Advanced Python Module :

Project: **Trading Bot-BTC :**

Made by:

Boujbair Oussamae

DATA-INE2

Supervised by:

Pr.LAANAYA HICHAM

INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATION

30 Junaury 2022
Rabat, Morocco

Part I

Abstract :

Cryptocurrencies are known for being incredibly volatile, with prices fluctuating dramatically even in the space of minutes. Investors also have the opportunity to take part in cryptocurrency trading around the world and at any hour of the day. Combined, these factors limit the effectiveness of human cryptocurrency trading in several ways.

First, investors in many cases cannot react quickly enough to changes in price to achieve the optimal trades that are theoretically available to them. Slowdowns in exchanges and transaction times further exacerbate this problem. Second, investors can simply not dedicate as much time to the cryptocurrency markets as necessary to always achieve the best trades. Doing so would require round-the-clock monitoring of cryptocurrency exchanges all over the globe.

Fortunately for many investors, there are solutions to these issues. One of the primary solutions is bots. Crypto trading bots are automated software that helps you to buy and sell cryptocurrencies at the correct time. The main goal of these software is to increase revenue and reduce losses and risks. These applications enable you to manage all crypto exchange account in one place. Many such programs allow you to trade for Ethereum, Litecoin, Bitcoin (BTC), and more with ease.

Part II

Introduction :

Crypto bots analyze data, predict risk and buy and sell assets as per their calculations. They watch the market and trade when certain market conditions are met. Most crypto trading bots work by connecting directly to a cryptocurrency exchange. After the bot successfully connects to an exchange, it starts watching the market and waiting for certain events or changes in prices. Once it detects an event, it will either send you a signal to take action or make a decision based on the rules you've defined and take the required action (buy/sell) itself. Some bots can even take into account historical data to make more accurate decisions.

1 Advantages of Bots in trading :

-Bots are used by traders to take advantage of the cryptocurrency markets that trade 24/7 all over the world.

-The advantage bots have over investors is they can react quicker.

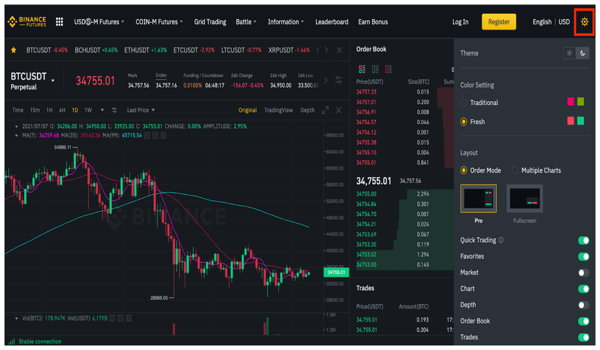
-Meanwhile, most investors also don't have the time to dedicate to always get the best trade—something that bots can do.

-One key type of bot is the arbitrage bot, which looks to take advantage of price discrepancies across exchanges.

2 Binance :

Binance is a cryptocurrency exchange which is the largest exchange in the world in terms of daily trading volume of cryptocurrencies.It was founded in 2017 and is registered in the Cayman Islands.

Binance was founded by Changpeng Zhao, a developer who had previously created high frequency trading software. Binance was initially based in China, but later moved its headquarters out of China following the Chinese government's increasing regulation of cryptocurrency.



Part III Project :

3 Install packages :

3.1 What Is Technical Analysis TA ?

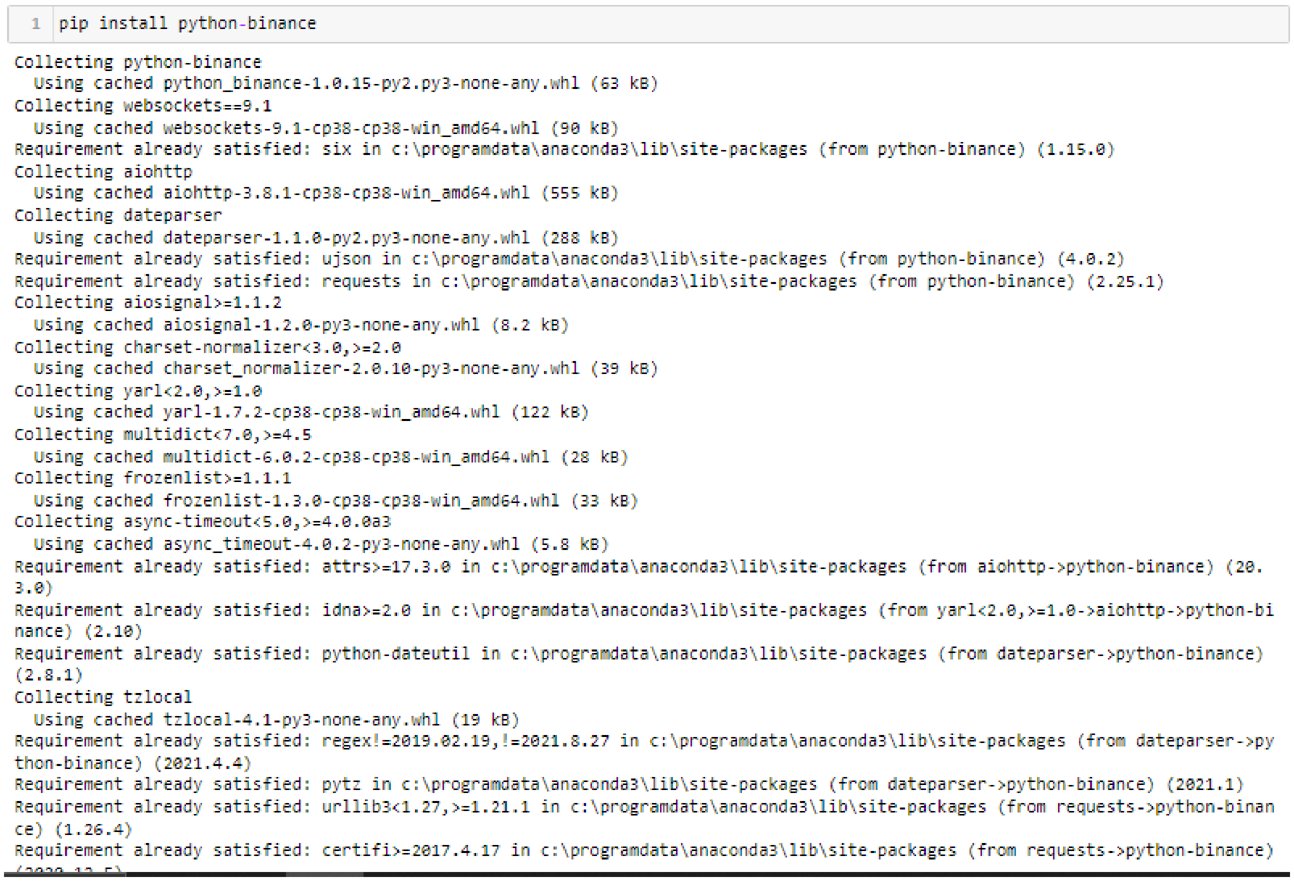
Technical analysis is a trading discipline employed to evaluate investments and identify trading opportunities by analyzing statistical trends gathered from trading activity, such as price movement and volume.

```
pip install ta
```

```
Requirement already satisfied: ta in e:\anaconda\lib\site-packages (0.9.0)
Requirement already satisfied: numpy in e:\anaconda\lib\site-packages (from ta) (1.20.1)
Requirement already satisfied: pandas in e:\anaconda\lib\site-packages (from ta) (1.2.4)
Requirement already satisfied: python-dateutil>=2.7.3 in e:\anaconda\lib\site-packages (from pandas->ta) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in e:\anaconda\lib\site-packages (from pandas->ta) (2021.1)
Requirement already satisfied: six>=1.5 in e:\anaconda\lib\site-packages (from python-dateutil>=2.7.3->pandas->ta) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

3.2 What is Python-Binance ?

The Binance API is a method that allows you to connect to the Binance servers via Python or several other programming languages. With it, you can automate your trading. More specifically, Binance has a RESTful API that uses HTTP requests to send and receive data.



