

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐHQGHN

Báo cáo kết quả bài tập: Triển khai hệ thống phân tích và xử lý dữ liệu

Tác giả: Bùi Minh Quân

Mã sinh viên: 23020415

buiminhquandz2005@gmail.com

<https://github.com/Bui-Minh-Quan/stock-bigdata>

Ngày 20 tháng 10 năm 2025

Mục lục

1	Giới thiệu	2
2	Hệ thống và công nghệ	3
2.1	Công nghệ sử dụng	3
2.2	Cấu trúc hệ thống	3
2.3	Triển khai hệ thống	4
3	Dữ liệu	6
3.1	Cấu trúc dữ liệu	6
3.2	Đọc dữ liệu cho việc xử lý	7
4	Xử lý và Phân tích Dữ liệu	9
4.1	Tiền xử lý dữ liệu	9
4.2	Phân tích tổng quan	10
4.3	Phân tích chỉ số thị trường tổng thể	14
4.4	Phân tích chi tiết một số mã cổ phiếu tiêu biểu	15
4.5	Phân tích tương quan giữa các cổ phiếu	17
5	Xây dựng mô hình học máy	18
5.1	Chuẩn bị dữ liệu huấn luyện	18
5.2	Mô hình 1: Dự đoán giá đóng cửa	19
5.3	Mô hình 2: Dự đoán xu hướng tăng/giảm	20
6	Kết luận và Hướng phát triển	22
6.1	Kết luận	22
6.2	Hướng phát triển	22

Chương 1

Giới thiệu

Bối cảnh: Trong bối cảnh thị trường chứng khoán ngày càng phát triển mạnh mẽ, lượng dữ liệu tài chính được tạo ra mỗi ngày là vô cùng lớn, bao gồm giá mở cửa, giá đóng cửa, khối lượng giao dịch, biến động giá, cùng nhiều chỉ số kỹ thuật khác. Việc lưu trữ, xử lý và phân tích các tập dữ liệu khổng lồ này đòi hỏi những công cụ và hệ thống có khả năng mở rộng, xử lý song song và đáng tin cậy.

Công nghệ **Big Data** ra đời nhằm đáp ứng nhu cầu đó. Trong bài thực hành này, em tiến hành triển khai một **môi trường xử lý dữ liệu chứng khoán** sử dụng các công nghệ dữ liệu lớn bao gồm **HDFS**, **Apache Spark** và **Jupyter Notebook**, được triển khai thông qua nền tảng **Docker Compose** để dễ dàng thiết lập, quản lý và mô phỏng cụm hệ thống phân tán.

Bài thực hành hướng đến việc:

- Xây dựng hệ thống lưu trữ và xử lý dữ liệu lớn trên nền tảng Docker.
- Thực hiện các bước tiền xử lý dữ liệu chứng khoán: làm sạch, tạo đặc trưng (*feature engineering*) và trực quan hóa dữ liệu.
- Áp dụng hai mô hình học máy đơn giản (*Linear Regression* và *Logistic Regression*) nhằm dự đoán xu hướng biến động giá cổ phiếu.
- Đánh giá kết quả mô hình và thảo luận hướng mở rộng trong tương lai.

Chương 2

Hệ thống và công nghệ

2.1 Công nghệ sử dụng

Hệ thống được xây dựng dựa trên các công nghệ mã nguồn mở phổ biến trong lĩnh vực dữ liệu lớn, bao gồm:

- **Hadoop Distributed File System (HDFS):** dùng để lưu trữ dữ liệu phân tán, đảm bảo tính chịu lỗi và khả năng mở rộng.
- **Apache Spark:** dùng để xử lý và phân tích dữ liệu song song trên cụm máy ảo.
- **Jupyter Notebook:** cung cấp giao diện thân thiện cho việc tương tác, chạy thử và trực quan hóa dữ liệu.
- **Docker & Docker Compose:** được sử dụng để triển khai toàn bộ hệ thống dưới dạng các container độc lập, dễ dàng khởi động và quản lý.

2.2 Cấu trúc hệ thống

Hệ thống được mô phỏng dưới dạng một **cụm Big Data (Cluster)** hoàn chỉnh, bao gồm hai phần chính:

1. **Cụm Hadoop (HDFS)** với 1 *NameNode* và 4 *DataNode*, chịu trách nhiệm lưu trữ dữ liệu chúng khoán phân tán.
2. **Cụm Spark** gồm 1 *Spark Master* và 4 *Spark Worker*, thực hiện các tác vụ xử lý và phân tích dữ liệu.

Ngoài ra, hệ thống còn có các dịch vụ hỗ trợ:

- **ResourceManager, NodeManager, HistoryServer:** phục vụ việc quản lý tài nguyên và theo dõi tiến trình xử lý trong hệ thống Hadoop YARN.
- **Jupyter Notebook:** kết nối trực tiếp đến cụm Spark, cho phép người dùng viết mã Python và chạy các tác vụ xử lý dữ liệu trên Spark thông qua giao diện notebook.

2.3 Triển khai hệ thống

Toàn bộ các dịch vụ được định nghĩa trong tệp `docker-compose.yml`. Mỗi dịch vụ tương ứng với một container riêng biệt, kết nối với nhau thông qua mạng nội bộ `hadoop`. Các thành phần chính gồm:

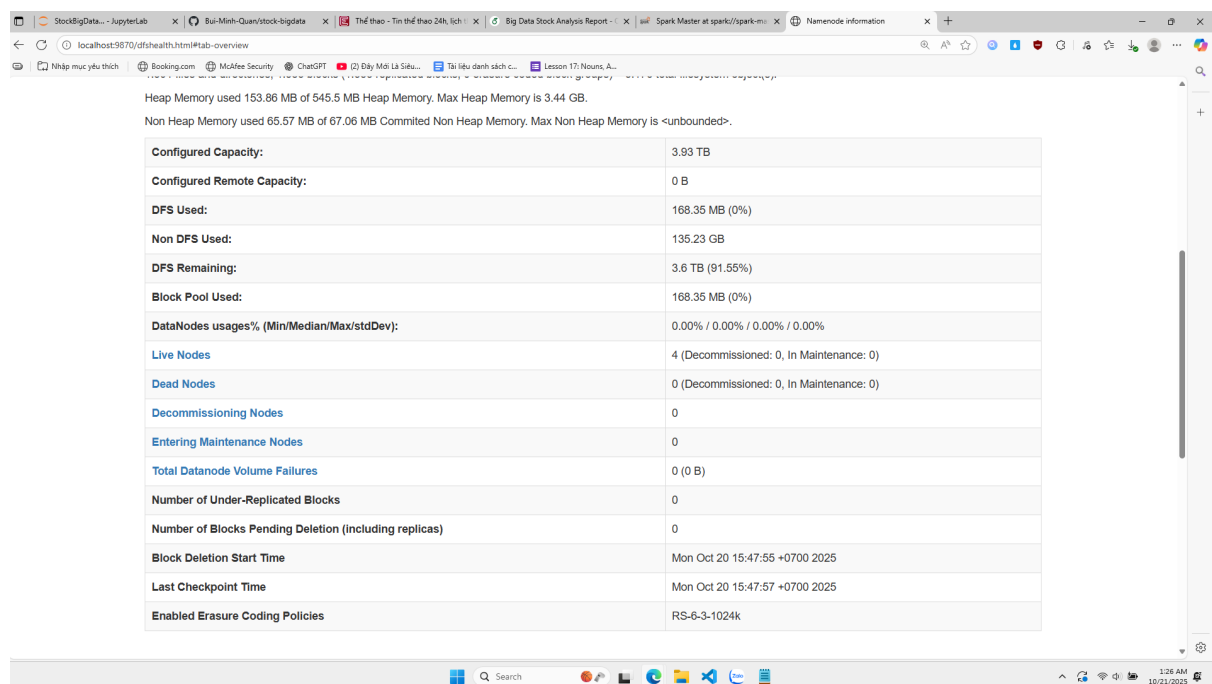
- **NameNode**: quản lý cấu trúc thư mục và thông tin metadata của toàn bộ hệ thống tệp HDFS.
- **DataNode1–4**: lưu trữ thực tế các khối dữ liệu, thực hiện việc đọc/ghi khi Spark hoặc người dùng yêu cầu.
- **Spark Master**: điều phối các tác vụ tính toán, phân công công việc cho các Spark Worker.
- **Spark Worker1–4**: thực thi các tác vụ được Spark Master giao.
- **Jupyter Notebook**: môi trường tương tác để chạy mã PySpark, kết nối tới Spark qua địa chỉ `spark://spark-master:7077`.

