Sigma Internship Coding Challenge

Algorithmic Trading Model for Portfolio Optimization

PADHMA PRAKASH A (21PT17)

PSG COLLEGE OF TECHNOLOGY, COIMBATORE

Goal:

To build a minimalist trading workflow and involves pulling daily close price data for Apple stock for the year 2023 and implementing a simple model to make buy orders

Overview :

Installation of QuantRocket:

- Since QuantRocket runs on Docker, I installed Docker desktop with WSL 2 backend.
- Start > Docker by running "**docker run hello-world** " on windows powershell for installation verification.
- Then downloaded necessary compose files required for QunatRocket which will run the following containers.

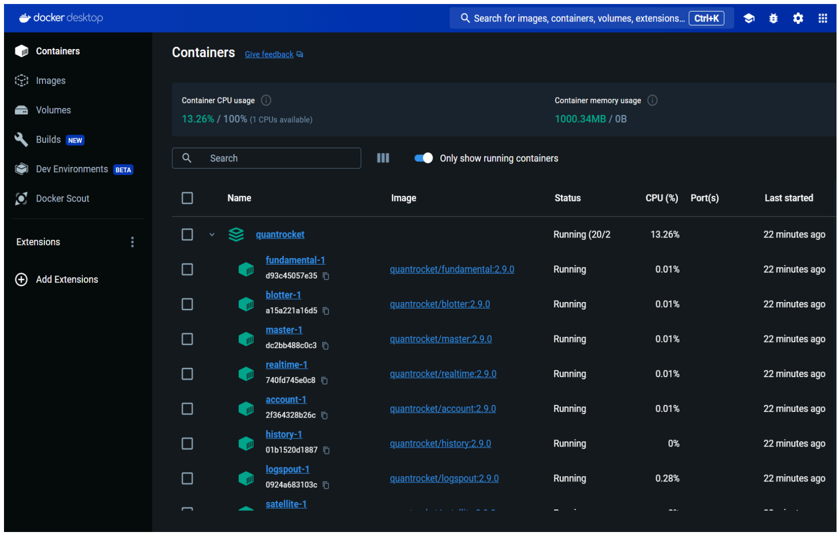
```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\padhm> cd ~/quantrocket
PS C:\Users\padhm\quantrocket> docker compose -p quantrocket up -d
[+] Running 20/20
 Container quantrocket-ibgrouter-1         Running
 Container quantrocket-jupyter-1           Running
 Container quantrocket-license-service-1   Running
 Container quantrocket-codeload-1          Running
 Container quantrocket-satellite-1         Running
 Container quantrocket-postgres-1          Running
 Container quantrocket-moonshot-1          Running
 Container quantrocket-db-1                Running
 Container quantrocket-ibg1-1              Running
 Container quantrocket-history-1           Running
 Container quantrocket-account-1           Running
 Container quantrocket-countdown-1         Running
 Container quantrocket-flightlog-1         Running
 Container quantrocket-master-1            Running
 Container quantrocket-blotter-1           Running
 Container quantrocket-fundamental-1       Running
 Container quantrocket-realtime-1          Running
 Container quantrocket-zipline-1           Running
 Container quantrocket-houston-1           Running
 Container quantrocket-logspout-1          Running
PS C:\Users\padhm\quantrocket>
```

Then started the docker engine and accessed Jupyter environment in browser at,

http://localhost:1969



Data Collection (Pulling daily close prices for Apple Stock) process of fetching historical stock price data for Apple (AAPL) using QuantRocket :

(Check "**Fetch_DataQR.ipynb**" in the repository.)

Setting License:

Received my license key which is a unique identifier that authorizes the usage of QuantRocket services.

licenseKey = "f785a9ee-dfa4-11ee-948d-5d738746c51e"

set_license(licenseKey)

Getting license Profile:

get_license_profile()

retrieves and prints information about the currently set license profile

Creating a US Stock Database:

create_usstock_db("usstock-1d")

creates a QuantRocket database named "usstock-1d" specifically for storing historical daily stock price data for U.S. stocks.

Collecting Historical data:

collect_history("usstock-1d")

#collects historical data for all U.S. stocks and stores it in the "usstock-1d" database.