```
pip install python-binance
```

```
Collecting python-binance
  Using cached python_binance-1.0.15-py2.py3-none-any.whl (63 kB)
Collecting websockets==9.1
  Using cached websockets-9.1-cp38-cp38-win_amd64.whl (90 kB)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from python-binance) (1.15.0)
Collecting aiohttp
  Using cached aiohttp-3.8.1-cp38-cp38-win_amd64.whl (555 kB)
Collecting dateparser
  Using cached dateparser-1.1.0-py2.py3-none-any.whl (288 kB)
Requirement already satisfied: ujson in c:\programdata\anaconda3\lib\site-packages (from python-binance) (4.0.2)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from python-binance) (2.25.1)
Collecting aiosignal>=1.1.2
  Using cached aiosignal-1.2.0-py3-none-any.whl (8.2 kB)
Collecting charset-normalizer<3.0,>=2.0
  Using cached charset_normalizer-2.0.10-py3-none-any.whl (39 kB)
Collecting yarl<2.0,>=1.0
  Using cached yarl-1.7.2-cp38-cp38-win_amd64.whl (122 kB)
Collecting multidict<7.0,>=4.5
  Using cached multidict-6.0.2-cp38-cp38-win_amd64.whl (28 kB)
Collecting frozenlist>=1.1.1
  Using cached frozenlist-1.3.0-cp38-cp38-win_amd64.whl (33 kB)
Collecting async-timeout<5.0,>=4.0.0a3
  Using cached async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Requirement already satisfied: attrs>=17.3.0 in c:\programdata\anaconda3\lib\site-packages (from aiohttp->python-binance) (20.
3.0)
Requirement already satisfied: idna>=2.0 in c:\programdata\anaconda3\lib\site-packages (from yarl<2.0,>=1.0->aiohttp->python-bi
nance) (2.10)
Requirement already satisfied: python-dateutil in c:\programdata\anaconda3\lib\site-packages (from dateparser->python-binance)
(2.8.1)
Collecting tzlocal
  Using cached tzlocal-4.1-py3-none-any.whl (19 kB)
Requirement already satisfied: regex!=2019.02.19,!=2021.8.27 in c:\programdata\anaconda3\lib\site-packages (from dateparser->py
thon-binance) (2021.4.4)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from dateparser->python-binance) (2021.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->python-binan
ce) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests->python-binance)
```

4 Libraries used :

Pandas, Numpy, ta, Seaborn, Matplotlib, ...

```python
import pandas as pd
from binance.client import Client
import ta
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

import os
print(os.listdir(r"C:\Users\Oussama\Desktop\python"))
```

```
['.ipynb_checkpoints', 'archive (1).zip', 'athlete_events.csv', 'BTC-USD.csv', 'BTC1.csv', 'c.csv', 'covid-19-datasets-1200x90
0.jpg', 'Covid_19_Dataset_Morocco.csv', 'data_Bitcoin.csv', 'Data_F.csv', 'data_MAR_2.csv', 'df10.csv', 'df9.csv', 'df_BT.csv',
'new3.csv', 'new_cases_analysis.ipynb', 'noc_regions.csv', 'Olympics.ipynb', 'Olympics_Boujbair.ipynb', 'our_data.csv', 'projet
_SC.ipynb', 'sere_temp1.ipynb', 'serie_temp - Copie (2).ipynb', 'serie_temp - Copie.ipynb', 'serie_temp.ipynb', 'Untitled.ipyn
b', 'Untitled1.ipynb', 'Untitled2.ipynb', 'Untitled3.ipynb', 'Untitled4.ipynb', 'Untitled5.ipynb', 'Untitled6.ipynb', 'Untitled
7.ipynb', 'VosQuestions.ipynb']
```

5 Dataset :

5.1 Download Data

Download and retrieve our dataset from the binance platform, this dataset is made up of a table of 12 columns: Time, open, close, high, low, volume .... since 01/01/2018 until the hour code execution.

```python
recup_data = Client().get_historical_klines("BTCUSDT", Client.KLINE_INTERVAL_1HOUR,"01 January 2018")
BTC_data = pd.DataFrame(recup_data, columns = ['Time','Open','High','Low','Close','Volume','close_time','quote_av','trades',
```

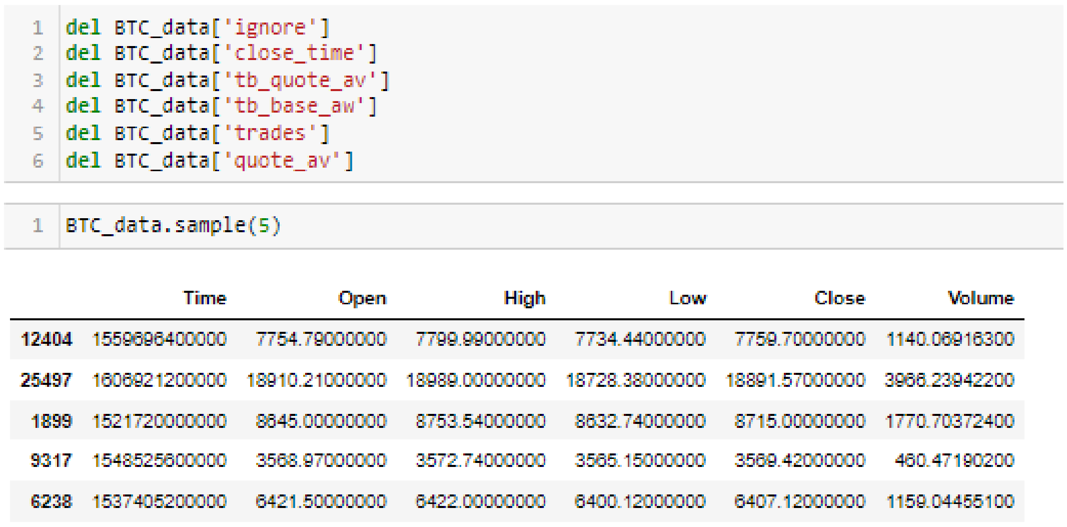
```python
BTC_data
```

	Time	Open	High	Low	Close	Volume	close_time	quote_av	trades	tb_base_a
0	1514764800000	13715.65000000	13715.65000000	13400.01000000	13529.01000000	443.35619900	1514768399999	5993909.83603800	5228	228.52192
1	1514768400000	13528.99000000	13595.89000000	13155.38000000	13203.06000000	383.69700600	1514771999999	5154521.55513544	4534	180.84040
2	1514772000000	13203.00000000	13418.43000000	13200.00000000	13330.18000000	429.06457200	1514775599999	5710192.01852959	4887	192.23793
3	1514775600000	13330.26000000	13611.27000000	13290.00000000	13410.03000000	420.08703000	1514779199999	5657448.45730419	4789	137.91840
4	1514779200000	13434.98000000	13623.29000000	13322.15000000	13601.01000000	340.80732900	1514782799999	4588046.99615741	4563	172.95763
...	...	...	...	...	...	...	...	...	...	
35632	1643468400000	37620.04000000	37735.67000000	37268.44000000	37566.82000000	1881.67209000	1643471999999	70637969.49429300	51754	889.31896
35633	1643472000000	37567.98000000	37603.57000000	37317.37000000	37538.19000000	1102.74218000	1643475599999	41323390.93888780	32444	550.48690
35634	1643475600000	37538.19000000	37720.77000000	37532.15000000	37646.31000000	784.47184000	1643479199999	29522144.27770180	27730	349.29032
35635	1643479200000	37646.31000000	37904.19000000	37556.56000000	37806.23000000	772.06693000	1643482799999	29138395.49170340	31389	428.81978
35636	1643482800000	37806.24000000	37940.19000000	37781.49000000	37891.83000000	305.89610000	1643486399999	11578326.31702000	12413	168.24119

35637 rows × 12 columns

5.2 Organize the Data for analysis:

Clean Dataset, Remove some columns :

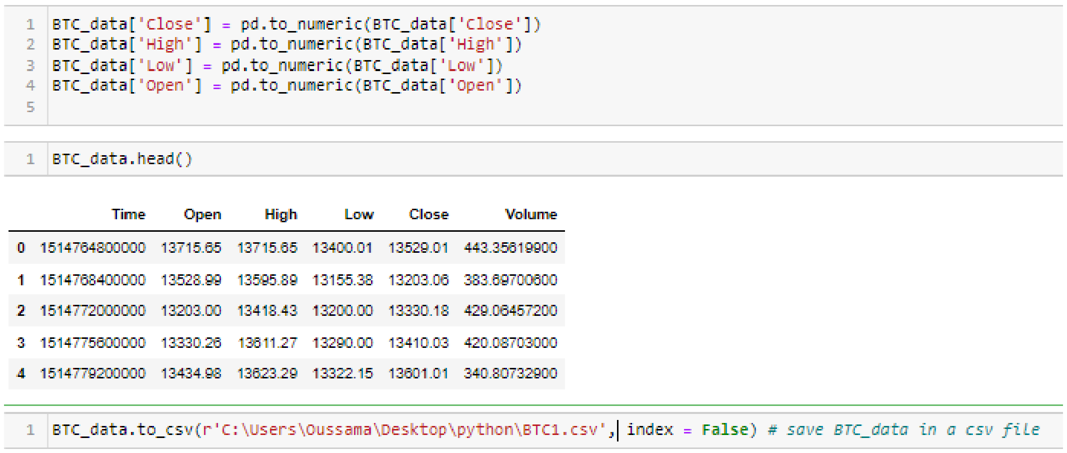


