TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT

VIỆN KỸ THUẬT CÔNG NGHỆ

BÁO CÁO KẾT THÚC HỌC PHẦN
THỰC HÀNH CHUYÊN ĐỀ DỮ LIỆU LỚN

BÁO CÁO TỔNG KẾT THỰC HÀNH

Sinh viên thực hiện: Nguyễn Hữu Nghĩa

MSSV: 2124802050013

Lớp: D21TTNT01

GVHD: ThS. Nguyễn Thế Bảo

Học kỳ I năm học 2024 – 2025

10/2024

BÌNH DƯƠNG - 2024

TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT
VIỆN KỸ THUẬT CÔNG NGHỆ

BÁO CÁO KẾT THÚC HỌC PHẦN
THỰC HÀNH CHUYÊN ĐỀ DỮ LIỆU LỚN

BÁO CÁO TỔNG KẾT THỰC HÀNH

Sinh viên thực hiện: Nguyễn Hữu Nghĩa

MSSV: 2124802050013

Lớp: D21TTNT01

GVHD: ThS. Nguyễn Thế Bảo

Học kỳ I năm học 2024 – 2025

10/2024

BÌNH DƯƠNG - 2024

MỤC LỤC

DANH MỤC HÌNH

CHƯƠNG 1. CÀI ĐẶT HADOOP - SPARK TRÊN WINDOWS

1.1. Chuẩn bị môi trường

Để làm việc với PySpark trên Windows cần chuẩn bị các môi trường sau:

- Hệ điều hành Windows: Đảm bảo bạn đang sử dụng một phiên bản Windows hỗ trợ cài đặt và chạy PySpark.

- Cài đặt Java 8s: Tải Java JDK 8 từ trang chính thức của Oracle hoặc từ một nguồn uy tín khác.

- Cài đặt Python 3: Tải Python 3 từ trang chính thức của Python.

1.2. Cài đặt Java

- Truy cập trang web chính thức của Oracle hoặc OpenJDK để tải JDK. Chạy `jre-8u202-windows-i586.exe` để tiến hành cài đặt. Sau khi cài đặt xong chạy lệnh để kiểm tra: `java -version.`

- Hoặc cài java qua Command Prompt: Nhấn `Windows + R`, gõ cmd, rồi nhấn `Enter.`

```
cd đường_dẫn_đến_file_cài_đặt
jdk-<version>-windows-x64.exe /s
```

Image 1 Cài đặt java

1.3. Cài đặt python

Truy cập đường dẫn https://www.python.org/downloads/release/python-383/ và tải tập tin **Windows x86 executable installer**

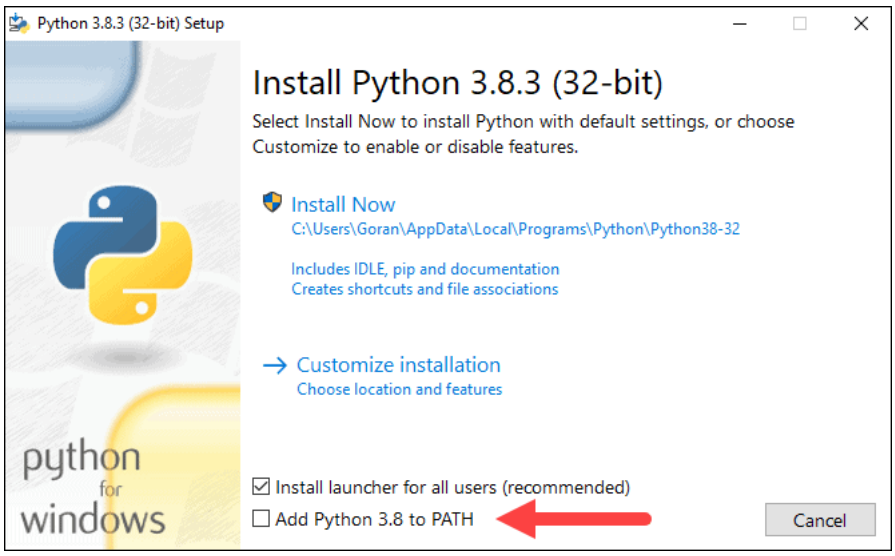


Image 2 Install Python

Hoặc mở cmd nhập lệnh:

```
curl -O
https://www.python.org/ftp/python/3.x.x/python-
3.x.x-amd64.exe
```

1.4. Cài đặt Apache Spark

- Truy cập đường dẫn https://spark.apache.org/downloads.html
- Chọn phiên bản Spark: 3.5.2 (có thể chọn phiên bản khác)
- Chọn gói Pre-built for Apache Hadoop 2.7 để nhận bản build sẵn
- Nhấp vào liên kết spark-3.5.2-bin-hadoop2.7.tgz để tải về

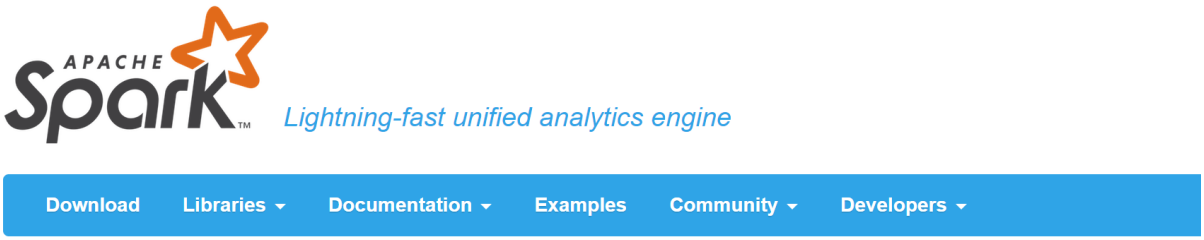


Image 3 Download Apache Spark

- Giải nén và copy vào folder **C:\spark\spark-3.5.2-bin-hadoop2.7**

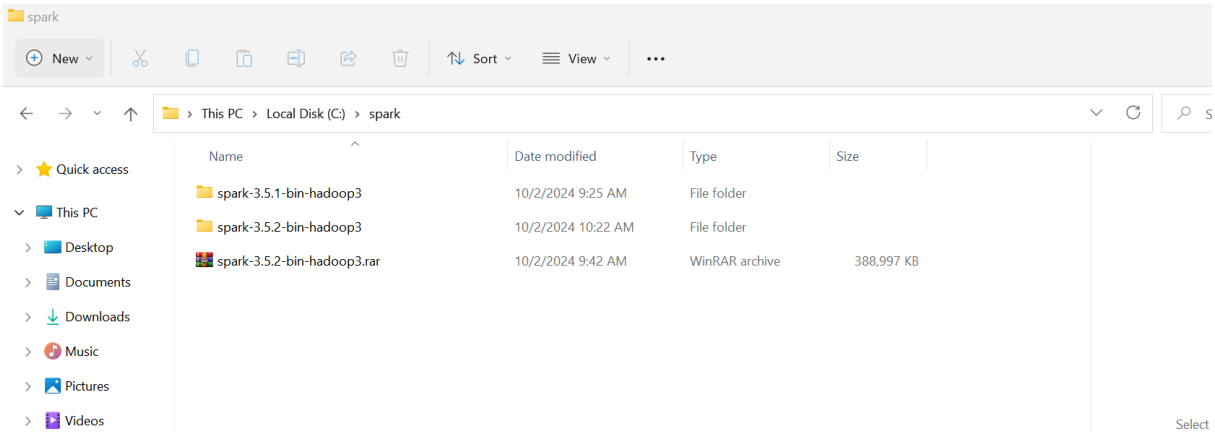


Image 4: Giải nén apache spark

1.5. Cài đặt Hadoop

- Tạo folder chứa hadoop: **C:/hadoop/**
- Truy cập đường dẫn và tải về https://github.com/cdarlint/winutils/blob/master/
- Giải nén tập tin **winutils-master.zip**
- Chọn Hadoop có version tương ứng với Spark bên trên
- Copy vào **C:/hadoop/**

