UNIVERSITY OF ECONOMICS AND LAW

BUILDING SIMPLE TECHNICAL ANALYSIS DASHBOARD IN PYTHON

Student: Tran Thanh Phuc

Student Code: K194141740

Class: K19414C

Teacher: Ngo Phu Thanh

TABLE OF CONTENTS

1. REASONS FOR CHOOSING THE TOPIC

Phân tích kỹ thuật là phương pháp dựa vào biểu đồ, đồ thị diễn biến giá cả và khối lượng giao dịch nhằm phân tích các biến động cung – cầu để giúp cho nhà đầu tư quyết định thời điểm nên mua vào, bán ra trên thị trường. Trong đó, để phân tích kỹ thuật hiệu quả và dễ hiểu hơn thì bảng điều khiển tương tác là vô cùng cần thiết. Bảng điều khiển tương tác là một công cụ trực quan hóa dữ liệu cho phép các nhóm kinh doanh theo dõi, phân tích và hiển thị các số liệu thuộc nhiều loại khác nhau. Chính vì sự quan trọng của bảng điều khiển tương tác đối với các nhà đầu tư sử dụng phân tích kỹ thuật nên tôi quyết định sẽ xây dựng một bản điều khiển tương tác cho phân tích kỹ thuật. Ngoài ra, sẽ rất bất tiện và khó sử dụng đối với nhiều người nếu phải tương tác với bản điều kiển thông qua giao diện của phần mềm lập trình. Vì thế tôi đã xây dựng bảng điều khiển này bằng ngôn ngữ Python và thể hiện kết quả thông qua giao diện web để dễ sử dụng đối với tất cả mọi người. Tuy nhiên, do trình độ còn thấp nên tôi sẽ chỉ xây dựng một bản điều khiển tương tác cho phân tích kỹ thuật đơn giản.

Một lý do khác, bảng điều khiển tương tác có tính áp dụng thực tế rất cao. Dùng chỉ xây dựng một bản điều khiển đơn giản, không có nhiều chức năng nhưng nó cũng là bước đầu tiên và giúp em có kinh nghiệm trong việc xây dựng một bản tương tác nâng cao hơn trong tương lai.

2. MEANING OF THE TOPIC.

Bài viết này nhằm mục đích sử dụng ngôn ngữ Python để xây dựng một Interactive dashboard đơn giản dùng cho phân tích kỹ thuật đối với cổ phiếu và tiền điện tử thông qua trang web. Ngoài ra, dashboard cũng hỗ trợ trong việc sử dụng các chỉ báo bằng cách đưa ra các tính hiệu mua hoặc bán cho từng chỉ báo. Qua bài viết này, các bạn sẽ được theo dõi quá trình xây dựng một Interactive dashboard đơn giản bằng Python, qúa trình lấy dữ liệu, xây dự các chỉ báo phân tích kỹ thuật, đưa ra kết quả và trực quan hóa chúng trên web bằng biểu đồ. Tuy nhiên, lưu ý rằng đây chỉ là bài tập và mục đích chính là ứng dụng Python vào tài chính. Những tín hiệu đưa ra có hoàn toàn có thể sai. Vì thế, không nên sử dụng chúng để đưa ra các quyết định đầu tư.

3. OBJECT

- Tài sản:

Trong bài này tôi đã đưa vào 3 đối tượng tài sản dùng để phân tích:

+ Cổ phiếu trên sàn HOSE Việt Nam

+ Cổ phiếu trên sàn NASDAQ Hòa Kỳ

+ Tiền điện tử

Tuy nhiên vì số lượng từng đối tượng trong mỗi loại là rất lớn nên giới hạn chỉ lấy 100 đối tượng có xếp hạng vốn hóa cao nhất. Vì thế chúng ta có 3 tài sản phân tích ở đây:

+ Top 100 cổ phiếu có giá trị vốn hóa lớn nhất trên sàn HOSE tại Việt Nam

+ Top 100 cổ phiếu có giá trị vốn hóa lớn nhất trên sàn NASDAQ tại Mỹ

+ Top 100 tiền điện tử có giá trị vốn hóa lớn nhất

- Các chỉ báo:

+ Bollinger Bands

+ Moving Average (MA)

+ Moving Average Convergence Divergence (MACD)

+ Money Flow Index (MFI)

+ On-Balance Volume (OBV)

+ Relative Strength Index (RSI).

Trong mỗi chỉ báo sẽ còn các đối tượng để tùy chỉnh cho chỉ báo đó như là: Period, Over Bought, Over Sold, …

- Tín hiệu:

+ Buy Signal

+ Sell Signal

- Thời gian:

+ Ngày bắt đầu

+ Ngày kết thúc

4. LIBRARY

Bảng điều khiển này được xây dựng dựa trên ngôn ngữ Python và sau đây là các thư viện được tôi sử dụng trong quá trình làm nó.

- Streamlit

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. ... Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

- Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

- Pandas

Pandas is an opensource Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem.

- Datetime

Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

- Pandas_datareader

Pandas Datareader is a Python package that allows us to create a pandas DataFrame object by using various data sources from the internet. It is commonly used to work with real-time data sets of stock prices and financial metrics such as: Yahoo Finance, Google Finance, Morningstar, World Bank, …

- VnStock

A package bringing an easy way to access to Vietnam Stock data. In order to use this package, user need obtain Vietstock cookies.

- Requests

The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

- BeautifulSoup

Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

- Cufflinks

Cufflinks is another library that connects the Pandas data frame with Plotly enabling users to create visualizations directly from Pandas. The library binds the power of Plotly with the flexibility of Pandas for easy plotting.

- Plotly

The plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

+ plotly.offline: A function creates a standalone HTML that is saved locally and opened inside your web browser.

+ plotly.graph_objects: A module (typically imported as go) contains an automatically-generated hierarchy of Python classes which represent non-leaf nodes in this figure schema. The term "graph objects" refers to instances of these classes. The primary classes defined in the plotly.

+ plotly.subplots: Allows multiple charts to be placed next to each other.

5. DATA SOURCE

Yahoo Finance: Dữ liệu được lấy thông qua thư viện pandas_datareader. Dữ liệu lấy được sẽ bao gồm Date, giá cao nhất trong ngày (High), giá thấp nhất trong ngày (Low), giá mở cửa (Open), giá đóng cửa (Close), khối lượng giao dịch (Volume), giá đóng của điều chỉnh (Adj Close).

Trandingview.com: Dữ liệu tên và mã cổ phiếu của các công ty thuộc Top 100 cổ phiếu có giá trị vốn hóa lớn nhất trên sàn HOSE. Dữ liệu được crawl từ trang https://www.tradingview.com/markets/stocks-vietnam/market-movers-large-cap/ thông qua hai thư viện Request và BeautifulSoup.

Wikipedia.org: Dữ liệu tên và mã cổ phiếu của các công ty thuộc Top 100 cổ phiếu có giá trị vốn hóa lớn nhất trên sàn NASDAQ (Nasdaq-100). Dữ liệu được crawl từ trang https://en.wikipedia.org/wiki/Nasdaq-100 thông qua hai thư viện Request và BeautifulSoup.

Coin360.com: Dữ liệu tên và mã của các tiền điện tử thuộc Top 100 tiền điện tử có giá trị vốn hóa lớn nhất. Dữ liệu được crawl từ trang https://coin360.com/coin/ thông qua hai thư viện Request và BeautifulSoup.

6. EXPLANATION OF THE IMPLEMENTATION PROCESS

6.1 Note

Ở phần này tôi muốn hướng dẫn cách khởi động bảng điều khiển tương tác và nhắc nhở người dùng cài đặt đầy đủ các thư viện cần thiết trước khi bắt đầu.

```
########## Note
# Before you start, You should make sure to install all of them to avoid errors.
# After running this file, you must go to the terminal and run the command: streamlit run "file"
# In this case: streamlit run K194141740_Tran-Thanh-Phuc_Last-Term.py
# After running that command successfully, Local URL and Network URL will appear.
# Your computer will redirect to the Local URL through your browser.
# Then, you will see this Interactive Dashboard Technical Analysis.
```

6.2 Import Library

Ở đây là bước tôi import the libraries mà tôi đã nêu ở trên vào trong bài và cài đặt 1 số thuộc tính cho bài.