Downloading Historical data for Apple Stock (AAPL)

prices = get_prices("usstock-free-1d", universes="usstock-free-active", start_date="2023-01-01",end_date="2023-12-31" ,fields=["Close"])

#downloads historical daily close prices for Apple stock (AAPL) from the "usstock-1d" database for the specified date range (from January 1, 2023, to December 31, 2023). The data is filtered to include only the "Close" field. The resulting data is saved to a CSV file named "AppleStock.csv".

A	B	C
Field	Date	FIBBG000B9XRY4
Close	2023-01-03	124.2163
Close	2023-01-04	125.4975
Close	2023-01-05	124.1666
Close	2023-01-06	128.7352
Close	2023-01-09	129.2616
Close	2023-01-10	129.8377
Close	2023-01-11	132.5788
Close	2023-01-12	132.4994
Close	2023-01-13	133.8402
Close	2023-01-17	135.0121
Close	2023-01-18	134.2871
Close	2023-01-19	134.3467
Close	2023-01-20	136.9289
Close	2023-01-23	140.1468
Close	2023-01-24	141.5571
Close	2023-01-25	140.8917
Close	2023-01-26	142.9774
Close	2023-01-27	144.9339

<u>Model Logic for state classifications and % returns :</u>

(Check "Model_main.ipynb" in the repository.)

Uses daily close prices (p(d)) and calculates percentage returns (r(d))

%Returns: r(d) = [p(d) - p(d-1)] / [p(d-1)]

Classified the state (s(d)) based on percentage returns.

If r(d) >= 0.1, s(d) = +1 (Bull state)

Else if -0.1 < r(d) < 0.1, s(d) = 0 (Flat state)

Else s(d) = -1 (Bear state)

Obtained V(N) based on the states classified.

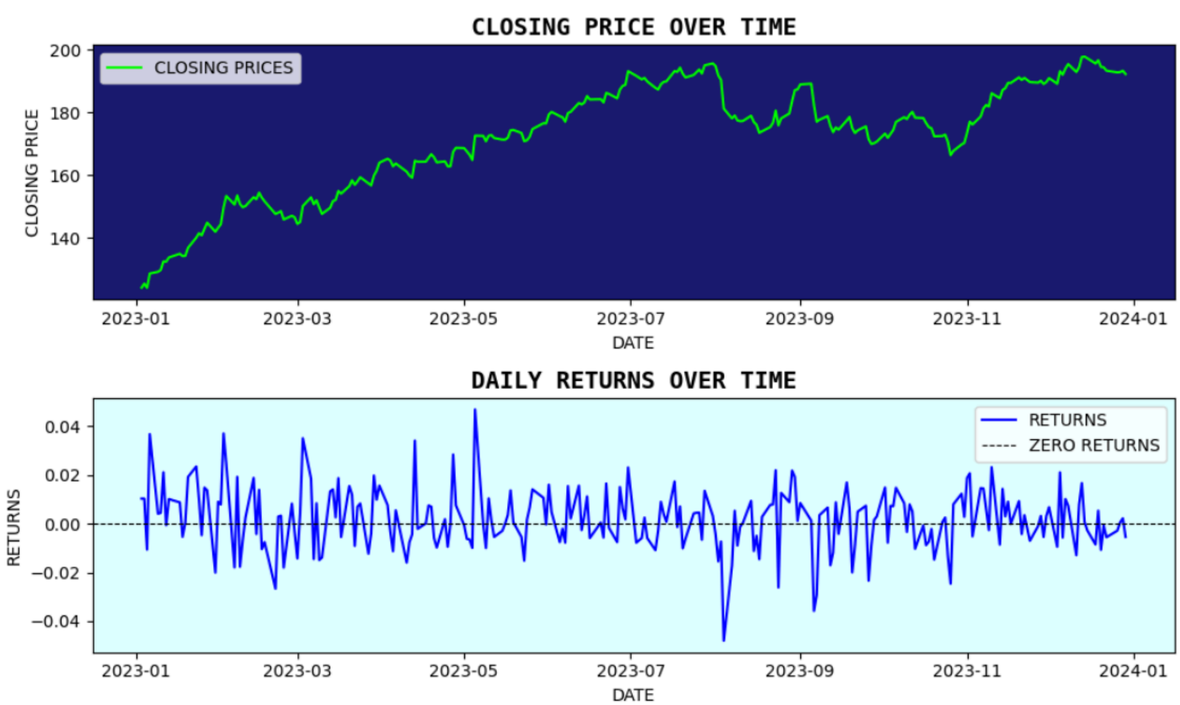
If s(d+1) = 1 & s(d) = 0 ---> V(d+1) = V(d) + 1