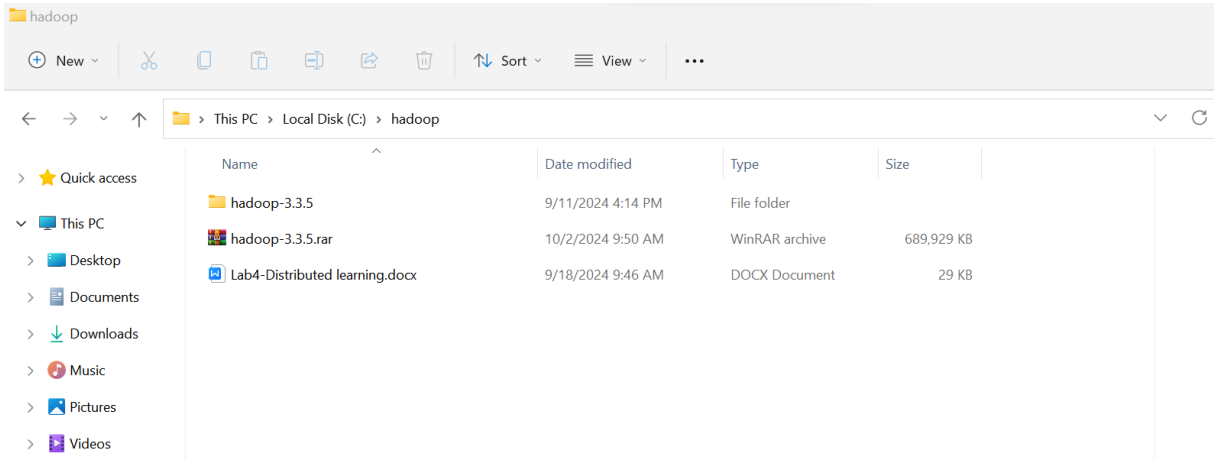


Image 5: Thiết lập Hadoop

1.6. Thiết lập biến môi trường

- **SPARK_HOME**=C:\spark\spark-3.5.2-bin-hadoop3
- **HADOOP_HOME**=C:\hadoop\hadoop-2.7.7