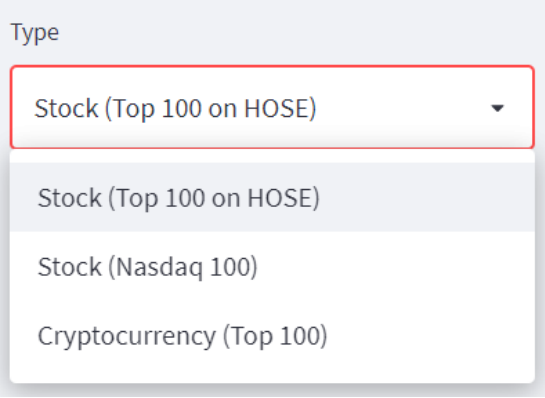
```
# import the libraries
from matplotlib.pyplot import title
import streamlit as st
st.set_page_config(layout="wide")  # set auto wide mode when run
import numpy as np
import pandas as pd
import datetime as dt
import pandas_datareader as web
from vnstock_data.all_exchange import VnStock
import requests
from bs4 import BeautifulSoup  # library to parse HTML documents
import cufflinks as cf
cf.go_offline()  # configure it for offline use
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

6.3 Get the names and tickers of stocks and cryptocurrencies

Sau khi đã import the libraries thì tôi bắt đầu tạo hộp chọn Type trên sidebar cho người dùng lựa chọn tài sản họ muốn phân tích.

```
# get Name, Stock Code and Crypto Code
option_type = st.sidebar.selectbox("Type", ["Stock (Top 100 on HOSE)", "Stock (Nasdaq 100)",\
    "Cryptocurrency (Top 100)"])
```

Hình ảnh của hộp chọn Type.



Nếu người dùng chọn “Stock (Nasdaq 100)” tức là họ chọn Top 100 cổ phiếu doanh nghiệp có vốn hóa lớn nhất trên sàn NASDAQ. Sau đó tôi dùng hai thư viện Requests và BeautifulSoup để truy cập vào trang https://en.wikipedia.org/wiki/Nasdaq-100 để lấy về dữ liệu tên và mã cổ phiếu.

Lưu ý:

Bạn có thể học cách cào dữ liệu từ web qua các hướng dẫn trên mạng hoặc qua video này: https://www.youtube.com/watch?v=ICXR9nDbudk&t=285s&ab_channel=JieJenn .

Ngoài ra, vì dữ liệu lấy có nhiều thông tin không cần thiết nên cần được xử lý và làm sạch. Sau đó tôi chuyển dữ liệu về dạng dataframe để dễ sử dụng hơn. Ngoài ra, mỗi trang lấy về dữ liệu khác nhau

nên cách xử lý chúng cũng có nhiều điểm khác nhau. Các bạn nên chạy những dòng code đó sau đó xem dữ liệu lấy về của từng trường hợp để hiểu rõ chúng có những vấn đề gì. Sau đó, đọc những dòng code sẽ giúp các bạn hiểu được cách tôi xử lý chúng.

```
30  if option_type == "Stock (Nasdaq 100)":
31      # get the response in the form of html
32      url = "https://en.wikipedia.org/wiki/Nasdaq-100"
33      table_id = "constituents"
34      response = requests.get(url)
35
36      # parse data from the html into a beautifulsoup object
37      soup = BeautifulSoup(response.text, "html.parser")
38      indiatable=soup.find('table',{'id':table_id})
39      df_company=pd.read_html(str(indiatable))
40
41      # convert list to dataframe
42      df_company=pd.DataFrame(df_company[0])
43      ticker = df_company["Ticker"].tolist()
44      ticker = sorted(ticker)
45      # get name company
46      name = df_company[["Company", "Ticker"]]
47      name.index = name["Ticker"]
48      del name["Ticker"]
49      name.rename(columns={"Company": "Name"}, inplace=True)
```

Đây là dữ liệu tôi lấy được từ trang web sau khi đã xử lý.

Lưu ý: Đây chỉ là một phần của dữ liệu được dùng để minh họa. Tôi không thể đưa toàn bộ dữ liệu vào đây được vì khích thước quá lớn.

	Name
ATVI	Activision Blizzard
ADBE	Adobe
AMD	Advanced Micro Devices
ABNB	Airbnb
ALGN	Align Technology
GOOGL	Alphabet (Class A)
GOOG	Alphabet (Class C)
AMZN	Amazon
AEP	American Electric Power
AMGN	Amgen

Nếu người dùng chọn “Cryptocurrency (Top 100)” nghĩa là họ chọn Top 100 tiền điện tử có vốn hóa lớn nhất. Tương tự ở trên, tôi cũng dùng hai thư viện Requests và BeautifulSoup để truy cập vào trang https://coin360.com/coin/ lấy về dữ liệu tên và mã tiền điện tử. Sau đó, tôi xử lý chúng và đưa về dạng dataframe để thuận tiện cho việc sử dụng.

```python
elif option_type == "Cryptocurrency (Top 100)":
    # get the response in the form of html
    url = "https://coin360.com/coin/"
    table_class = "TableView__Table"
    response = requests.get(url)

    # parse data from the html into a beautifulsoup object
    soup = BeautifulSoup(response.text, "html.parser")
    indiatable=soup.find('table',{'class':table_class})
    df_crypto=pd.read_html(str(indiatable))

    # convert list to dataframe
    df_crypto=pd.DataFrame(df_crypto[0])

    # get symbol
    ticker = df_crypto["Symbol"]
    ticker = ticker.values.tolist()
    for i in range(len(ticker)):
        if "?" in ticker[i]:
            ticker[i] = ticker[i].replace("?", "")
            ticker[i] += "-USD"
    ticker = sorted(ticker)
    # get name cryptocurrency
    name = df_crypto[["Name", "Symbol"]]
    for i in range(len(name["Symbol"])):
        if "?" in name["Symbol"][i]:
            name["Symbol"][i] = name["Symbol"][i].replace("?", "")
            name["Symbol"][i] += "-USD"
    name.index = name["Symbol"]
    del name["Symbol"]
```

Đây là dữ liệu tôi lấy được từ trang web sau khi đã xử lý.

	Name
BTC-USD	Bitcoin
ETH-USD	Ethereum
USDT-USD	Tether
BNB-USD	Binance Coin
USDC-USD	USD Coin
ADA-USD	Cardano
HEX-USD	Hex
XRP-USD	Ripple
LUNA-USD	Terra
SOL-USD	Solana

Cuối cùng, nếu người dùng chọn “Stock (Top 100 on HOSE)” tức là họ chọn Top 100 cổ phiếu doanh nghiệp có vốn hóa lớn nhất trên sàn HOSE. Tôi sử dụng hai thư viện Requests và BeautifulSoup để truy cập vào trang https://www.tradingview.com/markets/stocks-vietnam/market-movers-large-cap/

lấy về dữ liệu cần dùng là tên công ty và mã cổ phiếu. Sau đó tiếp tục xử lý và chỉ giữ lại dữ liệu cần thiết.

```
else:
    url = "https://www.tradingview.com/markets/stocks-vietnam/market-movers-large-cap/"
    table_class = "tv-data-table tv-screener-table"
    response = requests.get(url)

    # parse data from the html into a beautifulsoup object
    soup = BeautifulSoup(response.text, "html.parser")
    indiatable=soup.find('table',{'class':table_class})
    df_company=pd.read_html(str(indiatable))

    # convert list to dataframe
    df_company=pd.DataFrame(df_company[0])
    # get tickername column
    lst_ticker = df_company["Unnamed: 0"].tolist()
    # def list to string
    def listtostring(s):
        # initialize an empty string
        str1 = ""
        # traverse in the string
        for ele in s:
            str1 += (ele + " ")
        # return string
        return str1

