**MỤC LỤC**

[**LỜI MỞ ĐẦU 8**](#_Toc199335595)

[**I) CHƯƠNG I: TỔNG QUAN VỀ DỮ LIỆU LỚN VÀ GIỚI THIỆU ĐỀ TÀI 9**](#_Toc199335596)

[**1. Tổng quan về Dữ liệu lớn - Big Data 9**](#_Toc199335597)

[**a) Giới thiệu về dữ liệu lớn và sự bùng nổ của dữ liệu lớn 9**](#_Toc199335598)

[**b) 7 "Vs" của Big Data - Dữ liệu lớn 14**](#_Toc199335599)

[**c) Tầm quan trọng của việc phân tích và xử lý dữ liệu lớn trong đời sống 16**](#_Toc199335600)

[**d) Ví dụ và thực tiễn ứng dụng Big Data ở Việt Nam 17**](#_Toc199335601)

[**2. Giới thiệu đề tài 19**](#_Toc199335602)

[**3. Giới thiệu bộ dữ liệu 20**](#_Toc199335603)

[**II) APACHE HADOOP 22**](#_Toc199335604)

[**1) Tổng quan về Apache Hadoop 23**](#_Toc199335605)

[**a) Giới thiệu 23**](#_Toc199335606)

[**b) Mục tiêu chính 23**](#_Toc199335607)

[**c) Ưu điểm 23**](#_Toc199335608)

[**d) Các thành phần trong hệ sinh thái của Apache Hadoop 24**](#_Toc199335609)

[**2) Thực nghiệm 26**](#_Toc199335610)

[**3) Đánh giá và thêm một chút kiến thức về HDFS 51**](#_Toc199335611)

[**a) Đánh giá 51**](#_Toc199335612)

[**b) Cơ chế lưu trữ dữ liệu trên HDFS (block, replication) 52**](#_Toc199335613)

[**c) Block (Khối dữ liệu) 54**](#_Toc199335614)

[**d) Data Replication (Sao chép dữ liệu) 54**](#_Toc199335615)

[**e) Đọc và ghi dữ liệu trong HDFS 56**](#_Toc199335616)

[**f) Ưu điểm của cơ chế lưu trữ HDFS 56**](#_Toc199335617)

[**III) APACHE SPARK 56**](#_Toc199335618)

[**1) Apache Spark 56**](#_Toc199335619)

[**a) Giới thiệu 56**](#_Toc199335620)

[**b) Mục tiêu chính 58**](#_Toc199335621)

[**c) Các thành phần trong hệ sinh thái của Apache Spark 58**](#_Toc199335622)

[**d) Ưu điểm 60**](#_Toc199335623)

[**e) Thực nghiệm 60**](#_Toc199335624)

[**f) Tổng kết 64**](#_Toc199335625)

[**2) Truy vấn với Spark SQL 64**](#_Toc199335626)

[**a) Tổng quan về Spark SQL 64**](#_Toc199335627)

[**b) Thực nghiệm 66**](#_Toc199335628)

[**c) Tổng kết 95**](#_Toc199335629)

[**3) Xây dựng mô hình Machine Learning qua SparkMLlib 96**](#_Toc199335630)

[**a) Tổng quan về SparkMLlib 96**](#_Toc199335631)

[**b) Thực nghiệm 97**](#_Toc199335632)

[**c) Đánh giá và tổng kết 110**](#_Toc199335633)

[**IV) CHƯƠNG IV: ĐỀ XUẤT NHỮNG VẤN ĐỀ CÓ THỂ DÙNG DỮ LIỆU LỚN ĐỂ GIẢI QUYẾT VÀ KÉT LUẬN 116**](#_Toc199335634)

[**1) Đề xuất những vấn đề có thể áp dụng dữ liệu lớn để giải quyết 116**](#_Toc199335635)

[**2) Kết luận 118**](#_Toc199335636)

[**TÀI LIỆU THAM KHẢO 120**](#_Toc199335637)

**MỤC LỤC HÌNH VẼ**

[**Hình 1: Thống kê dữ liệu được tạo ra trong từng phút của năm 2023 11**](#_Toc199290281)

[**Hình 2: Thống kê dữ liệu được tạo ra từ năm 2010-2020 và dự đoán cho giai đoạn 2021-2025 12**](#_Toc199290282)

[**Hình 3: 7 đặc điểm, tính chất quan trọng của Big Data - Nguồn Internet 13**](#_Toc199290283)

[**Hình 4: Big data ứng dụng trong các trung tâm giám sát điều hành thông minh các tỉnh - Nguồn Internet 15**](#_Toc199290284)

[**Hình 5: Các thành phần trong hệ sinh thái Apache Hadoop – Nguồn Internet 22**](#_Toc199290285)

[**Hình 6: Thông tin về dữ liệu 25**](#_Toc199290286)

[**Hình 7: Cách tải bộ dữ liệu về 25**](#_Toc199290287)

[**Hình 8: Thông tin về bộ dữ liệu sau khi tải về 26**](#_Toc199290288)

[**Hình 9: Hai file Java SE với Hadoop sau khi tải về 27**](#_Toc199290289)

[**Hình 10: Giao diện cài đặt của Java SE, nhấn Next để tiếp tục 27**](#_Toc199290290)

[**Hình 11: Chọn Development Tools và nhấn Next 28**](#_Toc199290291)

[**Hình 12: Chờ đợi 28**](#_Toc199290292)

[**Hình 13: Sau khi cài xong sẽ có một setup khác hiện lên, ta nhấn Next 1 lần nữa 29**](#_Toc199290293)

[**Hình 14: Vẫn tiếp tục chờ đợi 29**](#_Toc199290294)

[**Hình 15: Xong rồi, đến lượt đi cấu hình thôi 30**](#_Toc199290295)

[**Hình 16: Đường dẫn tới cài đặt môi trường 31**](#_Toc199290296)

[**Hình 17: Ở mỗi đường dẫn user và hệ thống, ta sẽ nhấn New (Thêm đường dẫn mới vào) 32**](#_Toc199290297)

[**Hình 18: Nhập đường dẫn JAVA\_HOME vào 32**](#_Toc199290298)

[**Hình 19: Sau khi thêm vào, JAVA\_HOME sẽ hiển thị như này 33**](#_Toc199290299)

[**Hình 20: Ở phần PATH của đường dẫn hệ thống, ta sẽ nhấn cú đúp vào để hiện mục danh sách đường dẫn như trên, xong rồi thêm đường dẫn như trên hình 34**](#_Toc199290300)

[**Hình 21: Nếu mà hiện kết quả như này là ta đã cấu hình Java SE thành công 35**](#_Toc199290301)

[**Hình 22: Thư mục Hadoop trong ổ C: 35**](#_Toc199290302)

[**Hình 23: Ở mục thiết lập môi trường, ta thêm HADOOP\_HOME vào thôi 36**](#_Toc199290303)

[**Hình 24: Ở mục PATH trong đường dẫn hệ thống, ta sẽ thêm đường dẫn bin và sbin của Hadoop vào 37**](#_Toc199290304)

[**Hình 25: Khi nó hiện như này, có nghĩa là mình đã cấu hình thành công 38**](#_Toc199290305)

[**Hình 26: core-site.xml 39**](#_Toc199290306)

[**Hình 27: mapred-site.xml 40**](#_Toc199290307)

[**Hình 28: hdfs-site.xml 41**](#_Toc199290308)

[**Hình 29: yarn-site.xml 42**](#_Toc199290309)

[**Hình 30: hadoop-env.cmd 43**](#_Toc199290310)

[**Hình 31: Trang chủ tải xuống Winutils ở Github 44**](#_Toc199290311)

[**Hình 32: Thư mục bin có chứa Winutils sau khi giải nén và đè nén thư mục bin ở thư mục Hadoop 44**](#_Toc199290312)

[**Hình 33: Format NameNode 45**](#_Toc199290313)

[**Hình 34: Format DataNode 45**](#_Toc199290314)

[**Hình 35: Khởi động Hadoop 46**](#_Toc199290315)

[**Hình 36: Sử dụng lệnh jps 46**](#_Toc199290316)

[**Hình 37: Giao diện Hadoop khi vào localhost:8088 46**](#_Toc199290317)

[**Hình 38: Giao diện tổng quan cuả Hadoop khi vào localhost:9870 47**](#_Toc199290318)

[**Hình 39: Giao diện Datanodes của Hadoop khi vào localhost:9870 47**](#_Toc199290319)

[**Hình 40: Lệnh dùng để tạo thư mục BigData 48**](#_Toc199290320)

[**Hình 41: Sau khi tạo thư mục, thì thư mục BigData đã xuất hiện trên giao diện HDFS 48**](#_Toc199290321)

[**Hình 42: Lệnh dùng để tải file cục bộ lên HDFS 49**](#_Toc199290322)

[**Hình 43: Cuối cùng, bộ dữ liệu đã chính thúc được tải lên và lưu trữ trong HDFS 49**](#_Toc199290323)

[**Hình 44: Kiến trúc HDFS – Nguồn Internet 50**](#_Toc199290324)

[**Hình 45: Kiến trúc về Block Replication 53**](#_Toc199290325)

[**Hình 46: Lịch sử phát triển của Apache Spark 56**](#_Toc199290326)

[**Hình 47: Các thành phần trong hệ sinh thái Apache Spark 58**](#_Toc199290327)

[**Hình 48: Giao diện trang chủ Apache Spark 59**](#_Toc199290328)

[**Hình 49: Thư mục Spark 59**](#_Toc199290329)

[**Hình 50: Ở mục thiết lập đường dẫn, ta thêm SPARK\_HOME vào đường dẫn user (ta làm tương tự với đường dẫn hệ thống) 60**](#_Toc199290330)

[**Hình 51: Ở đường dẫn PATH của đường dẫn hệ thống, ta sẽ thêm các đường dẫn SPARK\_HOME vào 61**](#_Toc199290331)

[**Hình 52: Giao diện khởi động Apache Spark 62**](#_Toc199290332)

[**Hình 53: Giao diện Spark khi truy cập localhost:4040 62**](#_Toc199290333)

[**Hình 54: PySpark 64**](#_Toc199290334)

[**Hình 55: Các thư viện được khai báo 64**](#_Toc199290335)

[**Hình 56: Bắt đầu thiết lập code 65**](#_Toc199290336)

[**Hình 57: Đọc dữ liệu và tạo view với dữ liệu 66**](#_Toc199290337)

[**Hình 58: Thực hiện truy vấn Spark SQL 67**](#_Toc199290338)

[**Hình 59: Lệnh truy vấn đầu tiên 68**](#_Toc199290339)

[**Hình 60: Kết quả truy vấn đầu tiên 69**](#_Toc199290340)

[**Hình 61: Code truy vấn thứ hai 70**](#_Toc199290341)

[**Hình 62: Kết quả truy vấn thứ hai 70**](#_Toc199290342)

[**Hình 63: Code truy vấn thứ ba 71**](#_Toc199290343)

[**Hình 64: Kết quả code truy vấn thứ ba 72**](#_Toc199290344)

[**Hình 65: Code truy vấn thứ tư 73**](#_Toc199290345)

[**Hình 66: Kết quả truy vấn thứ tư 73**](#_Toc199290346)

[**Hình 67: Code truy vấn thứ năm 74**](#_Toc199290347)

[**Hình 68: Kết quả truy vấn thứ năm 74**](#_Toc199290348)

[**Hình 69: Code truy vấn thứ sáu 75**](#_Toc199290349)

[**Hình 70: Kết quả truy vấn thứ sáu 76**](#_Toc199290350)

[**Hình 71: Code truy vấn thứ bảy 77**](#_Toc199290351)

[**Hình 72: Kết quả truy vấn thứ bảy 78**](#_Toc199290352)

[**Hình 73: Code truy vấn thứ tám 79**](#_Toc199290353)

[**Hình 74: Kết quả truy vấn thứ tám 79**](#_Toc199290354)

[**Hình 75: Lệnh truy vấn thứ chin 81**](#_Toc199290355)

[**Hình 76: Kết quả truy vấn thứ chin (phần 1) 82**](#_Toc199290356)

[**Hình 77: Kết quả truy vấn thứ chin (phần 2) 83**](#_Toc199290357)

[**Hình 78: Code truy vấn thứ 10 84**](#_Toc199290358)

[**Hình 79: Kết quả truy vấn thứ 10 84**](#_Toc199290359)

[**Hình 80: Code truy vấn thứ 11 85**](#_Toc199290360)

[**Hình 81: Kết quả truy vấn thứ 11 86**](#_Toc199290361)

[**Hình 82: Code truy vấn thứ 12 87**](#_Toc199290362)

[**Hình 83: Kết quả truy vấn thứ 12 88**](#_Toc199290363)

[**Hình 84: Code truy vấn dữ liệu 13 89**](#_Toc199290364)

[**Hình 85: Kết quả truy vấn 13 89**](#_Toc199290365)

[**Hình 86: Code truy vấn thứ 14 90**](#_Toc199290366)

[**Hình 87: Kết quả truy vấn thứ 14 91**](#_Toc199290367)

[**Hình 88: Code truy vấn cuối cùng 92**](#_Toc199290368)

[**Hình 89: Kết quả truy vấn cuối cùng 92**](#_Toc199290369)

[**Hình 90: Khai báo thư viện cho bài toán dự đoán tiền boa 96**](#_Toc199290370)

[**Hình 91: Làm sạch dữ liệu, thống kê mô tả và tạo features cho bài toán dự đoán tiền boa 97**](#_Toc199290371)

[**Hình 92: Huấn luyện mô hình hồi quy dự đoán tiền boa 98**](#_Toc199290372)

[**Hình 93: Đánh giá mô hình dự đoán tiền boa 99**](#_Toc199290373)

[**Hình 94: Khai báo thư viện cho bài toán phân cụm 100**](#_Toc199290374)

[**Hình 95: Lọc dữ liệu, lấy mẫu và tạo features cho bài toán phân cụm 101**](#_Toc199290375)

[**Hình 96: Triển khai thuật toán K-means 102**](#_Toc199290376)

[**Hình 97: Huấn luyện mô hình phân cụm 103**](#_Toc199290377)

[**Hình 98: Khai báo thư viện cho bài toán dự đoán hình thức thanh toán 105**](#_Toc199290378)

[**Hình 99: Tiền xử lý dữ liệu và tạo features cho bài toán dự đoán hình thức thanh toán 105**](#_Toc199290379)

[**Hình 100: Huấn luyện và đánh giá mô hình bài toán dự đoán hình thức thanh toán 106**](#_Toc199290380)

[**Hình 101: Phân tích hệ số và hiển thị mẫu dự đoán cho bài toán dự đoán hình thức thanh toán 107**](#_Toc199290381)

[**Hình 102: Kết quả huấn luyện bài toán dự đoán tiền boa 108**](#_Toc199290382)

[**Hình 103: Kết quả huấn luyện bài toán phân cụm khu vực đón khách 110**](#_Toc199290383)

[**Hình 104: Kết quả huấn luyện phân loại hình thức thanh toán 112**](#_Toc199290384)

**MỤC LỤC BẢNG BIỂU**

[**Bảng 1: Chú giải về các giai đoạn phát triển của Big Data 11**](#_Toc199290715)

[**Bảng 2: Phân công công việc 19**](#_Toc199290716)

[**Bảng 3: Các thuộc tính dữ liệu 21**](#_Toc199290717)

[**Bảng 4: Kết quả truy vấn thứ tám 81**](#_Toc199290718)

[**Bảng 5: Kết quả truy vấn 13 90**](#_Toc199290719)

[**Bảng 6: Kết quả truy vấn cuối cùng 94**](#_Toc199290720)

# **LỜI MỞ ĐẦU**

Nhóm thuộc tập thể lớp chúng em xin chân thành cảm ơn thầy đã trang bị những kiến thức quý báu cho chúng em trong suốt quá trình học tập môn học phần . Đồng thời, chúng em cũng cảm ơn thầy vì đã tận tình hướng dẫn cho chúng em về định hướng thực hành đồ án môn học này.

Chính nhờ công giảng dạy, chỉ bảo chu đáo của thầy mà chúng em được có những kiến thức cơ bản để bước vào chuyên ngành và đặt những bước chân tiếp theo trên con đường học tập, đồng thời có thể vận dụng và sáng tạo ra những sản phẩm hữu ích góp phần phục vụ các lĩnh vực khác nhau của đời sống.

Chúng em xin chân thành cảm ơn thầy đã nhiệt tình hướng dẫn, chỉ bảo dẫn dắt cho chúng em trong suốt quá trình thực hiện đề tài này. Chúng em xin gửi lời cảm ơn đến gia đình, bạn bè, các anh chị tiền bối đã động viên và cung cấp các tài liệu hữu ích cho nhóm để chúng em hoàn thành đề tài một cách chỉnh chu nhất.

Mặc dù đã cố gắng nỗ lực thực hiện đề tài với quyết tâm cao nhưng bài làm chắc chắn không thể tránh khỏi thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp của các thầy cô để bài tập và kiến thức của chúng em ngày càng hoàn thiện hơn và rút kinh nghiệm trong những lần làm đề tài sau ạ.

Chúng em xin chân thành cảm ơn!

Ngày 28 tháng 11 năm 2024

1. **CHƯƠNG I: TỔNG QUAN VỀ DỮ LIỆU LỚN VÀ GIỚI THIỆU ĐỀ TÀI**
2. **Tổng quan về Dữ liệu lớn - Big Data**
3. **Giới thiệu về dữ liệu lớn và sự bùng nổ của dữ liệu lớn**

Mặc dù chỉ mới được biết đến rộng rãi trong một vài năm trở lại đây, nhưng thực tế, nguồn gốc của tập dữ liệu lớn có từ những năm 1960 - 1970, khi thế giới dữ liệu mới bắt đầu từ việc xây dựng những trung tâm dữ liệu đầu tiên và sự phát triển của cơ sở dữ liệu quan hệ.

Khoảng năm 2005, các tổ chức bắt đầu nhận ra tầm quan trọng của khối lượng lớn dữ liệu mà người dùng tạo ra thông qua Facebook, YouTube cũng như nhiều dịch vụ trực tuyến khác. Hadoop (một khung nguồn mở thiết kế riêng để lưu trữ, phân tích tập dữ liệu lớn) đã được phát triển cùng năm đó. Hệ thống quản lý dữ liệu không quan hệ (NoSQL) cũng bắt đầu trở nên phổ biến trong thời gian này.

Sự ra đời của các khung nguồn mở, chẳng hạn như Hadoop (gần đây hơn là Spark) là điều cần thiết cho sự phát triển của dữ liệu lớn. Chúng giúp cho Big data trở nên dễ dàng hơn trong ứng dụng cũng như lưu trữ rẻ.

Trong những năm kể từ đó, khối lượng dữ liệu lớn đã tăng vọt. Người dùng vẫn đang tiếp tục tạo ra lượng dữ liệu khổng lồ hàng ngày. Đồng thời, không chỉ con người mới làm việc đó. Với sự ra đời của Internet vạn vật (IoT), nhiều đối tượng, thiết bị được kết nối với internet, thu thập dữ liệu về mô hình sử dụng của khách hàng và hiệu suất sản phẩm. Sự xuất hiện của Máy học (ML) cũng góp phần tạo ra nhiều dữ liệu hơn.

Mặc dù dữ liệu lớn đã tiến xa nhưng tính hữu dụng của nó chỉ mới bắt đầu. Điện toán đám mây tăng cường phạm vi khả năng của Dữ liệu lớn. Đám mây giúp Big data mở rộng hoặc thu hẹp linh hoạt, nơi nhà phát triển chỉ cần tạo các cụm đặc biệt để kiểm tra một tập hợp con dữ liệu. Và khi cơ sở dữ liệu đồ thị ngày càng trở nên quan trọng, với khả năng hiển thị lượng dữ liệu khổng lồ, nó có thể giúp phân tích nhanh chóng, toàn diện.

Vậy Big Data - Dữ liệu lớn là cái gì? Khái niệm Dữ liệu lớn - Big Data được sử dụng để mô tả khối lượng khổng lồ của cả dữ liệu có cấu trúc và phi cấu trúc, lớn đến mức khó có thể xử lý bằng phương pháp thủ công.

Nói một cách đơn giản, dữ liệu lớn là các tập dữ liệu lớn hơn, phức tạp hơn, đặc biệt là thu thập từ nhiều nguồn dữ liệu mới. Những bộ dữ liệu này đồ sộ đến nỗi phần mềm xử lý dữ liệu truyền thống không thể quản lý chúng. Nhưng khối lượng dữ liệu khổng lồ này có thể được sử dụng để giải quyết các vấn đề thuộc nhiều lĩnh vực khác nhau mà trước đây chưa thể thể giải quyết được.

Khái niệm Big Data hiện nay đã trở nên tương đối quen thuộc. Nó đại diện cho số lượng dữ liệu này càng tăng lên, cũng như những loại dữ liệu đa dạng khác nhau đang được thu thập. Khi ngày càng nhiều thông tin trên thế giới được trao đổi online và số hóa, các nhà phân tích có thể bắt đầu sử dụng những thông tin đó làm dữ liệu. Những thứ như mạng xã hội, sách trực tuyến, âm nhạc, video,... đã làm tăng đáng kể lượng dữ liệu sẵn có để phân tích.

Mọi thứ hiện nay chúng ta thao tác online đều được lưu trữ và theo dõi như dữ liệu.