Để khởi chạy toàn bộ hệ thống, chỉ cần sử dụng lệnh:

```
docker compose up -d
```

Sau khi khởi động, người dùng có thể truy cập:

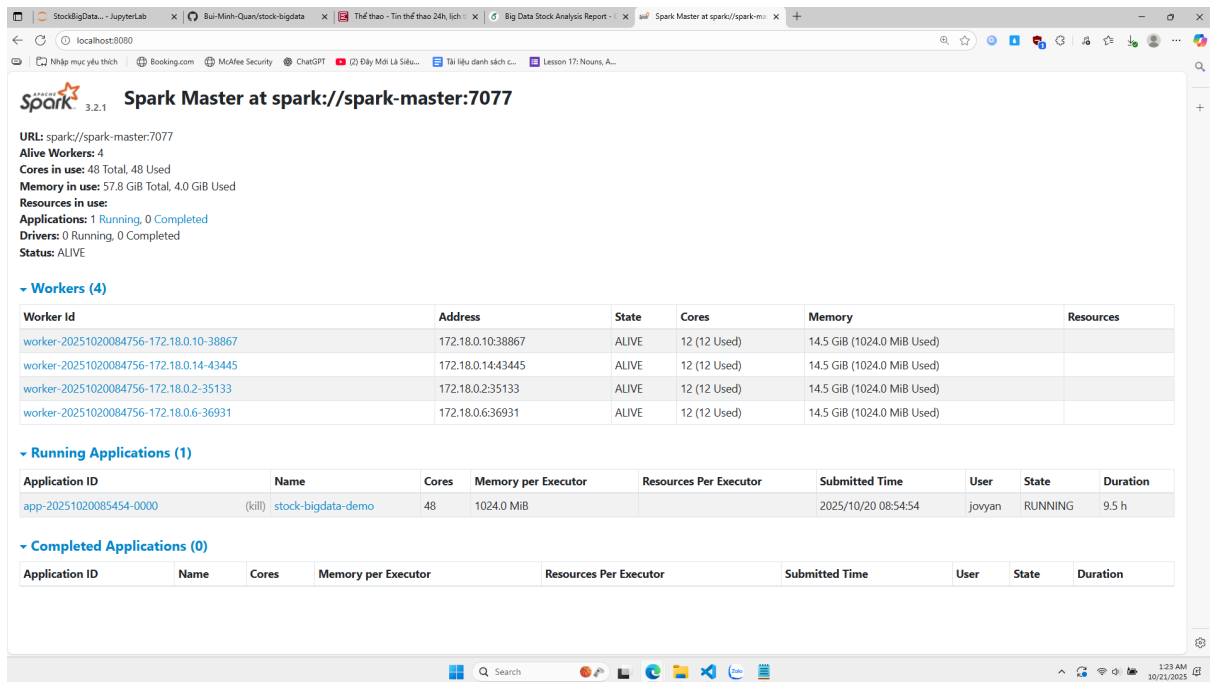
- Hadoop Web UI: `http://localhost:9870`
- Spark Master UI: `http://localhost:8080`
- Jupyter Notebook: `http://localhost:8888`



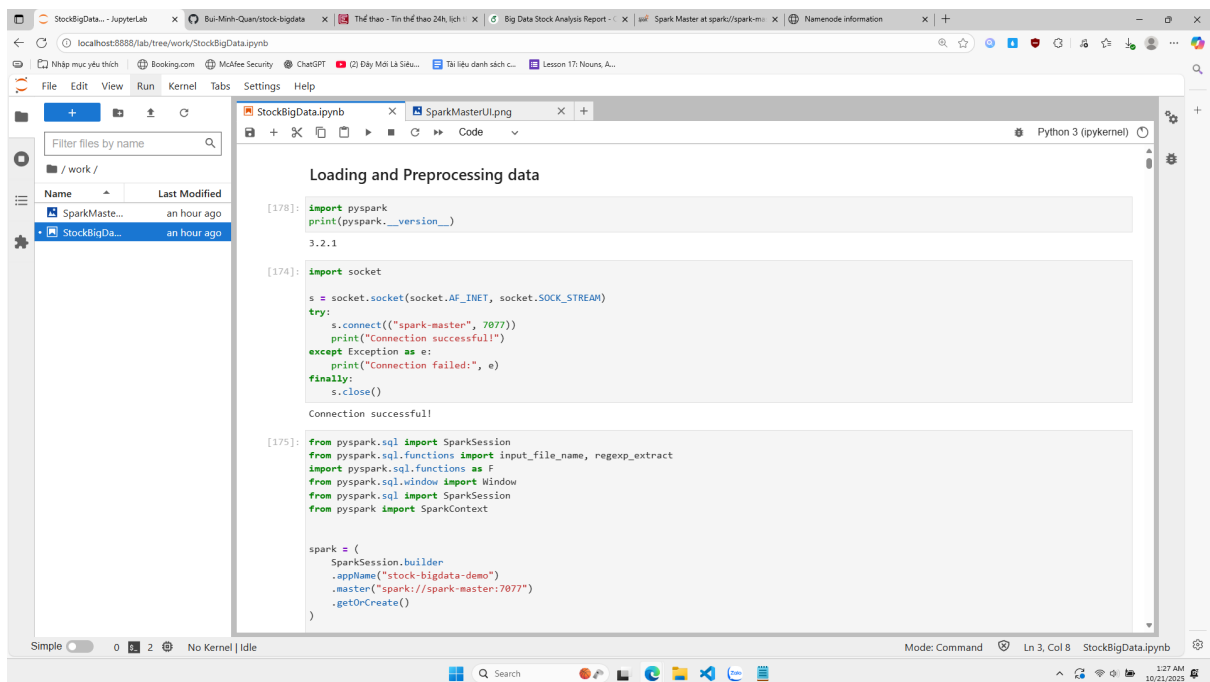
The screenshot displays the Hadoop NameNode Web UI. At the top, it shows memory usage: 'Heap Memory used 153.86 MB of 545.5 MB Heap Memory. Max Heap Memory is 3.44 GB.' and 'Non Heap Memory used 65.57 MB of 67.06 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.' Below this is a table of system metrics.

Configured Capacity:	3.93 TB
Configured Remote Capacity:	0 B
DFS Used:	168.35 MB (0%)
Non DFS Used:	135.23 GB
DFS Remaining:	3.6 TB (91.55%)
Block Pool Used:	168.35 MB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	4 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Mon Oct 20 15:47:55 +0700 2025
Last Checkpoint Time	Mon Oct 20 15:47:57 +0700 2025
Enabled Erasure Coding Policies	RS-6-3-1024k

Hình 2.1: Giao diện Hadoop NameNode Web UI



Hình 2.2: Giao diện Spark Master UI



Hình 2.3: Giao diện Jupyter Notebook

Chương 3

Dữ liệu

Dữ liệu được sử dụng trong đề tài được thu thập thông qua quá trình **cào dữ liệu (web scraping)** từ trang thông tin tài chính uy tín CafeF.vn, một trong những nguồn phổ biến cung cấp dữ liệu chứng khoán tại Việt Nam. Dữ liệu bao gồm các chỉ số giao dịch hàng ngày của các mã cổ phiếu được niêm yết trên các sàn HOSE, HNX và UPCOM, trong giai đoạn từ **năm 2005 đến năm 2025**. Sau khi thu thập, dữ liệu được xử lý sơ bộ và lưu trữ trên hệ thống **Hadoop Distributed File System (HDFS)** tại đường dẫn:

```
hdfs://namenode:9000/user/quan/stocks/*.csv
```

Tổng cộng có **1.584 tệp CSV**, trong đó mỗi tệp tương ứng với một mã cổ phiếu (symbol). Việc lưu trữ tách biệt theo từng mã giúp dễ dàng mở rộng, cập nhật và xử lý song song trong hệ thống **Apache Spark**.