    # get ticker and name
    ticker = []
    name_lst = []
    for i in range(len(lst_ticker)):
        get_ticker = lst_ticker[i].split(" ")
        if len(get_ticker[0]) == 1:
            ticker.append(get_ticker[1])
            name_lst.append(listtostring(get_ticker[2:]))
        else:
            ticker.append(get_ticker[0])
            name_lst.append(listtostring(get_ticker[1:]))

    name = pd.DataFrame()
    name["Name"] = name_lst
    name["Ticker"] = ticker
    name.index = name["Ticker"]
    del name["Ticker"]
    ticker = sorted(ticker)
```

Đây là dữ liệu tôi lấy được từ trang web sau khi đã xử lý.

	Name
VCB	JOINT STOCK COMMERCIAL BANK FOR FOREIGN TRADE OF VIET NAM
VIC	VINGROUP JOINT STOCK COMPANY
VHM	VINHOMES JOINT STOCK COMPANY
BID	JOINT STOCK COMMERCIAL BANK FOR INVESTMENT AND DEVELOPMENT OF VIETNAM
GAS	PETROVIETNAM GAS JOINT STOCK CORPORATION
HPG	HOA PHAT GROUP JOINT STOCK COMPANY
ACV	AIRPORTS CORPORATION OF VIETNAM
MSN	MASAN GROUP CORPORATION
TCB	NH TMCP KY THUONG VN
VNM	VIET NAM DAIRY PRODUCTS JOINT STOCK COMPANY

6.4 Create selectbox and slider in sidebar

Bước kế tiếp, tôi tạo ra những hộp chọn và thanh trượt trên sidebar để người dùng có thể lựa chọn:

- Hộp chọn Code

- Hộp chọn Indicator

- Thanh trượt Period

- Thanh trượt Slow Period: dùng cho chỉ báo MACD và MA

- Thanh trượt Period Signal: dùng cho chỉ báo MACD

- Thanh trượt Over Bought: dùng cho chỉ báo RSI và MFI

- Thanh trượt Over Sold: dùng cho chỉ báo RSI và MFI

- Hộp chọn Start Date

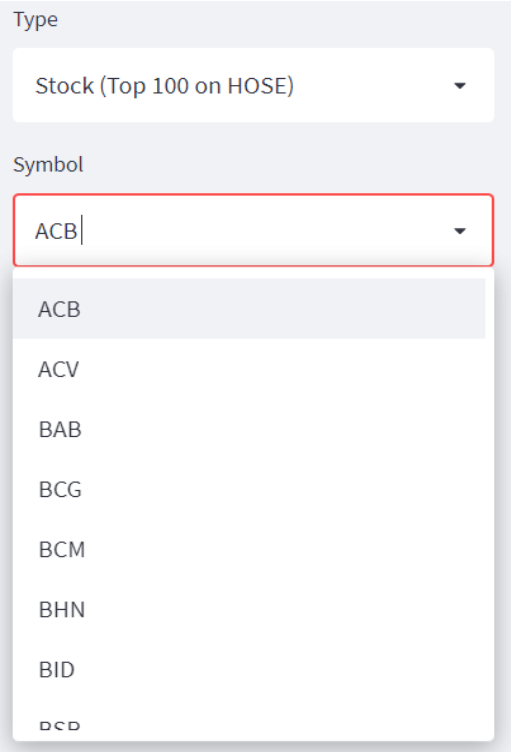
- Hộp chọn End Date

Nếu chỉ báo nào không cần thiết dùng đến hộp chọn nào đó thì hộp chọn đó sẽ được tự động gán giá trị bằng 0 và không được xuất hiện trên Dashboard.

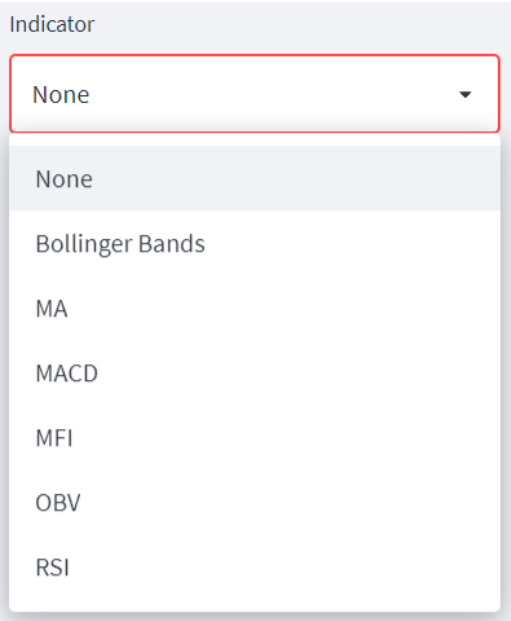
Lưu ý: Thanh trượt Period sẽ trở thành thanh trượt Fast Period khi một trong hai chỉ báo MA hoặc MACD được chọn.

```python
# create indicator selection
indicator = ["None","Bollinger Bands","MA", "MACD", "MFI", "OBV", "RSI"]
# create selectbox in sidebar
option_symbol = st.sidebar.selectbox("Symbol", ticker)
option_indi = st.sidebar.selectbox("Indicator", indicator)
option_periods = 0
option_periods_slow = 0
option_periods_signal = 0
option_overbought = 0
option_oversold = 0

if option_indi == "None":
    option_periods = 0
    option_periods_slow = 0
    option_periods_signal = 0
    option_overbought = 0
    option_oversold = 0
elif option_indi == "MACD":
    option_periods = st.sidebar.slider("Fast Period", min_value=2, max_value=251, value=12)
    option_periods_slow = st.sidebar.slider("Slow Period", min_value=2, max_value=251, value=26)
    option_periods_signal = st.sidebar.slider("MACD Period", min_value=2, max_value=251, value=9)
elif option_indi == "MA":
    option_periods = st.sidebar.slider("Fast Period", min_value=2, max_value=251, value=30)
    option_periods_slow = st.sidebar.slider("Slow Period", min_value=2, max_value=251, value=100)
elif option_indi == "OBV" or option_indi == "Bollinger Bands":
    option_periods = st.sidebar.slider("Period", min_value=2, max_value=251, value=14)
else:
    option_periods = st.sidebar.slider("Period", min_value=2, max_value=251, value=14)
    option_oversold = st.sidebar.slider("Over Sold", min_value=10, max_value=40, value=30, step=10)
    option_overbought = st.sidebar.slider("Over Bought", min_value=60, max_value=90, value=70, step=10)

option_start = st.sidebar.date_input("Start Date", dt.date(2021,1,1))
option_end = st.sidebar.date_input("End Date")
```

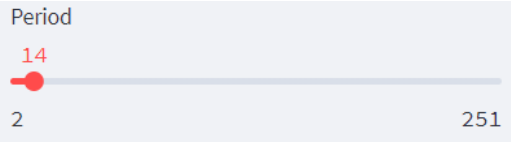
Hình ảnh của hộp chọn Code khi người dùng chọn phân tích Top 100 Stock in HOSE ở bên trên.



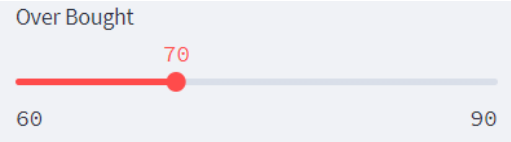
Hình ảnh của hộp chọn Indicator.



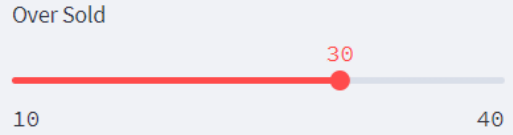
Hình ảnh của thanh trượt Period. Thanh trượt Slow Period cũng tương tự thanh trượt Period.



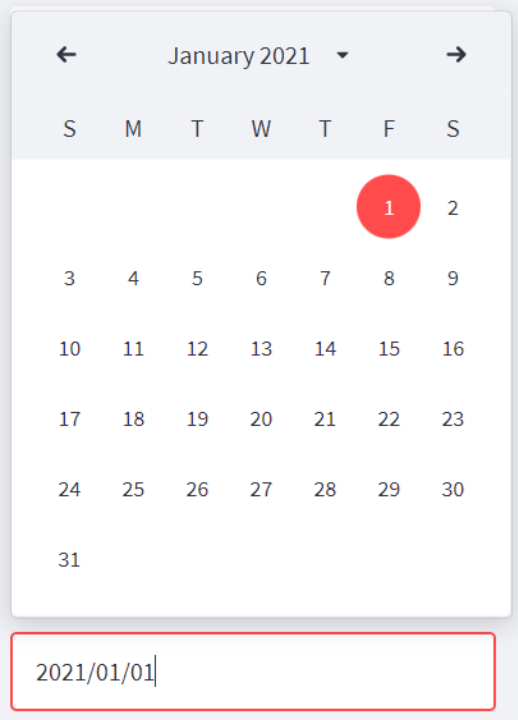
Hình ảnh của thanh trượt Over Bought.



Hình ảnh của thanh trượt Over Sold



Hình ảnh của hộp chọn Start Date. Hộp chọn End Date cũng tương tự hộp chọn Start Date.



6.5 Building data retrieval functions, indicators and data visualization

Phần nầy sẽ là một hàm khổng lồ chiếm phần lớn của bài code. Vì thế có rất nhiều thông tin trong phần này.

6.5.1 Lấy dữ liệu

Nếu như dữ liệu của Nasdaq 100 và Top 100 tiền điện tử có thể dễ dàng lấy được thông qua thư viện pandas_Datareader (gọi tắt là web) thì dữ liệu Top 100 cổ phiếu trên sàn HOSE lại khó lấy hơn nhiều. Thư viện VnStock yêu cầu cookie để có thể sử dụng. Ngoài ra, giá trị ngày cần phải được điều chỉnh theo quy tắc “month-date-year” trước khi đưa vào thư viện. Vì thế đối với thư viện VnStock, tôi cần có bước xử lý trước khi có thể lấy dữ liệu về sử dụng.

Lưu ý: Bạn có thể học cách sử dụng thư viện VnStock qua link này: https://vnstock-data-python.readthedocs.io/en/latest/#usage .