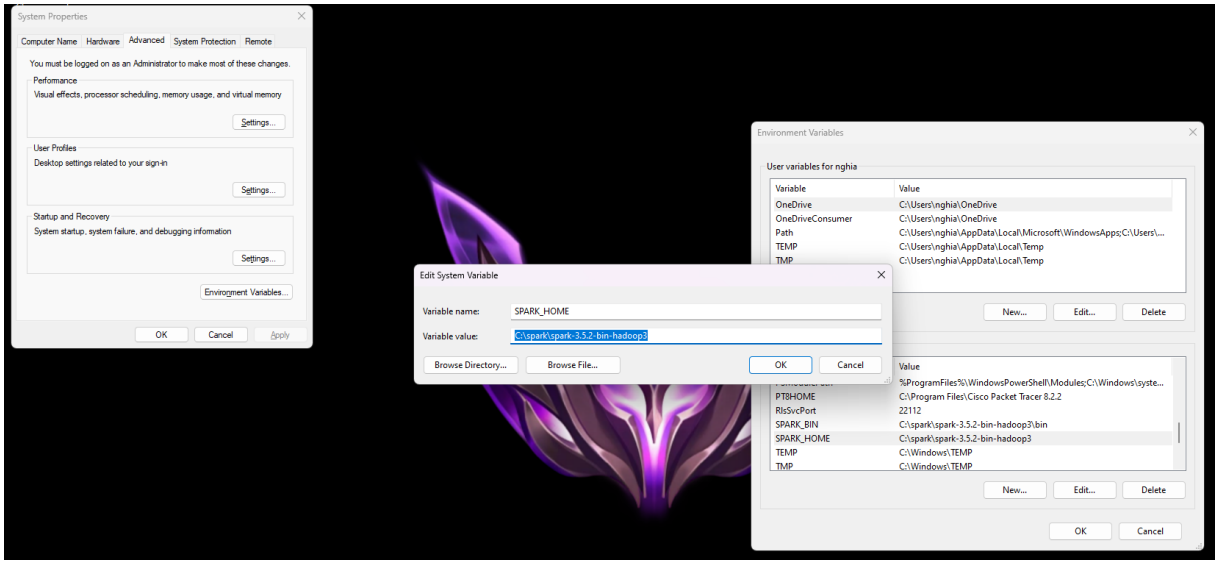


Image 6: Thiết lập biến môi trường

- Thêm Spark và Hadoop vào system PATH

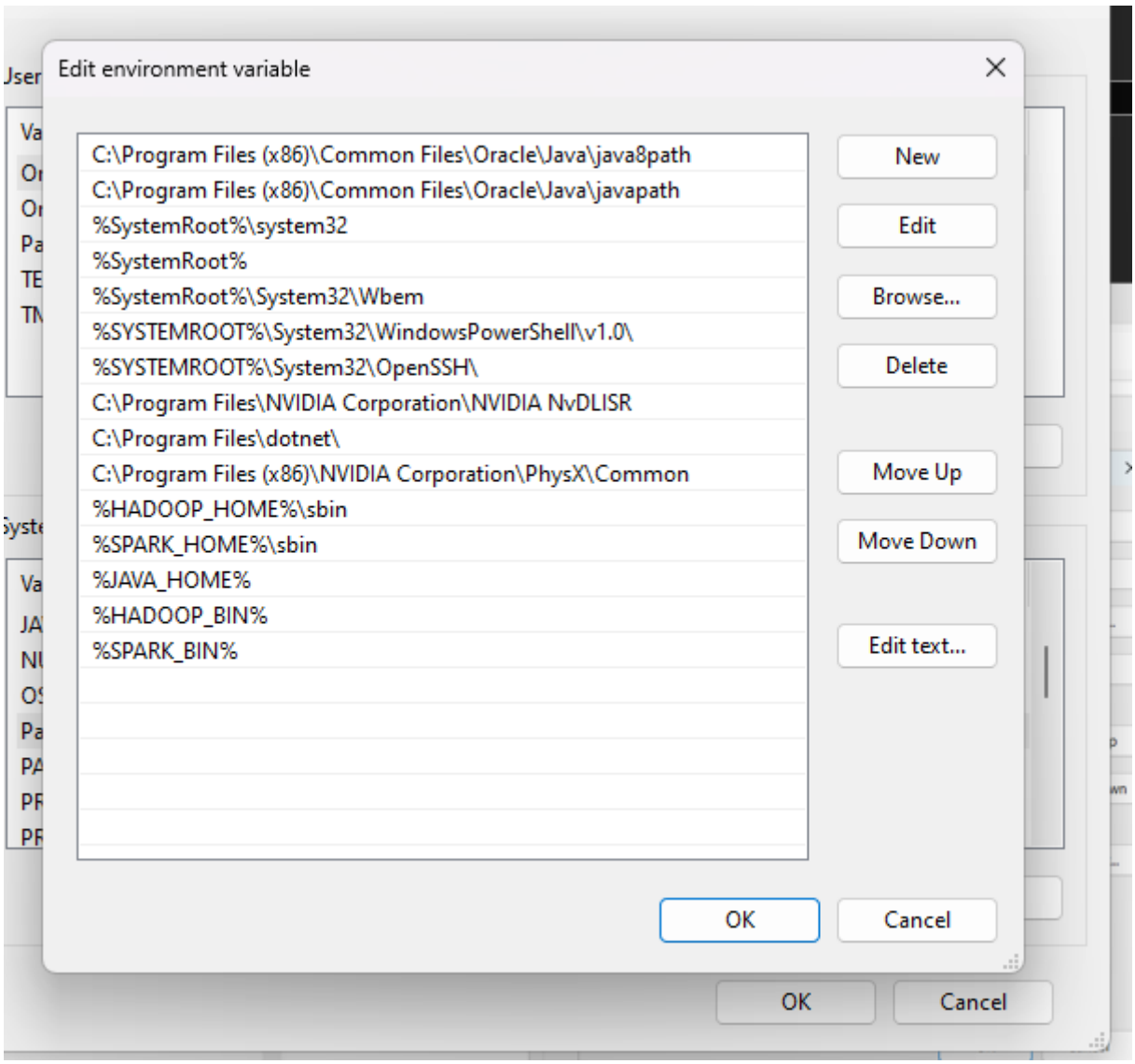


Image 7: Thêm spark và hadoop vào system path

1.7. Kiểm tra và khởi động Apache Spark

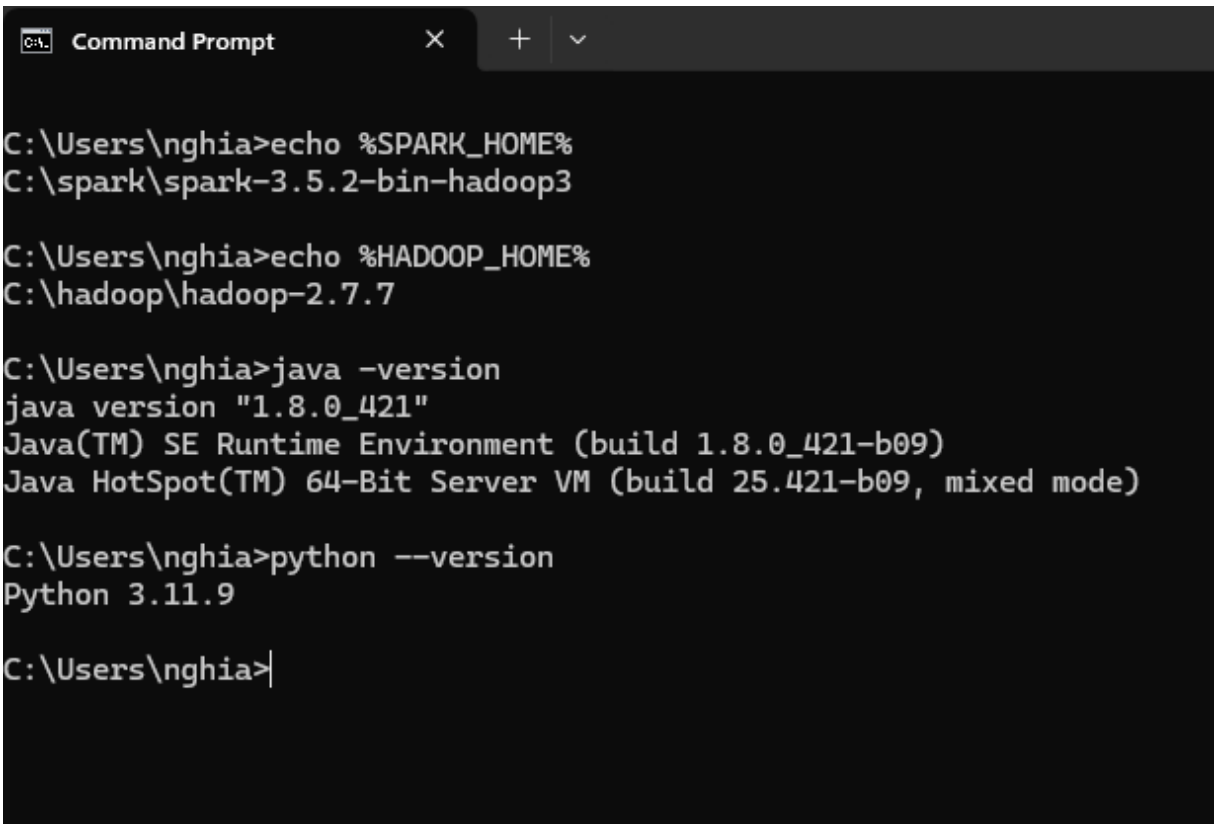


Image 8: Kiểm tra lại sau khi cài đặt spark, hadoop, java, python

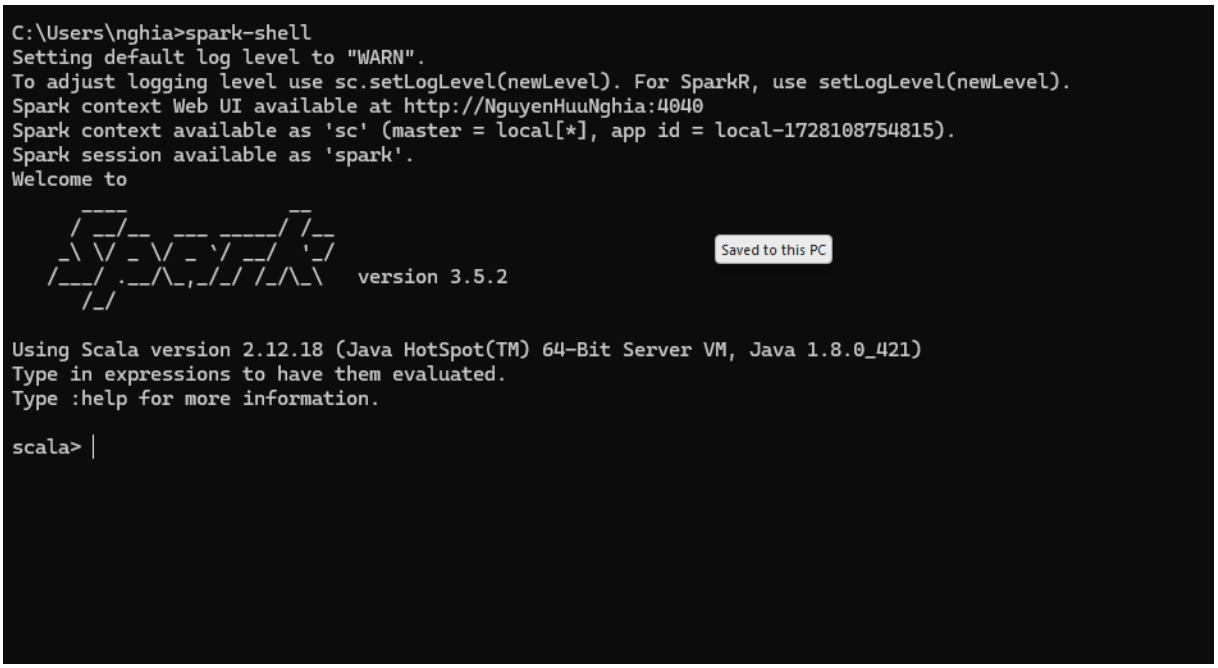


Image 9 Khởi động Spark-shell

Image 10 Khởi động Pyspark

1.8. Khởi động SPARK Master

- Mở 1 cửa sổ Command Prompt mới, chạy lệnh: `spark-class org.apache.spark.deploy.master.Master`