3.1 Cấu trúc dữ liệu

Mỗi tệp CSV có cùng định dạng, bao gồm sáu cột chính:

- **time**: Ngày giao dịch (định dạng YYYY-MM-DD)
- **open**: Giá mở cửa trong ngày
- **high**: Giá cao nhất trong ngày
- **low**: Giá thấp nhất trong ngày
- **close**: Giá đóng cửa
- **volume**: Khối lượng cổ phiếu được giao dịch trong ngày

Ví dụ dữ liệu mẫu (**AAV.csv**):

```
time,open,high,low,close,volume
2018-06-25,10.39,10.39,10.39,10.39,362000
2018-06-26,11.37,11.37,10.52,11.37,800200
2018-06-27,12.49,12.49,10.85,10.85,532700
```

```

2018-06-28,10.85,11.04,10.85,10.91,210300
2018-06-29,10.91,11.04,10.85,10.98,108100
2018-07-02,10.98,11.04,10.58,10.98,129000
2018-07-03,10.91,11.04,10.85,10.91,98800
2018-07-04,10.91,10.98,10.85,10.85,80500
2018-07-05,10.85,10.98,10.78,10.85,144700
2018-07-06,10.85,11.04,10.78,10.98,232100
2018-07-09,10.98,11.11,10.85,10.98,248300
...

```

3.2 Đọc dữ liệu cho việc xử lý

Dữ liệu được nạp trực tiếp từ HDFS bằng Apache Spark thông qua đoạn mã sau:

```

data_path = "hdfs://namenode:9000/user/quan/stocks/*.csv"

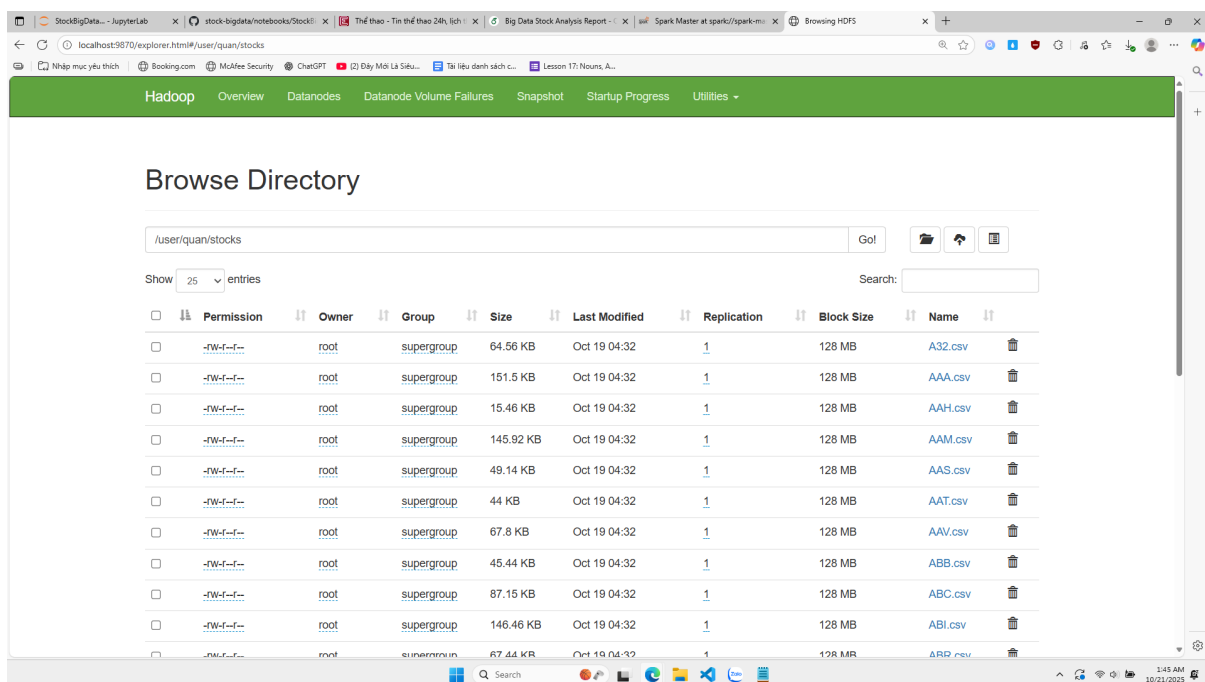
# Load all the data into a single data frame
df = spark.read.option("header", True).option("inferSchema", True).csv(data_path)

# parse time as date
df = df.withColumn("time", F.to_date("time", "yyyy-MM-dd"))

# Add a "symbol" column extracted from the file name
df = df.withColumn("symbol", regexp_extract(input_file_name(), r"([~/]+)\.csv$", 1))

```

Đoạn mã này cho phép hợp nhất toàn bộ 1.584 tệp CSV thành một **DataFrame** duy nhất, tạo điều kiện thuận lợi cho việc xử lý, phân tích và trực quan hóa dữ liệu ở về sau.



Hình 3.1: Danh sách dữ liệu tại giao diện HDFS


```
stock-bigdata / notebooks / StockBigData.ipynb
3789 lines (3789 loc) - 1.14 MB

In [5]: data_path = "hdfs://namenode:9000/user/quan/stocks/*.csv"

# Load all the data into a single data frame
df = spark.read.option("header", True).option("inferSchema", True).csv(data_path)

# parse time as date
df = df.withColumn("time", F.to_date("time", "yyyy-MM-dd"))

# add a "symbol" column extracted from the file name
df = df.withColumn("symbol", regexp_extract(input_file_name(), r"([^\.]*)\.csv$", 1))

df.printSchema()
df.show()

root
 |-- time: date (nullable = true)
 |-- open: double (nullable = true)
 |-- high: double (nullable = true)
 |-- low: double (nullable = true)
 |-- close: double (nullable = true)
 |-- volume: integer (nullable = true)
 |-- symbol: string (nullable = false)

+-----+-----+
| time|open|high| low|close|volume|symbol|
+-----+-----+
|2005-01-04|5.35|5.35|5.35| 5.35| 2060| GPD|
|2005-01-05|5.35|5.35|5.35| 5.35| 2090| GPD|
|2005-01-06|5.35|5.35|5.35| 5.35| 4110| GPD|
|2005-01-07| 5.3| 5.3| 5.3| 5.3| 2210| GPD|
|2005-01-10|5.25|5.25|5.25| 5.25| 2050| GPD|
|2005-01-11|5.25|5.25|5.25| 5.25| 360| GPD|
|2005-01-12|5.25|5.25|5.25| 5.25| 5300| GPD|
|2005-01-13| 5.2| 5.2| 5.2| 5.2| 1310| GPD|
|2005-01-14| 5.3| 5.3| 5.3| 5.3| 2390| GPD|
|2005-01-17| 5.3| 5.3| 5.3| 5.3| 410| GPD|
|2005-01-18| 5.3| 5.3| 5.3| 5.3| 900| GPD|
|2005-01-19| 5.3| 5.3| 5.3| 5.3| 5400| GPD|
|2005-01-20|5.25|5.25|5.25| 5.25| 2100| GPD|
|2005-01-21|5.35|5.35|5.35| 5.35| 5270| GPD|
|2005-01-24| 5.3| 5.3| 5.3| 5.3| 660| GPD|
|2005-01-25| 5.3| 5.3| 5.3| 5.3| 1100| GPD|
|2005-01-26| 5.3| 5.3| 5.3| 5.3| 600| GPD|
|2005-01-27|5.25|5.25|5.25| 5.25| 1470| GPD|
|2005-01-28| 5.3| 5.3| 5.3| 5.3| 10| GPD|
|2005-01-31| 5.3| 5.3| 5.3| 5.3| 100| GPD|
+-----+-----+
only showing top 20 rows
```

Hình 3.2: Dữ liệu được đọc trong Jupyter Notebook

Chương 4

Xử lý và Phân tích Dữ liệu

Mục tiêu: tiến hành xử lý, phân tích dữ liệu cổ phiếu được lưu trữ trên HDFS nhằm rút ra các đặc trưng tổng quan của thị trường và quan sát hành vi giá của một số mã cổ phiếu tiêu biểu.