```python
option_start = st.sidebar.date_input("Start Date", dt.date(2021,1,1))
option_end = st.sidebar.date_input("End Date")

def test(symbol, indicator, start, end, period, period_slow, period_signal, overbought, oversold, type):
    if type == "Stock (Top 100 on HOSE)":
        cookies = {"vts_usr_lg":"F1ED60F2507CE5F2E3E6A81669EA65857EB57ACB376840C73CDF10C43923C464AA42A0F7\
            019669C72DBE306E2BDACD63951BC63600B9AC754338EF5CAC16956B9E9909E830319150EE2F1D0FAF2A9484FE68AF\
                4A9FA8BECF01BAF54A18D919C5105332F7F3B70B1A9376E259F58913BFB14E2F6A5C5BC994C006E6F2DB0B156A4\
                    B7F463F048B957ED024016EE37543F5",
        "__RequestVerificationToken":"1_Waqh8PDWVN4qyRMz6Okrxr2hUEljm0tEJvbBY5AoaQ8PCctY1X6dsxvD0Do1xmxPlz8H\
            qlNGtBQHIX4owX4RE83MPwrEK0f09pleKX3HY1",
        "language":"en-US"
        }
        vndata = VnStock(cookies)
        start = str(start)
        end = str(end)
        date_start = start.split("-")
        start = date_start[1] + "-" + date_start[2] + "-" + date_start[0]
        date_end = end.split("-")
        end = date_end[1] + "-" + date_end[2] + "-" + date_end[0]
        df = vndata.price(f"{symbol}", start, end)

    else:
        df = web.DataReader(f"{symbol}", "yahoo", start, end)
```

Hình ảnh dữ liệu của cổ phiếu AMZN (Amazon).

	High	Low	Open	Close	Volume	Adj Close
2020-12-31T00:00:00	3,282.9199	3,241.2000	3,275.0000	3,256.9299	2957200	3,256.9299
2021-01-04T00:00:00	3,272.0000	3,144.0200	3,270.0000	3,186.6299	4411400	3,186.6299
2021-01-05T00:00:00	3,223.3799	3,165.0601	3,166.0100	3,218.5100	2655500	3,218.5100
2021-01-06T00:00:00	3,197.5100	3,131.1599	3,146.4800	3,138.3799	4394800	3,138.3799
2021-01-07T00:00:00	3,208.5400	3,155.0000	3,157.0000	3,162.1599	3514500	3,162.1599
2021-01-08T00:00:00	3,190.6399	3,142.2000	3,180.0000	3,182.7000	3537700	3,182.7000
2021-01-11T00:00:00	3,156.3799	3,110.0000	3,148.0100	3,114.2100	3683400	3,114.2100
2021-01-12T00:00:00	3,142.1399	3,086.0000	3,120.0000	3,120.8301	3514600	3,120.8301
2021-01-13T00:00:00	3,189.9500	3,122.0801	3,128.4399	3,165.8899	3321200	3,165.8899
2021-01-14T00:00:00	3,178.0000	3,120.5901	3,167.5200	3,127.4700	3070900	3,127.4700

Lưu ý: Kể từ lúc này trở đi, tôi sẽ dùng dữ liệu cổ phiếu AMZN để minh họa.

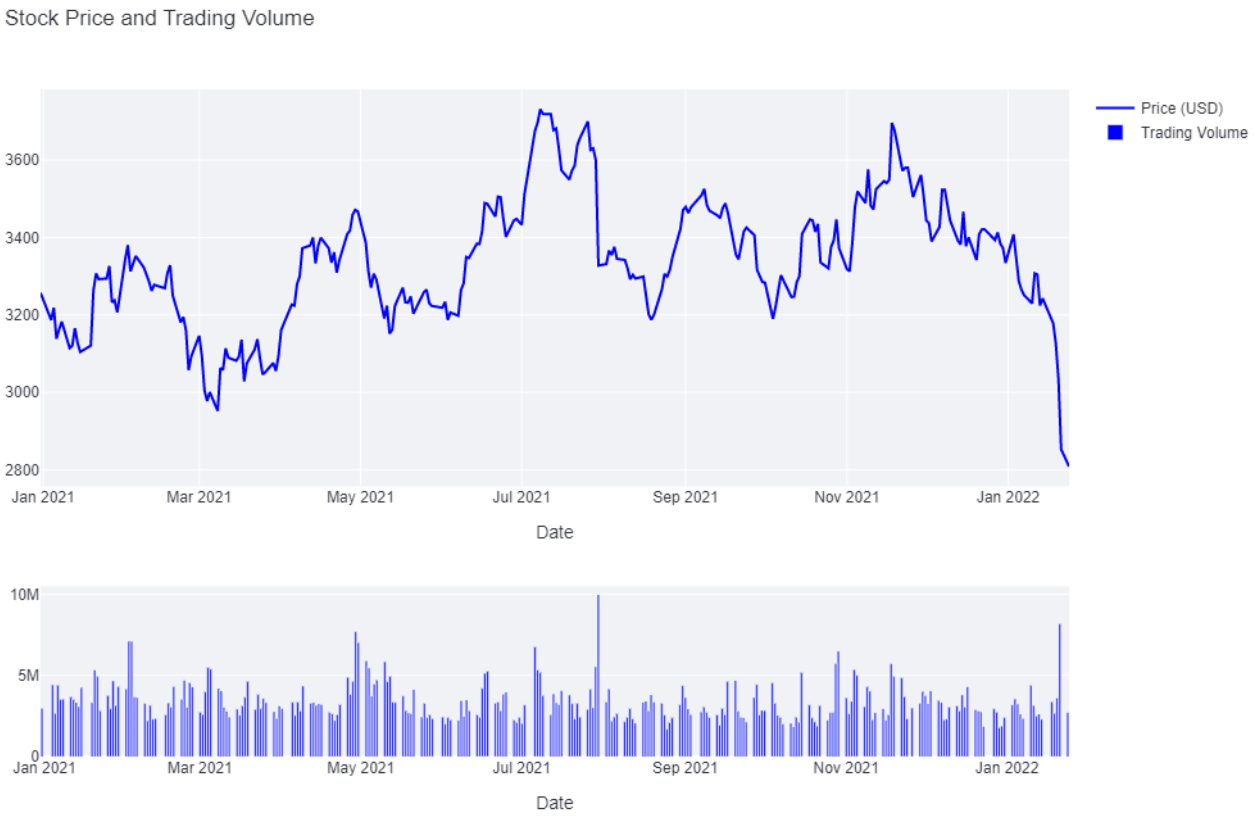
6.5.2 Giá và khối lượng giao dịch

Trong hộp chọn chỉ báo, nếu người dùng chọn “None” nghĩa là không dùng chỉ báo nào. Lúc này Dashboard sẽ hiện ra hai biểu đồ là biểu đồ giá và biểu đồ khối lượng giao dịch.

Lưu ý: Tất cả quá trình trực quan hóa dữ liệu trong bài viết này đều dùng thư viện Plotly. Bạn có thể học sử dụng thư viện Plotly qua các hướng dẫn trên mạng hoặc qua link này: https://plotly.com/ .