Ví dụ: Đọc sách trên Kindle sẽ sinh ra dữ liệu về việc chúng ta đang đọc sách gì, khi nào chúng ta đọc, chúng ta đọc trong bao lâu. Tương tự, nghe nhạc sẽ sinh ra dữ liệu về việc chúng ta đang nghe thể loại nhạc gì, chúng ta thường nghe khi nào. Điện thoại thông minh liên tục cập nhật dữ liệu về vị trí, tốc độ di chuyển và các ứng dụng đang hoạt động,...

Do đó, Big data cũng đề cập đến khả năng tận dụng, khai thác thông tin từ kho dữ liệu một cách hợp lý nhằm mang lại nhiều lợi ích thiết thực.

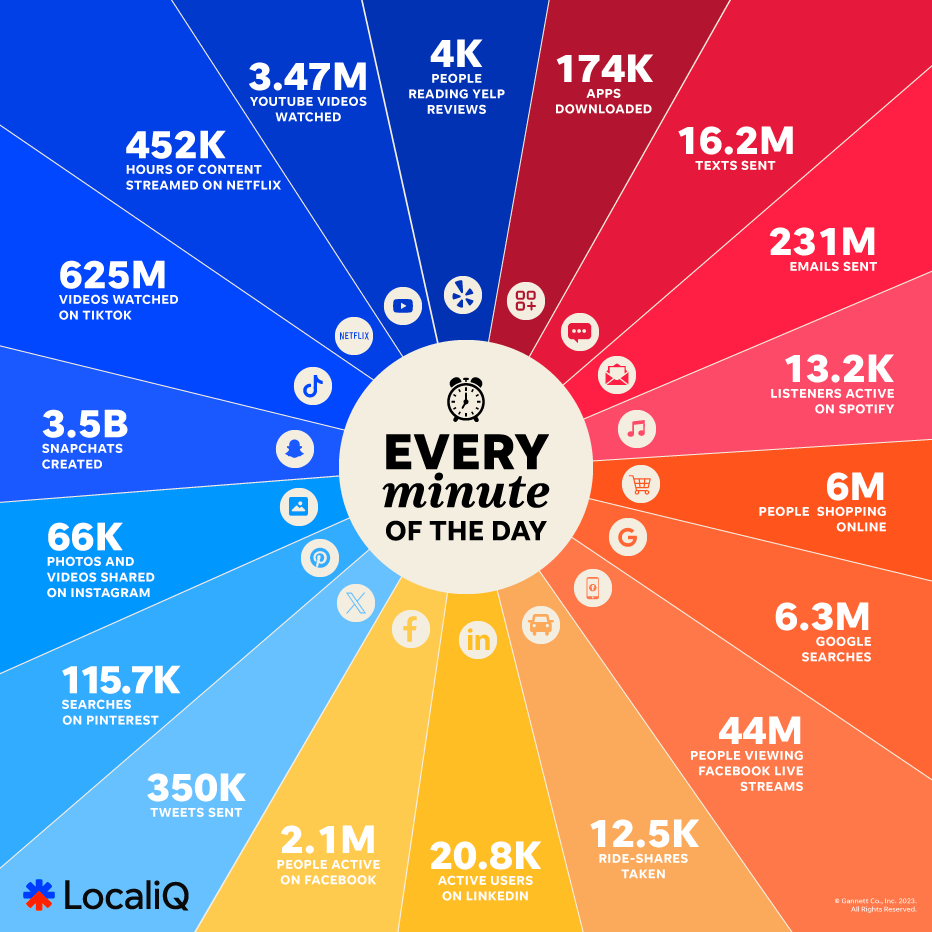
Một điều quan trọng chúng ta cần lưu ý, đó là khái niệm Big Data không chỉ là số lượng dữ liệu mà chúng ta đang tạo ra, nó còn bao gồm tất cả các dạng dữ liệu khác nhau: Text, video, lượt khách hàng ra vào, số giao dịch,…

Dưới đây là sự phát triển của Dữ liệu lớn qua các giai đoạn như sau:

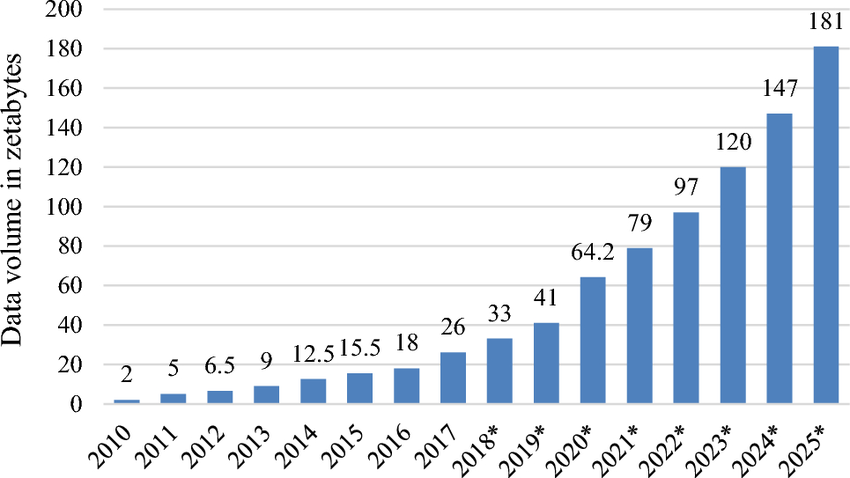
|  |  |  |
| --- | --- | --- |
| **Giai đoạn** | **Thời gian** | **Đặc điểm** |
| Giai đoạn 1 | 1970 – 2000 | Trong giai đoạn này, dữ liệu được lưu trữ, xử lý và khai thác sử dụng hệ quản trị cơ sở dữ liệu như hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Các doanh nghiệp cũng đã bắt đầu sử dụng data warehouse để lưu trữ những dữ liệu từ quá khứ. Phần lớn dữ liệu được lưu trữ, xử lý và phân tích trong giai đoạn này là dữ liệu có cấu trúc – dữ liệu có định dạng chuẩn hoá, có thể khai thác được bằng các câu lệnh truy vấn như SQL. |
| Giai đoạn 2 | 2000 – 2010 | Với sự ra đời của Internet và World Wide Web, dữ liệu không còn bị giới hạn dưới dạng bảng mà được mở rộng ra với rất nhiều hình thức lưu trữ khác nhau: địa chỉ IP, lịch sử duyệt Web, tỷ lệ nhấp chuột, nhật ký tìm kiếm,… Lượng người truy cập web sẽ tỉ lệ thuận với sự gia tăng lưu trữ các loại dữ liệu bán cấu trúc và không cấu trúc kể trên. Vậy nên, để hiểu hơn về khách hàng của mình trong bối cảnh thời đại mới, các doanh nghiệp đã phải tạo ra những cách thức tinh vi hơn để phân tích và trích xuất thông tin hữu ích từ các nguồn dữ liệu phi cấu trúc ngày càng gia tăng. |
| Giai đoạn 3 | 2010 đến hiện tại | Bước sang giai đoạn này, tuy ưu tiên hàng đầu của các doanh nghiệp vẫn là tập trung vào xử lý và phân tích dữ liệu phi cấu trúc trên web nhưng sự ra đời của các thiết bị di động đã tạo ra các xu hướng khai thác dữ liệu mới. Ví dụ, các thiết bị di động cho phép lưu trữ và phân tích dữ liệu không gian-thời gian cũng như dữ liệu hành vi của người dùng dựa trên các mô hình điều hướng web của họ. Các xu hướng mới nhất dựa trên công nghệ cảm biến như Internet vạn vật (IoT), Internet vạn vật công nghiệp (IIoT), Internet y tế vạn vật (IoMT), mạng cảm biến cơ thể (BSN), v.v., đang tạo ra khối lượng dữ liệu khổng lồ với tốc độ chưa từng thấy trước đây |

**Bảng 1: Chú giải về các giai đoạn phát triển của Big Data**

Có thể thấy qua lịch sử phát triển của Big Data, các kiểu dữ liệu mỗi ngày một thêm mới và ngày càng đa dạng. Song song với đó, khối lượng dữ liệu tạo ra sau mỗi năm cũng tăng nhanh, là hệ quả của số lượng dữ liệu đa dạng được tạo ra mỗi ngày. Ước tính đến năm 2025, số lượng dữ liệu sẽ đạt ngưỡng 181 zetabytes.



**Hình 1: Thống kê dữ liệu được tạo ra trong từng phút của năm 2023**

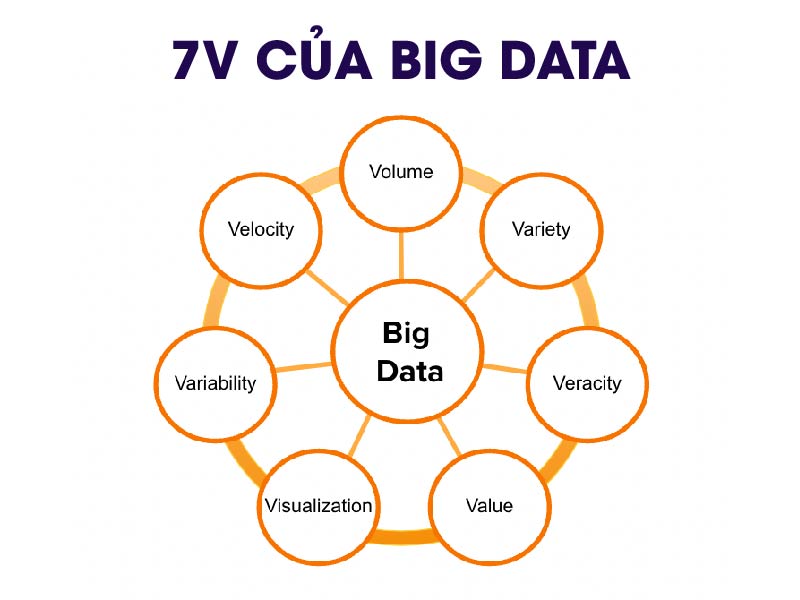


**Hình 2: Thống kê dữ liệu được tạo ra từ năm 2010-2020 và dự đoán cho giai đoạn 2021-2025**

1. **7 "Vs" của Big Data - Dữ liệu lớn**

Big Data là bao gồm 7 đặc điểm, tính chất quan trọng (7 "Vs") như sau:

* ***Khối lượng (Volume):*** Đúng như tên gọi, khối lượng của Big Data là rất lớn, thậm chí là cực lớn. Với sự phát triển mạnh mẽ của internet, thiết bị di động, mạng xã hội và công nghệ IoT (Internet of Things), khối lượng dữ liệu đang tạo ra tăng đáng kể hàng ngày, hàng giờ.
* ***Tốc độ (Velocity):*** Bên cạnh khối lượng dữ liệu lớn, tốc độ xử lý nhanh là điều kiện thiết yếu. Các ứng dụng phổ biến trong lĩnh vực Internet, tài chính, ngân hàng, y tế - chăm sóc sức khỏe,... phần lớn dữ liệu lớn được xử lý real-time (thời gian thực). Công nghệ quản lý dữ liệu lớn ngày một tiên tiến cho phép chúng ta xử lý ngay lập tức trước khi lưu trữ vào cơ sở dữ liệu.
* ***Đa dạng (Variety):*** Trước đây, hầu hết dữ liệu có thể được thu thập gọn gàng trong bảng, đó là dữ liệu có cấu trúc. Hiện nay, dữ liệu chủ yếu xuất hiện dưới dạng phi cấu trúc như bài viết trên mạng xã hội, âm thanh, video,... Công nghệ Big Data cho phép liên kết, phân tích đa dạng chủng loại dữ liệu với nhau.
* ***Độ chính xác (Veracity):*** Với sự đa dạng cùng số lượng lớn, độ chính xác là một trong những tính chất phức tạp nhất của Big Data. Tính chất này đề cập đến việc khai phá chất lượng tập dữ liệu và làm sạch dữ liệu đó một cách hệ thống. Từ đó làm tăng độ tin cậy, chính xác, giúp ích cho việc phân tích. Đây cũng là một bước quan trọng của Big Data.
* ***Giá trị thông tin (Value):*** Giá trị thông tin là tính chất quan trọng nhất của công nghệ Big Data. Ví dụ: Một doanh nghiệp muốn ứng dụng Big Data cho bài toán mô hình hoạt động kinh doanh của mình. Trước tiên, doanh nghiệp cần xác định rõ những giá trị thông tin hữu ích Big Data có thể mang lại trong việc giải quyết nhu cầu của doanh nghiệp.
* ***Tính khả biến (Variability):*** Quản lý và bối cảnh hóa dữ liệu theo cách cung cấp cấu trúc, ngay cả trong môi trường dữ liệu dễ biến đổi và không thể đoán trước.
* ***Hình dung (Visualization):*** Hình dung là rất quan trọng giúp người dùng hiểu rõ thông tin được phân tích, trích xuất từ kho dữ liệu. Sử dụng biểu đồ, đồ thị để trực quan hóa một lượng lớn dữ liệu phức tạp sẽ hiệu quả hơn nhiều trong việc truyền đạt ý nghĩa so với bảng tính, báo cáo chứa đầy các con số và công thức truyền thống.



**Hình 3: 7 đặc điểm, tính chất quan trọng của Big Data - Nguồn Internet**

1. **Tầm quan trọng của việc phân tích và xử lý dữ liệu lớn trong đời sống**

Big Data làm được gì hay tại sao lại cần đến Big Data chắc hẳn là những câu hỏi được nhiều người quan tâm. Nhìn chung, những lợi ích của Big Data bao gồm:

* *Cắt giảm chi phí:* Áp dụng công nghệ phân tích Big Data trên tất cả các quy trình trong tổ chức không chỉ giúp doanh nghiệp tự động loại bỏ sai sót mà còn có thể triển khai các giải pháp nhanh chóng, hiệu quả, tiết kiệm chi phí.
* *Tiết kiệm thời gian:* Việc tổng hợp, phân tích một lượng dữ liệu khổng lồ đòi hỏi nhiều thời gian và chi phí, đồng thời có độ trễ nhất định nếu xử lý thủ công. Big Data có khả năng thu thập, xử lý dữ liệu với tốc độ nhanh hơn, dễ dàng hơn, tiết kiệm thời gian đáng kể.
* *Tối ưu hóa sản phẩm:* Thông qua Big Data, các công ty có thể phân tích được mức giá có lợi nhuận tối đa trong những viễn cảnh kinh doanh khác nhau. Để làm hài lòng khách hàng và thu lại lợi nhuận, các công ty cần có những cải tiến sản phẩm phù hợp theo xu hướng của thị trường. Big Data tạo điều kiện cho họ thực hiện các thay đổi ấy.
* *Hỗ trợ con người đưa ra quyết định:* Nhờ vào khả năng khai thác, xử lý lượng dữ liệu khổng lồ, Big Data có thể xác định nhu cầu, dự đoán xu hướng, giúp cho việc đưa ra quyết định đúng đắn, kịp thời.

Nhờ những lợi ích trên, hiện nay có rất nhiều doanh nghiệp hoạt động trong các lĩnh vực khác nhau đã, đang ứng dụng công nghệ Big Data.

Nếu để ý một chút, chúng ta sẽ thấy khi mua sắm trên eBay, Amazon hoặc những sàn thương mại điện tử, chúng sẽ đưa ra những sản phẩm gợi ý tương tự. Ví dụ khi tìm kiếm áo thun, trang web sẽ tự động gợi ý chúng ta quần, phụ kiện liên quan.

Từ đâu mà những trang web này có thể đưa ra gợi ý như vậy? Đó là dữ liệu của khách hàng thao tác hàng ngày trên những trang web ấy. Doanh nghiệp khai thác hiệu quả Big Data không chỉ giúp tăng lợi nhuận cho chính mình mà còn tăng trải nghiệm mua sắm của người dùng.

Ngoài ra, Big Data cũng có thể được ứng dụng bởi các tổ chức, chính phủ trong việc dự đoán tỷ lệ thất nghiệp, xu hướng nghề nghiệp hiện tại và tương lai để đầu tư cho những hạng mục đó, cắt giảm chi tiêu, kích thích tăng trưởng kinh tế, thậm chí là ra phương án phòng ngừa trước một dịch bệnh nào đó,...

Tại các thành phố lớn trên thế giới, Big Data cũng được tích hợp vào trung tâm điều hành thông minh (IOC), hệ thống giám sát xử phạt,... Nhờ đó, nhà quản lý thành phố dễ dàng giám sát, đưa ra quyết định trong trường hợp khẩn cấp, đời sống của cư dân cũng được đảm bảo an toàn, hiện đại hơn.



**Hình 4: Big data ứng dụng trong các trung tâm giám sát điều hành thông minh các tỉnh - Nguồn Internet**

1. **Ví dụ và thực tiễn ứng dụng Big Data ở Việt Nam**

Big Data còn là một khái niệm tương đối mới mẻ tại Việt Nam, trong khi nhiều nước phát triển trên thế giới đã sử dụng Big Data từ những năm 1960. Một số lĩnh vực đã và đang đưa Dữ liệu lớn vào ứng dụng tại Việt Nam có thể kể đến:

* *Quản lý Nhà nước trực tuyến:* Ở Việt Nam, Big Data đang dần được số hóa nhằm hỗ trợ chính phủ thực hiện công tác quản lý nhà nước trực tuyến. Từ những bước cơ bản như làm Căn cước công dân gắn chip, hệ thống phản ánh hiện trường Social listening,... Chính phủ có thể quản lý một cách hệ thống và sử dụng hiệu quả thông tin của người dân trên toàn lãnh thổ Việt Nam. Big Data phục vụ cho số hóa có thể thay thế cho giấy tờ truyền thống, điển hình như Bảo hiểm Y tế , Bảo hiểm Xã hội điện tử hay sổ liên lạc điện tử,…
* *Thương mại điện tử:* Ngành thương mại điện tử ngày càng chiếm ưu thế hơn các mô hình cũ. Trong tương lai, thương mại điện tử sẽ phát triển, thậm chí hoàn toàn áp đảo thương mại truyền thống. Trong đó, Big Data là một công cụ vô cùng hữu ích. Big Data góp phần số hóa được hành động, thói quen, sở thích cũng như những mối quan tâm của khách hàng đối với doanh nghiệp. Các thông tin này giúp cho nhà cung ứng hàng hóa, dịch vụ tăng lợi nhuận nhờ đánh trúng nhu cầu, xây dựng chính sách phân phối, bán sản phẩm đến tay người tiêu dùng một cách hiệu quả. Đồng thời, doanh nghiệp cũng có thông tin để đưa ra các chiến dịch chăm sóc và nâng cao trải nghiệm khách hàng phù hợp. Mặt khác, bản thân người tiêu dùng có thể tiết kiệm thời gian, yên tâm trong trải nghiệm mua sắm của mình. Ngoài ra, doanh nghiệp còn có thể cải thiện kết quả kinh doanh dựa trên dự báo xu hướng hàng hóa trong tương lai nhờ vào Big Data.
* *Tài chính - Ngân hàng:* Hiện nay, hầu hết tổ chức ngân hàng, dịch vụ tài chính và bảo hiểm đều nỗ lực để áp dụng cách tiếp cận mới theo hướng khai thác dữ liệu nhằm phát triển, đổi mới sản phẩm. Các tổ chức đang thay đổi cách thức khai thác dữ liệu bằng cách thu thập một khối lượng dữ liệu khổng lồ, tiến hành phân tích, thực hiện bước đầu tiên trong quy trình khai thác Big Data. Khi khối lượng khách hàng tăng lên, nó ảnh hưởng đáng kể đến mức độ, khả năng cung cấp dịch vụ của từng tổ chức. Thực tiễn cho thấy việc phân tích dữ liệu đã đơn giản hóa quá trình theo dõi, đánh giá khách hàng tín dụng của ngân hàng và tổ chức tài chính dựa trên khối lượng lớn dữ liệu như hồ sơ cá nhân cũng như các thông tin bảo mật khác. Với sự giúp đỡ của Big data, ngân hàng có thể theo dõi hành vi của khách hàng, xác định nguồn dữ liệu cần thiết để thu thập phục vụ cho việc đưa ra giải pháp. Ngân hàng có khả năng truy cập trực tiếp nguồn thông tin, dữ liệu lịch sử dồi dào liên quan đến thói quen, hành vi chi tiêu, chi tiết về nguồn thu của khách hàng trong một năm, dịch vụ ngân hàng mà khách hàng sử dụng,… Điều này cung cấp cơ sở, cơ hội để các ngân hàng tiếp cận, phân tích dữ liệu sâu hơn. Hệ thống Big Data còn có khả năng tự động phát hiện những giao dịch bất thường. Từ đó, ngân hàng có hướng giải quyết nhanh chóng, đồng thời không tạo ra những lỗ hổng lớn trong ngành. Hệ thống ngân hàng sẽ luôn đảm bảo an ninh nhờ yếu tố Dữ liệu lớn giúp khai phá và áp dụng những tiến bộ về công nghệ. Kết hợp với các công nghệ Internet vạn vật (IoT - Internet of Things), Trí tuệ nhân tạo (AI - Artificial Intelligent), trong tương lai, chắc chắn chúng ta sẽ tiếp tục được chứng kiến sự phát triển và ứng dụng rộng rãi hơn nữa của Big Data - Công nghệ dữ liệu lớn.