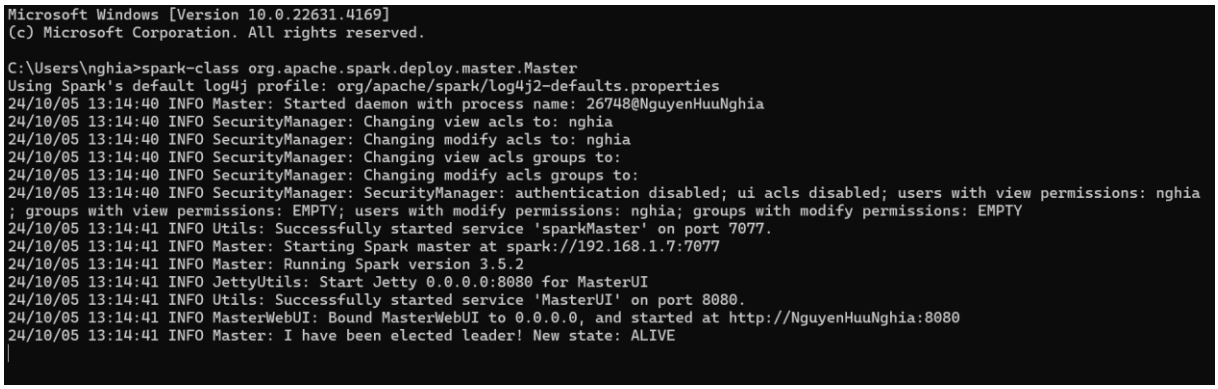


Image 11 Khởi động SPARK Master

Image 12 Kết quả trên web localhost

1.9. Khởi động SPARK Worker

```
C:\Users\nghia>spark-class org.apache.spark.deploy.worker.Worker spark://192.168.1.7:7077
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
24/10/05 13:19:11 INFO Worker: Started daemon with process name: 28932@NguyenHuuNghia
24/10/05 13:19:11 INFO SecurityManager: Changing view acls to: nghia
24/10/05 13:19:11 INFO SecurityManager: Changing modify acls to: nghia
24/10/05 13:19:11 INFO SecurityManager: Changing view acls groups to:
24/10/05 13:19:11 INFO SecurityManager: Changing modify acls groups to:
24/10/05 13:19:11 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: nghia
; groups with view permissions: EMPTY; users with modify permissions: nghia; groups with modify permissions: EMPTY
24/10/05 13:19:12 INFO Utils: Successfully started service 'sparkWorker' on port 63437.
24/10/05 13:19:12 INFO Worker: Worker decommissioning not enabled.
24/10/05 13:19:12 INFO Worker: Starting Spark worker 192.168.1.7:63437 with 8 cores, 14.8 GiB RAM
24/10/05 13:19:12 INFO Worker: Running Spark version 3.5.2
24/10/05 13:19:12 INFO Worker: Spark home: C:\spark\spark-3.5.2-bin-hadoop3
24/10/05 13:19:12 INFO ResourceUtils: ==============================================================
24/10/05 13:19:12 INFO ResourceUtils: No custom resources configured for spark.worker.
24/10/05 13:19:12 INFO ResourceUtils: ==============================================================
24/10/05 13:19:12 INFO JettyUtils: Start Jetty 0.0.0.0:8081 for WorkerUI
24/10/05 13:19:12 INFO Utils: Successfully started service 'WorkerUI' on port 8081.
24/10/05 13:19:12 INFO WorkerWebUI: Bound WorkerWebUI to 0.0.0.0, and started at http://NguyenHuuNghia:8081
24/10/05 13:19:12 INFO Worker: Connecting to master 192.168.1.7:7077...
24/10/05 13:19:12 INFO TransportClientFactory: Successfully created connection to /192.168.1.7:7077 after 37 ms (0 ms spent in bootstr
aps)
24/10/05 13:19:12 INFO Worker: Successfully registered with master spark://192.168.1.7:7077
```