```
180     if indicator == "None":  # stock price and trading volume
181
182         fig = make_subplots(rows=2, cols=1, row_heights=[0.7, 0.3])
183         fig.add_trace(go.Scatter(x=df.index, y=df["Adj Close"], mode="lines", name="Price (USD)", \
184             line=dict(color="blue")), row=1, col=1)
185         fig.add_trace(go.Bar(x=df.index, y=df["Volume"], name="Trading Volume", \
186             marker_color="blue"), row=2, col=1)
187         fig.update_xaxes(title="Date", row=1, col=1)
188         fig.update_xaxes(title="Date", row=2, col=1)
189         fig.update_layout(title="Stock Price and Trading Volume", autosize=False, width=1200, height=700)
190
191         return fig
```

Biểu đồ Giá cổ phiếu và Khối lượng giao dịch của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.



6.5.3 On-balance Volume (OBV)

Sau đây, tôi sẽ xây dựng mô hình của OBV Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

Ở đây tôi sẽ không hướng dẫn lý thuyết và cách sử dụng các chỉ báo vì nó không phải là nội dung chính của đề tài này. Vì thế, bạn có thể học cách sử dụng OBV Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/terms/o/onbalancevolume.asp .

Ngoài ra, tôi cũng sẽ không trực tiếp hướng dẫn xây dựng mô hình cho các chỉ báo vì chúng rất dài và khó có thể giải thích toàn bộ trong bài viết này được. Vì thế, tôi sẽ để lại link để các bạn có thể truy

cập và học cách xây dựng chúng. Hơn nữa, có thể mô hình của tôi không được tối ưu nên các bạn hoàn toàn có thể sử dụng những cách khác để xây dựng các chỉ báo.

Để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình OBV Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=MRGXd8eaWB4&t=803s&ab_channel=ComputerScience .

```
193     elif indicator == "OBV":  # create OBV indicator
194
195
196         OBV = []
197         OBV.append(0)
198         for i in range(1, len(df["Adj Close"])):
199             if df["Adj Close"][i] > df["Adj Close"][i-1]:
200                 OBV.append(OBV[-1] + df.Volume[i])
201             elif df["Adj Close"][i] < df["Adj Close"][i-1]:
202                 OBV.append(OBV[-1] - df.Volume[i])
203             else:
204                 OBV.append(OBV[-1])
205
206         # Store OBV and OBV_EMA
207         df_obv = df.copy()
208         df_obv["OBV"] = OBV
209         df_obv["OBV_EMA"] = df_obv["OBV"].ewm(span=period).mean()
210
211         def get_signal_OBV(signal, col1, col2):  # def signal
212             buy_signal = []
213             sell_signal = []
214             flag = 0
215             for i in range(len(signal)):
216                 if signal[col1][i] > signal[col2][i] and flag != 1:
217                     buy_signal.append(signal["Adj Close"][i])
218                     sell_signal.append(np.nan)
219                     flag = 1
220                 elif signal[col1][i] < signal[col2][i] and flag != -1:
221                     buy_signal.append(np.nan)
222                     sell_signal.append(signal["Adj Close"][i])
223                     flag = -1
224                 else:
225                     buy_signal.append(np.nan)
226                     sell_signal.append(np.nan)
227             return buy_signal, sell_signal
228
229         # get signal
230         df_obv["Buy"] = get_signal_OBV(df_obv, "OBV", "OBV_EMA")[0]
231         df_obv["Sell"] = get_signal_OBV(df_obv, "OBV", "OBV_EMA")[1]
```

```
232
233         # initialize figure with subplots
234         fig = make_subplots(rows=2, cols=1)
235         # add traces
236         # figure 1
237         fig.add_trace(go.Scatter(x=df_obv.index, y=df_obv["Adj Close"], mode="lines", name="Price"), row=1, col=1)
238         fig.add_trace(go.Scatter(x=df_obv.index, y=df_obv["Buy"], mode="markers", \
239             name="Buy Signal", marker=dict(color="green", symbol="arrow-up")), row=1, col=1)
240         fig.add_trace(go.Scatter(x=df_obv.index, y=df_obv["Sell"], mode="markers", \
241             name="Sell Signal", marker=dict(color="red", symbol="arrow-down")), row=1, col=1)
242         # figure 2
243         fig.add_trace(go.Scatter(x=df_obv.index, y=df_obv["OBV"], mode="lines", name="OBV", \
244             line=dict(color="orange")), row=2, col=1)
245         fig.add_trace(go.Scatter(x=df_obv.index, y=df_obv["OBV_EMA"], mode="lines", name="OBV_EMA", \
246             line=dict(color="purple")), row=2, col=1)
247         fig.update_xaxes(title="Date", row=1, col=1)
248         fig.update_yaxes(title="USD", row=1, col=1)
249         fig.update_xaxes(title="Date", row=2, col=1)
250         fig.update_layout(title="OBV Indicator", autosize=False, width=1200, height=900)
251
252         return fig
```

Biểu đồ áp dụng OBV Indicator (Period = 14) của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.

6.5.4 Bollinger Bands

Tôi sẽ tiếp tục xây dựng mô hình Bollinger Band Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

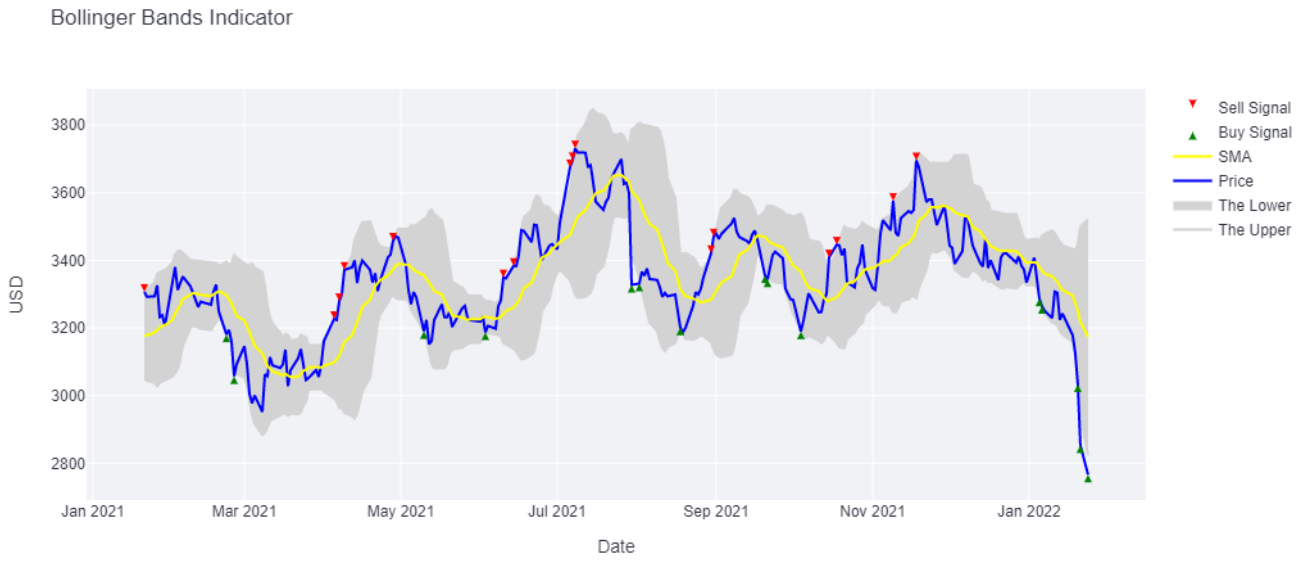
Bạn có thể học cách sử dụng Bollinger Band Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/articles/technical/102201.asp#:~:text=When%20using%20Bollinger%20Bands%C2%AE,represent%20the%20upper%20price%20target .

Ngoài ra, để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình Bollinger Band Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=gEIw2iUlFYc&t=1168s&ab_channel=ComputerScience .