```python
del BTC_data['ignore']
del BTC_data['close_time']
del BTC_data['tb_quote_av']
del BTC_data['tb_base_aw']
del BTC_data['trades']
del BTC_data['quote_av']
```

```python
BTC_data.sample(5)
```

	Time	Open	High	Low	Close	Volume
12404	1559696400000	7754.79000000	7799.99000000	7734.44000000	7759.70000000	1140.06916300
25497	1606921200000	18910.21000000	18989.00000000	18728.38000000	18891.57000000	3966.23942200
1899	1521720000000	8645.00000000	8753.54000000	8632.74000000	8715.00000000	1770.70372400
9317	1548525600000	3568.97000000	3572.74000000	3565.15000000	3569.42000000	460.47190200
6238	1537405200000	6421.50000000	6422.00000000	6400.12000000	6407.12000000	1159.04455100

Numiric columns Close, High, Low, Open :



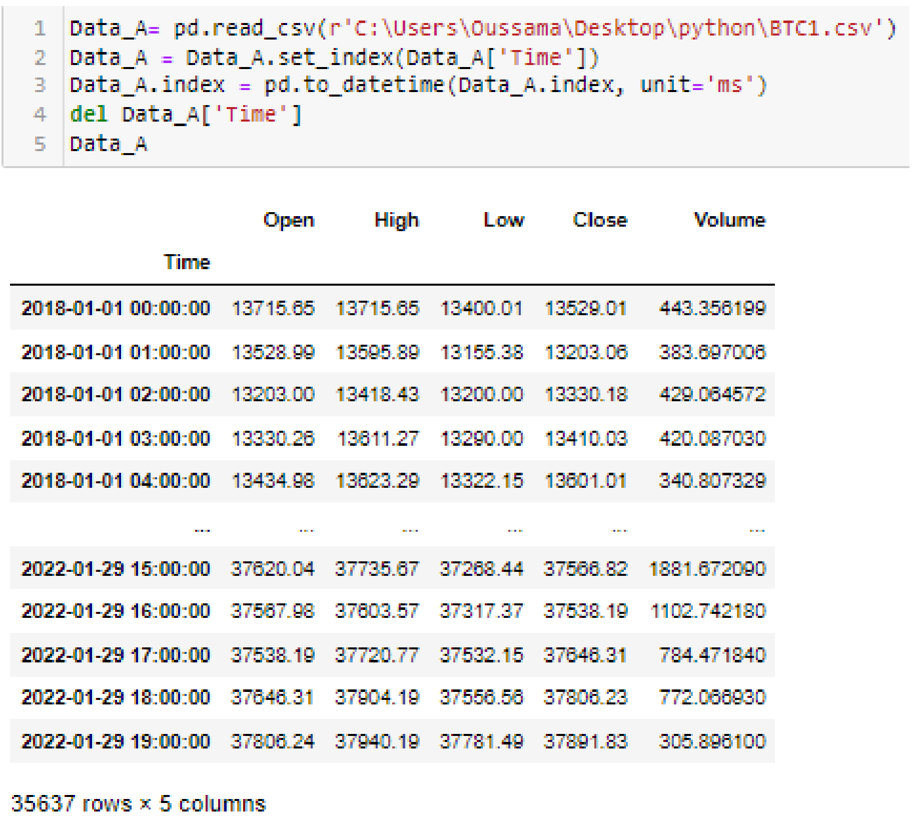
```python
BTC_data['Close'] = pd.to_numeric(BTC_data['Close'])
BTC_data['High'] = pd.to_numeric(BTC_data['High'])
BTC_data['Low'] = pd.to_numeric(BTC_data['Low'])
BTC_data['Open'] = pd.to_numeric(BTC_data['Open'])

```

```python
BTC_data.head()
```

	Time	Open	High	Low	Close	Volume
0	1514764800000	13715.65	13715.65	13400.01	13529.01	443.35619900
1	1514768400000	13528.99	13595.89	13155.38	13203.06	383.69700600
2	1514772000000	13203.00	13418.43	13200.00	13330.18	429.06457200
3	1514775600000	13330.26	13611.27	13290.00	13410.03	420.08703000
4	1514779200000	13434.98	13623.29	13322.15	13601.01	340.80732900

```python
BTC_data.to_csv(r'C:\Users\Oussama\Desktop\python\BTC1.csv', index = False) # save BTC_data in a csv file
```

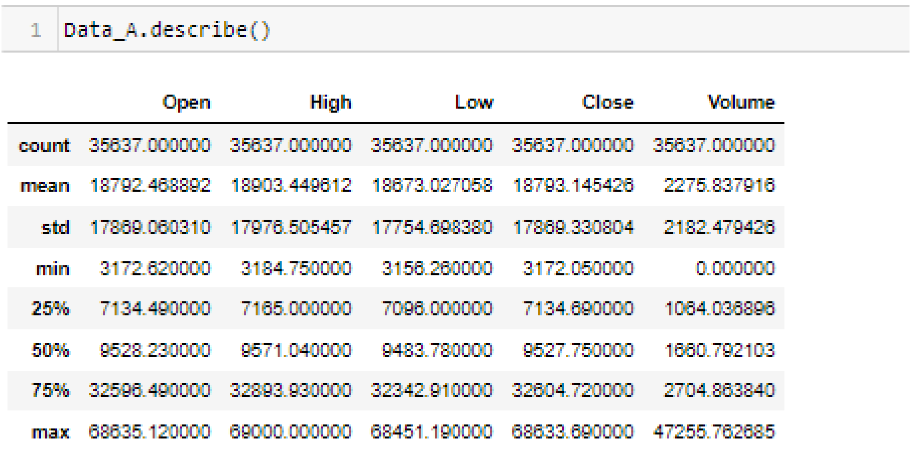
Convert Time and define it as index :