4.1 Tiền xử lý dữ liệu

Dữ liệu được đọc từ HDFS thông qua đường dẫn:

```
data_path = "hdfs://namenode:9000/user/quan/stocks/*.csv"
```

Trong đó, mỗi tệp tương ứng với một mã cổ phiếu. Để tiện cho việc phân tích, ta tiến hành:

- **Thêm cột `symbol`:** trích xuất tên mã cổ phiếu từ tên tệp bằng biểu thức chính quy.
- **Tạo cột `prev_close` và `return`:** sử dụng `Window function` trong PySpark để lấy giá đóng cửa của phiên trước và tính tỉ suất lợi nhuận hàng ngày:

$$return = \frac{close - prev_close}{prev_close}$$

```
[7]: w = Window.partitionBy("symbol").orderBy("time")
df = df.withColumn("prev_close", F.lag("close").over(w))
df = df.withColumn("return", (F.col("close") - F.col("prev_close")) / F.col("prev_close"))
df.show()
```

time	open	high	low	close	volume	symbol	prev_close	return
2021-03-24	9.51	9.51	9.36	9.51	1568200	AAT	null	null
2021-03-25	10.07	10.07	8.87	9.13	981400	AAT	9.51	-0.03995793901156667
2021-03-26	8.61	8.98	8.5	8.5	231300	AAT	9.13	-0.06900328587075583
2021-03-29	8.01	8.5	7.94	8.24	1741400	AAT	8.5	-0.03058823529411762
2021-03-30	8.24	8.8	8.2	8.8	761500	AAT	8.24	0.06796116504854374
2021-03-31	9.4	9.4	9.36	9.4	320300	AAT	8.8	0.06818181818181814
2021-04-01	9.88	10.03	9.43	10.03	947500	AAT	9.4	0.06702127659574457
2021-04-02	10.48	10.56	9.58	9.73	498100	AAT	10.03	-0.02991026919242...
2021-04-05	9.66	9.81	9.43	9.43	523800	AAT	9.73	-0.03083247687564...
2021-04-06	9.43	10.07	9.43	10.07	425300	AAT	9.43	0.0678685047720043
2021-04-07	10.37	10.63	9.96	10.11	798300	AAT	10.07	0.003972194637537154
2021-04-08	10.11	10.63	9.81	9.88	792900	AAT	10.11	-0.02274975272007...
2021-04-09	9.85	10.56	9.85	10.56	1336500	AAT	9.88	0.06882591093117406
2021-04-12	10.74	11.27	10.74	11.27	942700	AAT	10.56	0.0672348484848484
2021-04-13	11.87	11.9	11.31	11.72	1149700	AAT	11.27	0.03992901508429468
2021-04-14	11.61	11.87	11.61	11.64	997300	AAT	11.72	-0.00682593856655...
2021-04-15	11.68	12.13	11.53	11.9	770300	AAT	11.64	0.02233676975945015
2021-04-16	12.35	12.35	11.9	12.35	370600	AAT	11.9	0.03781512605042011
2021-04-19	12.65	12.65	11.98	12.13	509900	AAT	12.35	-0.01781376518218...
2021-04-20	11.98	12.95	11.94	12.58	771600	AAT	12.13	0.037098103874690785

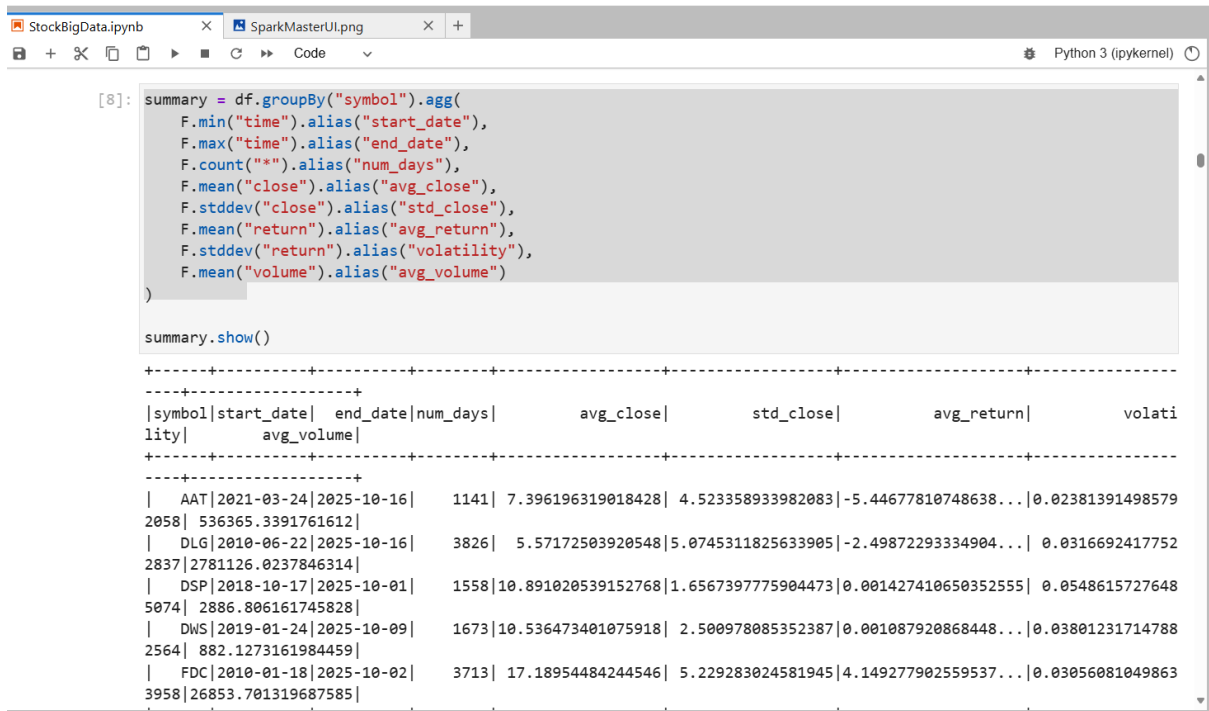
only showing top 20 rows

Hình 4.1: Dữ liệu sau khi được tiền xử lý

4.2 Phân tích tổng quan

Từ tập dữ liệu đã xử lý, ta thực hiện tính toán thống kê theo từng mã cổ phiếu:

- Thời gian bắt đầu và kết thúc dữ liệu
- Số ngày có dữ liệu (`num_days`)
- Giá đóng cửa trung bình và độ lệch chuẩn
- Lợi nhuận trung bình và độ biến động (`volatility`)
- Khối lượng giao dịch trung bình



Hình 4.2: Dữ liệu tổng hợp theo từng mã cổ phiếu

Từ đó, có thể xếp hạng các cổ phiếu (top 20) theo các tiêu chí:

- Độ biến động cao nhất
- Khối lượng giao dịch trung bình lớn nhất
- Lợi nhuận trung bình cao nhất
- Giá đóng cửa trung bình cao nhất

The screenshot shows a Jupyter Notebook interface with a file named 'StockBigData.ipynb'. The code cell [193] contains a print statement and a Spark SQL query to select the top 20 stocks by volatility. The output is a table with two columns: 'symbol' and 'volatility'.

```
[193]: print("20 stocks with highest volatility")
summary.select('symbol', 'volatility').orderBy(F.desc("volatility")).show()
```