```
    elif indicator == "Bollinger Bands":  # create Bollinger Bands indicator

        # calculate the SMA
        df["SMA"] = df["Adj Close"].rolling(window=period).mean()
        # get the Standard Deviation
        df["STD"] = df["Adj Close"].rolling(window=period).std()
        # calculate the upper Bollinger Bands
        df["Upper"] = df["SMA"] + 2*df["STD"]
        # calculate the lower Bollinger Bands
        df["Lower"] = df["SMA"] - 2*df["STD"]

        df_bb = df[period-1:]

        def get_signal_BollingerBands(data):  # def signal
            buy_signal = []
            sell_signal = []
            for i in range(len(data["Adj Close"])):
                if data["Adj Close"][i] > data["Upper"][i]:
                    buy_signal.append(np.nan)
                    sell_signal.append(data["Adj Close"][i])
                elif data["Adj Close"][i] < data["Lower"][i]:
                    buy_signal.append(data["Adj Close"][i])
                    sell_signal.append(np.nan)
                else:
                    buy_signal.append(np.nan)
                    sell_signal.append(np.nan)
            return buy_signal, sell_signal

        # get signal
        df_bb["Buy"] = get_signal_BollingerBands(df_bb)[0]
        df_bb["Sell"] = get_signal_BollingerBands(df_bb)[1]
```

```

        # initialize figure
        fig = go.Figure()
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["Upper"], fill=None, mode="lines", \
            name="The Upper", line=dict(color="lightgray")))
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["Lower"], fill="tonexty", mode="lines", \
            name="The Lower", fillcolor="lightgray", line=dict(color="lightgray")))
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["Adj Close"], mode="lines", name="Price", \
            line=dict(color="blue")))
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["SMA"], mode="lines", name="SMA", \
            line=dict(color="yellow")))
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["Buy"], mode="markers", \
            name="Buy Signal", marker=dict(color="green", symbol="arrow-up")))
        fig.add_trace(go.Scatter(x=df_bb.index, y=df_bb["Sell"], mode="markers", \
            name="Sell Signal", marker=dict(color="red", symbol="arrow-down")))
        fig.update_layout(title="Bollinger Bands Indicator", xaxis_title="Date", \
            yaxis_title="USD", autosize=False, width=1200, height=500)

        return fig
```

Biểu đồ áp dụng Bollinger Bands Indicator (Period = 14) của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.



6.5.5 Relative Strength Index (RSI)

Tôi sẽ tiếp tục xây dựng mô hình RSI Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

Bạn có thể học cách sử dụng RSI Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/terms/r/rsi.asp .

Ngoài ra, để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình RSI Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=oiheV1xXEtg&t=737s&ab_channel=NeuralNine .

```
    elif indicator == "RSI":  # create RSI indicator

        delta = df["Adj Close"].diff(1)
        delta.dropna(inplace=True)

        positive = delta.copy()
        negative = delta.copy()
        positive[positive < 0] = 0
        negative[negative > 0] = 0

        average_gain = positive.rolling(window=period).mean()
        average_loss = abs(negative.rolling(window=period).mean())
        relative_strength = average_gain / average_loss
        RSI = 100 - (100 / (1 + relative_strength))

        df_rsi = pd.DataFrame()
        df_rsi["Adj Close"] = df["Adj Close"].copy()
        df_rsi["RSI"] = RSI  # add RSI column

        def get_signal_RSI(dt, high, low):  # def signal
            buy_signals = []
            sell_signals = []
            for i in range(len(dt["RSI"])):
                if dt["RSI"][i] > high:
                    buy_signals.append(np.nan)
                    sell_signals.append(dt["Adj Close"][i])
                elif dt["RSI"][i] < low:
                    buy_signals.append(dt["Adj Close"][i])
                    sell_signals.append(np.nan)
                else:
                    buy_signals.append(np.nan)
                    sell_signals.append(np.nan)
            return buy_signals, sell_signals

        # get signal
        df_rsi["Buy"] = get_signal_RSI(df_rsi, overbought, oversold)[0]
        df_rsi["Sell"] = get_signal_RSI(df_rsi, overbought, oversold)[1]

```

```
339         # get signal
340         df_rsi["Buy"] = get_signal_RSI(df_rsi, overbought, oversold)[0]
341         df_rsi["Sell"] = get_signal_RSI(df_rsi, overbought, oversold)[1]
342 
343         # initialize figure with subplots
344         fig = make_subplots(rows=2, cols=1)
345         # add traces
346         # figure 1
347         fig.add_trace(go.Scatter(x=df_rsi.index, y=df_rsi["Adj Close"], mode="lines", \
348             name="Price"), row=1, col=1)
349         fig.add_trace(go.Scatter(x=df_rsi.index, y=df_rsi["Buy"], mode="markers", name="Buy Signal", \
350             marker=dict(color="green", symbol="arrow-up")), row=1, col=1)
351         fig.add_trace(go.Scatter(x=df_rsi.index, y=df_rsi["Sell"], mode="markers", name="Sell Signal", \
352             marker=dict(color="red", symbol="arrow-down")), row=1, col=1)
353         # figure 2
354         fig.add_trace(go.Scatter(x=df_rsi.index, y=df_rsi["RSI"], mode="lines", name="RSI"), row=2, col=1)
355         fig.add_hline(y=overbought, line=dict(color="green", dash="dash", width=1), row=2, col=1)
356         fig.add_hline(y=oversold, line=dict(color="green", dash="dash", width=1), row=2, col=1)
357         # update properties
358         fig.update_xaxes(title="Date", row=1, col=1)
359         fig.update_yaxes(title="USD", row=1, col=1)
360         fig.update_xaxes(title="Date", row=2, col=1)
361         fig.update_layout(title="RSI Indicator", autosize=False, width=1200, height=900)
362 
363         return fig
```

Biểu đồ áp dụng RSI Indicator của cổ phiếu AMZN (Period = 14, Over Bought = 70, Over Sold = 30) từ ngày 1/1/2021 đến ngày 24/1/2022.

6.5.6 Moving average (MA)

Tôi sẽ tiếp tục xây dựng mô hình MA Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

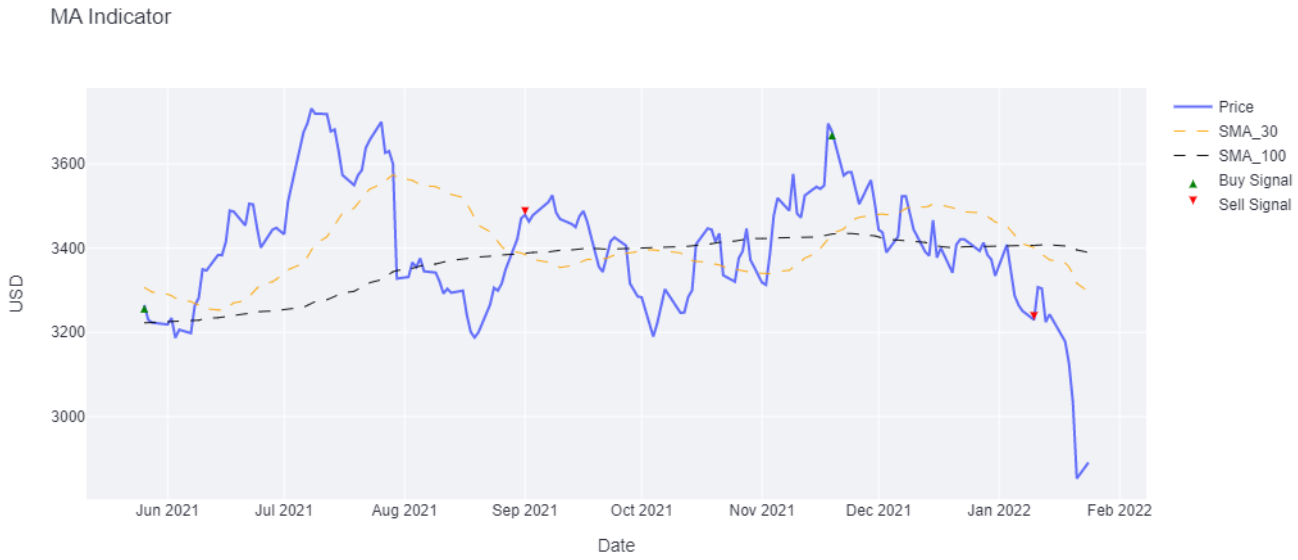
Bạn có thể học cách sử dụng MA Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/terms/m/movingaverage.asp .

Ngoài ra, để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình MA Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=FEDBsbTFG1o&ab_channel=NeuralNine .