1. **Giới thiệu đề tài**

Với sự phát triển ngày càng tiến bộ của Dữ liệu lớn, các công nghệ liên quan đến lưu trữ, xử lý và phân tích dữ liệu lớn liên tục ra đời để đáp ứng nhu cầu trong đời sống xã hội. Chủ đề báo cáo của nhóm em xin phép được giới thiệu hai apache framework nguồn mở nổi tiếng nhất mà những ai đã học qua Dữ liệu lớn đều phải biết, đó là Apache Hadoop và Apache Spark. Mục đích chính của Apache Spark như chúng ta đã biết đó là xử lý dữ liệu theo thời gian thực. Tuy vậy, để xử lý dữ liệu, hệ thống cần dữ liệu đầu vào từ thiết bị lưu trữ. Chính vì thế, Spark sẽ phải sử dụng HDFS, mà HDFS từ đâu ra? Chính xác là từ Apache Hadoop. Những chi tiết cụ thể này nhóm sẽ trình bày ở những nội dung sau. Dù đây không phải là lựa chọn duy nhất, nhưng là lựa chọn phổ biến nhất.

Ý tưởng của chủ đề báo cáo sẽ yêu cầu sẽ có 1 bộ dữ liệu lớn bất kỳ gồm số mẫu tin trên 100.000 mẫu tin, số thuộc tính (features) trên 10. Mình sẽ phải tiến hành cài đặt và cấu hình Apache Hadoop xong rồi tiến hành đẩy và lưu bộ dữ liệu lên HDFS. Xong rồi ta sẽ cài đặt và cấu hình Apache Spark để tiến hành lấy dữ liệu từ trên HDFS xuống và thực hiện truy vấn bộ dữ liệu này qua Spark SQL, xong rồi thực thi 1 bài toán mà nhóm yêu cầu thông qua thư viện Học máy trên SparkMLlib. Nếu được, nhóm sẽ đề xuất những vấn đề có thể áp dụng dữ liệu lớn để giải quyết.

Bài làm của nhóm được tổ chức thành 4 chương:

* Chương 1: Trình bày tổng quan về sự phát triển của Big Data cũng như những lợi ích mà Big Data có thể đem lại cho đời Từ đó xác định hướng đi của đề tài nghiên cứu cũng như đặt ra những mục tiêu cần đạt được.sống xã hội. Giới thiệu về bộ dữ liệu sẽ được sử dụng trong chủ đề tiểu luận này.
* Chương 2: Giới thiệu về Apache Hadoop và thực nghiệm cài đặt, cấu hình Apache Hadoop, rồi đẩy dữ liệu lên HDFS và lưu trữ dữ liệu.
* Chương 3: Giới thiệu về Apache Spark, Spark SQL, SparkMLlib, và thực nghiệm cài đặt, cấu hình Apache Spark, lấy dữ liệu từ HDFS về và thực hiện truy vấn bằng Spark SQL, xong rồi thực hiện yêu cầu xây dựng mô hình Machine Learning thông qua những thư viện trên SparkMLlib và đánh giá mô hình.
* Chương 4: Đề xuất những vấn đề có thể áp dụng dữ liệu lớn để giải quyết và đánh giá kết quả những vấn đề xuyên suốt chủ đề báo cáo tiểu luận.

Về phân công công việc:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Công việc** | **Đánh giá** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Bảng 2: Phân công công việc

1. **Giới thiệu bộ dữ liệu**

Trong báo cáo tiểu luận này, nhóm xin được giới thiệu bộ dữ liệu mang tên là “NYC Yellow Taxi Trip Data”, Đây là dữ liệu chứa thông tin chi tiết về các chuyến đi của taxi vàng tại thành phố New York, Mỹ. Dữ liệu được thu thập và công bố bởi Ủy ban Taxi và Limousine (TLC) của thành phố New York. Mỗi bản ghi trong bộ dữ liệu tương ứng với một chuyến đi của xe taxi vàng (Yellow Cab), bao gồm nhiều trường thông tin như thời gian, địa điểm đón/trả khách, quãng đường di chuyển, chi phí, loại thanh toán, số lượng hành khách,…

Các thuộc tính của những bản ghi này gồm 19 loại thuộc tính. Dưới đây là tất cả thuộc tính của bộ dữ liệu:

|  |  |
| --- | --- |
| **Trường dữ liệu** | **Mô tả** |
| VendorID | Mã số xác định nhà cung cấp TPEP đã cung cấp bản ghi.   1. Creative Mobile Technologies 2. VeriFone Inc. |
| tpep\_pickup\_datetime | Ngày và giờ khi đồng hồ tính cước được kích hoạt. |
| tpep\_dropoff\_datetime | Ngày và giờ khi đồng hồ tính cước được tắt. |
| Passenger\_count | Số lượng hành khách trong xe. Đây là giá trị do tài xế nhập. |
| Trip\_distance | Quãng đường chuyến đi đã thực hiện (tính bằng dặm) được báo cáo bởi đồng hồ tính cước. |
| Pickup\_longitude | Kinh độ tại vị trí đồng hồ tính cước được kích hoạt. |
| Pickup\_latitude | Vĩ độ tại vị trí đồng hồ tính cước được kích hoạt. |
| RateCodeID | Mã giá cuối cùng được áp dụng tại thời điểm kết thúc chuyến đi.   1. Giá tiêu chuẩn 2. JFK 3. Newark 4. Nassau hoặc Westchester 5. Giá thỏa thuận 6. Đi ghép nhóm |
| Store\_and\_fwd\_flag | Cờ này cho biết liệu bản ghi chuyến đi có được lưu trữ trong bộ nhớ của xe trước khi gửi cho nhà cung cấp hay không, còn gọi là “lưu và chuyển tiếp”, vì phương tiện không có kết nối tới máy chủ. Y = chuyến đi được lưu và chuyển tiếp  N = chuyến đi không được lưu và chuyển tiếp |
| Dropoff\_longitude | Kinh độ tại vị trí đồng hồ tính cước được tắt. |
| Dropoff\_latitude | Vĩ độ tại vị trí đồng hồ tính cước được tắt. |
| Payment\_type | Mã số cho biết hành khách đã thanh toán chuyến đi bằng hình thức nào.   1. Thẻ tín dụng 2. Tiền mặt 3. Không tính phí 4. Khiếu nại 5. Không rõ 6. Hủy chuyến |
| Fare\_amount | Tiền cước được tính dựa trên thời gian và quãng đường bởi đồng hồ tính cước. |
| Extra | Các khoản phụ phí và phí bổ sung khác nhau. Hiện tại, chỉ bao gồm phụ phí giờ cao điểm $0.50 và $1, cùng các khoản phụ phí qua đêm. |
| MTA\_tax | Thuế MTA trị giá $0.50 được tự động áp dụng dựa trên loại giá cước theo đồng hồ đang sử dụng. |
| Improvement\_surcharge | Phụ phí cải thiện 0,30 đô la được tính cho mỗi chuyến đi khi xe bắt đầu chạy. Khoản phụ phí này bắt đầu được áp dụng từ năm 2015. |
| Tip\_amount | Số tiền boa – Trường này chỉ tự động được điền cho các khoản boa bằng thẻ tín dụng. Tiền boa bằng tiền mặt không được tính vào. |
| Tolls\_amount | Tổng số tiền phí cầu đường phải trả trong chuyến đi. |
| Total\_amount | Tổng số tiền mà hành khách phải thanh toán. Không bao gồm tiền boa bằng tiền mặt. |

Bảng 3: Các thuộc tính dữ liệu

Về ứng dụng của bộ dữ liệu thì bộ dữ liệu này thường được sử dụng trong các bài toán phân tích dữ liệu lớn, học máy (machine learning), dự báo nhu cầu taxi theo thời gian, phân tích hành vi khách hàng, tối ưu hóa vận hành dịch vụ taxi, phân tích địa lý và các dự án liên quan đến đô thị thông minh.

Trong chủ đề tiều luận này, nhóm sẽ sử dụng bộ dữ liệu thông tin chi tiết về các chuyến đi taxi vàng trong thành phố New York được ghi lại và thu thập trong tháng 1 năm 2016. Dữ liệu được ghi dưới dạng file csv và có kích thước rất khổng lồ (1.71 GB, dung lượng thực tế là 1.59 GB), rất phù hợp cho bài toán lưu trữ, xử lý và phân tích dữ liệu lớn. Dữ liệu này bao gồm 19 thuộc tính dữ liệu và 10.713.835 bản ghi, mỗi bản ghi sẽ là tất cả thông tin được điền ứng với thuộc tính dữ liệu trên (có thể sẽ có 1 số dữ liệu gây nhiễu cần được lọc ra)

Mọi người có thể tải bộ dữ liệu này ở trên Kaggle: <https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data?select=yellow_tripdata_2016-01.csv>

1. **APACHE HADOOP**
2. **Tổng quan về Apache Hadoop**
3. Giới thiệu

Apache đã phát triển hệ thống mã mở Hadoop cùng với nhân là khung phần mềm MapReduce. Hadoop chạy trên môi trường Unix và được thiết kế để hỗ trợ các ứng dụng sử dụng được số lượng lớn dữ liệu cấu trúc và phi cấu trúc. Hệ thống phần mềm mã mở này được tối ưu cho tính tin cậy, khả năng mở rộng và tính toán phân tán.

Không giống như các hệ quản trị cơ sở dữ liệu truyền thống, Hadoop được thiết kế để làm việc với nhiều loại dữ liệu và dữ liệu nguồn. Công nghệ HDFS của Hadoop cho phép khối lượng lớn công việc được chia thành các khối dữ liệu nhỏ hơn được nhân rộng và phân phối trên các phần cứng của một cluster để xử lý nhanh hơn. Công nghệ này đã được sử dụng rộng rãi bởi một số trang web lớn nhất thế giới, chẳng hạn như Facebook, eBay, Amazon, Baidu, và Yahoo. Các nhà quan sát nhấn mạnh rằng Yahoo là một trong những nhà đóng góp lớn nhất đối với Hadoop.

1. **Mục tiêu chính**

Mục tiêu chính của Hadoop là:

- Lưu trữ dữ liệu khả mở, tiết kiệm chi phí:

+ Xử lý dữ liệu phân tán với mô hình lập trình đơn giản, thân thiện hơn như MapReduce

+ Hadoop thiết kế để mở rộng thông qua kỹ thuật scale-out, tăng số lượng máy chủ

+ Thiết kế để vận hành trên phần cứng phổ thông, có khả năng chống chịu lỗi phần cứng

- Xử lý dữ liệu mạnh mẽ

+ Thách thức: các thiết bị lưu trữ tốc độ chậm, máy tính thiếu tin cậy, lập trình song song phân tán không dễ dàng.

1. **Ưu điểm**

Thư viện phần mềm Apache Hadoop là một framework cho phép các xử lý phân tán các tập dữ liệu lớn trên các cụm nhiều máy tính sử dụng sử dụng mô hình lập trình đơn giản. Nó được thiết kế để mở rộng từ một vài máy chủ đơn đến hàng nghìn máy tính, mà mỗi máy đều cung cấp một đơn vị lưu trữ và tính toán. Thay vì dựa trên các máy tính có độ tin cậy cao, thì cái thư viện được thiết kế để tự nó phát hiện và xử lý lỗi ở tầng ứng dụng, do đó nó dung cấp một dịch vụ có độ tin cậy cao trên cụm nhiều máy tính mà mỗi máy đều có thể bị lỗi.

1. **Các thành phần trong hệ sinh thái của Apache Hadoop**

****

**Hình 5: Các thành phần trong hệ sinh thái Apache Hadoop – Nguồn Internet**

*Lõi (core) Hadoop*: bao gồm một tập hợp của các thành phần và các giao diện cung cấp chức năng truy cập vào các hệ thống tập tin phân tán và vào ra tổng quát. Các thành phần cốt lõi cũng cung cấp sự tối ưu hóa dựa vào phân nhóm địa lý của máy chủ nhằm giảm thiểu lưu lượng mạng giữa các máy chủ trong các cụm tính toán.

*Hệ thống tập tin phân tán* (HDFS - Hadoop Distribute File System): là hệ thống lưu trữ chính được sử dụng bởi các ứng dụng Hadoop. HDFS, như tên gọi của nó, một hệ thống tập tin phân tán cung cấp truy cập thông lượng cao vào dữ liệu của ứng dụng, tạo ra nhiều bản sao của khối dữ liệu và phân phối chúng trên các nút tính toán trong một cụm để cho phép tính toán song song, đáng tin cậy và nhanh chóng.

*MapReduce Framework*: là một mô hình lập trình và khung phát triển phần mềm cho phép viết các ứng dụng nhanh chóng xử lý một lượng lớn dữ liệu song song dựa vào một cụm lớn các máy tính gọi là các nút tính toán (node) dựa trên kiến trúc của nền tảng Hadoop. MapReduce sử dụng HDFS để truy cập vào các khối (phân đoạn tập tin) và lưu trữ kết quả rút gọn.

*Apache Pig*: là ngôn ngữ xử lí dòng dữ liệu. Apache Pig là một nền tảng cho việc phân tích dữ liệu lớn bao gồm một ngôn ngữ cấp cao để diễn tả các chương trình phân tích dữ liệu. Đặc điểm chính của chương trình Pig là cấu trúc của chúng có thể được song song hóa cho phép nó xử lý các tập hợp dữ liệu rất lớn, cú pháp đơn giản. Các tính năng xây dựng sẵn (built-in functionality) cung cấp một mức độ trừu trượng để cho phát triển các công việc Hadoop nhanh hơn và dễ dàng hơn để viết hơn so với MapReduce truyền thống.

*Apache Hive*: là một kho dữ liệu cơ sở hạ tầng được xây dựng trên Hadoop. Hive cung cấp các công cụ để cho phép tóm tắt dữ liệu, truy vấn không chuẩn (ad-hoc) và phân tích các bộ dữ liệu lớn được lưu trữ trong các tập tin Hadoop. Nó cung cấp một cơ chế để định cấu trúc cho loại dữ liệu này và cung cấp một ngôn ngữ truy vấn đơn giản gọi là Hive QL, dựa SQL, cho phép người sử dụng quen thuộc với SQL để truy vấn dữ liệu này.

*Apache Hbase*: là một cơ sở dữ liệu phân tán theo cột. HBase sử dụng HDFS cho việc lưu trữ cơ bản của nó. Nó ánh xạ dữ liệu HDFS vào một cơ sở dữ liệu có cấu trúc giống và cung cấp các giao diện lập trình được cho Java (Java API) truy cập vào CSDL này. Nó hỗ trợ hàng loạt kiểu tính toán sử dụng các truy vấn MapReduce và đọc ngẫu nhiên. HBase thường được sử dụng trong Hadoop khi có truy cập đọc/ ghi ngẫu nhiên, thời gian thực. Mục tiêu của nó là lưu trữ các bảng rất lớn đang chạy trên cụm thiết bị phần cứng.

*Apache Sqoop:* là một công cụ được thiết kế để hỗ trợ xuất và nhập dữ liệu hàng loạt vào HDFS từ các kho dữ liệu có cấu trúc như cơ sở dữ liệu quan hệ, kho dữ liệu doanh nghiệp và hệ thống NoSQL. Nó là một công cụ di chuyển dữ liệu dựa trên kiến ​​trúc trình kết nối hỗ trợ các plugin để cung cấp kết nối với các hệ thống bên ngoài mới.Hỗ trợ import tất cả các bảng, một bảng hay 1 phần của bảng vào HDFS. Thông qua Map only hoặc MapReduce job, kết quả là 1 thư mục trong HDFS chứ các tập tin văn bản phân tách các trường theo ký tự phân tách (vd. , hoặc \t).Hỗ trợ export dữ liệu ngược trở lại từ Hadoop ra bên ngoài.

*Apache Kafka*: là hệ thống message pub/sub phân tán (distributed messaging system). Bên pulbic dữ liệu được gọi là producer, bên subscribe nhận dữ liệu theo topic được gọi là consumer. Kafka có khả năng truyền một lượng lớn message theo thời gian thực, trong trường hợp bên nhận chưa nhận message vẫn được lưu trữ sao lưu trên một hàng đợi và cả trên ổ đĩa bảo đảm an toàn. Đồng thời nó cũng được replicate trong cluster giúp phòng tránh mất dữ liệu.

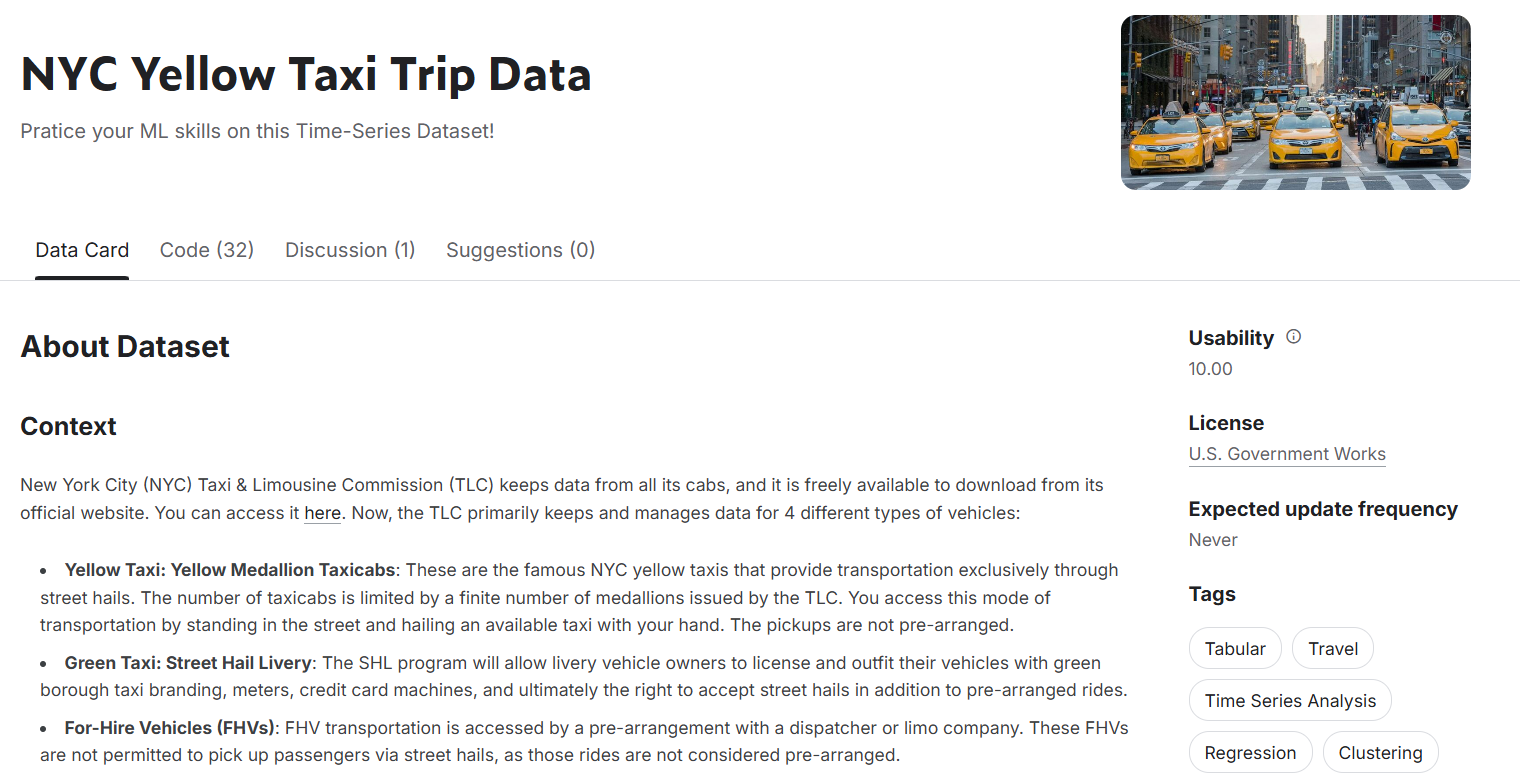
*Apache Oozie*: Oozie là một hệ thống lập lịch luồng công việc để quản lý các công việc thực thi trên cụm Hadoop. Luồng workflow của Oozie là đồ thị vòng có hướng (Directed Acyclical Graphs (DAGs)) của các khối công việc. Oozie hỗ trợ đa dạng các loại công việc như: thực thi MapReduce jobs, thực thi Pig hay Hive scripts, thực thi các chương trình Java hoặc Shell, tương tác với dữ liệu trên HDFS, chạy chương trình từ xa qua SSH, gửi nhận email

*Apache Zookeeper*: là một công cụ cấu hình cụm (cluster) và quản lý sự dãy hóa (serialization) rất hữu ích để xây dựng các cụm lớn các nút của Hadoop, dịch vụ hiệu năng cao cho các ứng dụng phân tán. Nó tập trung vào các dịch vụ như quản lí thông tin cấu hình, đặt tên, đồng bộ hóa phân tán cũng và các dịch vụ nhóm.

*YARN – Yet Another Resource Negotiator*: là một framework hỗ trợ phát triển ứng dụng phân tán YARN cung cấp daemons và APIs cần thiết cho việc phát triển ứng dụng phân tán, đồng thời xử lý và lập lịch sử dụng tài nguyên tính toán (CPU hay memory) cũng như giám sát quá trình thực thi các ứng dụng đó. YARN đóng vai trò cấp phát lượng tài nguyên phù hợp cho các ứng dụng khi có yêu cầu. YARN được đưa ra từ Hadoop 2.0, cho phép MapReduce và non MapReduce cùng chạy trên 1 cụm Hadoop. Với MapReduce job, vai trò của job tracker được thực hiện bởi application tracker.