```python
Data_A= pd.read_csv(r'C:\Users\Oussama\Desktop\python\BTC1.csv')
Data_A = Data_A.set_index(Data_A['Time'])
Data_A.index = pd.to_datetime(Data_A.index, unit='ms')
del Data_A['Time']
Data_A
```

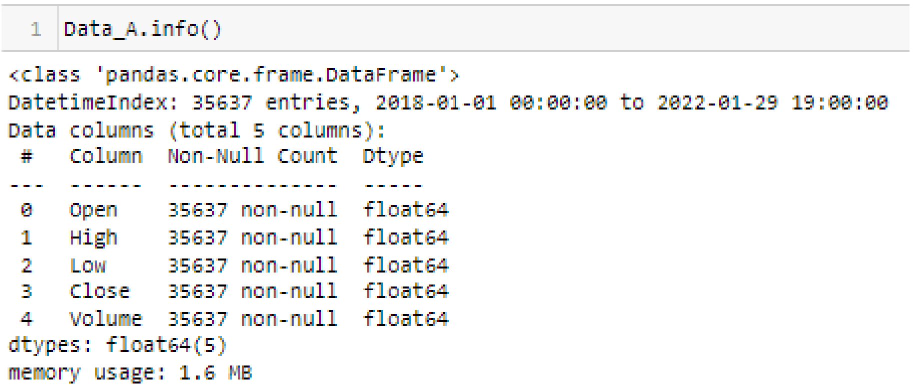
Time	Open	High	Low	Close	Volume
2018-01-01 00:00:00	13715.65	13715.65	13400.01	13529.01	443.356199
2018-01-01 01:00:00	13528.99	13595.89	13155.38	13203.06	383.697006
2018-01-01 02:00:00	13203.00	13418.43	13200.00	13330.18	429.064572
2018-01-01 03:00:00	13330.26	13611.27	13290.00	13410.03	420.087030
2018-01-01 04:00:00	13434.98	13623.29	13322.15	13601.01	340.807329
...	...	...	...	...	...
2022-01-29 15:00:00	37620.04	37735.67	37268.44	37566.82	1881.672090
2022-01-29 16:00:00	37567.98	37603.57	37317.37	37538.19	1102.742180
2022-01-29 17:00:00	37538.19	37720.77	37532.15	37646.31	784.471840
2022-01-29 18:00:00	37646.31	37904.19	37556.56	37806.23	772.066930
2022-01-29 19:00:00	37806.24	37940.19	37781.49	37891.83	305.896100

35637 rows × 5 columns



```python
Data_A.describe()
```

	Open	High	Low	Close	Volume
count	35637.000000	35637.000000	35637.000000	35637.000000	35637.000000
mean	18792.468892	18903.449612	18673.027058	18793.145426	2275.837916
std	17869.060310	17976.505457	17754.698380	17869.330804	2182.479426
min	3172.620000	3184.750000	3156.260000	3172.050000	0.000000
25%	7134.490000	7165.000000	7096.000000	7134.690000	1064.036896
50%	9528.230000	9571.040000	9483.780000	9527.750000	1660.792103
75%	32596.490000	32893.930000	32342.910000	32604.720000	2704.863840
max	68635.120000	69000.000000	68451.190000	68633.690000	47255.762685

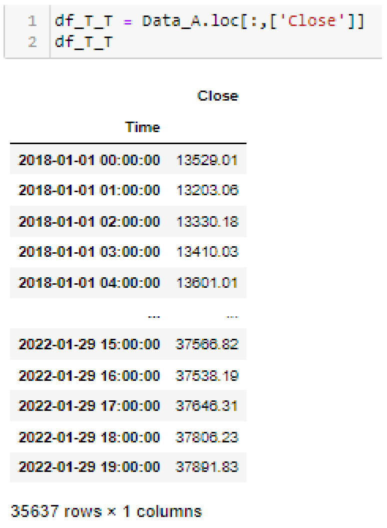


```python
Data_A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 35637 entries, 2018-01-01 00:00:00 to 2022-01-29 19:00:00
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Open    35637 non-null  float64
 1   High    35637 non-null  float64
 2   Low     35637 non-null  float64
 3   Close   35637 non-null  float64
 4   Volume  35637 non-null  float64
dtypes: float64(5)
memory usage: 1.6 MB
```

6 Training data, Testing data :

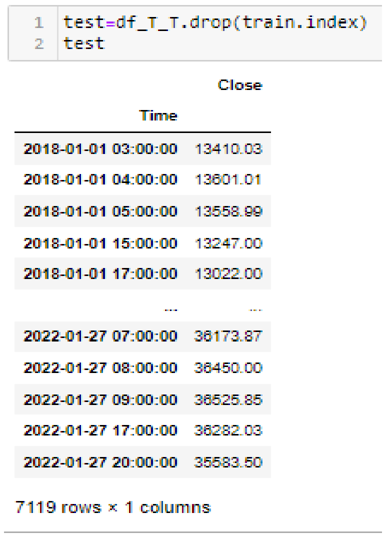
we divide the data into two parts: Training data 80% and Testing data 20%.



```python
df_T_T = Data_A.loc[:,['Close']]
df_T_T
```

Time	Close
2018-01-01 00:00:00	13529.01
2018-01-01 01:00:00	13203.06
2018-01-01 02:00:00	13330.18
2018-01-01 03:00:00	13410.03
2018-01-01 04:00:00	13601.01
...	...
2022-01-29 15:00:00	37566.82
2022-01-29 16:00:00	37538.19
2022-01-29 17:00:00	37646.31
2022-01-29 18:00:00	37806.23
2022-01-29 19:00:00	37891.83

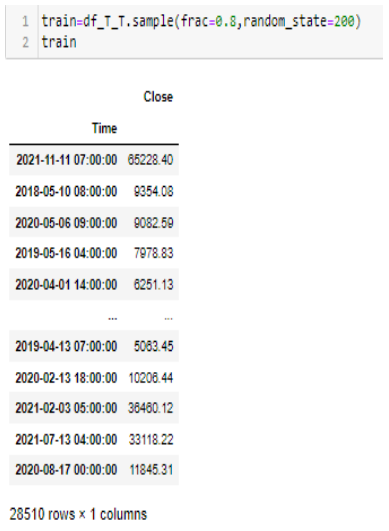
35637 rows × 1 columns