Image 13 Khởi động SPARK Worker

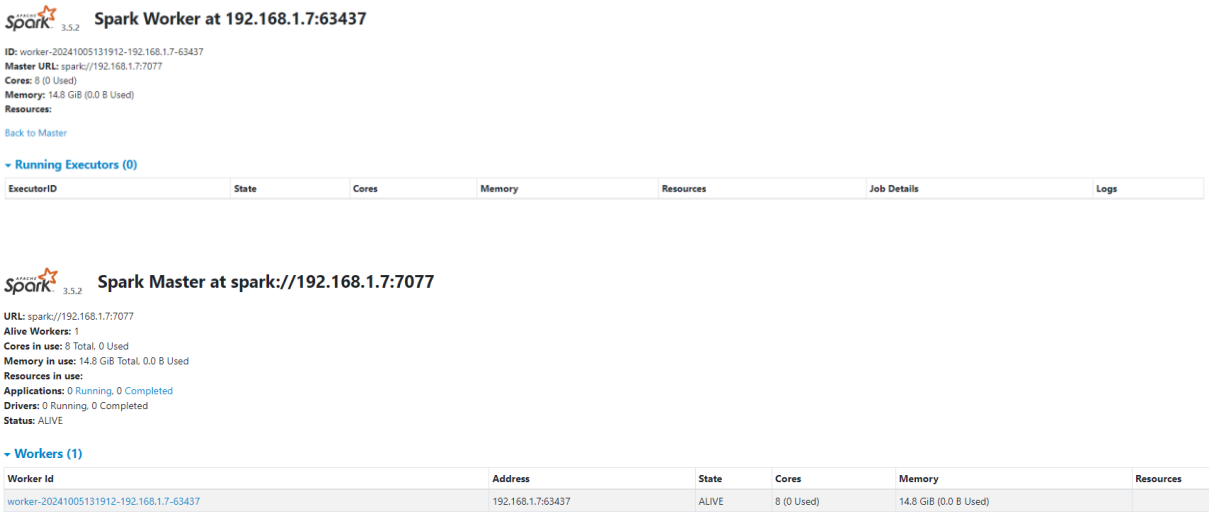


Image 14 Kết quả trên web localhost

1.10. Kết nối Shell với Spark Master

```
C:\Users\nghia>spark-shell --master spark://192.168.1.7:7077
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://NguyenHuuNghia:4040
Spark context available as 'sc' (master = spark://192.168.1.7:7077, app id = app-20241005132236-0000).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.2
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_421)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Image 15 Kết nối Shell với Spark Master

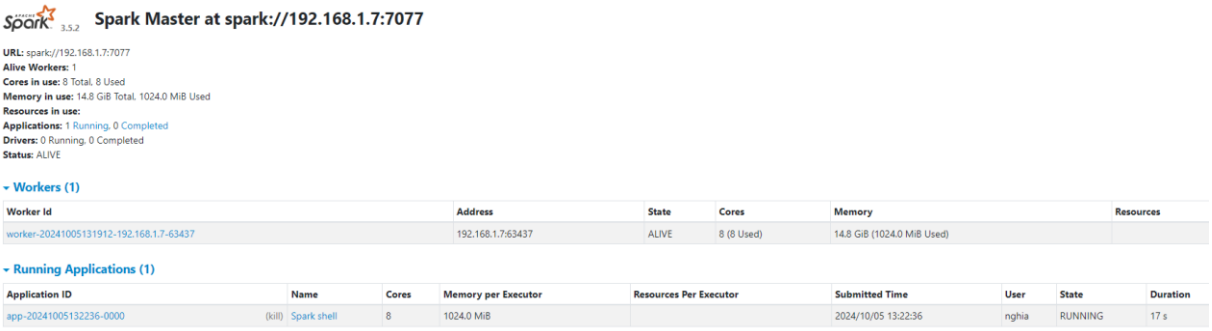


Image 16 Kết quả trên web localhost:8080

CHƯƠNG 2. LAUNCHING APPLICATION ON HADOOP – SPARK WINDOWS

2.1. Khởi động hệ thống: Hadoop, Spark.

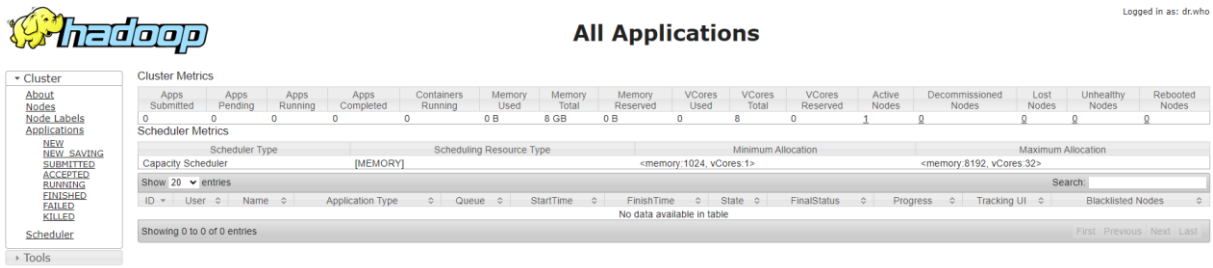
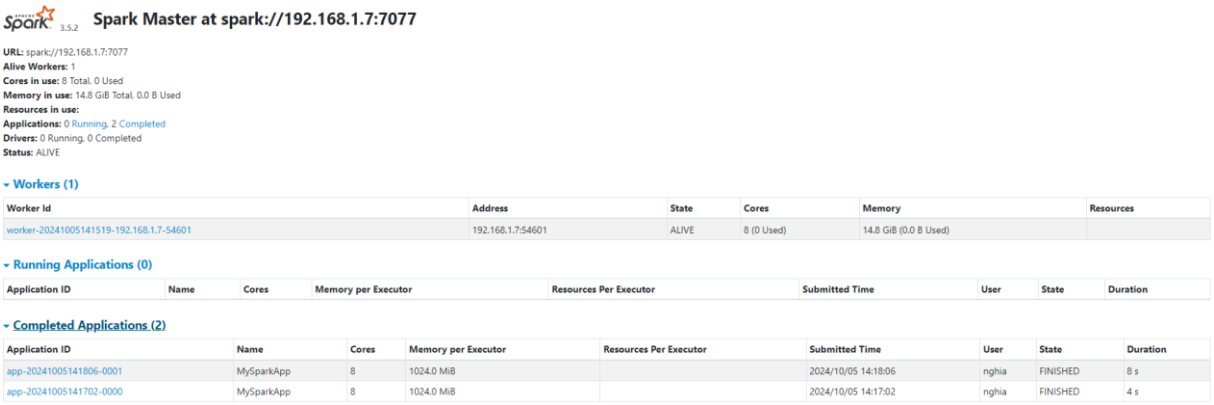


Image 17 Khởi động hệ thống: Hadoop, Spark.



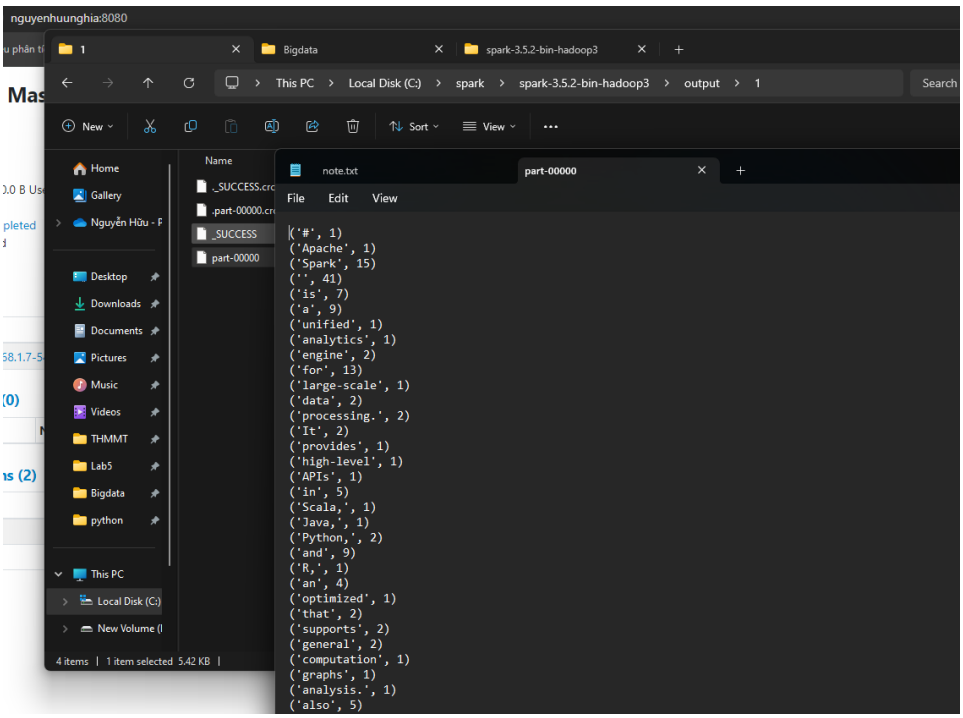


Image 18 Submit wordcount_demo.py và kết quả

▾ Running Applications (3)

Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241005134206-0003	(kill)	PythonALS	0	1024.0 MiB		2024/10/05 13:42:06	nghia	WAITING	8 s
app-20241005133735-0002	(kill)	MySparkApp	0	1024.0 MiB		2024/10/05 13:37:35	nghia	WAITING	4.6 min
app-20241005132236-0000	(kill)	Spark shell	8	1024.0 MiB		2024/10/05 13:22:36	nghia	RUNNING	20 min

Image 19 Khởi động hệ thống: Hadoop, Spark.

CHƯƠNG 3. XỬ LÝ DỮ LIỆU VĂN BẢN VỚI PYSPARK

3.1. Mô tả dữ liệu

B1: Khởi tạo ứng dụng PySpark

B2: Tạo RDD

B3: Xử lý trên RDD

```
import os
import shutil
from pyspark import SparkContext

# Khởi tạo SparkContext
sc = SparkContext("local", "Twitter Data Processing")

# Đọc dữ liệu từ file CSV
lines = sc.textFile("elonmusk_tweets.csv")

# Tách các dòng thành các từ
words = lines.flatMap(lambda line: line.split(" "))

# Đếm tần số của mỗi từ
wordFrequencies = words.map(lambda word: (word.lower(), 1)).reduceByKey(lambda a, b: a + b)

# Lấy 20 từ được nhắc đến nhiều nhất
topWords = wordFrequencies.takeOrdered(20, key=lambda x: -x[1])
print("\nTop 20 words:")
for word, count in topWords:
    print(f"{word}: {count}")

# Đếm tần số của các tài khoản được nhắc đến
mentions = lines.flatMap(lambda line: [word for word in line.split() if word.startswith('@')])
mentionFrequencies = mentions.map(lambda mention: (mention, 1)).reduceByKey(lambda a, b: a + b)
```

Image 20 Xử lý dữ liệu văn bản với pyspark

```
# Lấy 10 tài khoản được nhắc đến nhiều nhất
topMentions = mentionFrequencies.takeOrdered(10, key=lambda x: -x[1])
print("\nTop 10 mentioned accounts:")
for account, count in topMentions:
    print(f"{account}: {count}")