1. **Thực nghiệm**

* Bước 1: Chuẩn bị công cụ
* Bộ dữ liệu “NYC Yellow Taxi Trip Data”, mình sẽ vào đường dẫn Kaggle và tải xuống file “yellow\_tripdata\_2016-01.csv”

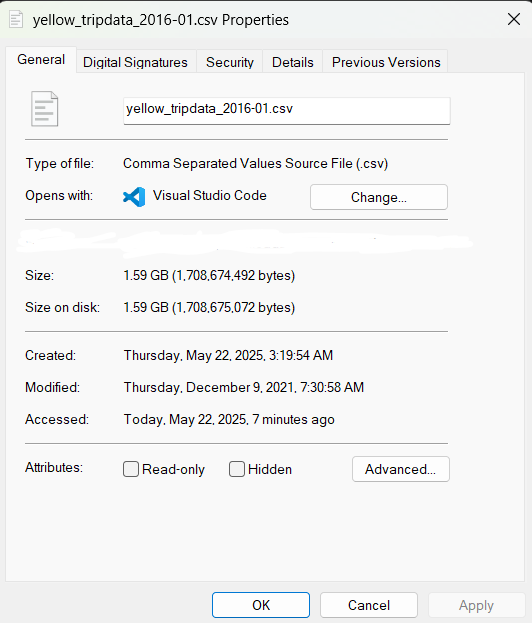


**Hình 6: Thông tin về dữ liệu**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

**Hình 7: Cách tải bộ dữ liệu về**



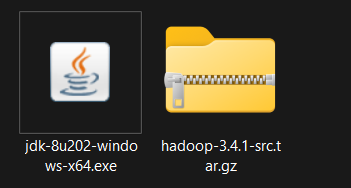
**Hình 8: Thông tin về bộ dữ liệu sau khi tải về**

* Java SE, là phiên bản Long-Term Support của nền tảng Java, cung cấp môi trường phát triển ổn định và hiệu quả. Hadoop được xây dựng bằng Java, do đó, Java SE là thành phần thiết yếu để chạy và cấu hình hệ thống Hadoop. Mọi người có thể tải ở link dưới đây:

<https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html>

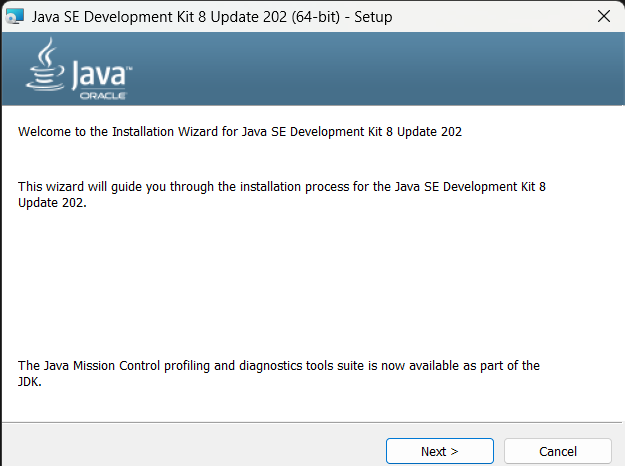
* Apache Hadoop, nền tảng xử lý dữ liệu phân tán, Nó cung cấp các tính năng mạnh mẽ như HDFS (Hadoop Distributed File System) và MapReduce, hỗ trợ lưu trữ và xử lý dữ liệu lớn với hiệu quả cao, đồng thời cải thiện khả năng mở rộng và sẵn sàng. Mọi người có thể tải ở link dưới đây:

<https://dlcdn.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1-src.tar.gz>

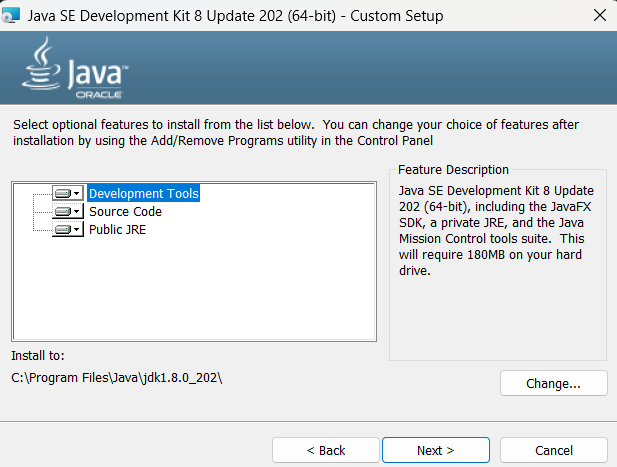


**Hình 9: Hai file Java SE với Hadoop sau khi tải về**

* Bước 2: Cài đặt và cấu hình Java SE
* Nhấn vào file cài đặt Java SE đã tải xuống và sẽ hiện giao diện cài đặt như sau:



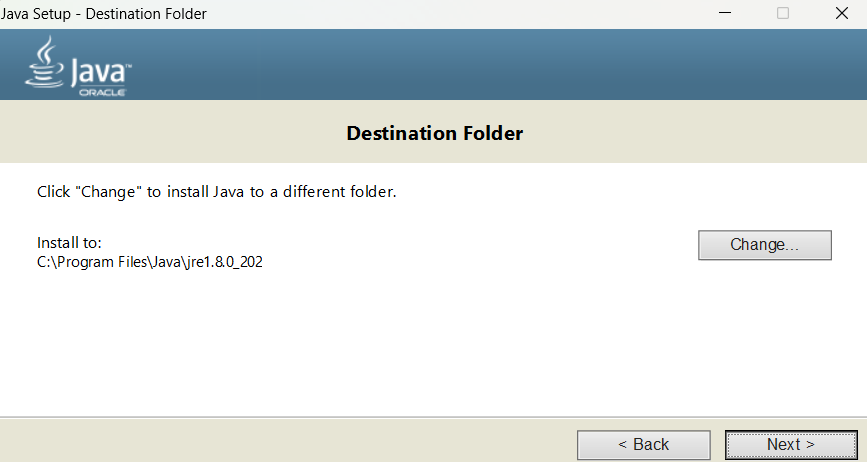
**Hình 10: Giao diện cài đặt của Java SE, nhấn Next để tiếp tục**



**Hình 11: Chọn Development Tools và nhấn Next**



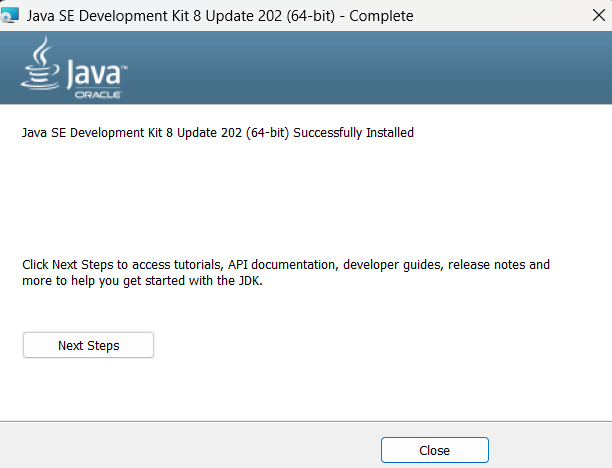
**Hình 12: Chờ đợi**



**Hình 13: Sau khi cài xong sẽ có một setup khác hiện lên, ta nhấn Next 1 lần nữa**

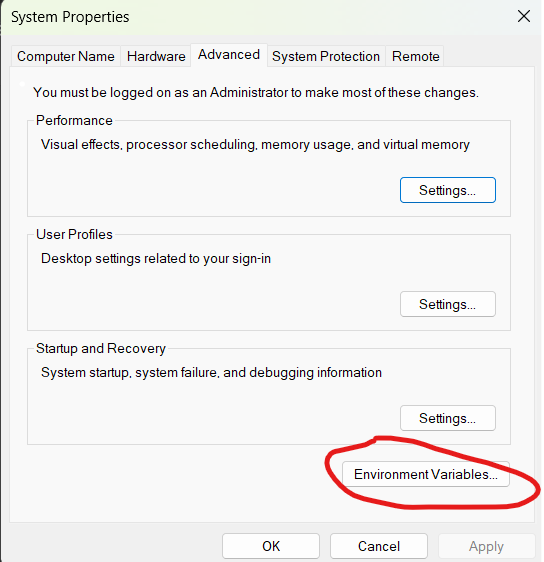


**Hình 14: Vẫn tiếp tục chờ đợi**



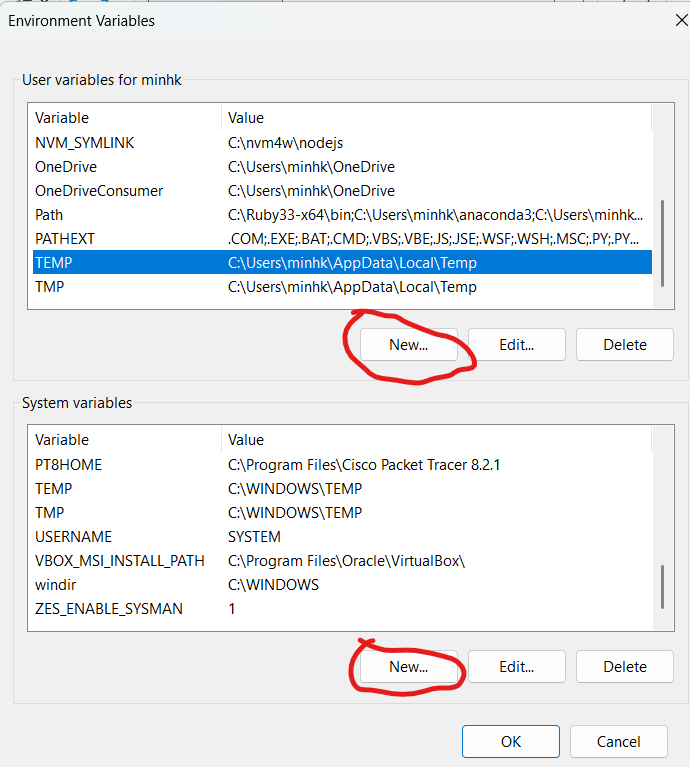
**Hình 15: Xong rồi, đến lượt đi cấu hình thôi**

* Sau khi cài đặt xong, ta tiến hành thiết lập môi trường cho Java SE, Bấm vào biểu tượng Window trên bàn phím và gõ: Edit the system environment variables, sau đó nhấn vào và sẽ có một hội thoại mở ra.

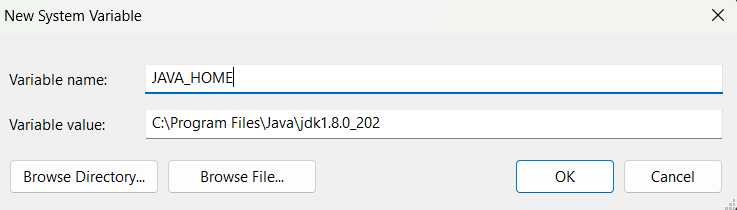


**Hình 16: Đường dẫn tới cài đặt môi trường**

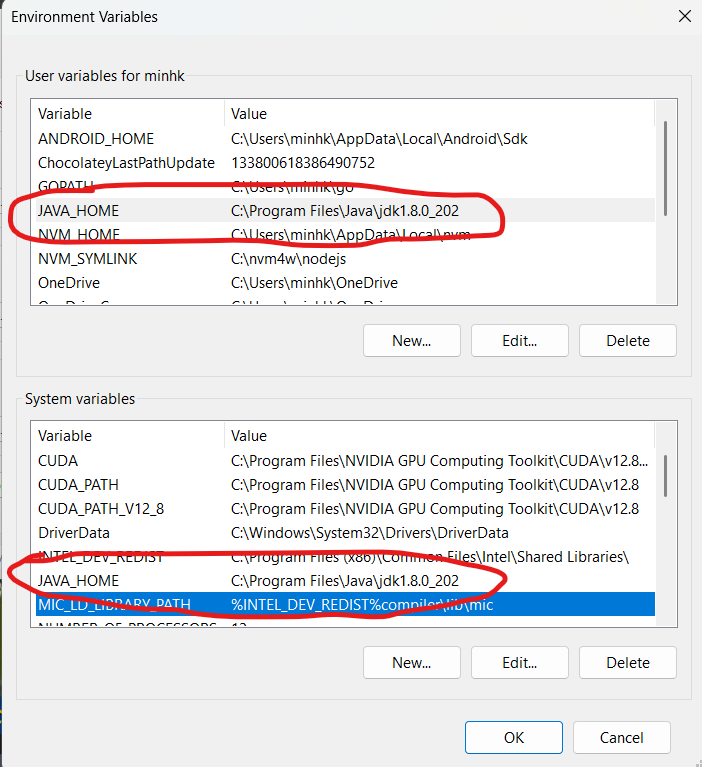
* Cấu hình JAVA\_HOME với đường dẫn JDK đã cài đặt, ta sẽ cấu hình đường dẫn (Environment Variables) để hệ điều hành nhận biết và cho phép bạn chạy các lệnh từ bất kỳ vị trí nào trong Command Prompt hoặc PowerShell mà không cần phải gõ đầy đủ đường dẫn.



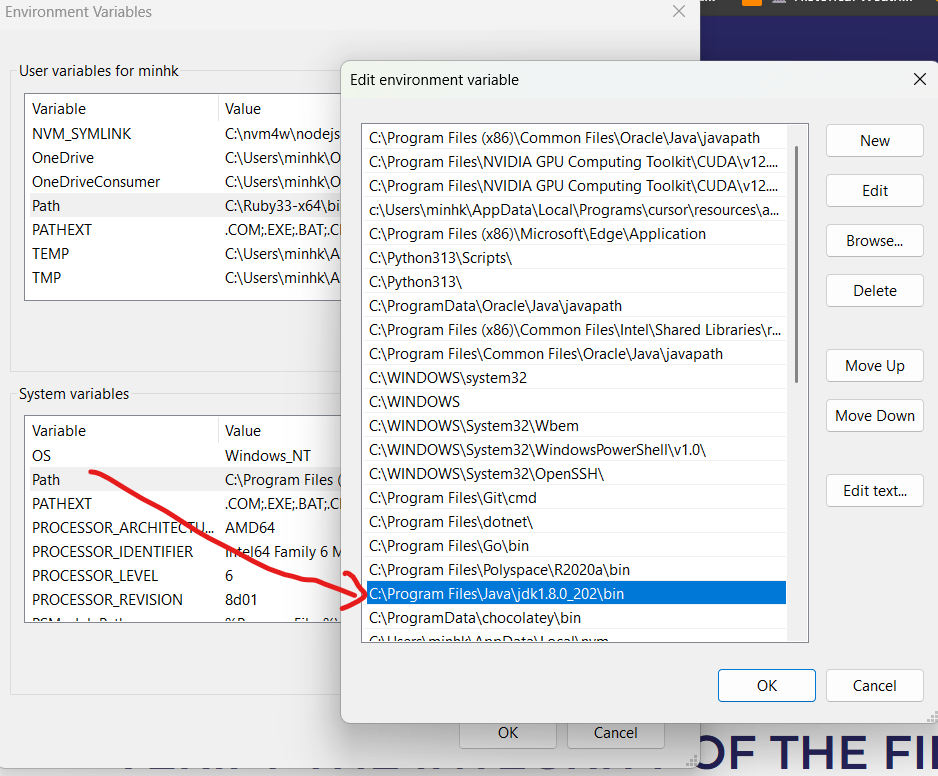
**Hình 17: Ở mỗi đường dẫn user và hệ thống, ta sẽ nhấn New (Thêm đường dẫn mới vào)**



**Hình 18: Nhập đường dẫn JAVA\_HOME vào**

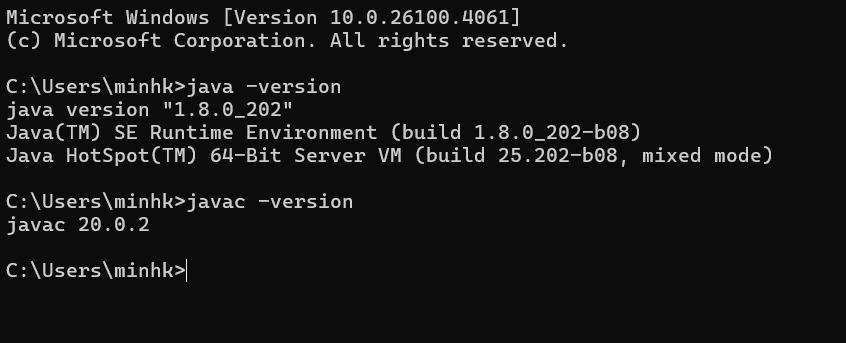


**Hình 19: Sau khi thêm vào, JAVA\_HOME sẽ hiển thị như này**



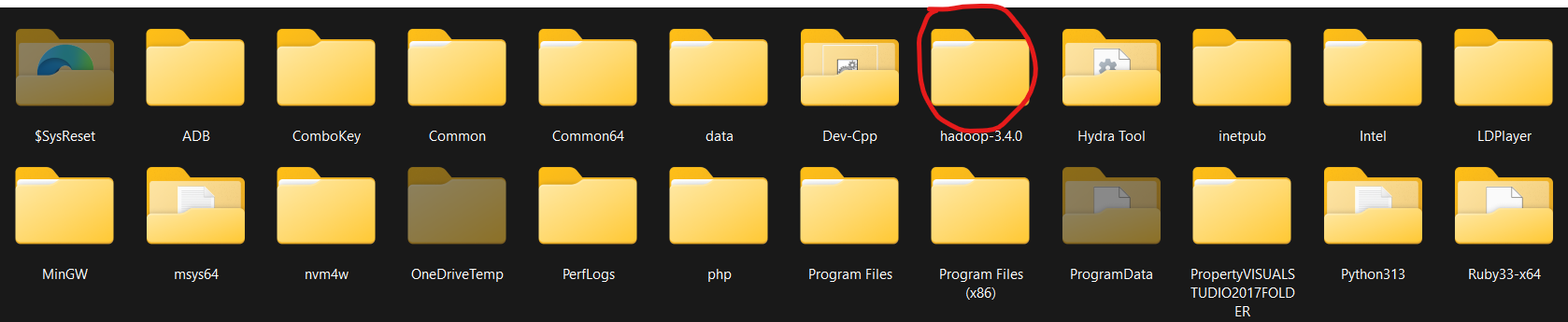
**Hình 20: Ở phần PATH của đường dẫn hệ thống, ta sẽ nhấn cú đúp vào để hiện mục danh sách đường dẫn như trên, xong rồi thêm đường dẫn như trên hình**

* Sau khi cấu hình xong, ta sẽ kiểm tra lại xem để xem Java SE đã được cấu hình hay chưa, mở Command Prompt (CMD) lên và nhập lệnh những lệnh sau đây.
* java -version : Lệnh này hiển thị phiên bản của Java Runtime Environment (JRE) đang được sử dụng. Nó cho biết phiên bản JVM (Java Virtual Machine) được cài đặt và hoạt động trên máy.
* javac -version : Lệnh này hiển thị phiên bản của Java Compiler (javac), công cụ được sử dụng để biên dịch mã nguồn Java thành bytecode (file .class).