```python
test=df_T_T.drop(train.index)
test
```

Time	Close
2018-01-01 03:00:00	13410.03
2018-01-01 04:00:00	13601.01
2018-01-01 05:00:00	13558.99
2018-01-01 15:00:00	13247.00
2018-01-01 17:00:00	13022.00
...	...
2022-01-27 07:00:00	36173.87
2022-01-27 08:00:00	36450.00
2022-01-27 09:00:00	36525.85
2022-01-27 17:00:00	36282.03
2022-01-27 20:00:00	35583.50

7119 rows × 1 columns



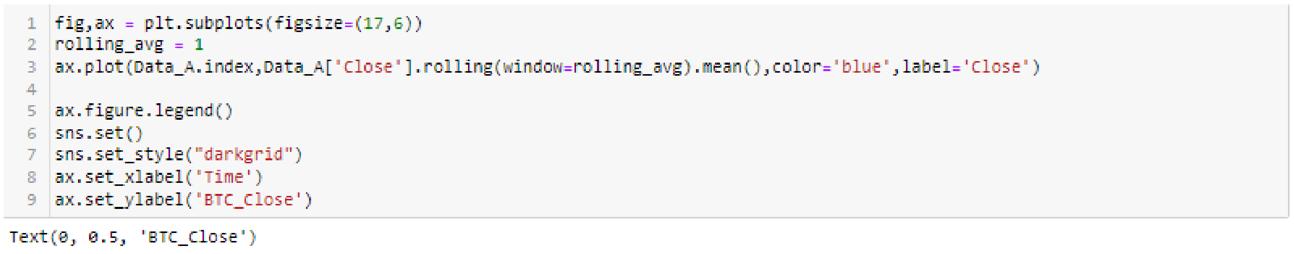
```python
train=df_T_T.sample(frac=0.8,random_state=200)
train
```

Time	Close
2021-11-11 07:00:00	65228.40
2018-05-10 08:00:00	9354.08
2020-05-06 09:00:00	9082.59
2019-05-16 04:00:00	7978.83
2020-04-01 14:00:00	6251.13
...	...
2019-04-13 07:00:00	5063.45
2020-02-13 18:00:00	10206.44
2021-02-03 05:00:00	36460.12
2021-07-13 04:00:00	33118.22
2020-08-17 00:00:00	11845.31

28510 rows × 1 columns

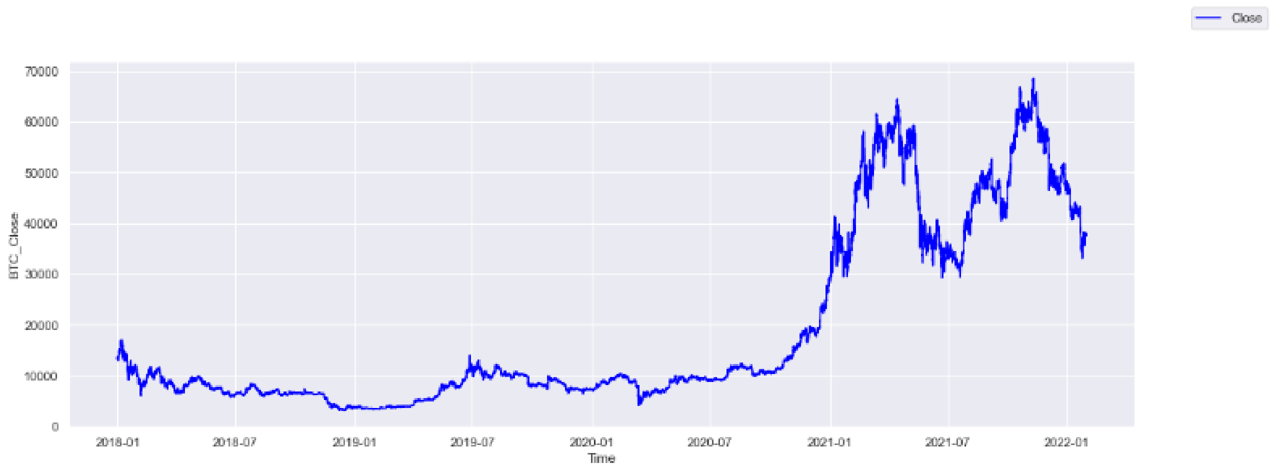
7 Data visualization :

We analyze Close price of BTC over time from 01/01/2018 to 29/01/2022. Let's use matplotlib to visualise the series.



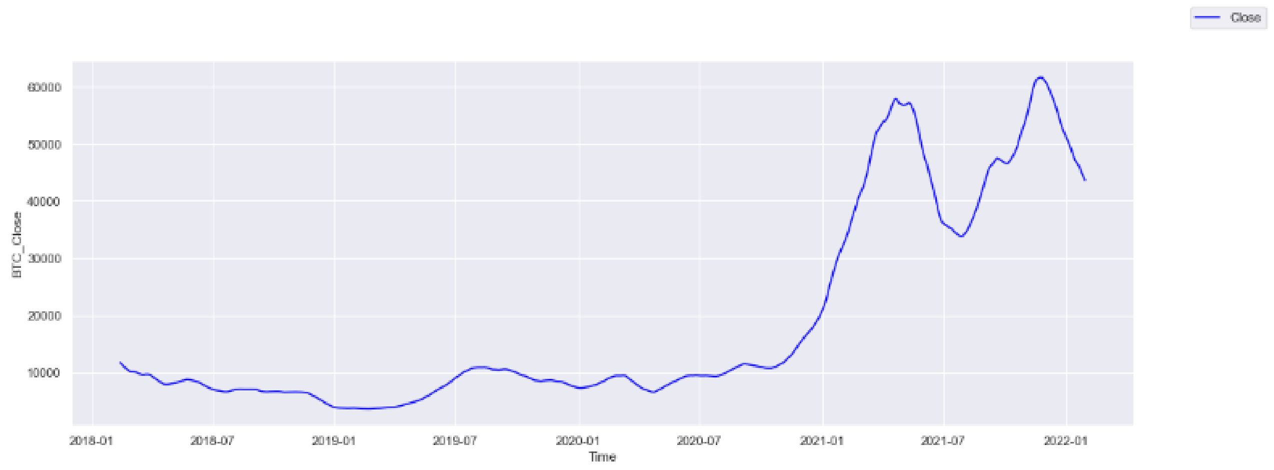
```python
fig,ax = plt.subplots(figsize=(17,6))
rolling_avg = 1
ax.plot(Data_A.index,Data_A['Close'].rolling(window=rolling_avg).mean(),color='blue',label='Close')

ax.figure.legend()
sns.set()
sns.set_style("darkgrid")
ax.set_xlabel('Time')
ax.set_ylabel('BTC_Close')
```

```
Text(0, 0.5, 'BTC_Close')
```



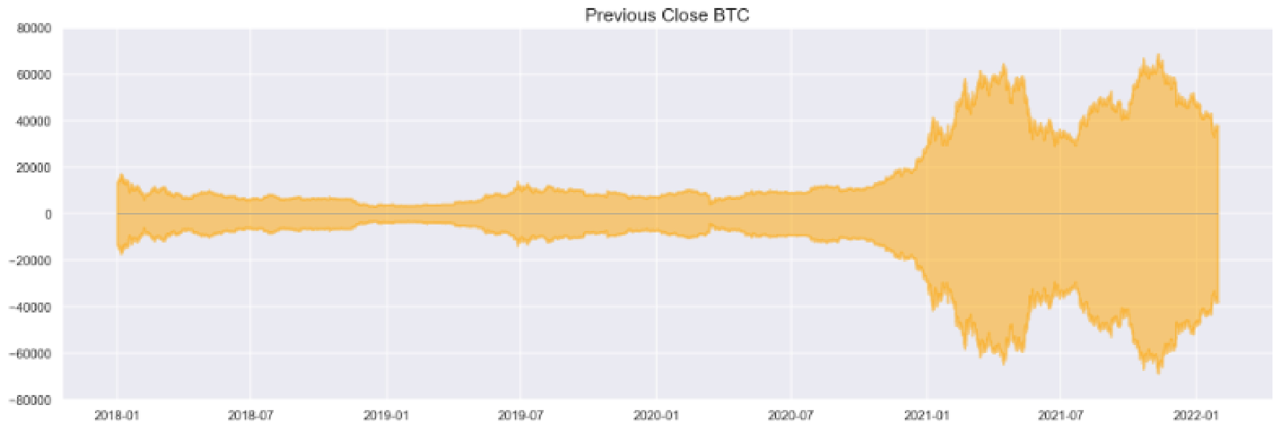
```
fig,ax = plt.subplots(figsize=(17,6))
rolling_avg = 1000
ax.plot(Data_A.index,Data_A['Close'].rolling(window=rolling_avg).mean(),color='blue',label='Close')

ax.figure.legend()
sns.set()
sns.set_style("darkgrid")
ax.set_xlabel('Time')
ax.set_ylabel('BTC_Close')
```

```
Text(0, 0.5, 'BTC_Close')
```



Since all values are positive, you can show this on both sides of the Y axis to emphasize the growth.

```
x = Data_A.index
y1 = Data_A['Close'].values

fig, ax = plt.subplots(1, 1, figsize=(19,6), dpi= 120)
plt.fill_between(x, y1=y1, y2=-y1, alpha=0.5, linewidth=2, color='orange')
plt.ylim(-80000, 80000)
plt.title('Previous Close BTC', fontsize=16)
plt.hlines(y=0, xmin=np.min(x), xmax=np.max(x), linewidth=.5)
plt.show()
```



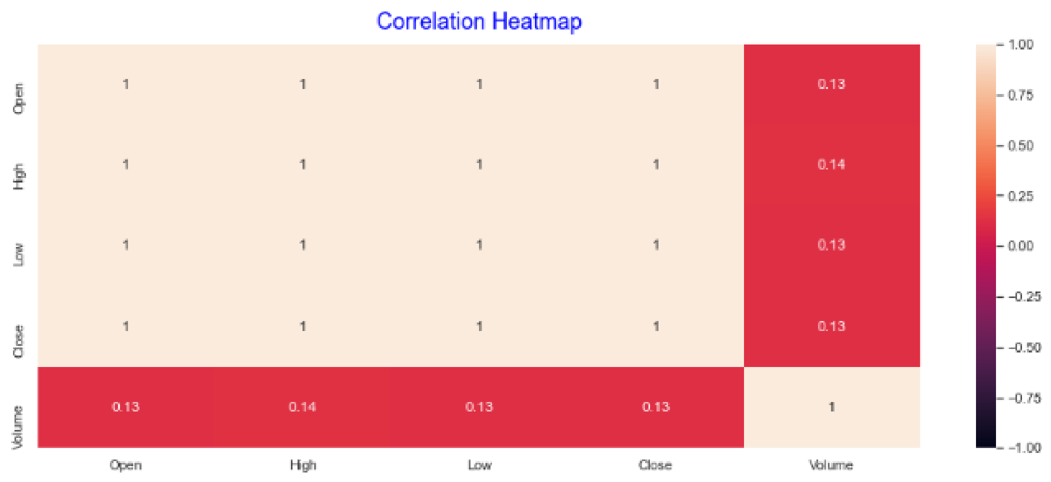
A correlation heatmap is a heatmap that shows a 2D correlation matrix between two discrete dimensions, using colored cells to represent data from usually a monochromatic scale. The values of the first dimension appear as the rows of the table while of the second dimension as a column.