20 stocks with highest volatility

symbol	volatility
DNN	1.7806845259321122
PTG	0.6765261938940811
PTX	0.33929109724093914
SGS	0.13035621084381577
NSS	0.11585282923734551
PHS	0.11468140721167194
L40	0.10939442386131405
VDG	0.09735043704561938
PSG	0.09636373679012492
NEM	0.09574464029952594
TNV	0.0948108088590712
XDH	0.08936299992786234
ING	0.08909086579890628
XMD	0.08806736608721671
G20	0.08341649962437453
HLA	0.08321998328109574
HAF	0.0827566824632022
MGR	0.08204509511192595
V11	0.0818635731864896
NCS	0.08183720442492819

Hình 4.3: Top 20 Cổ phiếu có độ biến động lớn nhất

The screenshot shows a Jupyter Notebook interface with a file named 'StockBigData.ipynb'. The code cell [194] contains a print statement and a Spark SQL query to select the top 20 stocks by average volume. The output is a table with two columns: 'symbol' and 'avg_volume'.

```
[194]: print("Top 20 most traded stocks by average volume")
summary.select("symbol", "avg_volume").orderBy(F.desc("avg_volume")).show()
```

Top 20 most traded stocks by average volume

symbol	avg_volume
VPB	1.2089127298873102E7
SHB	1.086813486786495E7
ROS	1.0106185350404313E7
HPG	9213093.66286098
FLC	8864954.158291457
TCB	8539684.48264642
NVL	8395333.136032756
POW	8176228.281118143
MBB	7366654.166140603
MSB	6989656.726666667
STB	6948509.0832466185
BSR	6757228.886842106
GEX	6747180.384028892
SSI	6703932.978864219
VND	6606459.449780532
VIX	6034048.278526048
TPB	6001234.099305926
HQC	5883687.147514698
HAG	5856310.554073368
LPB	5253243.3555333

Hình 4.4: Top 20 cổ phiếu có khối lượng giao dịch trung bình lớn nhất

```
StockBigData.ipynb SparkMasterUI.png Python 3 (ipykernel)

[195]: print("Top 20 stocks with highest average return")

summary.select('symbol', 'avg_return').orderBy(F.desc("avg_return")).show()

Top 20 stocks with highest average return
+-----+-----+
|symbol|      avg_return|
+-----+-----+
| DNN | 0.10187643867728115|
| PTG | 0.022516605067223746|
| PTX | 0.020804770588832684|
| D17 | 0.006908666929962545|
| ING | 0.006174833290713505|
| NEM | 0.006053119939721107|
| XDH | 0.004845783144520415|
| NSS | 0.004549897387364335|
| TAB | 0.004495892225623251|
| HAF | 0.003914422702014924|
| L40 | 0.003431057163298161|
| XMD | 0.003412427389281...|
| PAP | 0.003319224966372...|
| VMK | 0.003301561615746921|
| IDP | 0.003130159062752...|
| MGG | 0.003074979532918...|
| VDG | 0.003039389896637828|
| CMM | 0.003024430755495...|
| GEE | 0.002973404262169...|
| SGS | 0.002961615490725003|
+-----+-----+
only showing top 20 rows
```

Hình 4.5: Top 20 cổ phiếu có tỷ lệ lợi nhuận trả về trung bình cao nhất

```
StockBigData.ipynb SparkMasterUI.png Python 3 (ipykernel)

[196]: print("Top 20 stocks with highest average close price")

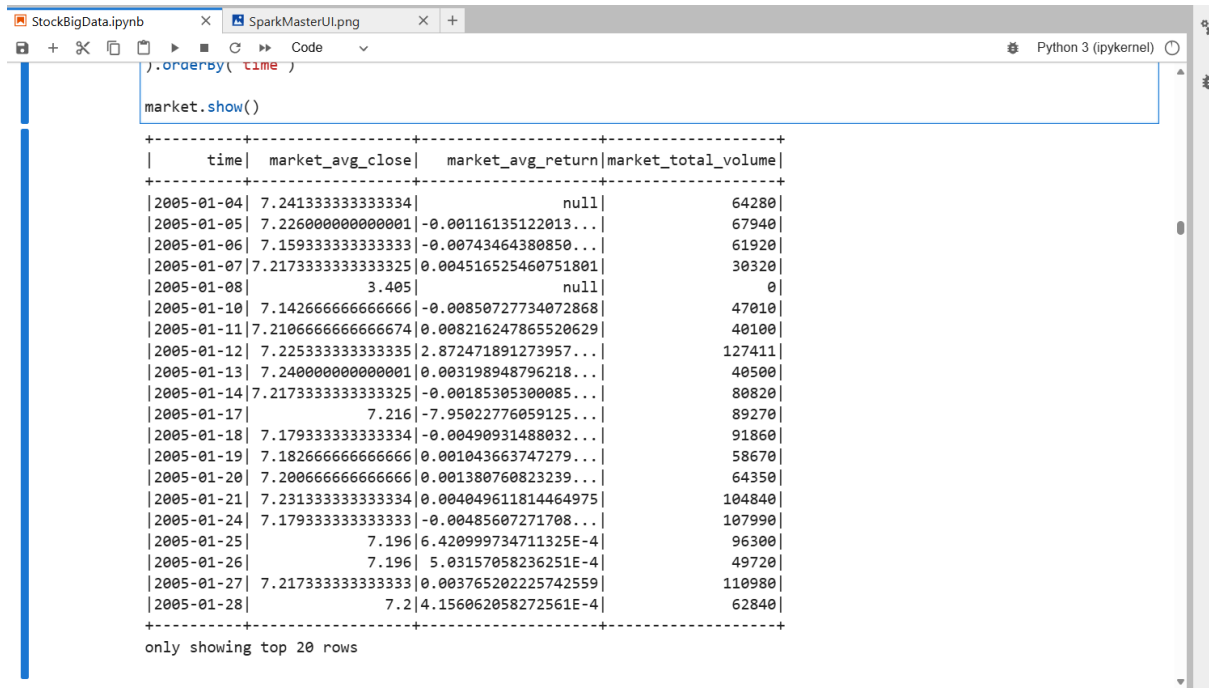
summary.select('symbol', 'avg_close').orderBy(F.desc("avg_close")).show()

Top 20 stocks with highest average close price
+-----+-----+
|symbol|      avg_close|
+-----+-----+
| VNZ | 570.3321585903076|
| IDP | 170.73524401913895|
| GAB | 162.26692185007704|
| CMF | 155.9605956258722|
| HLB | 146.42398588235326|
| VCF | 117.37923998910384|
| NTC | 116.43990471869311|
| VJC | 114.84265740740706|
| VEF | 93.76860228198858|
| WCS | 92.71073196699435|
| PAT | 79.50996398559437|
| FOC | 75.47418713450284|
| SSH | 75.41078846153847|
| SAB | 73.45524841915108|
| CAB | 71.78435185185117|
| THD | 70.76186936936944|
| SQC | 69.8372323617538|
| SCG | 67.11890374331553|
| SGN | 67.10511201629356|
| SLS | 64.92438828967676|
+-----+-----+
```