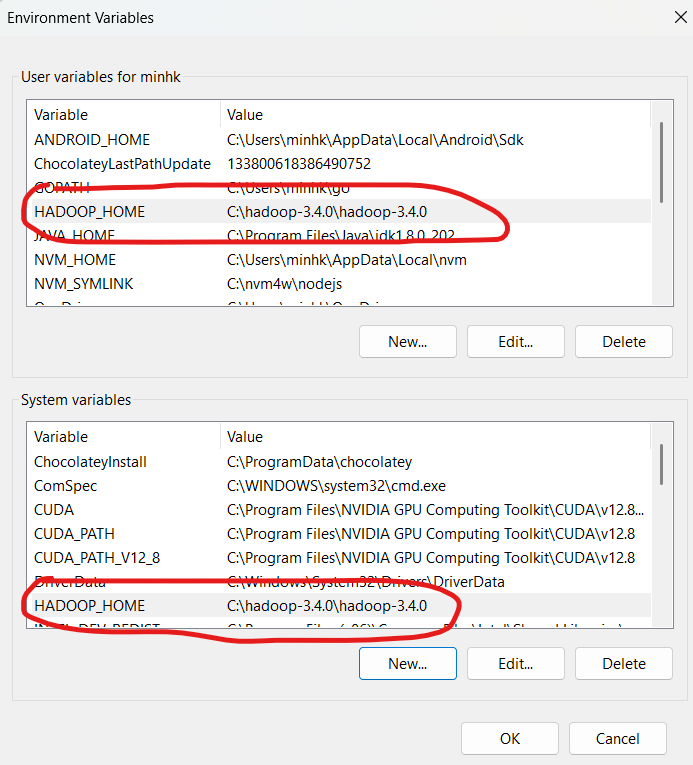
**Hình 21: Nếu mà hiện kết quả như này là ta đã cấu hình Java SE thành công**

* Bước 3: Tiếp theo, ta sẽ tiến hành cài đặt và cấu hình Hadoop
* Đầu tiên, ta sẽ giải nén thư mục Hadoop và để nó vào trong ổ C:

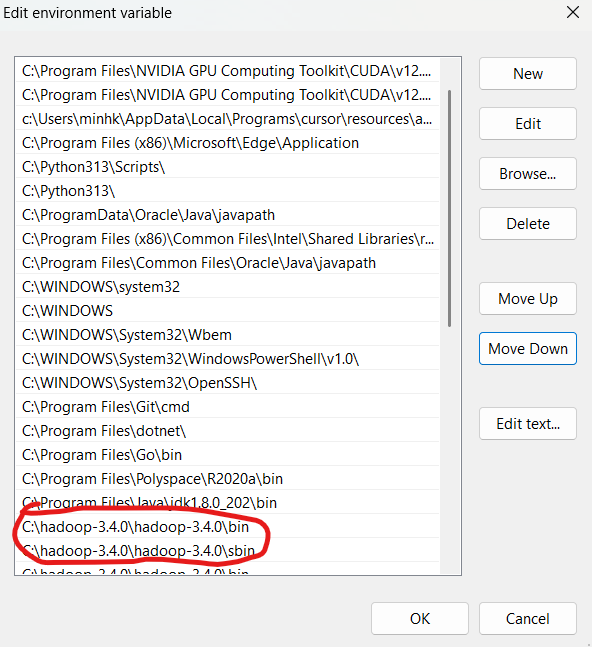


**Hình 22: Thư mục Hadoop trong ổ C:**

* Tương tự với Java SE, ta sẽ thiết lập cấu hình đường dẫn cho Hadoop, dưới tên gọi HADOOP\_HOME

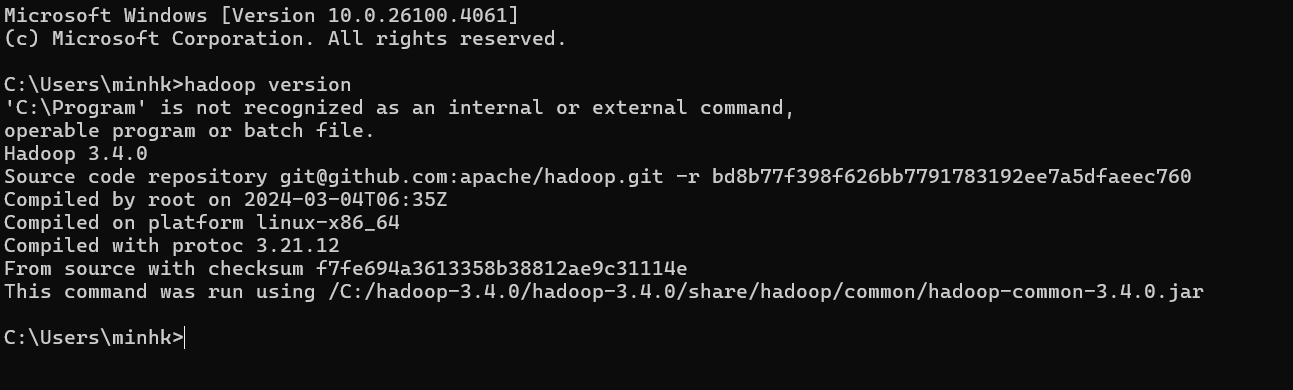


**Hình 23: Ở mục thiết lập môi trường, ta thêm HADOOP\_HOME vào thôi**



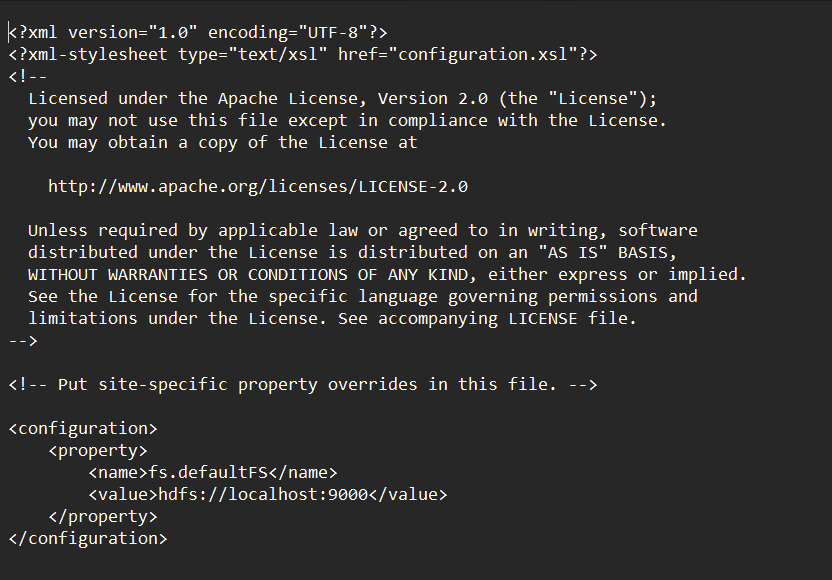
**Hình 24: Ở mục PATH trong đường dẫn hệ thống, ta sẽ thêm đường dẫn bin và sbin của Hadoop vào**

* Với thư mục bin, nó chứa các tập lệnh (scripts) và công cụ dòng lệnh (command-line tools) để tương tác với Hadoop. Những lệnh phổ biến bạn có thể chạy từ thư mục này là: Hadoop - Chạy các công cụ Hadoop như HDFS (Hadoop Distributed File System) và YARN. HDFS - Dùng để quản lý HDFS, ví dụ tạo, đọc, xóa các file trong HDFS. YARN - Quản lý các tài nguyên của cluster và điều phối các ứng dụng. Với mục đích dành cho các thao tác chung liên quan đến Hadoop, thường được sử dụng bởi người dùng hoặc quản trị viên để tương tác với hệ thống
* Còn thư mục sbin, nó chứa các tập lệnh khởi động và quản lý các dịch vụ Hadoop trên một cluster. Ví dụ: start-dfs.sh - Khởi động dịch vụ HDFS, stop-dfs.sh - Dừng dịch vụ HDFS, start-yarn.sh - Khởi động dịch vụ YARN, stop-yarn.sh - Dừng dịch vụ YARN, mr-jobhistory-daemon.sh - Quản lý JobHistory cho MapReduce. Với mục đích là dùng để khởi động, dừng hoặc quản lý các thành phần cốt lõi của Hadoop.
* Sau khi thiết lập cấu hình Hadoop xong, mở CMD lên và nhập hadoop version lên để kiểm tra



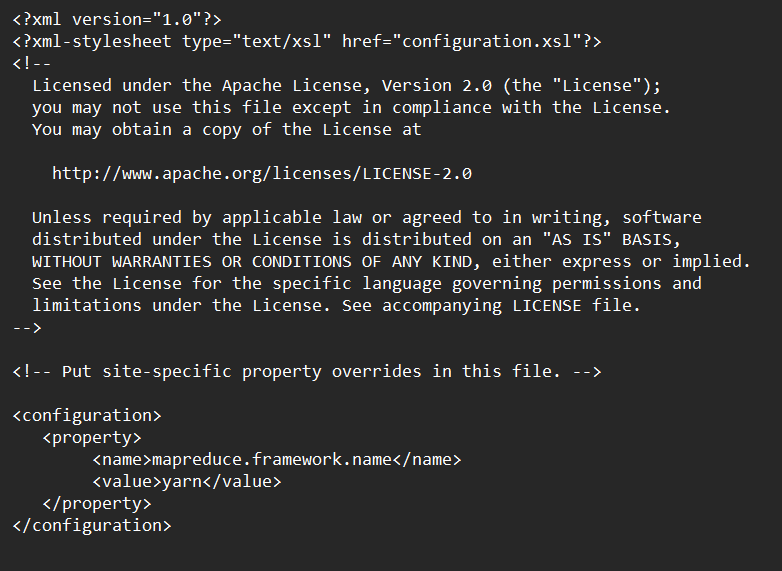
Hình : Khi nó hiện như này, có nghĩa là mình đã cấu hình thành công

* Tiếp theo, ta sẽ tiến hành vào thư mục C:/hadoop/etc/hadoop để lần lượt chỉnh sửa những file này như sau:



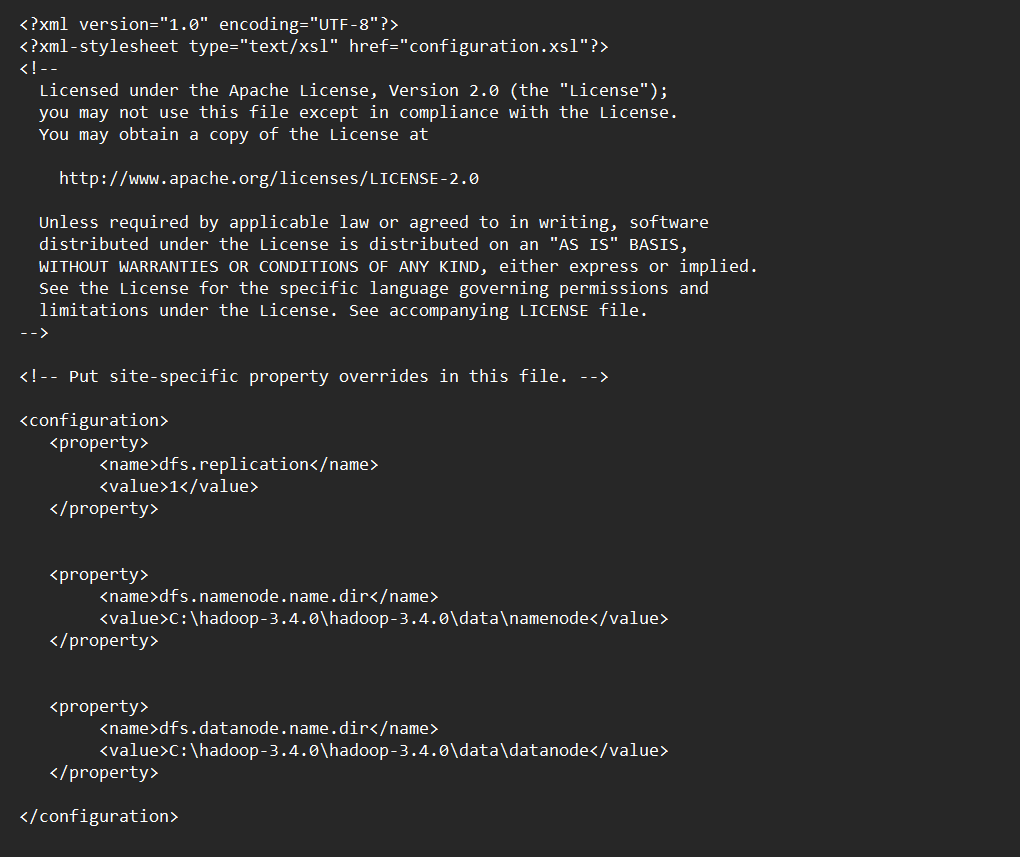
**Hình 26: core-site.xml**

* core-site.xml: gồm có fs.defaultFS - là thuộc tính định nghĩa hệ thống tệp mặc định mà Hadoop sử dụng để lưu trữ và truy cập dữ liệu. Còn giá trị hdfs://localhost:9000 chỉ ra rằng hệ thống tệp mặc định là HDFS (Hadoop Distributed File System) và nó đang chạy trên máy cục bộ (localhost) với cổng 9000.



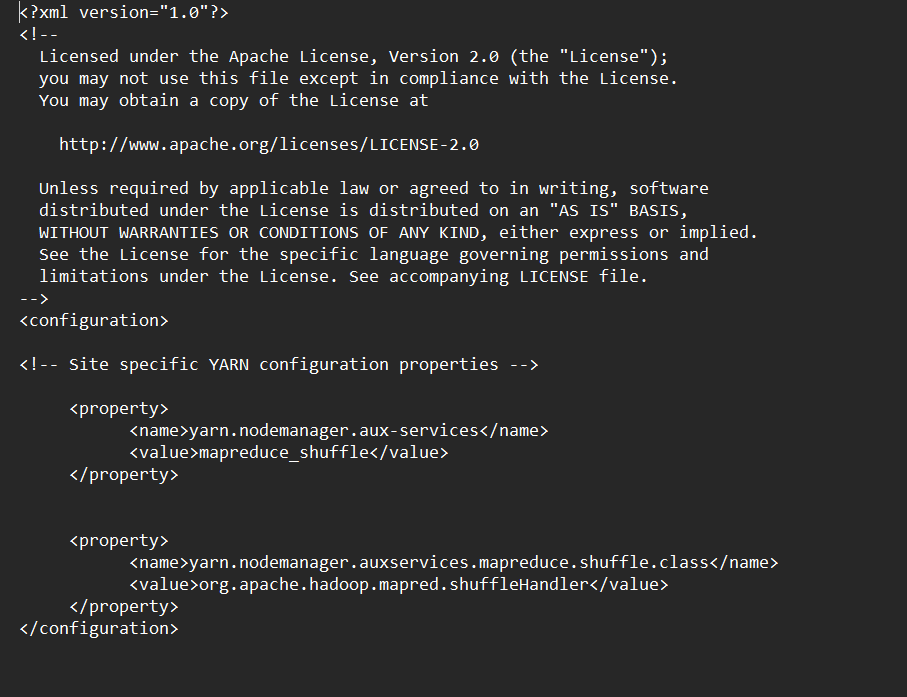
Hình : mapred-site.xml

* mapred-site.xml: có mapreduce.framework.name - thuộc tính này chỉ định framework thực thi các tác vụ MapReduce. Giá trị yarn chỉ ra rằng Hadoop sẽ sử dụng YARN (Yet Another Resource Negotiator) làm framework quản lý tài nguyên và thực thi MapReduce.



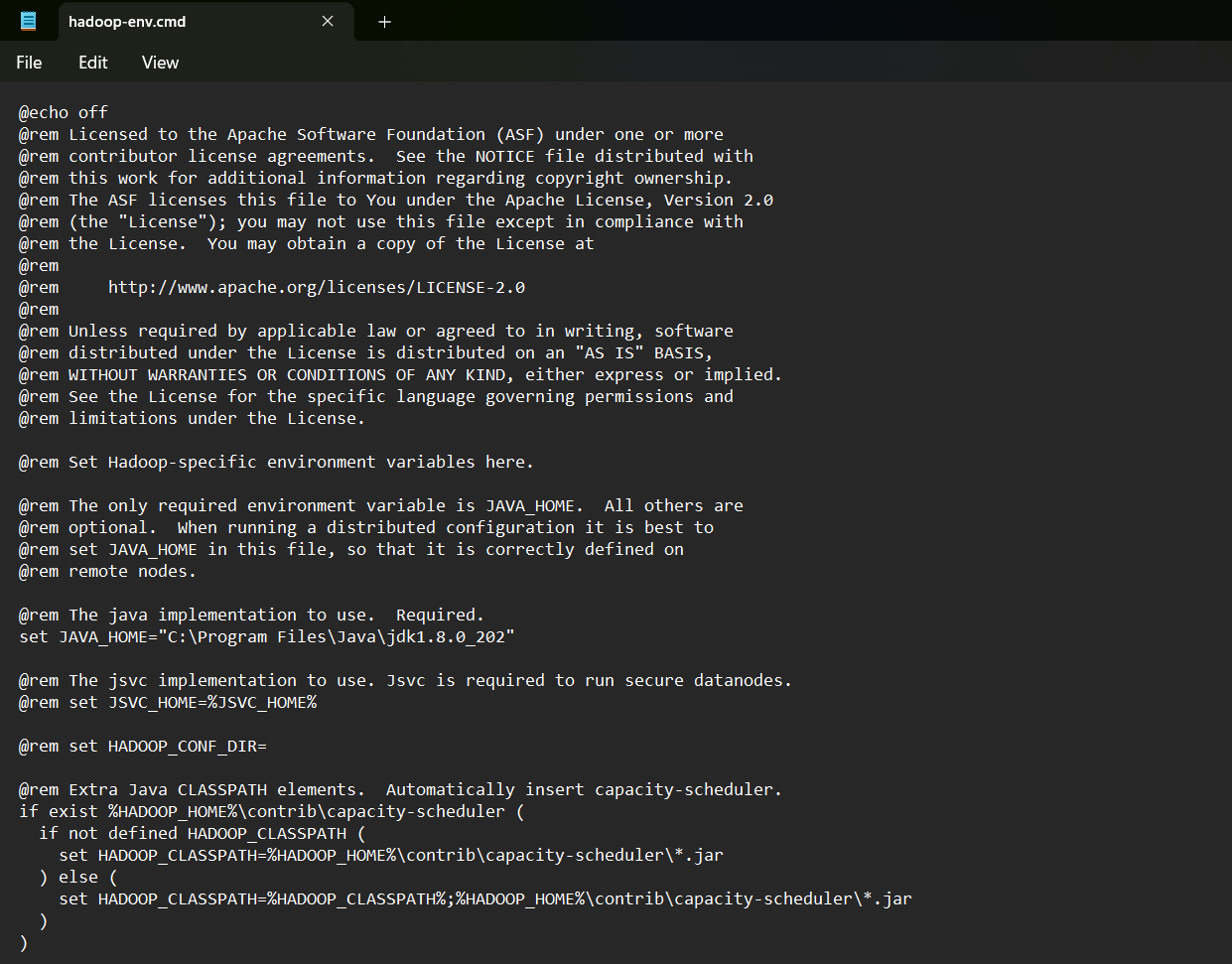
Hình : hdfs-site.xml

* hdfs-site.xml: Trước hết là tạo thư mục “data” trong “C:\hadoop”, xong rồi tạo thư mục con “datanode” trong “C:\hadoop\data”, cuối cùng tạo thư mục con “namenode” trong “C:\hadoop\data”. dfs.replication gồm số giá trị là 1 và đây là số bản sao (replicas) của mỗi khối dữ liệu được lưu trữ trong HDFS. Trong trường hợp này, mỗi khối dữ liệu chỉ được lưu trên 1 DataNode (không có bản sao dự phòng). Sử dụng 1 thường chỉ áp dụng cho môi trường thử nghiệm hoặc single-node (máy đơn) setup vì không có dữ liệu dự phòng khi một DataNode bị lỗi. Trong môi trường cluster thực tế, giá trị thường là 3 hoặc cao hơn. Còn dfs.namenode.name.dir sẽ hay nằm trong /hadoop/data/namenode. Là đường dẫn đến thư mục trên ổ đĩa nơi NameNode lưu trữ metadata của HDFS. Metadata bao gồm thông tin về cấu trúc thư mục, vị trí các khối dữ liệu, và các bản sao. Còn dfs.datanode.data.dir cũng nằm trong /hadoop/data/datanode. Đường dẫn đến thư mục trên ổ đĩa nơi DataNode lưu trữ các khối dữ liệu thực tế.



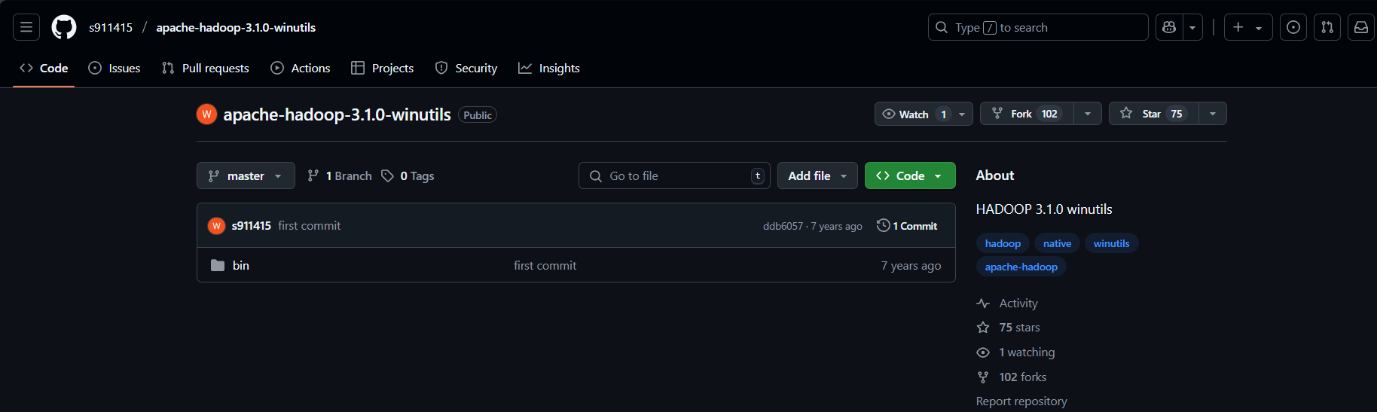
Hình : yarn-site.xml

* yarn-site.xml: yarn.nodemanager.aux-services thì ta sẽ gán mapreduce\_shuffle. Ta định nghĩa các dịch vụ phụ trợ (auxiliary services) mà NodeManager phải cung cấp. Trong trường hợp này, dịch vụ mapreduce\_shuffle được kích hoạt. MapReduce Shuffle là một bước quan trọng trong quy trình MapReduce, nơi dữ liệu trung gian từ giai đoạn Map được chuyển đến giai đoạn Reduce. Dịch vụ này đảm bảo rằng dữ liệu được phân phối đúng cách giữa các nút (nodes). Còn yarn.nodemanager.auxservices.mapreduce.shuffle.class sẽ gán giá trị: org.apache.hadoop.mapred.ShuffleHandler. Nó sẽ chỉ định lớp Java được sử dụng để xử lý dịch vụ mapreduce\_shuffle. Lớp ShuffleHandler là thành phần cốt lõi chịu trách nhiệm thực hiện logic của quá trình Shuffle.