```
Data_A.corr()
```

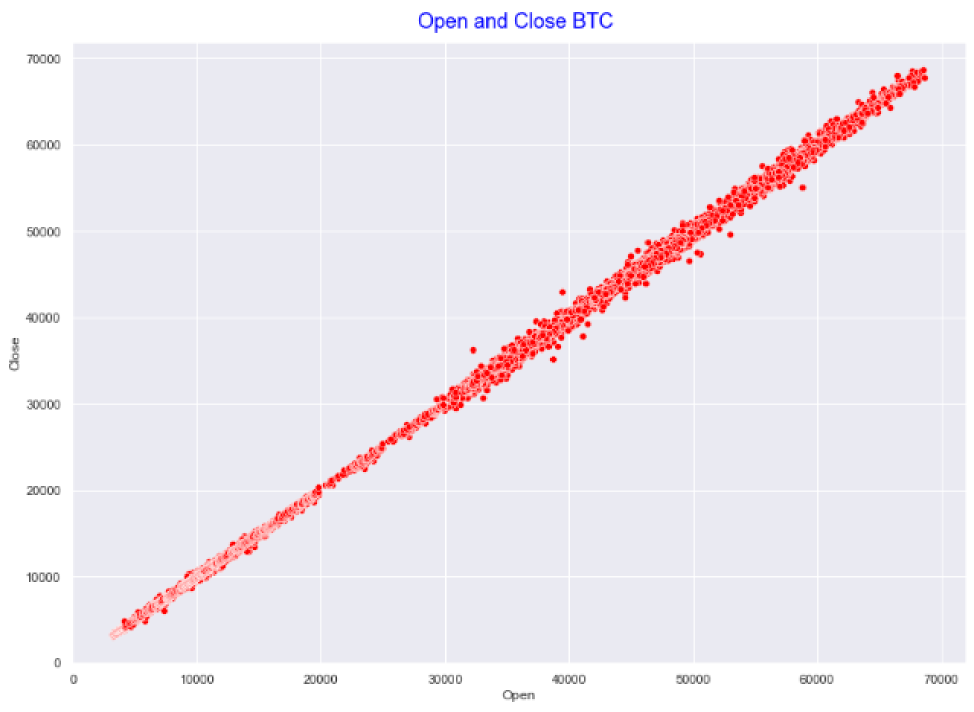
	Open	High	Low	Close	Volume
Open	1.000000	0.999960	0.999938	0.999926	0.132344
High	0.999960	1.000000	0.999907	0.999960	0.135666
Low	0.999938	0.999907	1.000000	0.999950	0.127203
Close	0.999926	0.999960	0.999950	1.000000	0.131835
Volume	0.132344	0.135666	0.127203	0.131835	1.000000

```
plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(Data_A.corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12, color='blue')
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



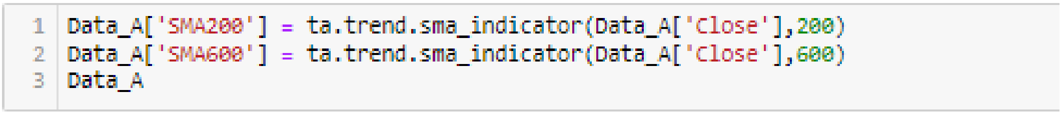
```
plt.figure(figsize=(14, 10))
ax = sns.scatterplot(x="Open", y="Close", data=Data_A, color='red')
plt.title('Open and Close BTC ',fontdict={'fontsize':18}, pad=12, color='blue');
```



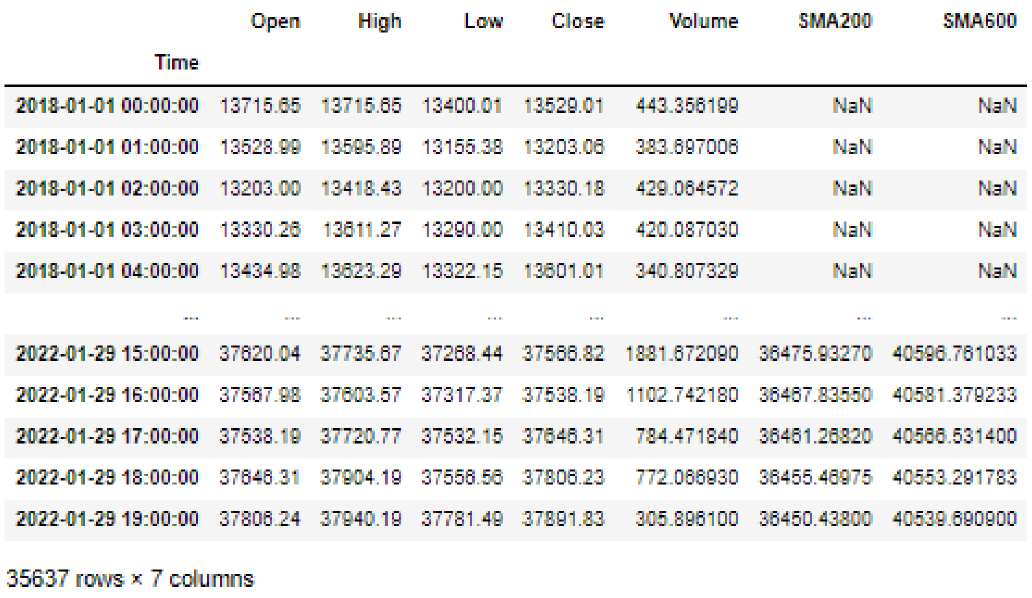
8 Trading bot strategy :

Many traders use moving averages to place their trades.

The issue with this, as for almost all indicators, is if you want to trade on a short time frame it is very likely that you will miss the exact right time to execute the trade.
In our situation we will use Moving Average 200 and 600.



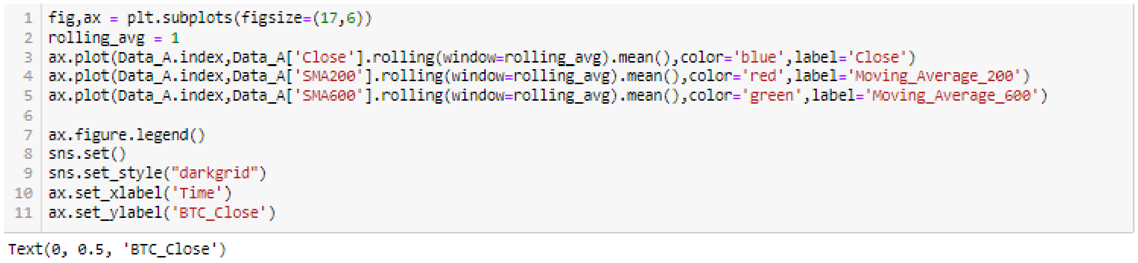
```python
Data_A['SMA200'] = ta.trend.sma_indicator(Data_A['Close'],200)
Data_A['SMA600'] = ta.trend.sma_indicator(Data_A['Close'],600)
Data_A
```