```
365     elif indicator == "MA":  # create MA indicator
366
367         ma_1 = period
368         ma_2 = period_slow
369         df[f"SMA_{ma_1}"] = df["Adj Close"].rolling(window=ma_1).mean()
370         df[f"SMA_{ma_2}"] = df["Adj Close"].rolling(window=ma_2).mean()
371         df = df.iloc[ma_2:]
372
373         # create signal
374         buy_signal = []
375         sell_signal = []
376         trigger = 0
377         for i in range(len(df)):
378             if df[f"SMA_{ma_1}"].iloc[i] > df[f"SMA_{ma_2}"].iloc[i] and trigger != 1:
379                 buy_signal.append(df["Adj Close"].iloc[i])
380                 sell_signal.append(np.nan)
381                 trigger = 1
382             elif df[f"SMA_{ma_1}"].iloc[i] < df[f"SMA_{ma_2}"].iloc[i] and trigger != -1:
383                 buy_signal.append(np.nan)
384                 sell_signal.append(df["Adj Close"].iloc[i])
385                 trigger = -1
386             else:
387                 buy_signal.append(np.nan)
388                 sell_signal.append(np.nan)
389
390         # get signal
391         df["Buy"] = buy_signal
392         df["Sell"] = sell_signal
```

```
393
394         # initialize figure
395         fig = go.Figure()
396         fig.add_trace(go.Scatter(x=df.index, y=df["Adj Close"], mode="lines", name="Price"))
397         fig.add_trace(go.Scatter(x=df.index, y=df[f"SMA_{ma_1}"], mode="lines", name=f"SMA_{ma_1}", \
398             line=dict(color="orange", dash="dash", width=0.9)))
399         fig.add_trace(go.Scatter(x=df.index, y=df[f"SMA_{ma_2}"], mode="lines", name=f"SMA_{ma_2}", \
400             line=dict(color="black", dash="dash", width=0.9)))
401         fig.add_trace(go.Scatter(x=df.index, y=df["Buy"], mode="markers", \
402             name="Buy Signal", marker=dict(color="green", symbol="arrow-up")))
403         fig.add_trace(go.Scatter(x=df.index, y=df["Sell"], mode="markers", \
404             name="Sell Signal", marker=dict(color="red", symbol="arrow-down")))
405         fig.update_layout(title="MA Indicator", xaxis_title="Date", yaxis_title="USD", \
406             autosize=False, width=1200, height=500)
407
408         return fig
```

Biểu đồ áp dụng MA Indicator (Fast Period = 30, Slow Period = 100) của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.



6.5.7 Moving average convergence divergence (MACD)

Tôi sẽ tiếp tục xây dựng mô hình MACD Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

Bạn có thể học cách sử dụng MACD Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/terms/m/macd.asp .

Ngoài ra, để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình MACD Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=kz_NJERCgm8&t=823s&ab_channel=ComputerScience .

```
    # build MACD indicator
    elif indicator == "MACD":
        ShortEMA = df["Adj Close"].ewm(span=period, adjust=False).mean()
        LongEMA = df["Adj Close"].ewm(span=period_slow, adjust=False).mean()
        MACD = ShortEMA - LongEMA
        signal = MACD.ewm(span=period_signal, adjust=False).mean()

        df["MACD"] = MACD
        df["Signal Line"] = signal

        def buy_sell(signal):  # def signal
            buy = []
            sell = []
            flag = 0

            for i in range(0, len(signal)):
                if signal["MACD"][i] > signal["Signal Line"][i]:
                    sell.append(np.nan)
                    if flag != 1:
                        buy.append(signal["Adj Close"][i])
                        flag = 1
                    else:
                        buy.append(np.nan)
                elif signal["MACD"][i] < signal["Signal Line"][i]:
                    buy.append(np.nan)
                    if flag != -1:
                        sell.append(signal["Adj Close"][i])
                        flag = -1
                    else:
                        sell.append(np.nan)
                else:
                    buy.append(np.nan)
                    sell.append(np.nan)

            return buy, sell

        # get signal
        df["Buy_Signal_Price"] = buy_sell(df)[0]
        df["Sell_Signal_Price"] = buy_sell(df)[1]
```

```
449
450         # initialize figure with subplots
451         fig = make_subplots(rows=2, cols=1)
452         # add traces
453         # figure 1
454         fig.add_trace(go.Scatter(x=df.index, y=df["Adj Close"], mode="lines", name="Price"), \
455             row=1, col=1)
456         fig.add_trace(go.Scatter(x=df.index, y=df["Buy_Signal_Price"], mode="markers", \
457             name="Buy Signal", marker=dict(color="green", symbol="arrow-up")), row=1, col=1)
458         fig.add_trace(go.Scatter(x=df.index, y=df["Sell_Signal_Price"], mode="markers", \
459             name="Sell Signal", marker=dict(color="red", symbol="arrow-down")), row=1, col=1)
460         # figure 2
461         fig.add_trace(go.Scatter(x=df.index, y=df["MACD"], mode="lines", name="MACD", \
462             line=dict(color="purple")), row=2, col=1)
463         fig.add_trace(go.Scatter(x=df.index, y=df["Signal Line"], mode="lines", name="Signal Line", \
464             line=dict(color="orange")), row=2, col=1)
465         # update properties
466         fig.update_xaxes(title="Date", row=1, col=1)
467         fig.update_yaxes(title="USD", row=1, col=1)
468         fig.update_xaxes(title="Date", row=2, col=1)
469         fig.update_layout(title="MACD Indicator", autosize=False, width=1200, height=900)
470
471         return fig
```

Biểu đồ áp dụng MACD Indicator (Fast Period = 12, Slow Period = 26, MACD Period = 9) của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.

6.5.8 Money Flow Index (MFI)

Tôi sẽ tiếp tục xây dựng mô hình MFI Indicator, đưa ra các tín hiệu Buy hoặc Sell từ chỉ báo này và trực quan hóa chúng.

Lưu ý:

Bạn có thể học cách sử dụng MFI Indicator trên mạng hoặc qua link sau: https://www.investopedia.com/terms/m/mfi.asp .

Ngoài ra, để xem hướng dẫn chi tiết cách mà tôi xây dựng mô hình MFI Indicator, bạn có thể xem qua link sau: https://www.youtube.com/watch?v=tF1Lz4WBQwM&t=906s&ab_channel=ComputerScience .

```
    # create MFI indicator
    else:

        typical_price = (df["Adj Close"] + df["High"] + df["Low"]) / 3 # caculate typical price
        money_flow = typical_price * df["Volume"]  # caculate money flow
        positive_flow = []
        negative_flow = []

        # loop through the typical price
        for i in range(1, len(typical_price)):
            if typical_price[i] > typical_price[i-1]:
                positive_flow.append(money_flow[i-1])
                negative_flow.append(0)
            elif typical_price[i] < typical_price[i-1]:
                negative_flow.append(money_flow[i - 1])
                positive_flow.append(0)
            else:
                positive_flow.append(0)
                negative_flow.append(0)

        # get all of the positive and negative money flows within the time period
        positive_mf = []
        negative_mf = []
        for i in range(period-1, len(positive_flow)):
            positive_mf.append(sum(positive_flow[i + 1 - period:i + 1]))
        for i in range(period-1, len(negative_flow)):
            negative_mf.append(sum(negative_flow[i + 1 - period:i + 1]))
```

```

         # caculate MFI
         mfi = 100 * (np.array(positive_mf)/(np.array(positive_mf)+np.array(negative_mf)))
         df_mfi = pd.DataFrame()
         df_mfi["MFI"] = mfi
         new_df = df[period:]
         new_df["MFI"] = mfi

         def get_signal(data, high, low):  # def signal
             buy_signals = []
             sell_signals = []
             for i in range(len(data["MFI"])):
                 if data["MFI"][i] > high:
                     buy_signals.append(np.nan)
                     sell_signals.append(data["Adj Close"][i])
                 elif data["MFI"][i] < low:
                     buy_signals.append(data["Adj Close"][i])
                     sell_signals.append(np.nan)
                 else:
                     buy_signals.append(np.nan)
                     sell_signals.append(np.nan)
             return buy_signals, sell_signals

         # get signal
         new_df["Buy"] = get_signal(new_df, overbought, oversold)[0]
         new_df["Sell"] = get_signal(new_df, overbought, oversold)[1]
```