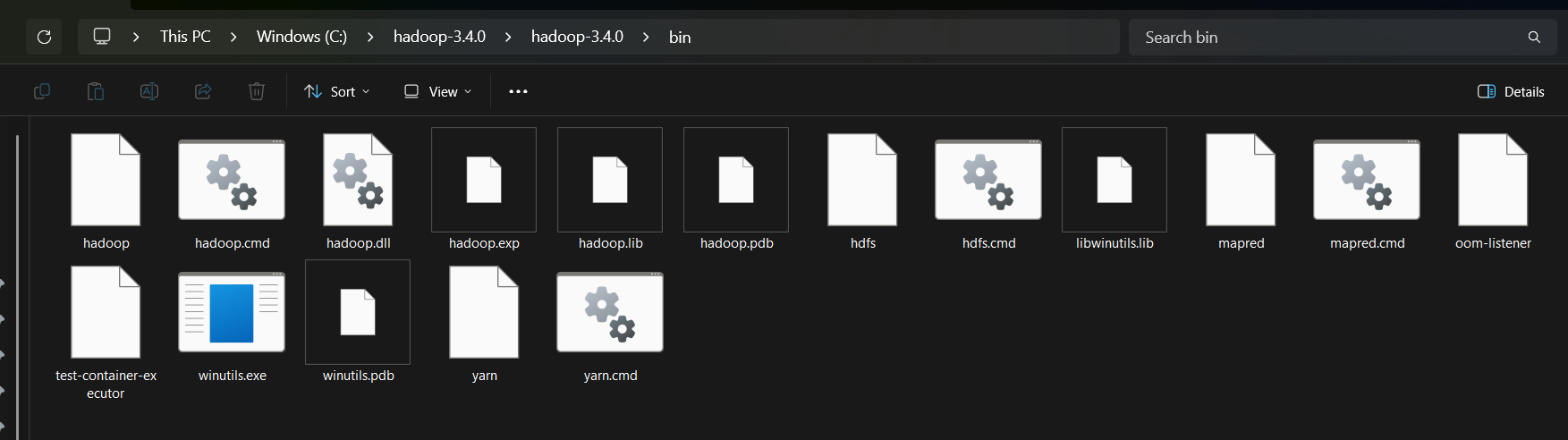


Hình : hadoop-env.cmd

* hadoop-env.cmd: Cấu hình file này sẽ cho hadoop biết Java đã được cài đặt ở đâu. Ta sẽ mở file này lên và tìm tới lệnh: set JAVA\_HOME=%JAVA\_HOME%
* Bước 4: Apache Hadoop vốn là framework mã nguồn mở được hỗ trợ cho Linux, chính vì thế chúng ta phải cấu hình Hadoop Configurations để nó chạy trên Windows
* Trước hết ta phải vào link tải: <https://github.com/s911415/apache-hadoop-3.1.0-winutils>. Winutils là công cụ cần thiết để Hadoop hoạt động đúng trên hệ điều hành Windows. Khi tải về giải nén ra thấy thư mục con có tên bin ở bên trong. Xong rồi chép thư mục bin này trong thư mục bin của C:\hadoop\bin, ghi đè lên nội dung cũ nếu có.

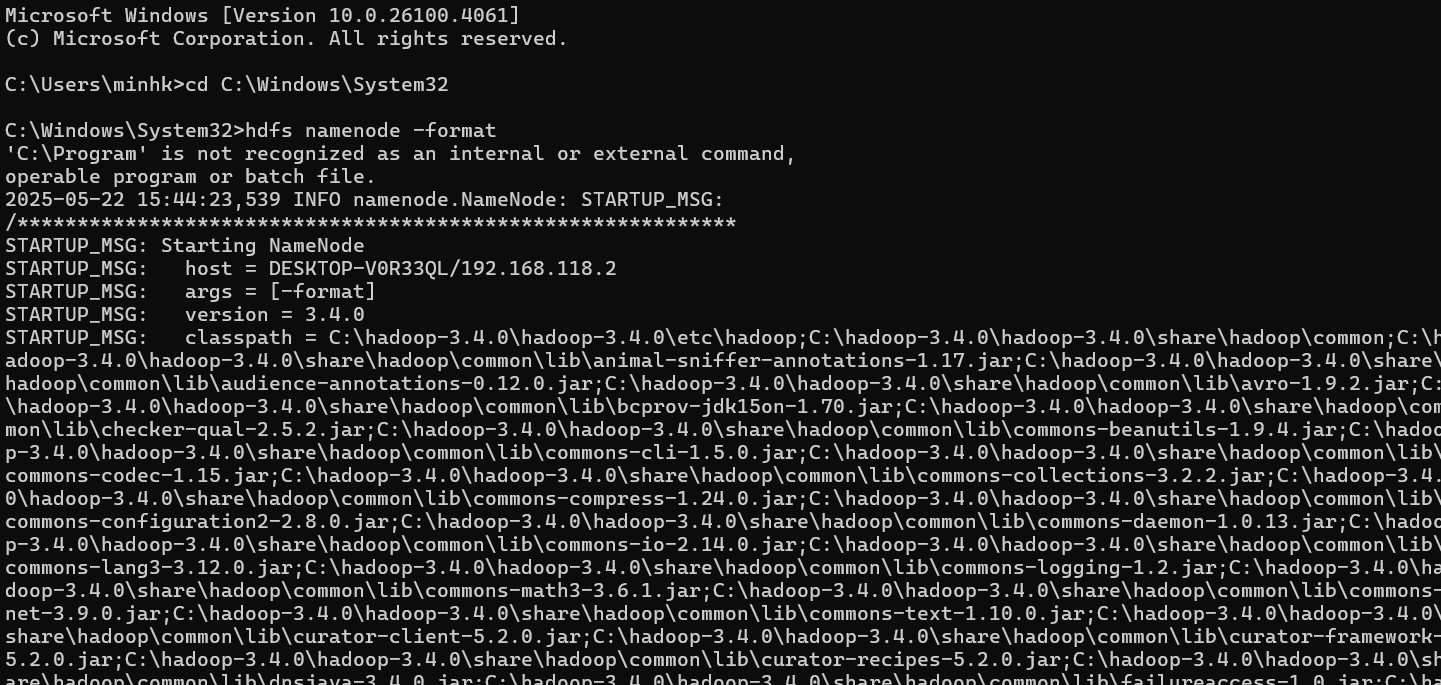


Hình : Trang chủ tải xuống Winutils ở Github

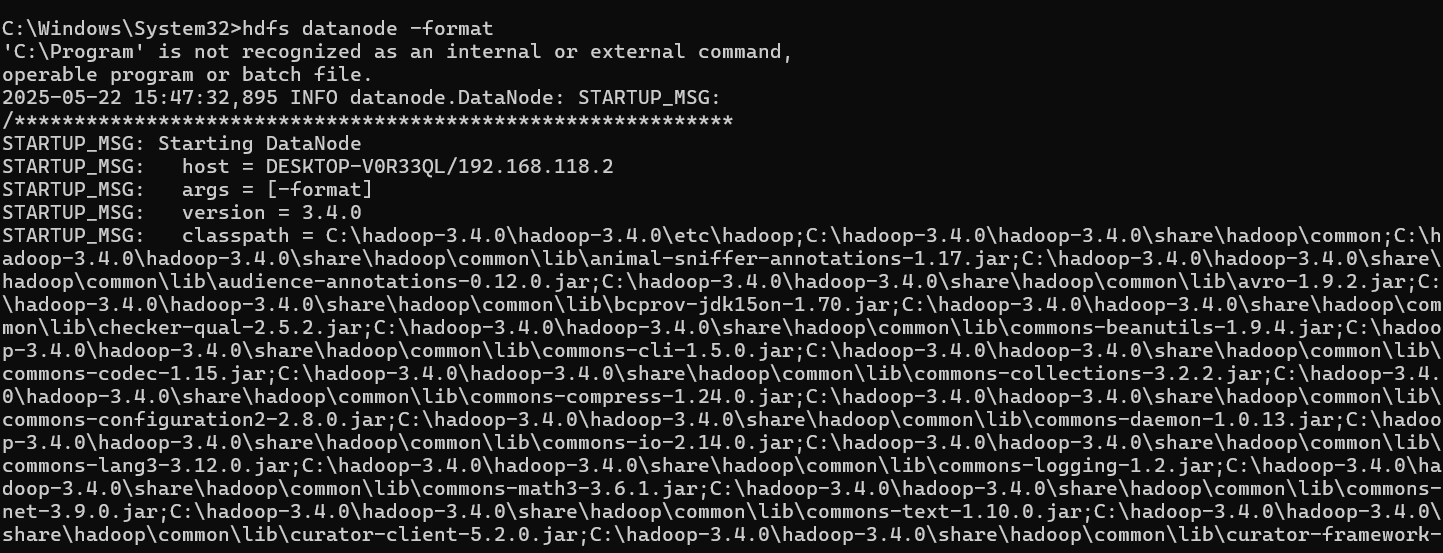


Hình : Thư mục bin có chứa Winutils sau khi giải nén và đè nén thư mục bin ở thư mục Hadoop

* Bước 5: Cuối cùng, ta sẽ format lại NameNode và DataNode
* Tại sao? NameNode lưu trữ thông tin metadata về cấu trúc và vị trí lưu trữ dữ liệu trong HDFS. Còn DataNode lưu trữ dữ liệu thực tế. Việc format này là cần thiết để chuẩn bị hệ thống cho lần sử dụng đầu tiên
* Mở Command Prompt lên, gõ những lệnh sau đây
* hdfs namenode -format dùng để khởi tạo metadata của NameNode. Cần thiết khi thiết lập hệ thống HDFS mới.
* hdfs datanode -format dùng để khởi tạo cấu trúc lưu trữ dữ liệu trên DataNode. Cần thiết khi cấu hình DataNode mới hoặc tái tạo DataNode.
* Việc format là cần thiết khi thiết lập hệ thống mới hoặc tái cấu trúc hệ thống để đảm bảo dữ liệu không bị lỗi hoặc xung đột.

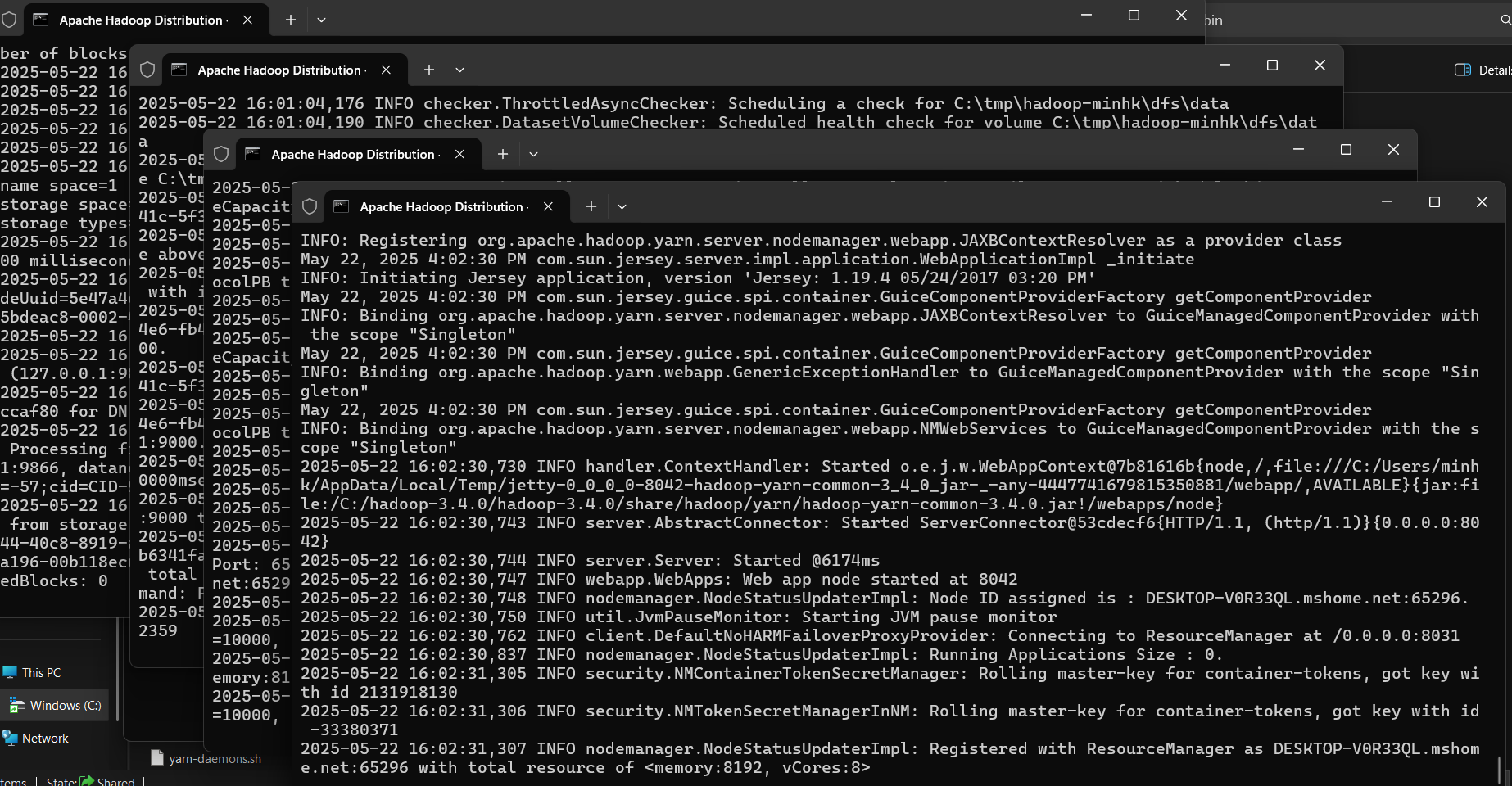


Hình : Format NameNode

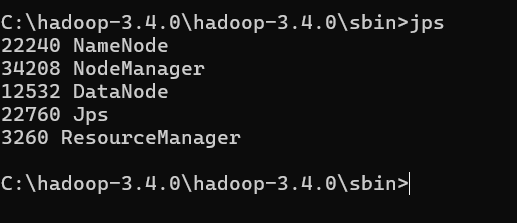


Hình : Format DataNode

* Bước 6: Như vậy ta đã hoàn tất việc cấu hình hệ thông phân tán HDFS. Ta sẽ thử nghiệm chạy tất cả các thành phần với lệnh: start-all.cmd. Hoặc có thể chạy từng thành phần bằng các lệnh: start-dfs.cmd và start-yarn.cmd. Để kiểm tra những thành phần đang chạy, gõ lệnh: jps

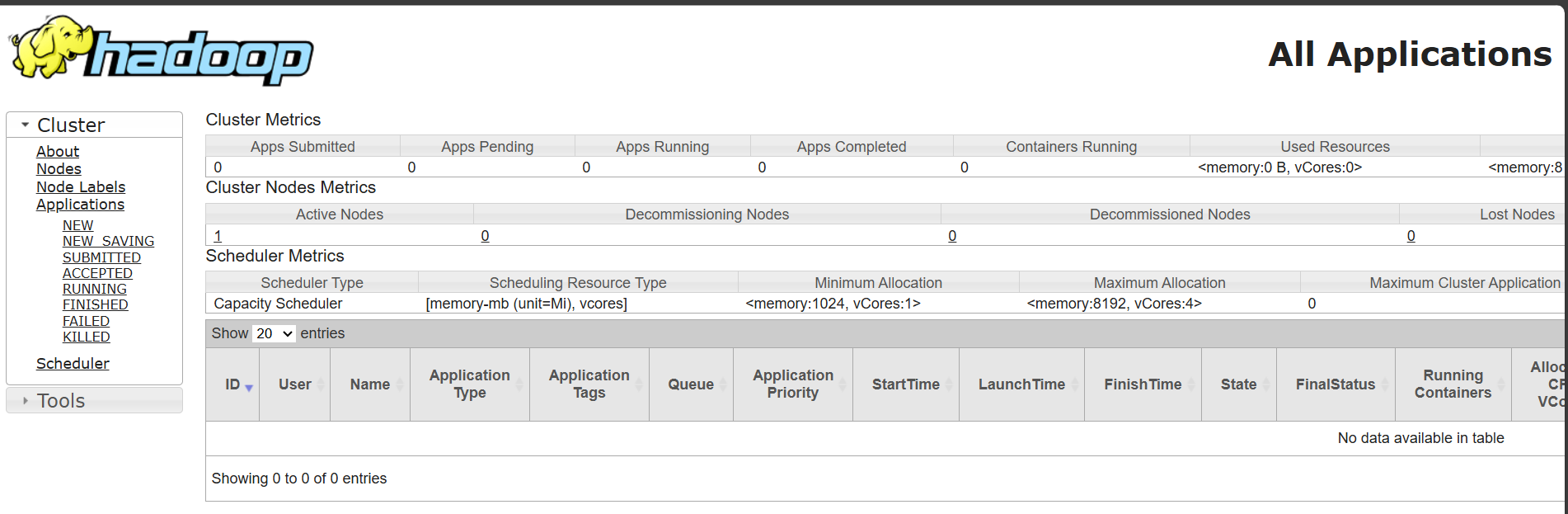


Hình : Khởi động Hadoop



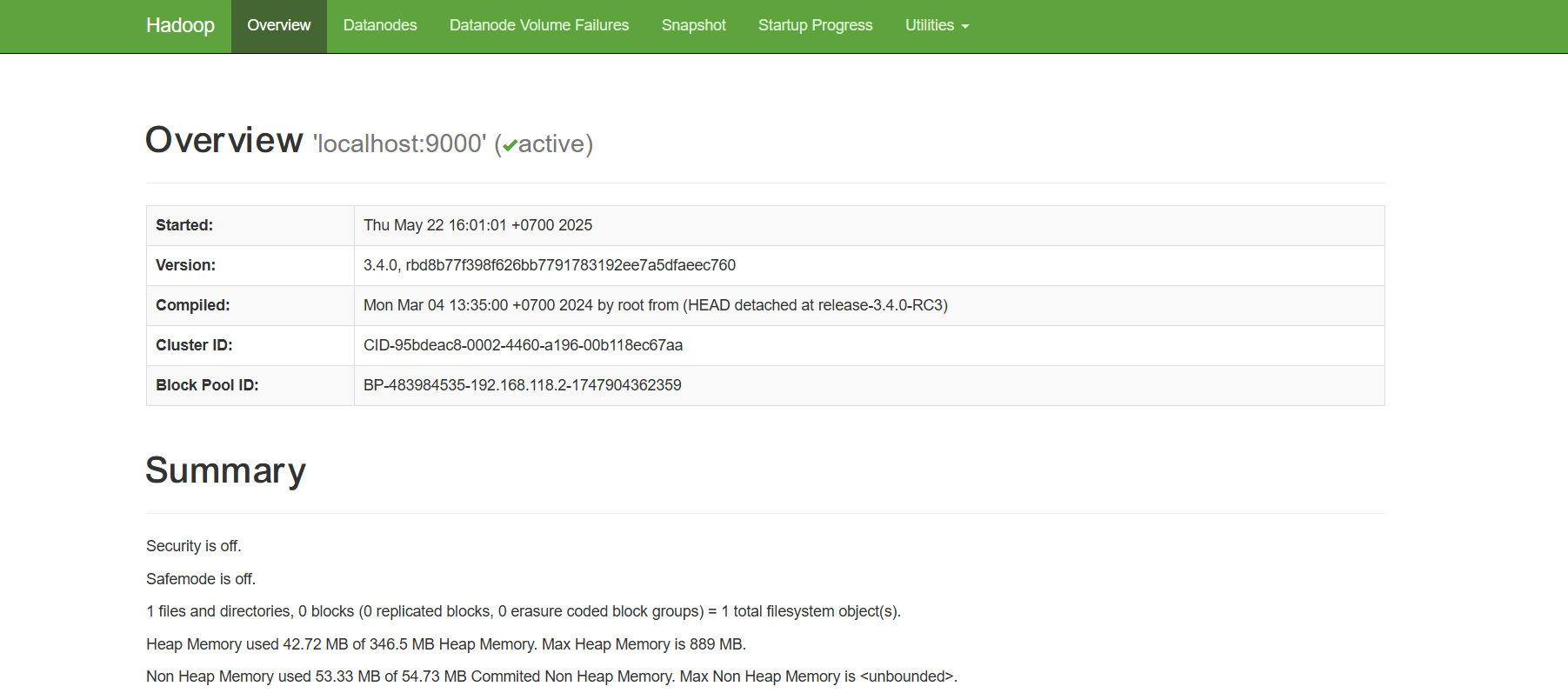
Hình : Sử dụng lệnh jps

* Các thành phần được khởi chạy đầy đủ thì đã thành công. Sau đó ta mở trình duyệt và vào các đường dẫn: <http://localhost:8088> để hiển thị thông tin về các ứng dụng đang chạy và tài nguyên được sử dụng