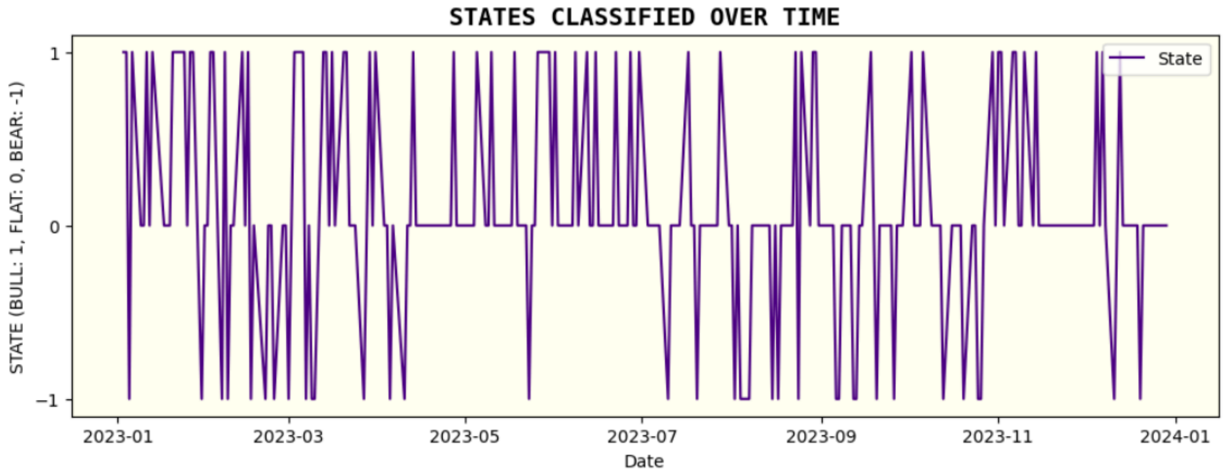
Else if s(d+1) = -1 & s(d) = 0 ---> V(d+1) = V(d) - 1

Else (in all other cases) ---> V(d+1) = V(d)

DATA ANALYSIS AND VISUALIZATION:

Compelling plots for better understanding about the data fetched.





Implementation of value function V(N):

PORTFOLIO VALUE OBTAINED (PERFORMING BUY TRADES ON ALL DAYS)

Placing buy order:

```
if curr_state == 1 and prev_state == 0:
    pfVAL += 1
elif curr_state == -1 and prev_state == 0:
    pfVAL -= 1
```

Note: This is not an optimal solution. Our objective is to **maximize** the Portfolio value

Day 241	18
Day 242	18
Day 243	17
Day 244	17
Day 245	17
Day 246	17
Day 247	17
Day 248	17
Day 249	17

Portfolio Value obtained: 17

OPTIMUM SOLUTION OBTAINED (FINDING IN STREAMING MANNER)

Performing Buy order trade when the current day is "**FLAT**" and followed by a "**BULL**"

```
if curr_state == 1 and prev_state == 0:
    pfVAL += 1
    n_ord += 1
    buy_indices.append(i)
```

FINAL PORTFOLIO VALUE = **40**

OPTIMAL BUY INDICES = **[6, 8, 12, 16, 21, 28, 30, 41, 50, 52, 59, 61, 69, 79, 85, 88, 94, 100, 103, 108, 110, 113, 117, 120, 123, 133, 142, 160, 164, 177, 187, 191, 207, 209, 212, 216, 218, 232, 234, 238]**

FINDING TRANSITION DISTRIBUTIONS IN STREAMED MANNER:

- Used generator function to yield intermediate results.
- Updated transition counts and probabilities as new data points arrive.
- Continuously calculated probabilities based on streaming data.

Bull	0.333333	0.333333	0.333333

Transition Distribution(5):

	Bear	Flat	Bull
Bear	0	0	1
Flat	0	1	0
Bull	0.333333	0.333333	0.333333

...... Intermediate results ......

FINAL TRANSITION DISTRIBUTION:

	Bear	Flat	Bull
Bear	0.142857	0.742857	0.114286
Flat	0.146497	0.598726	0.254777
Bull	0.122807	0.666667	0.210526