```
526
527         # initialize figure with subplots
528         fig = make_subplots(rows=2, cols=1)
529         # add traces
530         # figure 1
531         fig.add_trace(go.Scatter(x=new_df.index, y=new_df["Adj Close"], mode="lines", name="Price"), row=1, col=1)
532         fig.add_trace(go.Scatter(x=new_df.index, y=new_df["Buy"], mode="markers", name="Buy Signal", \
533             marker=dict(color="green", symbol="arrow-up")), row=1, col=1)
534         fig.add_trace(go.Scatter(x=new_df.index, y=new_df["Sell"], mode="markers", name="Sell Signal", \
535             marker=dict(color="red", symbol="arrow-down")), row=1, col=1)
536         # figure 2
537         fig.add_trace(go.Scatter(x=new_df.index, y=new_df["MFI"], mode="lines", name="MFI"), row=2, col=1)
538         fig.add_hline(y=overbought, line=dict(color="green", dash="dash", width=1), row=2, col=1)
539         fig.add_hline(y=oversold, line=dict(color="green", dash="dash", width=1), row=2, col=1)
540         # update properties
541         fig.update_xaxes(title="Date", row=1, col=1)
542         fig.update_yaxes(title="USD", row=1, col=1)
543         fig.update_xaxes(title="Date", row=2, col=1)
544         fig.update_layout(title="MFI Indicator", autosize=False, width=1200, height=900)
545
546         return fig
```

Biểu đồ áp dụng MFI Indicator (Period = 14, Over Bought = 70, Over Sold = 30) của cổ phiếu AMZN từ ngày 1/1/2021 đến ngày 24/1/2022.

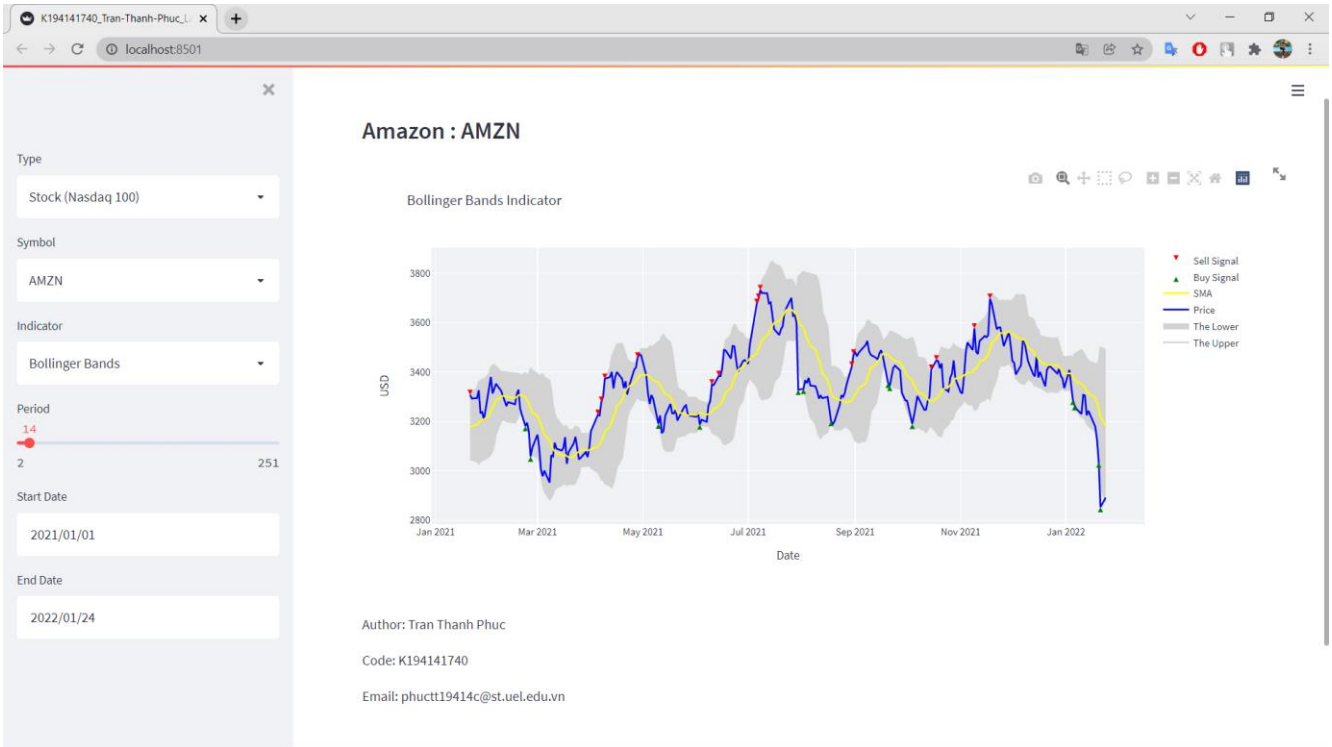
6.6 Visualize the results on the web

Sau khi đã hoàn thành hàm khổng lồ cho việc load dữ liệu, xây dựng các chỉ báo và đưa ra các biểu đồ, tôi sẽ đưa biểu đồ mà được hàm trả về lên trang web bằng thư viện Streamlit. Đầu tiên, tôi sẽ cho xuất hiện tên và mã cổ phiếu của công ty (hoặc tiền điện tử) trước. Sau đó, tôi tiếp tục đưa biểu đồ chứa các thuộc tính mà người dùng đã lựa chọn lên web thông qua lệnh "st.plotly_chart()" để đưa hàm mà tôi đã xây dựng bên trên vào để trả về kết quả là biểu đồ. Cuối cùng, tôi để lại một vài thông tin cá nhân ở cuối trang web.

```
548   # visualize the results on the web
549   sub = "" + name.loc[option_symbol].at["Name"]
550   st.subheader(f"{sub} : {option_symbol}")
551   st.plotly_chart(test(option_symbol, option_indi, option_start, option_end, option_periods, option_periods_slow, \
552       option_periods_signal, option_overbought, option_oversold, option_type), use_container_width=True)
553   st.write("Author: Tran Thanh Phuc")
554   st.write("Code: K194141740")
555   st.write("Email: phuctt19414c@st.uel.edu.vn")
```

Dưới này là hình ảnh sau khi mọi thứ đã được trực quan hóa lên web nếu người dùng chọn Bollinger Bands Indicator và cổ phiếu AMZN.

Lưu ý: Tôi chỉ minh họa một chỉ báo và cổ phiếu. Vì thế nếu bạn chọn chỉ báo khác hoặc cổ phiếu (tiền điện tử) khác thì sẽ có nhiều khác biệt. Tuy nhiên, về giao diện thì chúng tương tự nhau.



Đây cũng là lúc một bản điều khiển tương tác cho phân tích kỹ thuật cơ bản được hoàn thành. Thông qua giao diện của trình duyệt web, người dùng có thể sử dụng nó dễ dàng.

7. CONCLUSION

Qua tất cả những phần trên, tôi đã trình bày toàn và giải thích toàn bộ quá trình tôi thực hiện chủ đề "Build A Simple Technical Analysis Dashboard In Python". Các phần bao gồm: Lý do chọn đề tài, Ý nghĩa đề tài, Các đối tượng trong bài, Các thư viện, Nguồn dữ liệu, Diễn giải quá trình thực hiện

8. REFERENCE

Jie Jenn. (2020). Web Scraping Wikipedia tables using Python. [Online Video]. 20 April 2020. Available from: https://www.youtube.com/watch?v=ICXR9nDbudk&t=285s&ab_channel=JieJenn. [Accessed: 25 January 2022].

Dat, V., 2021. Getting started - Vietnam Stock Data. [online] Vnstock-data-python.readthedocs.io. Available at: <https://vnstock-data-python.readthedocs.io/en/latest/#usage> [Accessed 25 January 2022].

Part Time Larry. (2021). Streamlit - Building Financial Dashboards with Python. [Online Video]. 20 February 2021. Available from: https://www.youtube.com/watch?v=0ESc1bh3eIg&ab_channel=PartTimeLarry. [Accessed: 25 January 2022].

Computer Science. (2020). On-Balance-Volume (OBV) Stock Trading Strategy Using Python. [Online Video]. 12 May 2020. Available from: https://www.youtube.com/watch?v=MRGXd8eaWB4&t=803s&ab_channel=ComputerScience. [Accessed: 25 January 2022].

Computer Science. (2020). Stock Trading Using Bollinger Bands & Python. [Online Video]. 11 October 2020. Available from: https://www.youtube.com/watch?v=gEIw2iUlFYc&t=1168s&ab_channel=ComputerScience. [Accessed: 25 January 2022].

NeuralNine. (2020). Technical Stock Analysis (RSI) in Python. [Online Video]. 2 October 2020. Available from: https://www.youtube.com/watch?v=oiheV1xXEtg&t=737s&ab_channel=NeuralNine. [Accessed: 25 January 2022].

NeuralNine. (2021). Algorithmic Trading Strategy in Python. [Online Video]. 4 July 2021. Available from: https://www.youtube.com/watch?v=FEDBsbTFG1o&ab_channel=NeuralNine. [Accessed: 25 January 2022].

Computer Science. (2020). Algorithmic Trading Strategy Using MACD & Python. [Online Video]. 22 June 2020. Available from: https://www.youtube.com/watch?v=kz_NJERCgm8&t=823s&ab_channel=ComputerScience. [Accessed: 25 January 2022].

Computer Science. (2020). Algorithmic Trading Using Money Flow Index (MFI) and Python. [Online Video]. 1 October 2020. Available from: https://www.youtube.com/watch?v=tF1Lz4WBQwM&t=906s&ab_channel=ComputerScience. [Accessed: 25 January 2022].