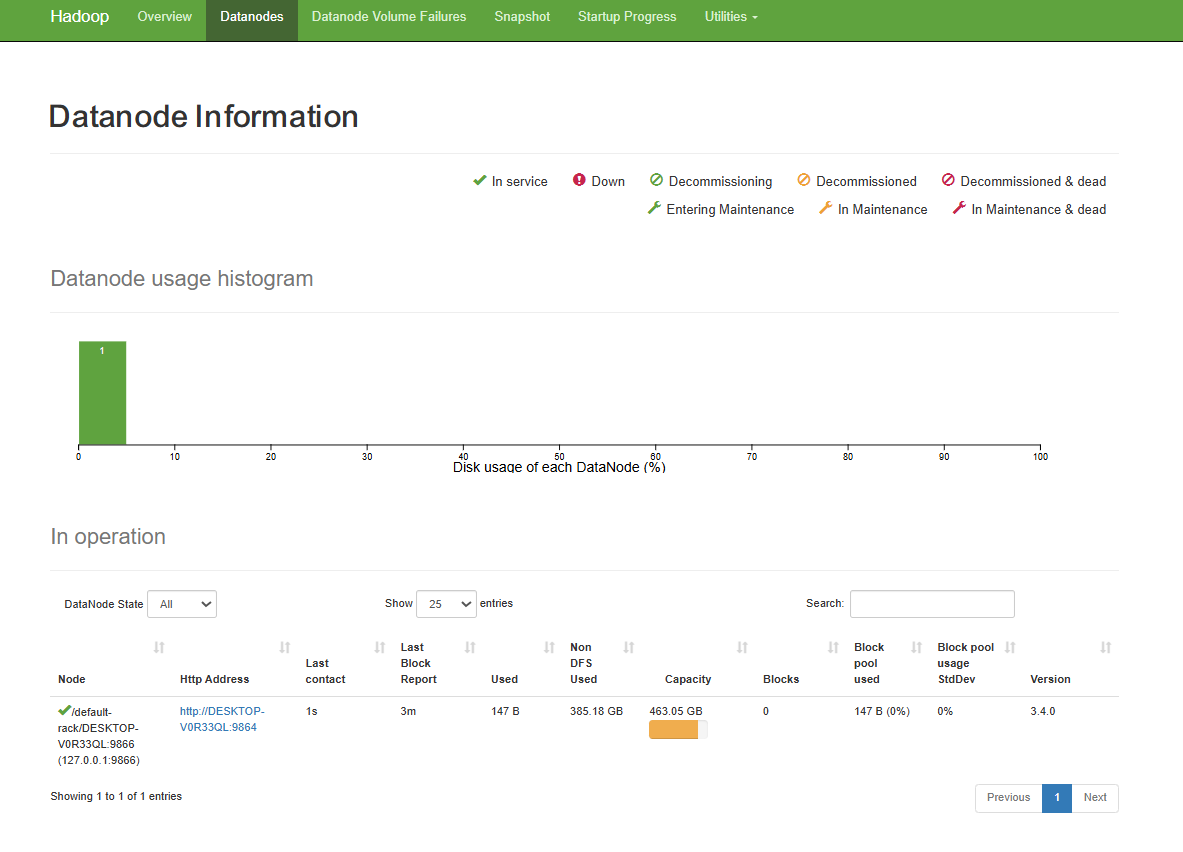


Hình : Giao diện Hadoop khi vào localhost:8088

* Ta sẽ truy cập vào http://localhost:9870 - nơi hiển thị thông tin về HDFS, bao gồm các tệp và trạng thái DataNode



Hình : Giao diện tổng quan cuả Hadoop khi vào localhost:9870

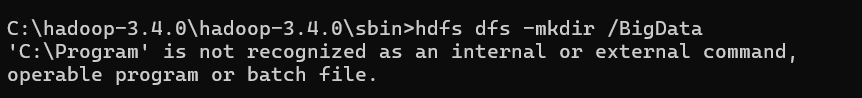


Hình : Giao diện Datanodes của Hadoop khi vào localhost:9870

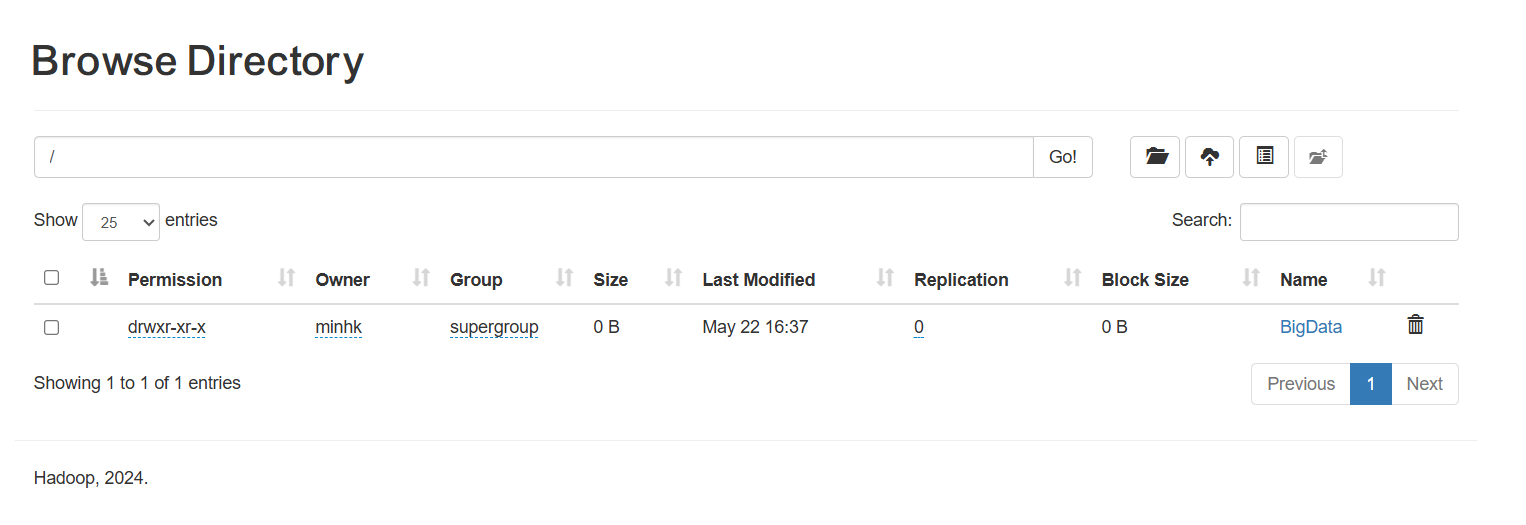
* Bước 7: Ở bước cuối cùng, ta sẽ tiến hành đẩy file bộ dữ liệu đã nêu ở trên vào kho HDFS của Hadoop
* Dưới đây là lệnh thao tác với file và thư mục trong HDFS

|  |  |
| --- | --- |
| Commands | Chức năng |
| hdfs dfs -mkdir /đường\_dẫn | Tạo thư mục mới trong HDFS |
| hdfs dfs -ls /đường\_dẫn | Liệt kê các file trong thư mục HDFS tại đường dẫn chỉ định |
| hdfs dfs -rmdir /đường\_dẫn | Xóa thư mục trống trong HDFS |
| hdfs dfs -rm /đường\_dẫn | Xóa file hoặc thư mục trong HDFS |
| hdfs dfs -rm -r /đường\_dẫn | Xóa thư mục và xóa tất cả các nội dung bên trong |
| Hdfs dfs -copyFromLocal /đường\_dẫn\_local /đường\_dẫn\_hdfs | Sao chép file hoặc thư mục từ máy cục bộ vào HDFS |
| hdfs dfs -copyToLocal /đường\_dẫn\_hdfs /đường\_dẫn\_local | Sao chép file hoặc thư mục từ HDFS về máy cục bộ |
| hdfs dfs -put /đường\_dẫn\_local /đường\_dẫn\_hdfs | Tải file từ máy cục bộ lên HDFS |
| hdfs dfs -get /đường\_dẫn\_hdfs /đường\_dẫn\_local | Tải file từ HDFS về máy cục bộ |
| hdfs dfs -cat /đường\_dẫn/file | Hiển thị nội dung của file trong HDFS |

* Bây giờ ta sẽ tiến hành tạo thư mục BigData

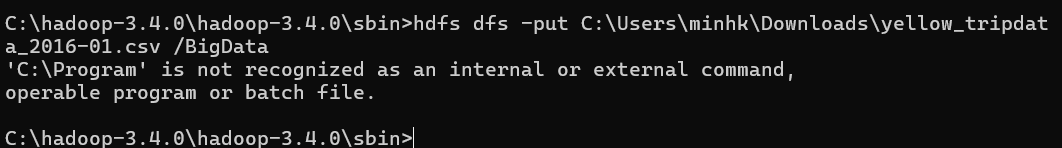


Hình : Lệnh dùng để tạo thư mục BigData

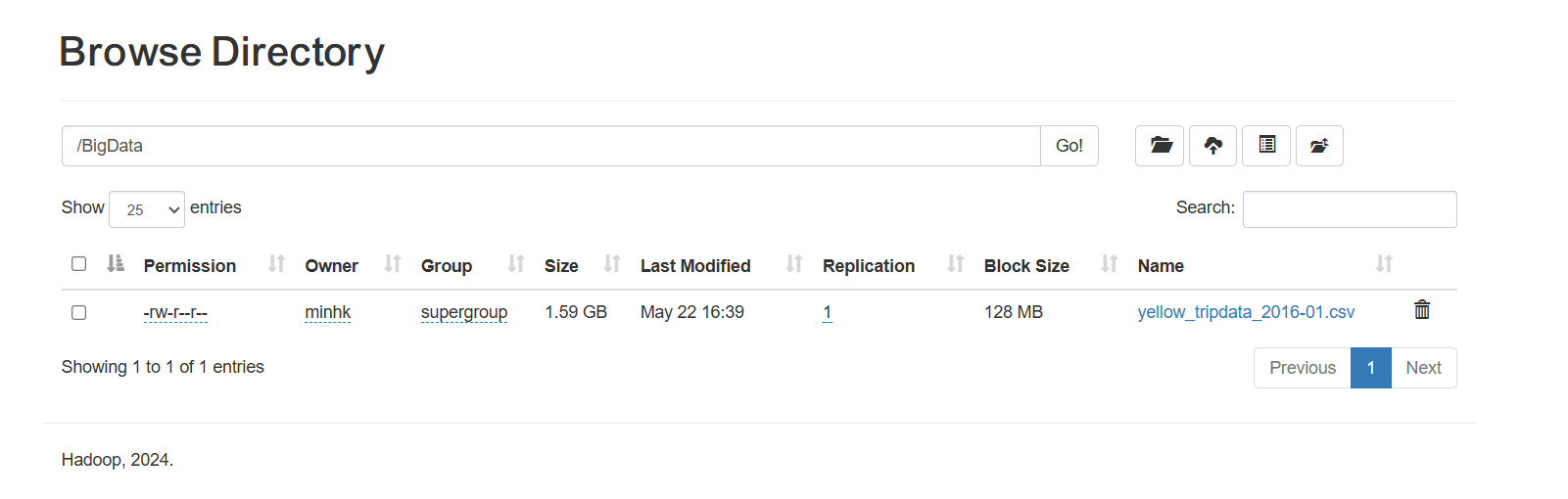


Hình : Sau khi tạo thư mục, thì thư mục BigData đã xuất hiện trên giao diện HDFS

* Ta sẽ tải file “yellow\_tripdata\_2016-01.csv” từ máy cục bộ lên thư mục BigData trong HDFS



Hình : Lệnh dùng để tải file cục bộ lên HDFS



Hình : Cuối cùng, bộ dữ liệu đã chính thúc được tải lên và lưu trữ trong HDFS

1. **Đánh giá và thêm một chút kiến thức về HDFS**
2. **Đánh giá**

* Như vậy, nhóm đã cài đặt và cấu hình Apache Hadoop đúng phiên bản, đảm bảo cả NameNode, DataNode, ResourceManager và NodeManager khởi động ổn định.
* Các file cấu hình (core-site.xml, hdfs-site.xml, yarn-site.xml,…) được chỉnh đúng đường dẫn và thông số cơ bản.
* Bộ dữ liệu đã được đẩy lên HDFS thành công, không gặp lỗi I/O hay phân mảnh không đều. Dữ liệu đã được phân tán đều trên các DataNode (block replication theo thiết lập).
* Quá trình hdfs dfs -put hoàn thành không lỗi, dung lượng trên HDFS khớp với dự kiến.
* Mỗi bước (cài đặt, cấu hình, khởi dịch vụ, upload dữ liệu) đều được ghi chép chi tiết.

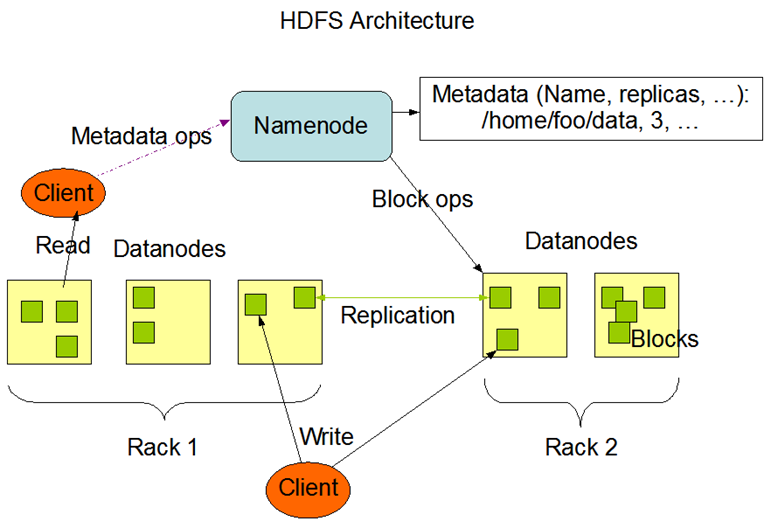
Như vậy nhóm đã tải lên và lưu trữ bộ dữ liệu thành công trên Apache Hadoop, tiếp theo nhóm sẽ cài đặt và cấu hình Apache Spark và tiến hành xử lý dữ liệu. Trước hết, nhóm sẽ lý giả một chút kiến thức về HDFS.

1. **Cơ chế lưu trữ dữ liệu trên HDFS (block, replication)**

HDFS là từ viết tắt của Hadoop Distributes File System, là một hệ thống lưu trữ dữ liệu phân tán được thiết kế chạy trên phần cứng thông thường để lưu trữ các tệp dữ liệu có kích thước lớn trên nhiều nút máy tính trong một mạng Hadoop.

HDFS cũng tương tự như những hệ thống file phân tán khác. Tuy nhiên, sự khác biệt ở đây là HDFS có khả năng chịu lỗi cao (fault-tolerant) và được thiết kế để deploy trên các phần cứng rẻ tiền. HDFS cũng cung cấp khả năng truy cập high throughput từ ứng dụng và thích hợp với các ứng dụng có tập dữ liệu lớn.

HDFS là một phần của nền tảng Hadoop và là một phần quan trọng của việc xử lý dữ liệu lớn.



Hình : Kiến trúc HDFS – Nguồn Internet

NameNode: Đây là máy chủ quản lý metadata, bao gồm vị trí của từng block trên các DataNode và thông tin về hệ số sao chép của chúng. NameNode không lưu trữ dữ liệu thực tế mà chỉ lưu trữ thông tin về nơi mà dữ liệu được lưu trữ. NameNode là điểm duy nhất quản lý metadata. Nếu NameNode hỏng mà không có bản sao sự phòng (secondary NameNode) thì hệ thống HDFS sẽ bị gián đoạn.

DataNode: Các file trong HDFS được chia thành nhiều khổi nhỏ (blocks) và các khối này được lưu trữ trên các DataNode. DataNode chịu trách nhiệm quản lý, lưu trữ và phục vụ các yêu cầu đọc/ghi từ các block. Các DataNode gửi heartbeat (tín hiệu) thường xuyên đến NameNode để báo rằng DataNode đang hoạt động bình thường. HDFS đảm bảo rằng mỗi block dữ liệu được sao chép (replication) trên nhiều DataNode để tăng độ tin cậy. Nếu một DataNode bị hỏng, NameNode sẽ sao chép dữ liệu từ các DataNode khác.

Quy trình ghi dữ liệu:

- Client gửi yêu cầu đến NameNode:

+) Khi Client muốn lưu một file, nó liên hệ với NameNode để biết cách chia file thành các block và lưu trữ trên DataNode.

+) NameNode trả về danh sách các DataNode có thể lưu trữ các block.

- Client ghi dữ liệu vào DataNode:

+) File được chia thành các block (theo kích thước mặc định, thường là 128 MB).

+) Client bắt đầu ghi các block lên DataNode đầu tiên.

- Replication (sao chép):

+) Sau khi DataNode đầu tiên nhận được block, nó sẽ tự động sao chép block này đến các DataNode khác (theo hệ số sao chép - mặc định là 3 bản).

+) Việc sao chép diễn ra đồng thời giữa các DataNode mà không cần quay lại Client.

Quy trình đọc dữ liệu:

- Client gửi yêu cầu đến NameNode:

+) Client yêu cầu NameNode cung cấp thông tin về vị trí các block của file cần đọc.

+) NameNode trả về danh sách các DataNode chứa các block của file.

- Client đọc dữ liệu trực tiếp từ DataNode:

+) Dựa vào thông tin nhận được từ NameNode, Client kết nối trực tiếp đến các DataNode để đọc dữ liệu.

+) Nếu một DataNode bị lỗi, Client sẽ chuyển sang đọc từ bản sao của block trên DataNode khác.

1. **Block (Khối dữ liệu)**

- Kích thước khối (block size): Mỗi file trong HDFS được chia thành các khối dữ liệu (blocks) có kích thước cố định, thường là 128 MB hoặc 256 MB (có thể thay đổi theo cấu hình). Kích thước khối lớn giúp giảm số lượng block và tăng hiệu suất đọc/ghi dữ liệu.

- Phân chia file thành blocks: Khi một file lớn được lưu trữ, HDFS sẽ chia nhỏ nó thành nhiều block theo kích thước đã định. Mỗi block được lưu trữ độc lập trên các node khác nhau trong cluster (cụm máy).

- Địa chỉ của block: Các block này không chứa metadata về file hoặc vị trí. Thay vào đó, thông tin metadata được lưu trữ trên NameNode, nơi chứa thông tin vị trí của từng block.

Ví dụ: Nếu một file có kích thước 600 MB và kích thước block là 128 MB, file sẽ được chia thành 5 block:

Block 1: 128 MB

Block 2: 128 MB

Block 3: 128 MB

Block 4: 128 MB

Block 5: 88 MB (phần còn lại)

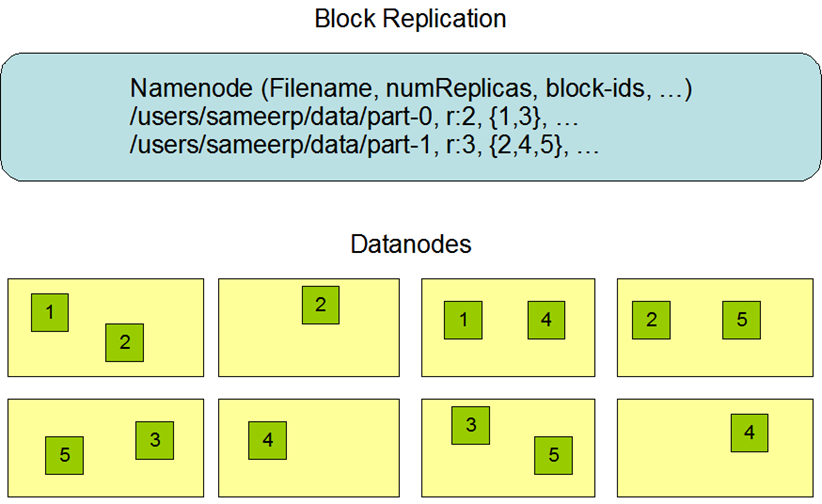
1. **Data Replication (Sao chép dữ liệu)**

HDFS được thiết kế để lưu trữ đáng tin cậy các tệp rất lớn trên các máy trong một cụm lớn. Nó lưu trữ mỗi tệp dưới dạng một chuỗi các khối; tất cả các khối trong một tệp ngoại trừ khối cuối cùng đều có cùng kích thước. Các khối của một tệp được sao chép để có khả năng chịu lỗi. Kích thước khối và hệ số sao chép có thể định cấu hình cho mỗi tệp. Một ứng dụng có thể chỉ định số lượng bản sao của một tệp. Hệ số sao chép có thể được chỉ định tại thời điểm tạo tệp và có thể thay đổi sau đó. Các tệp trong HDFS là ghi một lần và chỉ có một trình ghi tại bất kỳ thời điểm nào.

NameNode đưa ra mọi quyết định liên quan đến việc sao chép các khối. Nó định kỳ nhận Heartbeat và Blockreport từ mỗi DataNode trong cụm. Nhận Heartbeat ngụ ý rằng DataNode đang hoạt động bình thường. Blockreport chứa danh sách tất cả các khối trên DataNode.

- Heartbeat là tín hiệu định kỳ mà mỗi DataNode gửi đến NameNode để thông báo rằng nó vẫn đang hoạt động bình thường.

- Blockreport là báo cáo định kỳ mà mỗi DataNode gửi đến NameNode. Báo cáo này chứa danh sách chi tiết tất cả các khối dữ liệu (data blocks) hiện có trên DataNode.



Hình : Kiến trúc về Block Replication

- Hệ số sao chép (Replication Factor): Để đảm bảo tính sẵn sàng và an toàn cho dữ liệu, mỗi block trong HDFS được sao chép nhiều lần trên các node khác nhau. Mặc định, hệ số sao chép là 3 (có thể thay đổi). Điều này có nghĩa là mỗi block sẽ có 3 bản sao nằm trên các máy khác nhau trong cluster.

- Cơ chế sao chép: Khi một block được tạo, NameNode sẽ xác định 3 DataNode khác nhau để lưu trữ 3 bản sao của block đó. Nếu một DataNode bị hỏng hoặc không khả dụng, hệ thống vẫn có thể lấy dữ liệu từ các DataNode khác có chứa bản sao của block.