# Lưu kết quả vào một tệp văn bản
savingPath = "file:///C:/spark/spark-3.5.2-bin-hadoop3/output/lab3"
if os.path.isdir(savingPath):
    shutil.rmtree(savingPath, ignore_errors=True)

wordFrequencies.saveAsTextFile(savingPath)

# Đóng SparkContext
sc.stop()
```

Image 21 Xử lý dữ liệu văn bản với pyspark

3.2. Bài tập xử lý dữ liệu:

Cho file văn bản elonmusk_tweets.csv chứa các dòng tweets của Elon Musk từ 2011-2017. Dữ liệu được chia sẻ bởi Adam Helsinger. Từ file dữ liệu trên, hãy thực hiện Viết chương trình lab3.py thực hiện các xử lý sau với PySpark:

- Liệt kê 20 từ được nhắc đến nhiều nhất

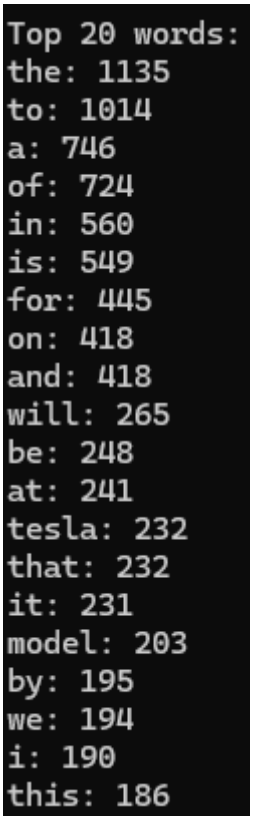


Image 22 Liệt kê 20 từ được nhắc nhiều nhất

- Liệt kê 10 tài khoản được nhắc đến nhiều nhất

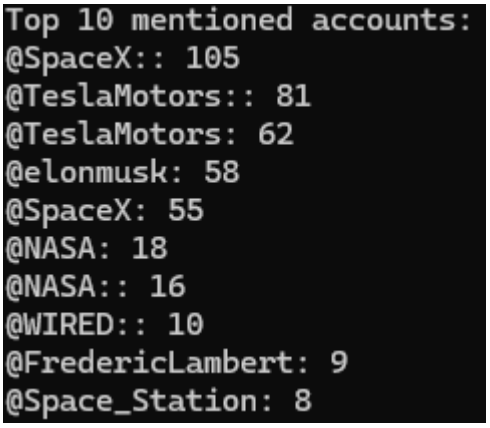


Image 23 liệt kê 5 tài khoản được nhắc nhiều nhất

CHƯƠNG 4. ỨNG DỤNG SPARK ĐỂ TRIỂN KHAI CÁC MÔ HÌNH HỌC MÁY (MACHINE LEARNING) PHÂN TÁN

4.1 Kmeans_pyspark

Soucer code:

```
import pandas as pd
import numpy as np
from pyspark.ml.clustering import KMeans
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler

# Khởi tạo Spark Session
spark = SparkSession.builder.appName("Distributed KMeans Example").getOrCreate()

# Đọc dữ liệu iris vào Spark DataFrame
iris_data = spark.createDataFrame(pd.read_csv('E:/TTNT/2024-2025/Bigdata/lab4/iris.csv', header=None, names=['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'label']))
print("First 10 rows:")
iris_data.show(10)

# Tạo vector đặc trưng từ DataFrame
assembler = VectorAssembler(inputCols=["sepal-length", "sepal-width", "petal-length", "petal-width"], outputCol="features")
irisFeatures = assembler.transform(iris_data)
irisFeatures.show(5)

# Huấn luyện mô hình K-means với K=3
kmeans = KMeans().setK(3).setSeed(0)
model = kmeans.fit(irisFeatures)

# In ra tâm điểm của các cụm
centers = model.clusterCenters()
np.set_printoptions(precision=2)
print("Centers learned by Spark ML: ")
for center in centers:
    print(center)

# Đóng Spark Session
spark.stop()
```

Kết quả:

- **Submit lên spark-master**

▾ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241005142031-0002 (kill)	Distributed KMeans Example	8	1024.0 MiB		2024/10/05 14:20:31	nghia	RUNNING	3 s

▾ Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241005141806-0001	MySparkApp	8	1024.0 MiB		2024/10/05 14:18:06	nghia	FINISHED	8 s
app-20241005141702-0000	MySparkApp	8	1024.0 MiB		2024/10/05 14:17:02	nghia	FINISHED	4 s

```
+------------+-----------+------------+-----------+-----------+
|sepal-length|sepal-width|petal-length|petal-width|      label|
+------------+-----------+------------+-----------+-----------+
|         5.1|        3.5|         1.4|        0.2|Iris-setosa|
|         4.9|        3.0|         1.4|        0.2|Iris-setosa|
|         4.7|        3.2|         1.3|        0.2|Iris-setosa|
|         4.6|        3.1|         1.5|        0.2|Iris-setosa|
|         5.0|        3.6|         1.4|        0.2|Iris-setosa|
|         5.4|        3.9|         1.7|        0.4|Iris-setosa|
|         4.6|        3.4|         1.4|        0.3|Iris-setosa|
|         5.0|        3.4|         1.5|        0.2|Iris-setosa|
|         4.4|        2.9|         1.4|        0.2|Iris-setosa|
|         4.9|        3.1|         1.5|        0.1|Iris-setosa|
+------------+-----------+------------+-----------+-----------+
only showing top 10 rows
```

```
Centers learned by Spark ML:
[5.9  2.75 4.39 1.43]
[5.01 3.42 1.46 0.24]
[6.85 3.07 5.74 2.07]
```

Image 24 kết quả Kmeans-pyspark

4.2 Kmeans_scikit-learn

Soucer code:

```python
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Nạp dữ liệu
iris_data = pd.read_csv('E:/TTNT/2024-2025/Bigdata/lab4/iris.csv', header=None, names=['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'label'])

# Tạo vector đặc trưng
X = iris_data[['sepal-length', 'sepal-width', 'petal-length', 'petal-width']].values

# Huấn luyện mô hình K-means với K=3
kmeans_sklearn = KMeans(n_clusters=3, random_state=0)
kmeans_sklearn.fit(X)

# In ra tọa độ điểm của các cụm
centers_sklearn = kmeans_sklearn.cluster_centers_
np.set_printoptions(precision=2)
print("Centers learned by Scikit-learn: ")
for center in centers_sklearn:
    print(center)

# Vẽ biểu đồ các cụm
plt.scatter(X[:, 0], X[:, 1], c=kmeans_sklearn.labels_, cmap='viridis', marker='o')
plt.scatter(centers_sklearn[:, 0], centers_sklearn[:, 1], c='red', s=200, alpha=0.75, label='Centroids')
plt.title('K-means Clustering (Scikit-learn)')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.legend()
plt.show()
```

Kết quả:

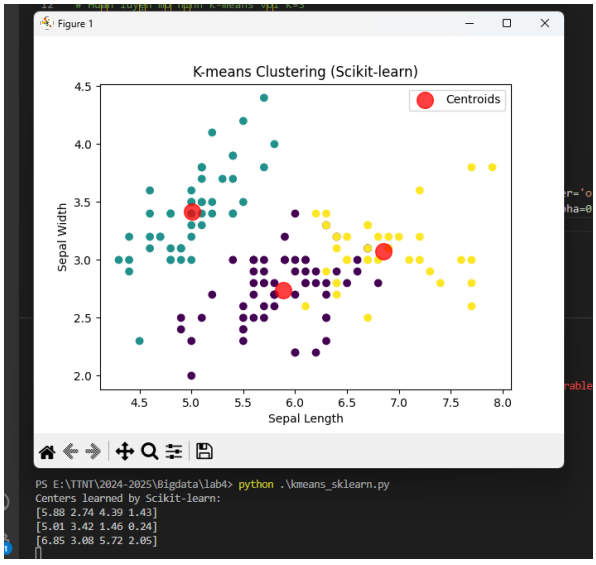


Image 25 Kết quả Kmeans-sklearn

4.3 So sánh tâm cụm