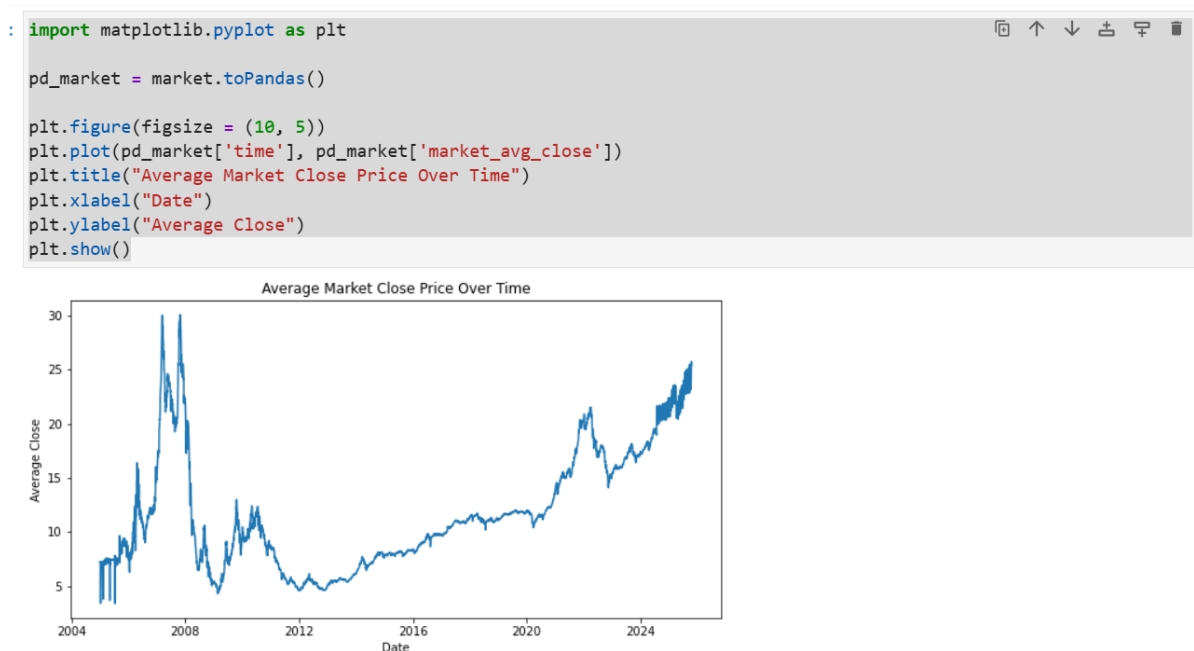
Hình 4.6: Top 20 cổ phiếu có giá đóng cửa cao nhất

4.3 Phân tích chỉ số thị trường tổng thể

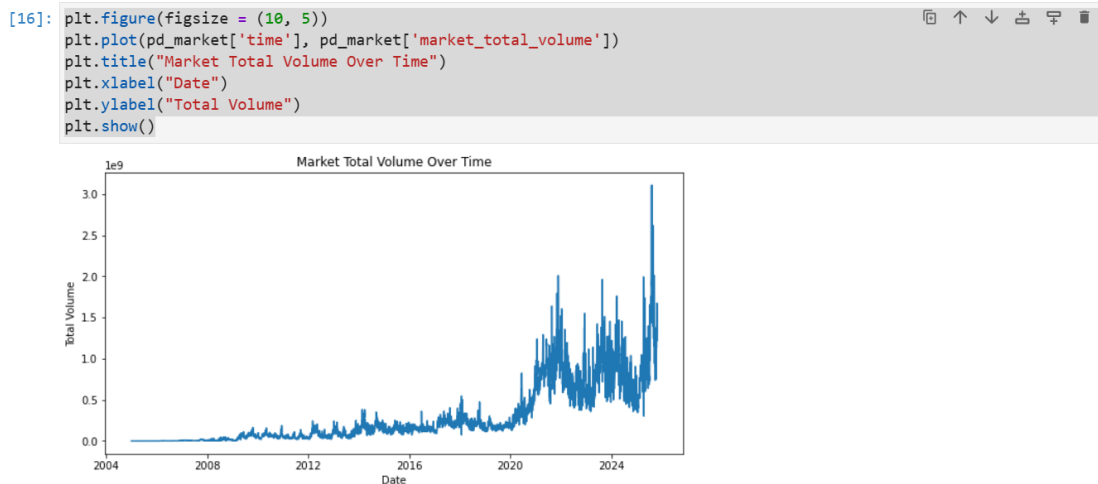
Ta tổng hợp dữ liệu toàn thị trường theo từng ngày:

$$market = \text{mean}(\text{close}), \text{mean}(\text{return}), \text{sum}(\text{volume})$$


Hình 4.7: Dữ liệu của thị trường tổng hợp chung theo từng ngày



Hình 4.8: Trung bình giá cổ phiếu trên thị trường theo từng ngày

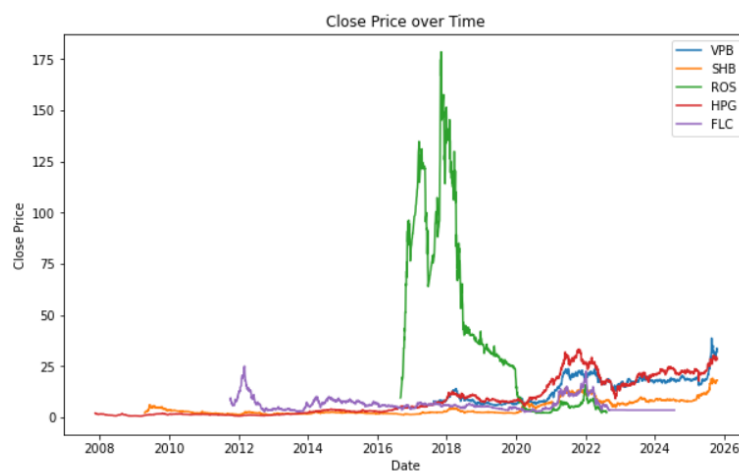


Hình 4.9: Tổng khối lượng giao dịch cổ phiếu trên thị trường theo ngày

4.4 Phân tích chi tiết một số mã cổ phiếu tiêu biểu

Ta chọn ra 5 mã có khối lượng giao dịch lớn nhất gồm: **VPB**, **SHB**, **ROS**, **HPG**, **FLC** để quan sát chi tiết. Các chỉ số được tính gồm:

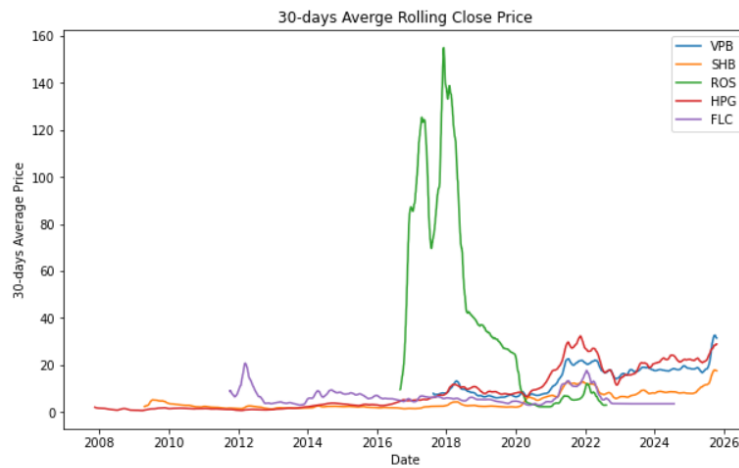
- Trung bình trượt 7 ngày và 30 ngày cho giá đóng cửa
- Phân bố cho lợi nhuận trả về