	Open	High	Low	Close	Volume	SMA200	SMA600
Time							
2018-01-01 00:00:00	13715.65	13715.65	13400.01	13529.01	443.356199	NaN	NaN
2018-01-01 01:00:00	13528.99	13595.89	13155.38	13203.06	383.697006	NaN	NaN
2018-01-01 02:00:00	13203.00	13418.43	13200.00	13330.18	429.064572	NaN	NaN
2018-01-01 03:00:00	13330.26	13611.27	13290.00	13410.03	420.087030	NaN	NaN
2018-01-01 04:00:00	13434.98	13623.29	13322.15	13601.01	340.807329	NaN	NaN
...	...	...	...	...	...	...	...
2022-01-29 15:00:00	37620.04	37735.67	37268.44	37566.82	1881.672090	36475.93270	40596.761033
2022-01-29 16:00:00	37567.98	37603.57	37317.37	37538.19	1102.742180	36467.83550	40581.379233
2022-01-29 17:00:00	37538.19	37720.77	37532.15	37646.31	784.471840	36461.26820	40566.531400
2022-01-29 18:00:00	37646.31	37904.19	37556.56	37806.23	772.066930	36455.46975	40553.291783
2022-01-29 19:00:00	37806.24	37940.19	37781.49	37891.83	305.896100	36450.43800	40539.690900

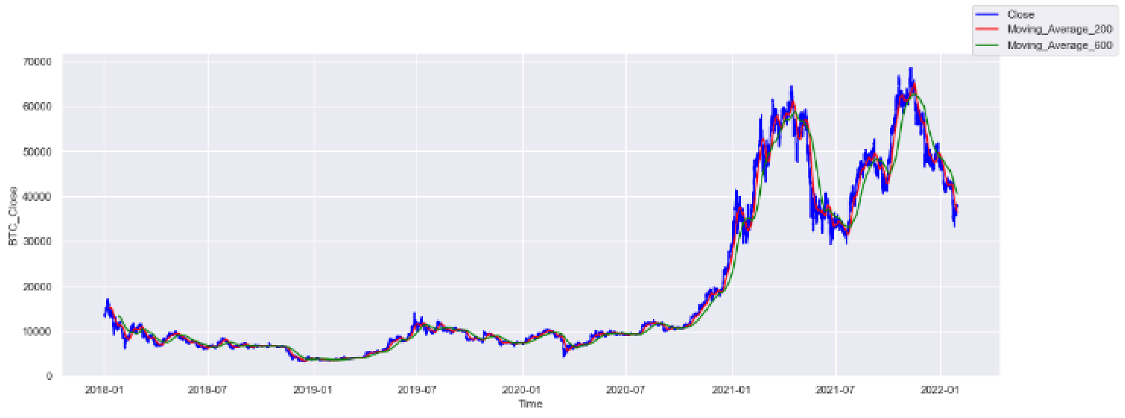
35637 rows × 7 columns

Visualize SMA200 and SMA600 graphics :



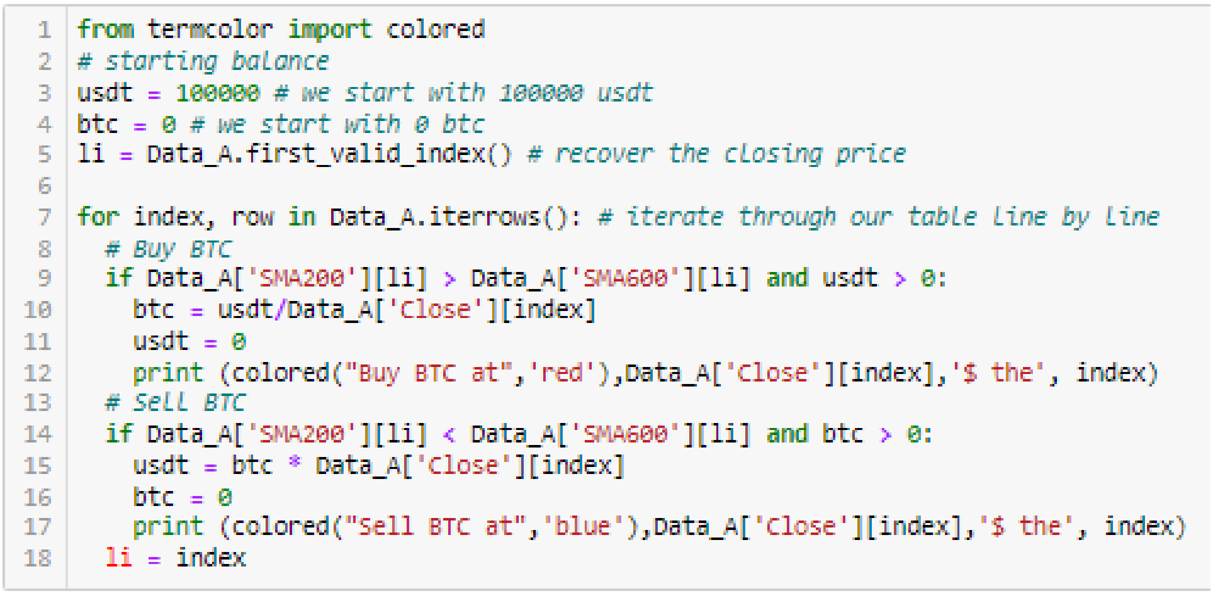
```python
fig,ax = plt.subplots(figsize=(17,6))
rolling_avg = 1
ax.plot(Data_A.index,Data_A['Close'].rolling(window=rolling_avg).mean(),color='blue',label='Close')
ax.plot(Data_A.index,Data_A['SMA200'].rolling(window=rolling_avg).mean(),color='red',label='Moving_Average_200')
ax.plot(Data_A.index,Data_A['SMA600'].rolling(window=rolling_avg).mean(),color='green',label='Moving_Average_600')

ax.figure.legend()
sns.set()
sns.set_style("darkgrid")
ax.set_xlabel('Time')
ax.set_ylabel('BTC_Close')
```

```
Text(0, 0.5, 'BTC_Close')
```

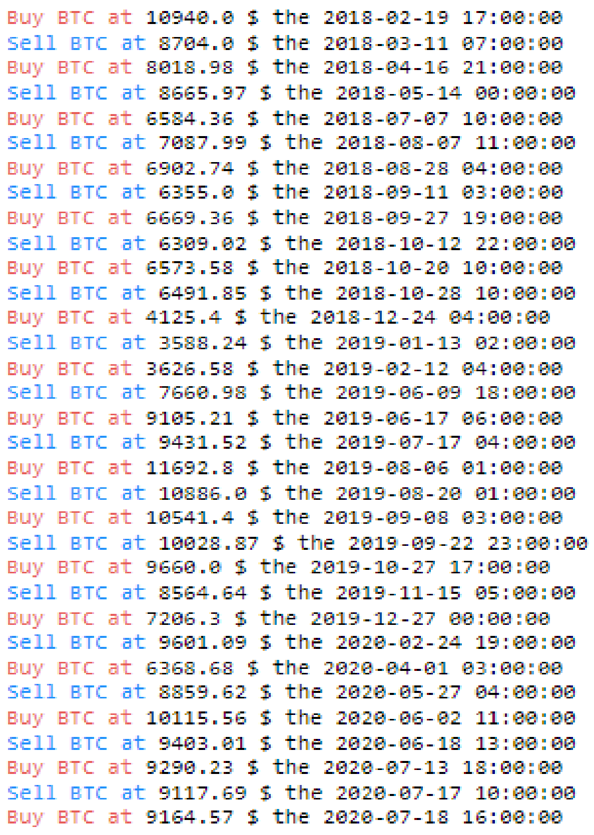


Define the trading bot strategy :
We start with 100000 USDT and 0 BTC.