```
Centers learned by Spark ML:
[5.9  2.75 4.39 1.43]
[5.01 3.42 1.46 0.24]
[6.85 3.07 5.74 2.07]
```

```
Centers learned by Scikit-learn:
[5.88 2.74 4.39 1.43]
[5.01 3.42 1.46 0.24]
[6.85 3.08 5.72 2.05]
```

- Dựa vào kết quả trên, có thể thấy rằng chênh lệch Kmeans_spark, và Kmeans_scikitlearn không đáng kể

4.5 Nhận xét

- Hiệu suất: Nếu bạn làm việc với tập dữ liệu lớn, PySpark sẽ hiệu quả hơn vì nó được thiết kế cho xử lý phân tán. Scikit-learn thường hoạt động tốt trên dữ liệu nhỏ hơn.

- Dễ sử dụng: Scikit-learn có thể dễ sử dụng hơn cho người mới bắt đầu nhờ vào API trực quan và tài liệu phong phú.

- Khả năng mở rộng: PySpark cho phép bạn mở rộng xử lý dữ liệu lớn mà không gặp vấn đề về hiệu suất.

CHƯƠNG 5. PHÂN TÍCH MẠNG XÃ HỘI

Cho file văn bản ***data/twitter_following.txt*** chứa thông tin về việc tài khoản người dùng theo dõi tài khoản khác trên cùng mạng xã hội. Mỗi dòng file văn bản có dạng:

<user_id1> <user_id2>

cho biết người dùng với user_id1 theo dõi người dùng user_id2. Nếu user_id2 cũng theo dõi user_id1 thì cặp tài khoản này được gọi là theo dõi lẫn nhau (mutual followers).

Yêu cầu:

Vận dụng cả 2 phương pháp (1) truyền thống và (2) dữ liệu lớn thực hiện các xử lý sau:

- Thống kê số lượng người theo dõi (followers) của mỗi tài khoản người dùng.
- Liệt kê top 5 người dùng có nhiều theo dõi nhất.
- Liệt kê tất cả các cặp người dùng theo dõi lẫn nhau trong file dữ liệu được cho.

Cài đặt thư viện networkx

```
C:\Users\PC>pip install networkx
Collecting networkx
  Downloading networkx-3.3-py3-none-any.whl.metadata (5.1 kB)
Downloading networkx-3.3-py3-none-any.whl (1.7 MB)
   ---------------------------------------- 1.7/1.7 MB 941.7 kB/s eta 0:00:00
Installing collected packages: networkx
Successfully installed networkx-3.3

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: C:\Users\PC\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra
8p0\python.exe -m pip install --upgrade pip
```

Image 26 Cài đặt thư viện networkx

Chạy Phương pháp Truyền thống

Soucer code:

```python
import networkx as nx
import matplotlib.pyplot as plt

# Đọc dữ liệu từ file và tạo danh sách các cạnh
edges = []
with open('twitter_followings.txt', 'r') as f:
    for line in f:
        id1, id2 = line.strip().split()
        edges.append((id1, id2))

# Tạo đồ thị có hướng từ danh sách các cạnh
G = nx.DiGraph(edges)

# Tìm mutual followers
mutual_followers = []
for u, v in G.edges():
    if G.has_edge(v, u):
        if (v, u) not in mutual_followers:
            mutual_followers.append((u, v))

# In các cặp mutual followers
print('Các cặp tài khoản theo dõi lẫn nhau:')
for u, v in mutual_followers:
    print(f'{u} <-> {v}')

# **Bổ sung 1: Thiết lập vị trí các node để đồ thị đẹp hơn**
pos = nx.spring_layout(G, k=0.15, iterations=20)

# **Bổ sung 2: Vẽ tất cả các cạnh không phải mutual followers**
non_mutual_edges = [edge for edge in G.edges() if edge not in mutual_followers and (edge[1], edge[0]) not in mutual_followers]
nx.draw_networkx_nodes(G, pos, node_size=800, node_color='#3498DB')
nx.draw_networkx_labels(G, pos, font_size=12, font_color='white')
```

```python
# Vẽ các cạnh không phải mutual followers
nx.draw_networkx_edges(
    G,
    pos,
    edgelist=non_mutual_edges,
    edge_color='#FF5733',
    arrows=True,
    arrowstyle='-|>',
    arrowsize=15,
    label='Following'
)

# **Bổ sung 3: Vẽ các cạnh mutual followers với màu khác**
nx.draw_networkx_edges(
    G,
    pos,
    edgelist=mutual_followers,
    edge_color='green',
    arrows=True,
    arrowstyle='-|>',
    arrowsize=15,
    label='Mutual Following',
    connectionstyle='arc3, rad=0.1'  # Để tránh các cạnh chồng lấp
)

# **Bổ sung 4: Tạo legend cho đồ thị**
import matplotlib.patches as mpatches

following_patch = mpatches.Patch(color='#FF5733', label='Following')
mutual_patch = mpatches.Patch(color='green', label='Mutual Following')
plt.legend(handles=[following_patch, mutual_patch])

# **Bổ sung 5: Tùy chỉnh hiển thị đồ thị**
plt.title('Twitter Followers Graph')
plt.axis('off')  # Tắt hiển thị trục
plt.tight_layout()

# **Bổ sung 6: Hiển thị đồ thị**
plt.show()
```

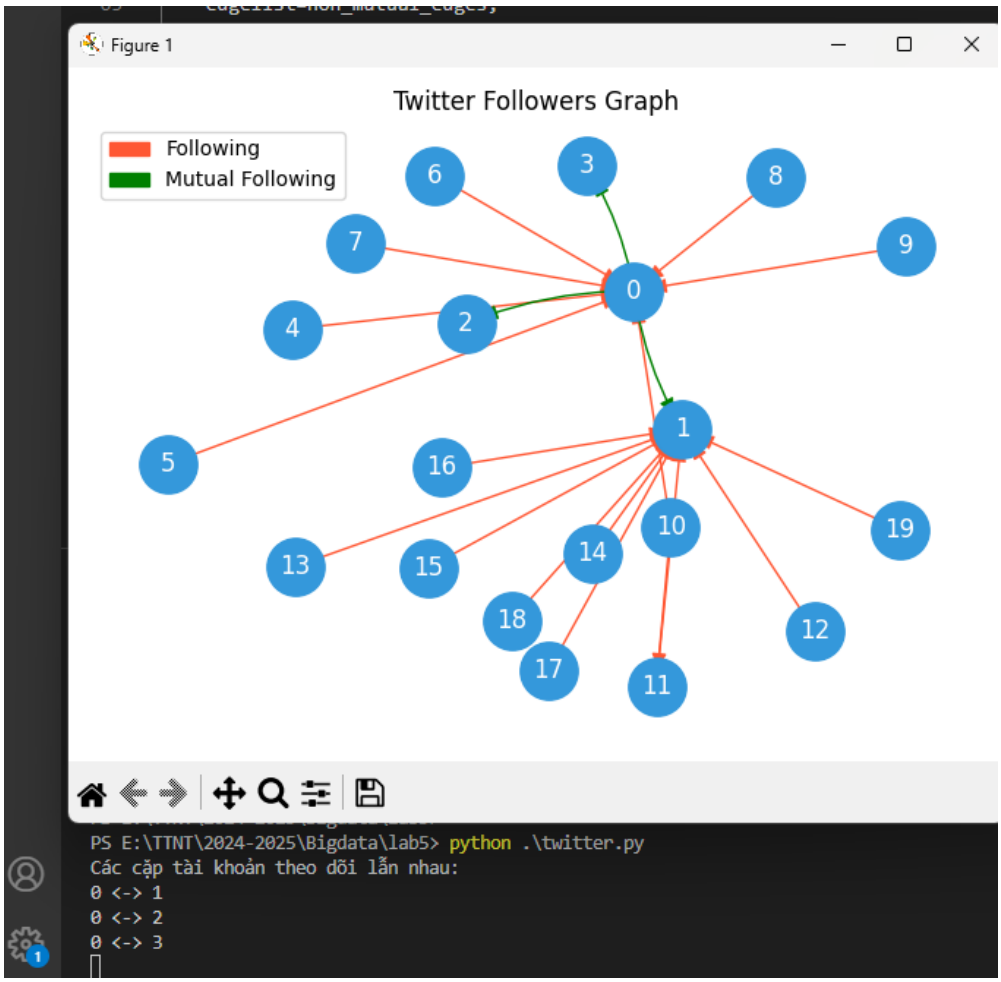


Image 27 Kết quả phương pháp truyền thống.

Chạy Phương pháp Dữ liệu Lớn

Soucer code:

```
import sys
import io

# Thiết lập mã hóa đầu ra là UTF-8
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, split

# Khởi tạo Spark Session
spark = SparkSession.builder.appName("Twitter Analysis").getOrCreate()

# Đọc dữ liệu từ file
df = spark.read.text('Twitter_followings.txt')

# Chia dữ liệu thành hai cột user_id1 và user_id2
df = df.withColumn('user_id1', split(df['value'], ' ')[0]) \
       .withColumn('user_id2', split(df['value'], ' ')[1]) \
       .drop('value')
```

```
# Thống kê số lượng người theo dõi cho mỗi tài khoản
followers_count_df = df.groupBy('user_id2').agg(count('user_id1').alias('followers_count'))

# Liệt kê top 5 người dùng có nhiều theo dõi nhất
top_followers_df = followers_count_df.orderBy(col('followers_count').desc()).limit(5)
print("Top 5 nguoi dung co nhieu theo doi nhat:")
top_followers_df.show()

# Tìm cặp tài khoản theo dõi lẫn nhau
mutual_followers_df = df.alias('a').join(df.alias('b'),
    (col('a.user_id1') == col('b.user_id2')) & (col('a.user_id2') == col('b.user_id1')),
    'inner').select(col('a.user_id1').alias('user1'), col('a.user_id2').alias('user2'))

# Liệt kê các cặp tài khoản theo dõi lẫn nhau
print('Cac cap tai khoan theo doi lan nhau:')
mutual_followers_df.show(truncate=False)

# Dừng Spark Session
spark.stop()
```

- Submit lên spark-master

▾ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241005150000-0003 (kill)	Twitter Analysis	8	1024.0 MiB		2024/10/05 15:00:00	nghia	RUNNING	6 s

▾ Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241005142031-0002	Distributed KMeans Example	8	1024.0 MiB		2024/10/05 14:20:31	nghia	FINISHED	23 s
app-20241005141806-0001	MySparkApp	8	1024.0 MiB		2024/10/05 14:18:06	nghia	FINISHED	8 s
app-20241005141702-0000	MySparkApp	8	1024.0 MiB		2024/10/05 14:17:02	nghia	FINISHED	4 s

```
Top 5 nguoi dung co nhieu theo doi nhat:
+--------+---------------+
|user_id2|followers_count|
+--------+---------------+
|       0|             10|
|       1|             10|
|      11|              1|
|       3|              1|
|       2|              1|
+--------+---------------+

Cac cap tai khoan theo doi lan nhau:
+-----+-----+
|user1|user2|
+-----+-----+
|0    |1    |
|0    |2    |
|0    |3    |
|1    |0    |
|2    |0    |
|3    |0    |
+-----+-----+
```

Image 28 Kết quả phương pháp dữ liệu lớn

KẾT LUẬN

Báo cáo tổng kết thực hành chuyên đề dữ liệu lớn đã trình bày những khía cạnh quan trọng trong việc cài đặt và sử dụng Hadoop, Spark, cũng như ứng dụng của chúng trong phân tích và xử lý dữ liệu. Qua các chương, chúng ta đã thực hiện các bước cụ thể để thiết lập môi trường, xử lý dữ liệu văn bản và phân tích mạng xã hội, đồng thời áp dụng hai phương pháp truyền thống và dữ liệu lớn.

Cài đặt và Thiết lập Môi Trường: Việc cài đặt Hadoop và Spark trên Windows đã được thực hiện thành công, giúp chúng ta có nền tảng vững chắc để làm việc với dữ liệu lớn. Điều này cũng minh họa cho sự cần thiết của việc thiết lập biến môi trường và cấu hình phù hợp để các công cụ này hoạt động hiệu quả.

Xử Lý Dữ Liệu với PySpark: Qua việc thực hiện các bài tập xử lý dữ liệu văn bản, chúng ta đã có cái nhìn sâu sắc hơn về cách PySpark xử lý các tập dữ liệu lớn một cách hiệu quả. Việc thống kê tần suất từ ngữ và tài khoản đã cho thấy khả năng xử lý nhanh chóng của PySpark so với phương pháp truyền thống.

Phân Tích Mạng Xã Hội: Chương phân tích mạng xã hội đã giúp chúng ta hiểu rõ hơn về cách mà các tài khoản người dùng tương tác với nhau trên nền tảng mạng xã hội. Việc xác định số lượng người theo dõi, tìm kiếm top người dùng có nhiều theo dõi nhất, cũng như phát hiện các cặp tài khoản theo dõi lẫn nhau đã cho thấy tính ứng dụng cao của các kỹ thuật phân tích mạng.

So Sánh Hai Phương Pháp: Qua việc áp dụng cả phương pháp truyền thống và dữ liệu lớn, chúng ta nhận thấy rằng PySpark vượt trội hơn trong việc xử lý các tập dữ liệu lớn, trong khi phương pháp truyền thống dễ hiểu và phù hợp với dữ liệu nhỏ hơn.

Định Hướng Tương Lai

Từ những kinh nghiệm và kiến thức thu được, chúng ta có thể mở rộng nghiên cứu sang các lĩnh vực khác như học máy (machine learning) hoặc phân tích dự đoán, nhằm khai thác tối đa giá trị từ dữ liệu lớn. Sự phát triển của công nghệ sẽ tiếp tục mở ra nhiều cơ hội và thách thức, đòi hỏi những người làm trong lĩnh vực này không ngừng học hỏi và thích nghi.

TÀI LIỆU THAM KHẢO

[1] File bài giảng

[2] https://www.who.int/vietnam/vi/health-topics/cardiovascular-disease

[3] https://en.wikipedia.org/wiki/ID3_algorithm

[4] https://en.wikipedia.org/wiki/C4.5_algorithm

[5] https://machinelearningcoban.com/2017/08/31/evaluation/

[6] https://machinelearningcoban.com/2018/01/14/id3/

[7] https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

[8]https://www.kaggle.com/code/tanmay111999/heart-failure-prediction-cv-score-90-5-models