '2020-1-1','1D')
VCB 1D with SMA, SuperTrend
Price
SMA9
SMA20
SMA50
SMA200
Sell Super Trend
Buy Super Trend
MACD
MACD Signal
Volume
VCB 1D with MACD
vnstock('VCB','2020-1-1','1W')
VCB 1W with SMA, SuperTrend
Volume
VCB 1W with MACD

- **Please check my code for a better chart viewing.**

REFERENCES

1. What Is an Indicator?. (2022). Retrieved 24 January 2022, from https://www.investopedia.com/terms/i/indicator.asp
2. Simple Moving Average (SMA) Definition. (2022). Retrieved 24 January 2022, from https://www.investopedia.com/terms/s/sma.asp
3. The Powerful Supertrend Indicator :2 strategies for Intraday trading. (2017). Retrieved 24 January 2022, from https://www.elearnmarkets.com/blog/supertrend-indicator-strategy-trading/
4. Moving Average Convergence Divergence (MACD) . (2022). Retrieved 24 January 2022, from https://www.investopedia.com/terms/m/macd.asp
5. BinanceEDA/Binance EDA-Tutorial.ipynb at main · nicknochnack/BinanceEDA. (2022). Retrieved 24 January 2022, from https://github.com/nicknochnack/BinanceEDA/blob/main/Binance%20EDA-Tutorial.ipynb & https://www.youtube.com/watch?v=4aqx5P2Y38U&t=572s
6. Subplots. (2019). Retrieved 24 January 2022, from https://plotly.com/python/subplots/
7. Stock Market | Investopedia. (2022). Retrieved 24 January 2022, from https://www.investopedia.com/terms/s/stockmarket.asp
8. GitHub - twopirllc/pandas-ta: Technical Analysis Indicators - Pandas TA is an easy to use Python 3 Pandas Extension with 130+ Indicators. (2022). Retrieved 24 January 2022, from https://github.com/twopirllc/pandas-ta