Hình 4.10: Trung bình giá trị 5 mã cổ phiếu được chọn theo ngày

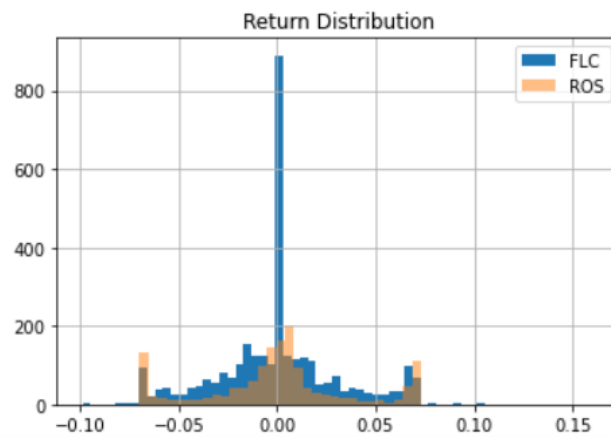


Hình 4.11: Trung bình trượt 7 ngày của 5 mã cổ phiếu

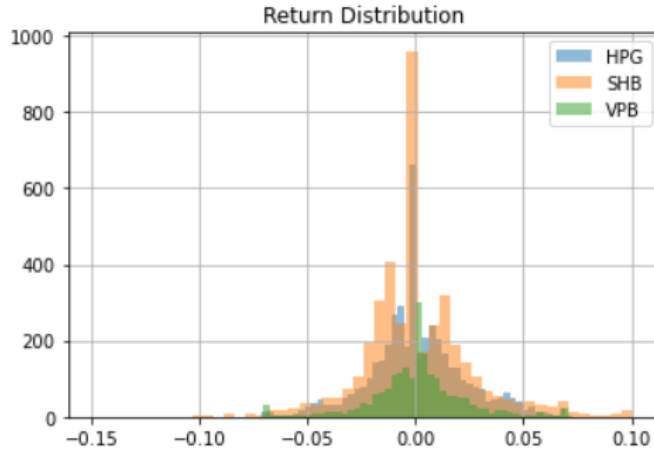


Hình 4.12: Trung bình trượt 30 ngày của 5 mã cổ phiếu

[62]: <matplotlib.legend.Legend at 0x7fef6b5cf280>



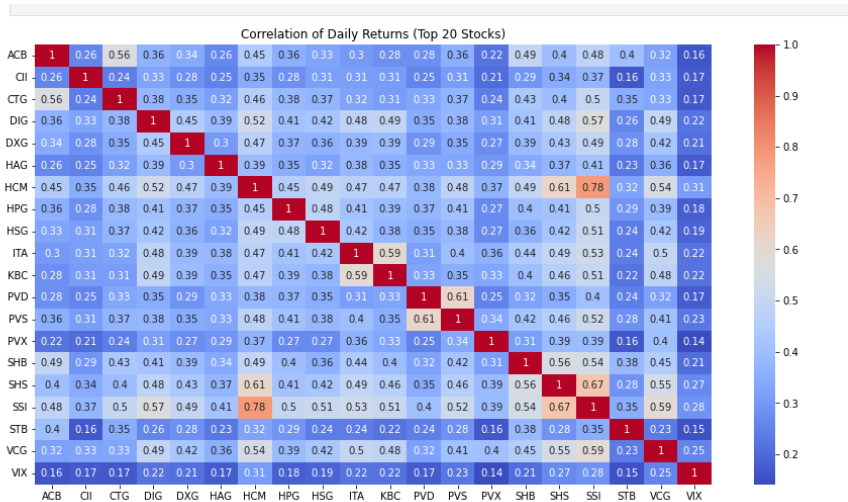
Hình 4.13: Phân bố giá trị lợi nhuận trả về của cổ phiếu FLC và ROS



Hình 4.14: Phân bố giá trị lợi nhuận trả về của cổ phiếu HPG, SHB và VPB

4.5 Phân tích tương quan giữa các cổ phiếu

Ta chọn ra 20 mã cổ phiếu có khối lượng giao dịch lớn nhất và dữ liệu ổn định (trước 2010 đến sau 2020). Lợi nhuận ngày (**return**) được pivot thành ma trận thời gian–mã cổ phiếu, sau đó tính **ma trận tương quan Pearson**. Kết quả được biểu diễn dưới dạng *heatmap* như sau:



Hình 4.15: Hệ số tương quan giữa giá trị trả về của 20 mã cổ phiếu

Chương 5

Xây dựng mô hình học máy

Thóm tắt hành xây dựng hai mô hình học máy nhằm dự đoán thông tin về cổ phiếu, cụ thể như sau:

- **Mô hình 1:** Dự đoán *giá đóng cửa (close)* của cổ phiếu trong ngày giao dịch tiếp theo.
- **Mô hình 2:** Dự đoán *xu hướng biến động giá* của cổ phiếu trong ngày giao dịch tiếp theo (tăng hoặc không tăng, được mã hoá lần lượt là 1 và 0).

Cả hai mô hình đều sử dụng dữ liệu đầu vào là thông tin của 7 ngày giao dịch gần nhất, bao gồm các đặc trưng: `open`, `high`, `low`, `close`, `volume`. Để đơn giản hoá và phục vụ mục đích minh hoạ, mô hình được huấn luyện và đánh giá trên hai mã cổ phiếu phổ biến tại Việt Nam là **HCM** và **SSI**.

5.1 Chuẩn bị dữ liệu huấn luyện

Trước hết, dữ liệu chuỗi thời gian được chuyển đổi sang dạng *supervised learning* bằng cách chia thành các mẫu huấn luyện, trong đó mỗi mẫu gồm dữ liệu của 7 ngày trước đó để dự đoán giá hoặc xu hướng của ngày tiếp theo. Quá trình này được thực hiện bằng hàm `prepare_supervised_data`, được mô tả trong mã nguồn dưới đây:

```
def prepare_supervised_data(df, feature_cols, label_col="close", window=7):
    assembler = VectorAssembler(inputCols=feature_cols, outputCol='feature_raw')
    df = assembler.transform(df)
    scaler = StandardScaler(inputCol='feature_raw', outputCol='features_scaled',
                             withMean=True, withStd=True)
    df = scaler.fit(df).transform(df)
    df = df.select('time', 'features_scaled', label_col).toPandas()

    features = np.stack(df['features_scaled'].to_numpy())
    targets = df[label_col].to_numpy()

    X, y = [], []
    for i in range(window, len(features)):
        X.append(features[i - window:i].flatten())
        y.append(df[label_col].iloc[i])
    return np.array(X), np.array(y)
```

5.2 Mô hình 1: Dự đoán giá đóng cửa

Đối với bài toán hồi quy, ta sử dụng thuật toán **Linear Regression** trong thư viện `scikit-learn`. Mã huấn luyện được thực hiện như sau:

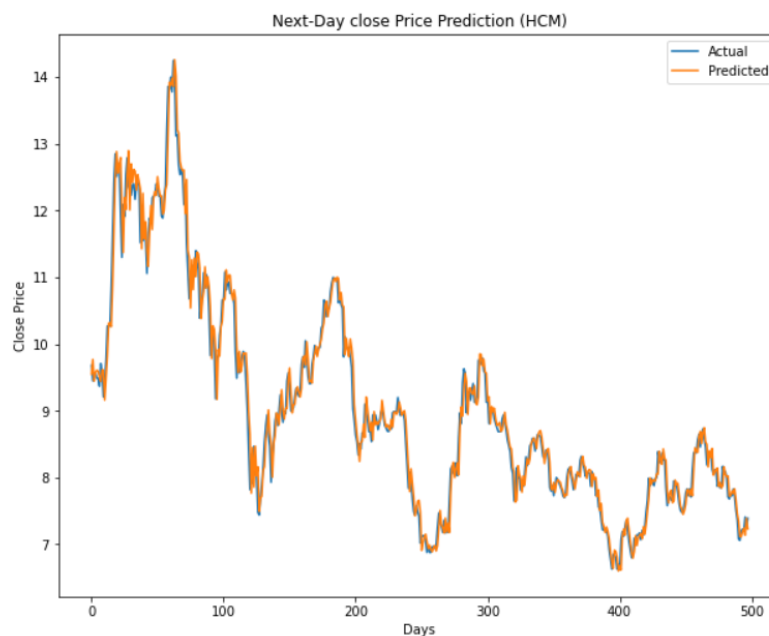
```
def train_regression_model(X, y, test_size=0.2):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    metrics = {
        "rmse": mean_squared_error(y_test, y_pred, squared=False),
        "r2": r2_score(y_test, y_pred)
    }
    return model, y_pred, y_test, metrics
```