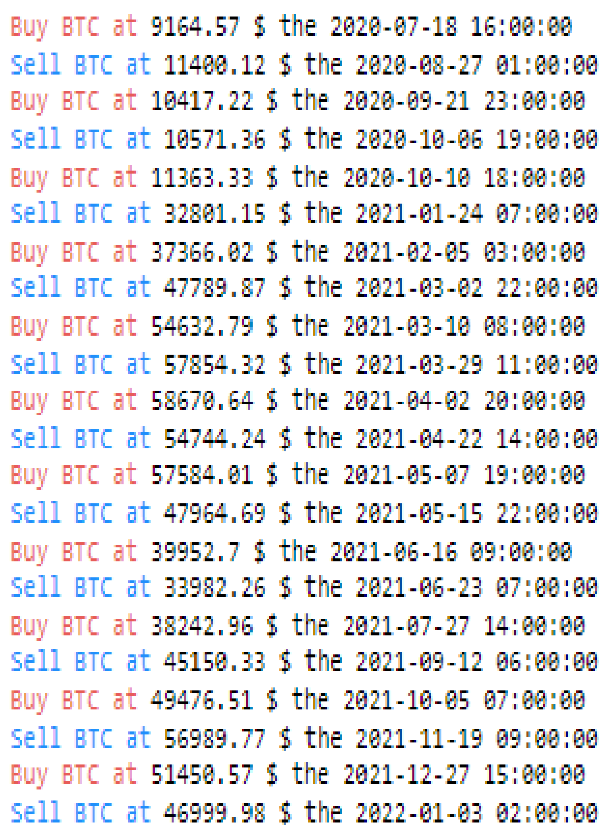
Firstly, we iterate through our table line by line, secondly we test if the graph of SMA200 is above of SMA600 graph and we have a balance usdt, then we buy BTC.
If the graph of SMA600 is above of SMA200 graph and we have a balance BTC, then we sell BTC.



```python
from termcolor import colored
# starting balance
usdt = 100000 # we start with 100000 usdt
btc = 0 # we start with 0 btc
li = Data_A.first_valid_index() # recover the closing price

for index, row in Data_A.iterrows(): # iterate through our table line by line
  # Buy BTC
  if Data_A['SMA200'][li] > Data_A['SMA600'][li] and usdt > 0:
    btc = usdt/Data_A['Close'][index]
    usdt = 0
    print (colored("Buy BTC at",'red'),Data_A['Close'][index],'$ the', index)
  # Sell BTC
  if Data_A['SMA200'][li] < Data_A['SMA600'][li] and btc > 0:
    usdt = btc * Data_A['Close'][index]
    btc = 0
    print (colored("Sell BTC at",'blue'),Data_A['Close'][index],'$ the', index)
  li = index
```

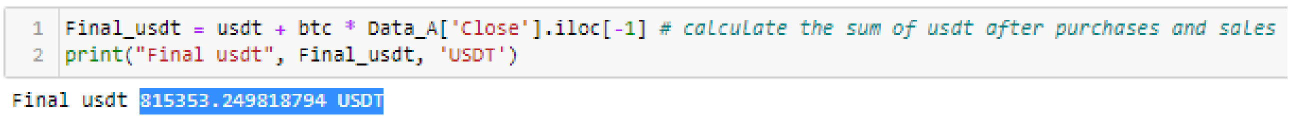


```
Buy BTC at 10940.0 $ the 2018-02-19 17:00:00
Sell BTC at 8704.0 $ the 2018-03-11 07:00:00
Buy BTC at 8018.98 $ the 2018-04-16 21:00:00
Sell BTC at 8665.97 $ the 2018-05-14 00:00:00
Buy BTC at 6584.36 $ the 2018-07-07 10:00:00
Sell BTC at 7087.99 $ the 2018-08-07 11:00:00
Buy BTC at 6902.74 $ the 2018-08-28 04:00:00
Sell BTC at 6355.0 $ the 2018-09-11 03:00:00
Buy BTC at 6669.36 $ the 2018-09-27 19:00:00
Sell BTC at 6309.02 $ the 2018-10-12 22:00:00
Buy BTC at 6573.58 $ the 2018-10-20 10:00:00
Sell BTC at 6491.85 $ the 2018-10-28 10:00:00
Buy BTC at 4125.4 $ the 2018-12-24 04:00:00
Sell BTC at 3588.24 $ the 2019-01-13 02:00:00
Buy BTC at 3626.58 $ the 2019-02-12 04:00:00
Sell BTC at 7660.98 $ the 2019-06-09 18:00:00
Buy BTC at 9105.21 $ the 2019-06-17 06:00:00
Sell BTC at 9431.52 $ the 2019-07-17 04:00:00
Buy BTC at 11692.8 $ the 2019-08-06 01:00:00
Sell BTC at 10886.0 $ the 2019-08-20 01:00:00
Buy BTC at 10541.4 $ the 2019-09-08 03:00:00
Sell BTC at 10028.87 $ the 2019-09-22 23:00:00
Buy BTC at 9660.0 $ the 2019-10-27 17:00:00
Sell BTC at 8564.64 $ the 2019-11-15 05:00:00
Buy BTC at 7206.3 $ the 2019-12-27 00:00:00
Sell BTC at 9601.09 $ the 2020-02-24 19:00:00
Buy BTC at 6368.68 $ the 2020-04-01 03:00:00
Sell BTC at 8859.62 $ the 2020-05-27 04:00:00
Buy BTC at 10115.56 $ the 2020-06-02 11:00:00
Sell BTC at 9403.01 $ the 2020-06-18 13:00:00
Buy BTC at 9290.23 $ the 2020-07-13 18:00:00
Sell BTC at 9117.69 $ the 2020-07-17 10:00:00
Buy BTC at 9164.57 $ the 2020-07-18 16:00:00
```



```
Buy BTC at 9164.57 $ the 2020-07-18 16:00:00
Sell BTC at 11400.12 $ the 2020-08-27 01:00:00
Buy BTC at 10417.22 $ the 2020-09-21 23:00:00
Sell BTC at 10571.36 $ the 2020-10-06 19:00:00
Buy BTC at 11363.33 $ the 2020-10-10 18:00:00
Sell BTC at 32801.15 $ the 2021-01-24 07:00:00
Buy BTC at 37366.02 $ the 2021-02-05 03:00:00
Sell BTC at 47789.87 $ the 2021-03-02 22:00:00
Buy BTC at 54632.79 $ the 2021-03-10 08:00:00
Sell BTC at 57854.32 $ the 2021-03-29 11:00:00
Buy BTC at 58670.64 $ the 2021-04-02 20:00:00
Sell BTC at 54744.24 $ the 2021-04-22 14:00:00
Buy BTC at 57584.01 $ the 2021-05-07 19:00:00
Sell BTC at 47964.69 $ the 2021-05-15 22:00:00
Buy BTC at 39952.7 $ the 2021-06-16 09:00:00
Sell BTC at 33982.26 $ the 2021-06-23 07:00:00
Buy BTC at 38242.96 $ the 2021-07-27 14:00:00
Sell BTC at 45150.33 $ the 2021-09-12 06:00:00
Buy BTC at 49476.51 $ the 2021-10-05 07:00:00
Sell BTC at 56989.77 $ the 2021-11-19 09:00:00
Buy BTC at 51450.57 $ the 2021-12-27 15:00:00
Sell BTC at 46999.98 $ the 2022-01-03 02:00:00
```

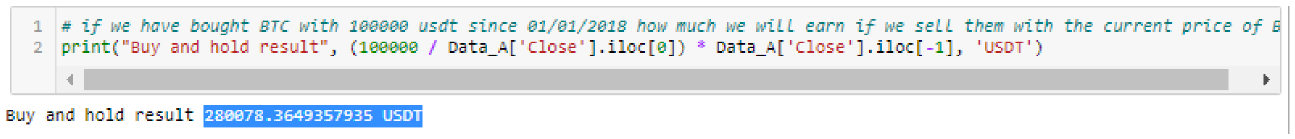
Using the strategy we win 715353.249818794 USDT, if we start with 100000 USDT



```python
Final_usdt = usdt + btc * Data_A['Close'].iloc[-1] # calculate the sum of usdt after purchases and sales
print("Final usdt", Final_usdt, 'USDT')
```

```
Final usdt 815353.249818794 USDT
```

If we have bought BTC with 100000 usdt since 01/01/2018 how much we will earn if we sell them with the current price of BTC.(280078.3649357935 USDT)



```python
# if we have bought BTC with 100000 usdt since 01/01/2018 how much we will earn if we sell them with the current price of B
print("Buy and hold result", (100000 / Data_A['Close'].iloc[0]) * Data_A['Close'].iloc[-1], 'USDT')
```

```
Buy and hold result 280078.3649357935 USDT
```

Here is the GitHub repo link:

https://github.com/Boujbair/Trading_Bot_Python

END