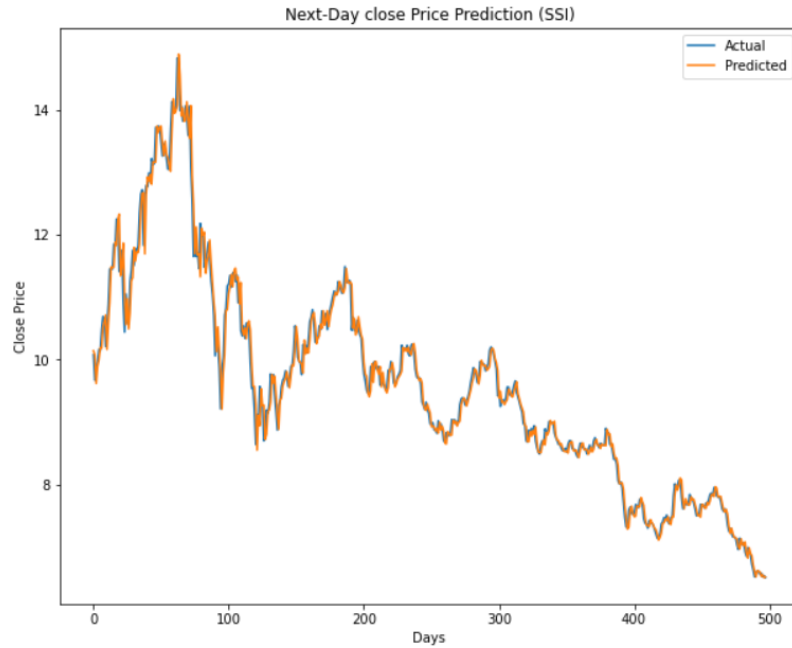
Kết quả huấn luyện:

- Mã **HCM**: RMSE = 0.2621, R^2 = 0.9737
- Mã **SSI**: RMSE = 0.2276, R^2 = 0.9825

Các chỉ số trên cho thấy mô hình hồi quy tuyến tính có khả năng mô phỏng tốt biến động giá đóng cửa của hai mã cổ phiếu trong tập dữ liệu thử nghiệm.



Hình 5.1: Kết quả dự đoán cho mã HCM của Mô hình 1



Hình 5.2: Kết quả dự đoán cho mã SSI của mô hình 1

5.3 Mô hình 2: Dự đoán xu hướng tăng/giảm

Đối với bài toán phân loại, ta sử dụng thuật toán **Logistic Regression**. Dữ liệu được gán nhãn 1 nếu lợi nhuận (**return**) dương và 0 nếu ngược lại.

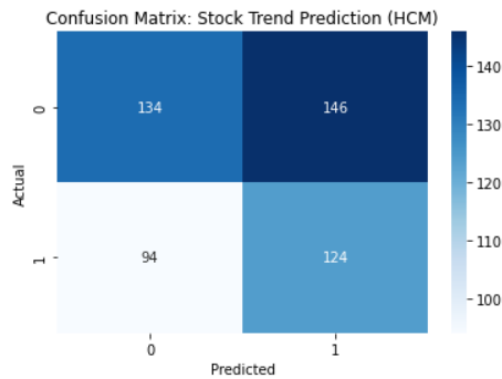
```
def train_logistics_regression_model(X, y, test_size=0.2):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    metrics = {
        "accuracy": accuracy_score(y_test, y_pred),
        "report": classification_report(y_test, y_pred)
    }
    return model, y_pred, y_test, metrics
```

Kết quả huấn luyện:

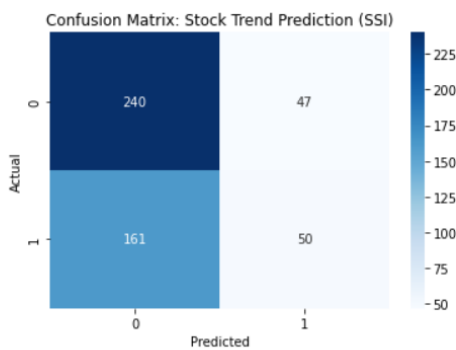
- Mã **HCM**: Độ chính xác (*Accuracy*) = 0.518
- Mã **SSI**: Độ chính xác (*Accuracy*) = 0.582

Logistics Regression Model Accuracy for HCM stock Symbol: 0.5180722891566265				
Logistics Regression Model Report for HCM stock Symbol:				
	precision	recall	f1-score	support
0	0.59	0.48	0.53	280
1	0.46	0.57	0.51	218
accuracy			0.52	498
macro avg	0.52	0.52	0.52	498
weighted avg	0.53	0.52	0.52	498



Hình 5.3: Kết quả dự đoán cho mã HCM của mô hình 2

Logistics Regression Model Accuracy for SSI stock Symbol: 0.5823293172690763				
Logistics Regression Model Report for SSI stock Symbol:				
	precision	recall	f1-score	support
0	0.60	0.84	0.70	287
1	0.52	0.24	0.32	211
accuracy			0.58	498
macro avg	0.56	0.54	0.51	498
weighted avg	0.56	0.58	0.54	498



Hình 5.4: Kết quả dự đoán cho mã SSI của mô hình 2

Kết quả trên cho thấy mô hình Logistic Regression chỉ đạt độ chính xác trung bình, do tính chất nhiễu và khó dự đoán của dữ liệu cổ phiếu ngắn hạn. Tuy nhiên, mô hình vẫn thể hiện được khả năng học được xu hướng tổng quát, phù hợp với mục tiêu minh họa của đề tài.

Chương 6

Kết luận và Hướng phát triển

6.1 Kết luận

Trong báo cáo này, ta đã xây dựng một hệ thống phân tích dữ liệu chứng khoán sử dụng nền tảng **Big Data** dựa trên **Hadoop** và **Spark**. Hệ thống cho phép lưu trữ, xử lý và phân tích khối lượng lớn dữ liệu cổ phiếu được thu thập trong giai đoạn từ năm 2005 đến năm 2025, bao gồm hơn 1500 mã cổ phiếu khác nhau tại thị trường Việt Nam.

Các bước chính được thực hiện bao gồm:

- Xây dựng hạ tầng lưu trữ dữ liệu trên hệ thống phân tán **HDFS**.
- Thực hiện xử lý, thống kê, và phân tích dữ liệu bằng **PySpark**.
- Trực quan hóa xu hướng thị trường và các chỉ số đặc trưng của cổ phiếu theo thời gian.
- Xây dựng và huấn luyện hai mô hình học máy: mô hình hồi quy dự đoán giá đóng cửa, và mô hình phân loại dự đoán xu hướng tăng giảm của cổ phiếu trong ngày tiếp theo.

Kết quả cho thấy các mô hình học máy đã đạt được độ chính xác và mức sai số tương đối tốt, thể hiện khả năng nắm bắt xu hướng thị trường dựa trên dữ liệu lịch sử. Hệ thống hoạt động ổn định, có khả năng mở rộng, và chứng minh tính khả thi của việc áp dụng công nghệ dữ liệu lớn vào phân tích tài chính.

6.2 Hướng phát triển

Mặc dù hệ thống đã hoàn thiện được những chức năng cơ bản và cho kết quả khả quan, vẫn còn nhiều hướng mở rộng tiềm năng trong tương lai:

- **Kết hợp dữ liệu phi cấu trúc:** Bổ sung thêm nguồn dữ liệu văn bản như tin tức tài chính, báo cáo kinh tế, hoặc các bài viết trên phương tiện truyền thông, giúp mô hình hiểu rõ hơn các yếu tố ảnh hưởng đến biến động giá.
- **Ứng dụng mô hình học sâu (Deep Learning):** Nâng cấp mô hình dự đoán bằng các kiến trúc hiện đại như *LSTM*, *GRU*, hoặc *Transformer* để khai thác đặc trưng chuỗi thời gian hiệu quả hơn.

- **Tích hợp mô hình ngôn ngữ lớn (LLM):** Sử dụng các mô hình ngôn ngữ tiên tiến để xử lý và trích xuất thông tin từ văn bản, hỗ trợ ra quyết định dựa trên cả dữ liệu định lượng và định tính.
- **Phát triển hệ thống dự báo thời gian thực:** Tối ưu quy trình thu thập và xử lý dữ liệu để hệ thống có thể cập nhật và dự đoán liên tục, phục vụ cho các bài toán phân tích tài chính trực tuyến.

Nhìn chung, kết quả nghiên cứu cho thấy việc áp dụng công nghệ Big Data và Machine Learning vào lĩnh vực tài chính là hoàn toàn khả thi và có tiềm năng lớn. Với việc mở rộng phạm vi dữ liệu và nâng cao năng lực mô hình, hệ thống có thể trở thành nền tảng hữu ích cho các nhà đầu tư và tổ chức tài chính trong việc phân tích, dự báo và ra quyết định trên thị trường chứng khoán Việt Nam.