- Sao chép tự động: Nếu một bản sao của block bị mất (do hỏng DataNode), NameNode sẽ tự động tạo bản sao mới từ các bản sao còn lại và lưu trữ trên một DataNode khác để đảm bảo rằng luôn có đủ số lượng bản sao theo hệ số sao chép.

1. **Đọc và ghi dữ liệu trong HDFS**

- Ghi dữ liệu: Khi người dùng ghi một file vào HDFS, file sẽ được chia thành các block và gửi đến các DataNode để lưu trữ theo hệ số sao chép. NameNode chỉ định các DataNode để lưu trữ bản sao của từng block.

- Đọc dữ liệu: Khi người dùng yêu cầu đọc một file, NameNode cung cấp thông tin vị trí của các block và DataNode chứa block đó. Người dùng sau đó có thể truy cập trực tiếp vào các DataNode để đọc dữ liệu.

1. **Ưu điểm của cơ chế lưu trữ HDFS**

- Tính sẵn sàng cao: Cơ chế sao chép dữ liệu giúp HDFS có thể duy trì tính sẵn sàng ngay cả khi một số DataNode gặp sự cố.

- Tốc độ xử lý song song: HDFS phân chia dữ liệu thành các block nhỏ giúp tăng khả năng xử lý song song khi đọc hoặc ghi dữ liệu.

- Khả năng mở rộng: Dữ liệu có thể được phân phối trên nhiều node, và số lượng DataNode có thể dễ dàng mở rộng để tăng khả năng lưu trữ.

1. **APACHE SPARK**
2. **Apache Spark**
3. Giới thiệu

Ngày nay có rất nhiều hệ thống đang sử dụng Hadoop để phân tích và xử lý dữ liệu lớn. Ưu điểm lớn nhất của Hadoop là được dựa trên một mô hình lập trình song song với xử lý dữ liệu lớn là MapReduce, mô hình này cho phép khả năng tính toán có thể mở rộng, linh hoạt, khả năng chịu lỗi, chi phí rẻ. Điều này cho phép tăng tốc thời gian xử lý các dữ liệu lớn nhằm duy trì tốc độ, giảm thời gian chờ đợi khi dữ liệu ngày càng lớn.

Dù có rất nhiều điểm mạnh về khả năng tính toán song song và khả năng chịu lỗi cao nhưng Apache Hadoop có một nhược điểm là tất cả các thao tác đều phải thực hiện trên ổ đĩa cứng điều này đã làm giảm tốc độ tính toán đi gấp nhiều lần.

Để khắc phục được nhược điểm này thì Apache Spark được ra đời. Apache Spark có thể chạy nhanh hơn 10 lần so với Haddop ở trên đĩa cứng và 100 lần khi chạy trên bộ nhớ RAM.

Vậy Apache Spark là gì? Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, biểu đồ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Lịch sử phát triển của Apache Spark

1. **Mục tiêu chính**

* Xử lý dữ liệu nhanh và tương tác
* Xử lý đồ thị
* Công việc lặp đi lặp lại
* Xử lý thời gian thực
* Joining Dataset
* Machine Learning
* Apache Spark là Framework thực thi dữ liệu dựa trên Hadoop HDFS. Apache Spark không thay thế cho Hadoop nhưng nó là một framework ứng dụng. Apache Spark tuy ra đời sau nhưng được nhiều người biết đến hơn Apache Hadoop vì khả năng xử lý hàng loạt và thời gian thực.

1. **Các thành phần trong hệ sinh thái của Apache Spark**

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

* ***Spark Core*** là nền tảng cho các thành phần còn lại và các thành phần này muốn khởi chạy được thì đều phải thông qua Spark Core do Spark Core đảm nhận vai trò thực hiện công việc tính toán và xử lý trong bộ nhớ (In-memory computing) đồng thời nó cũng tham chiếu các dữ liệu được lưu trữ tại các hệ thống lưu trữ bên ngoài.
* ***Spark SQL*** cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc (semi-structured data – thường là dữ liệu dữ liệu có cấu trúc nhưng không đồng nhất và cấu trúc của dữ liệu phụ thuộc vào chính nội dung của dữ liệu ấy). Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.
* ***Spark Streaming*** được sử dụng để thực hiện việc phân tích stream bằng việc coi stream là các mini-batches và thực hiệc kỹ thuật RDD transformation đối với các dữ liệu mini-batches này. Qua đó cho phép các đoạn code được viết cho xử lý batch có thể được tận dụng lại vào trong việc xử lý stream, làm cho việc phát triển lambda architecture được dễ dàng hơn. Tuy nhiên điều này lại tạo ra độ trễ trong xử lý dữ liệu (độ trễ chính bằng mini-batch duration) và do đó nhiều chuyên gia cho rằng Spark Streaming không thực sự là công cụ xử lý streaming giống như Storm hoặc Flink.
* ***MLlib (Machine Learning Library):*** MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn 9 lần so với phiên bản chạy trên Hadoop (Apache Mahout).
* ***GrapX:*** Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tảcác tính toán trong đồ thị bằng cách sử dụng Pregel Api.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, Hình chữ nhật

Nội dung do AI tạo ra có thể không chính xác.

Hình : Các thành phần trong hệ sinh thái Apache Spark

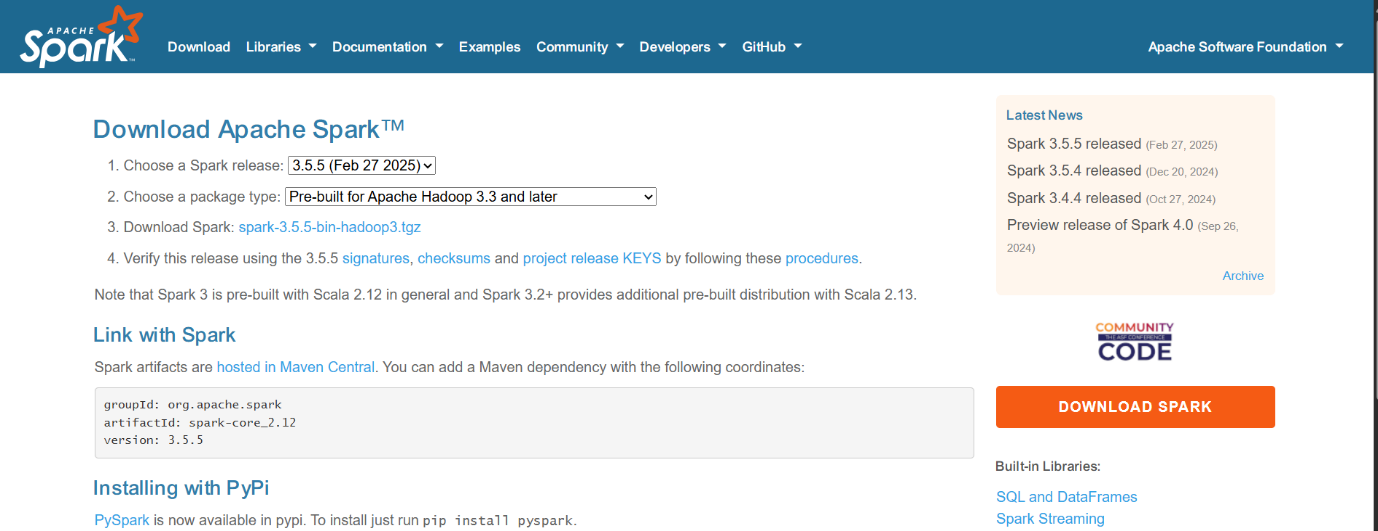
1. **Ưu điểm**

* Xử lý dữ liệu: Spark xử lý dữ liệu theo lô và thời gian thực
* Tính tương thích: Có thể tích hợp với tất cả các nguồn dữ liệu và định dạng tệp được hỗ trợ bởi cụm Hadoop.
* Hỗ trợ ngôn ngữ: hỗ trợ Java, Scala, Python và R.
* Phân tích thời gian thực:
* Apache Spark có thể xử lý dữ liệu thời gian thực tức là dữ liệu đến từ các luồng sự kiện thời gian thực với tốc độ hàng triệu sự kiện mỗi giây. Ví dụ: Data Twitter chẳng hạn hoặc luợt chia sẻ, đăng bài trên Facebook. Sức mạnh Spark là khả năng xử lý luồng trực tiếp hiệu quả.
* Apache Spark có thể được sử dụng để xử lý phát hiện gian lận trong khi thực hiện các giao dịch ngân hàng. Đó là bởi vì, tất cả các khoản thanh toán trực tuyến được thực hiện trong thời gian thực và chúng ta cần ngừng giao dịch gian lận trong khi quá trình thanh toán đang diễn ra.

1. **Thực nghiệm**

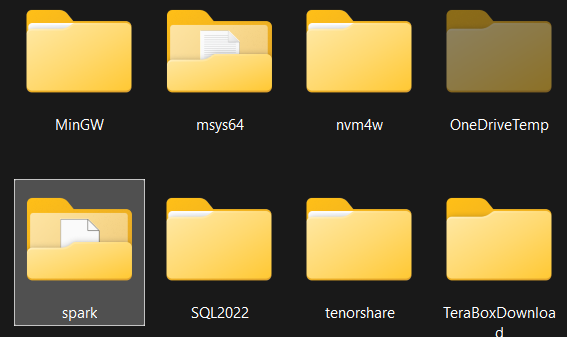
Cách cài đặt Spark về cơ bản là giống như cách mọi người cài đặt Hadoop.

* Bước 1: Lên trang web chính thức: <https://spark.apache.org/downloads.html> để tải Spark về.



Hình : Giao diện trang chủ Apache Spark

* Chọn “package type” là “Pre-built for Apache Hadoop 3.3 and later”, và tải.
* Bước 2: Giải nén tệp đã tải về. Thông thường tên đó sẽ là “spark-3.5.3-bin-hadoop3.tgz” (hoặc phiên bản khác, tùy vào số phiên bản đã chọn trước đó.). Đổi tên thư mục sang tên “spark”, chuyển thư mục sang ổ C: và ta sẽ có một thư mục như sau:

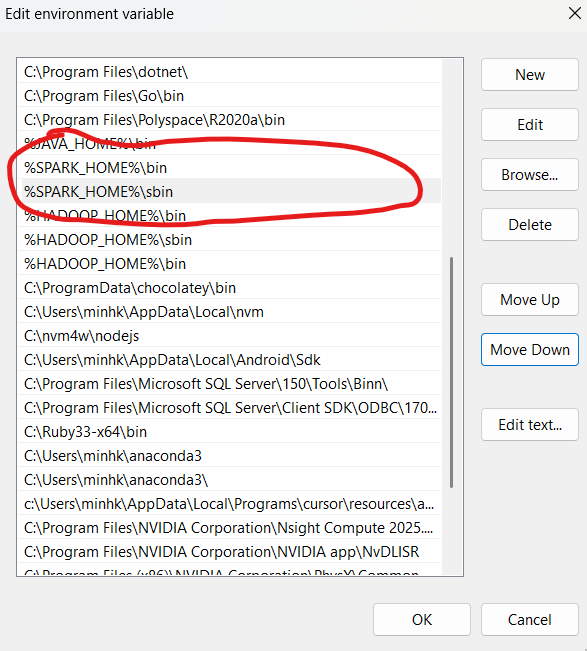


Hình : Thư mục Spark

* Bước 3: Như trên Hadoop, từ thanh tìm kiếm, gõ ‘môi trường’ (hoặc ‘environment’), sẽ ra kết quả “Sửa các biến môi trường” (hoặc “Edit environment variables” nếu máy dùng tiếng Anh).
* Như đã làm với Java SE và Hadoop, ta sẽ cấu hình đường dẫn Spark với tên gọi là SPARK\_HOME



Hình : Ở mục thiết lập đường dẫn, ta thêm SPARK\_HOME vào đường dẫn user (ta làm tương tự với đường dẫn hệ thống)



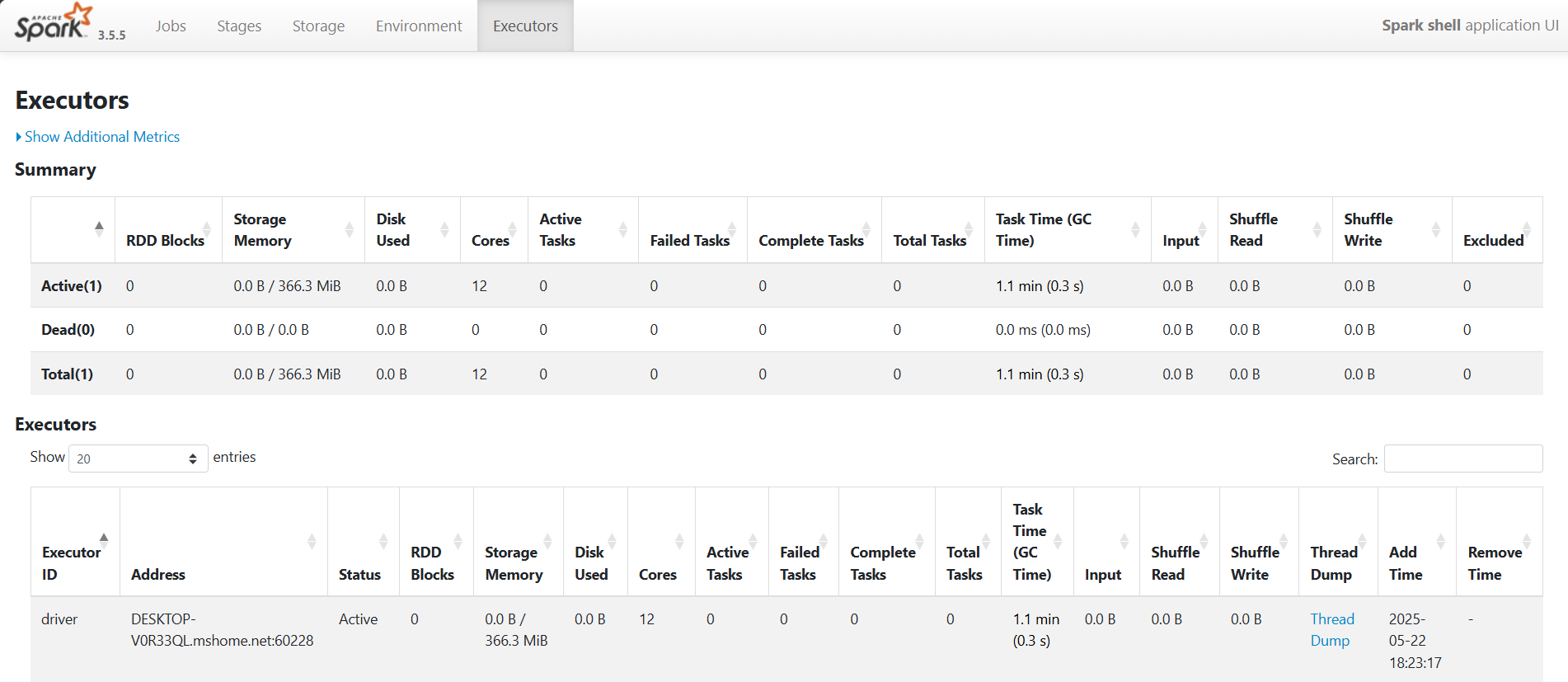
Hình : Ở đường dẫn PATH của đường dẫn hệ thống, ta sẽ thêm các đường dẫn SPARK\_HOME vào

* Bước 4: Chạy Spark. Trong Command Prompt, gõ câu lệnh sau: spark-shell, và màn hình hiện ra như sau:



Hình : Giao diện khởi động Apache Spark

* Ta sẽ gõ tiếp vào localhost:4040, sẽ hiện ra một trang:



Hình : Giao diện Spark khi truy cập localhost:4040

1. **Tổng kết**

Chúng ta đã cài đặt thành công Spark, một công cụ hữu ích cho Hadoop

trong việc quản lý, xử lý và phân tích dữ liệu lớn. Trong các phần tiếp theo, nhóm sẽ thực hiện truy vấn Spark SQL với tập dữ liệu trên.

1. **Truy vấn với Spark SQL**
2. **Tổng quan về Spark SQL**

Spark SQL là một thành phần quan trọng của Apache Spark, cung cấp một cách tiếp cận linh hoạt và hiệu quả để thực hiện các truy vấn và biến đổi dữ liệu bằng ngôn ngữ SQL trên dữ liệu phân tán trong Spark.

Dưới đây là một số điểm quan trọng cần hiểu rõ về Spark SQL:

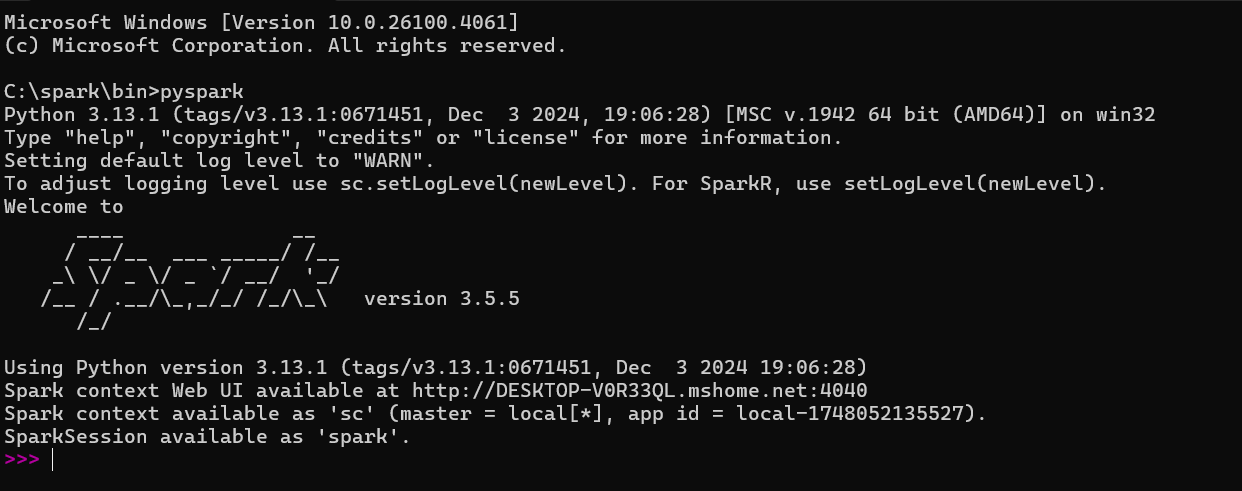
* ***SQL-like Interface:*** Spark SQL cho phép sử dụng ngôn ngữ SQL để truy vấn và biến đổi dữ liệu, giống như mọi người làm việc với một cơ sở dữ liệu quan hệ truyền thống. Điều này làm cho việc làm quen và sử dụng Spark SQL trở nên dễ dàng đối với những người đã quen với SQL.
* ***DataFrames Integration:*** Spark SQL tích hợp chặt chẽ với DataFrames trong Apache Spark. Thực tế, mọi DataFrame trong Spark đều có một bảng tương ứng trong Spark SQL, và chúng ta có thể sử dụng các phương thức DataFrame cũng như SQL để truy vấn dữ liệu.
* ***Dữ liệu có cấu trúc:*** Spark SQL thích hợp cho việc xử lý dữ liệu có cấu trúc. Chúng ta có thể sử dụng Spark SQL để truy vấn và biến đổi dữ liệu trong các bảng có cấu trúc được định nghĩa trước, giúp bạn dễ dàng thực hiện các tác vụ phân tích dữ liệu và báo cáo. Dữ liệu có cấu trúc thường là dữ liệu được tổ chức dưới dạng bảng có các cột và hàng, tương tự như trong một cơ sở dữ liệu quan hệ. Chúng ta có thể sử dụng Apache Spark và Spark SQL để đọc và xử lý tập dữ liệu, sau khi đọc dữ liệu vào DataFrame thì sẽ thực hiện các truy vấn và biến đổi dữ liệu.
* ***Hỗ trợ nhiều định dạng dữ liệu:*** Spark SQL hỗ trợ nhiều định dạng dữ liệu phổ biến như JSON, CSV, Parquet, Avro và Hive. Điều này cho phép chúng ta đọc và ghi dữ liệu từ nhiều nguồn khác nhau và thực hiện các phương pháp phân tích dữ liệu trên chúng.
* ***Tối ưu hóa hiệu suất:*** Spark SQL được tối ưu hóa để tận dụng các tính năng in-memory của Apache Spark, giảm thiểu việc truy cập dữ liệu từ đĩa và tối ưu hóa việc thực thi truy vấn SQL trên dữ liệu phân tán.
* ***Tích hợp với các thư viện và công cụ khác:*** Spark SQL tích hợp chặt chẽ với các công cụ và thư viện khác trong hệ sinh thái Spark như MLlib, GraphX và Streaming, giúp chúng ta kết hợp các phương tiện phân tích dữ liệu khác nhau để thực hiện các nhiệm vụ phức tạp trong lĩnh vực Big Data.

Tóm lại, Spark SQL cung cấp một cách tiếp cận linh hoạt và hiệu quả để thực hiện các truy vấn và biến đổi dữ liệu bằng ngôn ngữ SQL trên dữ liệu phân tán trong Apache Spark. Điều này làm cho việc xử lý và phân tích dữ liệu trở nên dễ dàng và hiệu quả hơn trong lĩnh vực Big Data.

1. **Thực nghiệm**

Với bộ dữ liệu đã có trên HDFS, chúng ta sẽ bắt đầu tiến hành xử lý dữ liệu thông qua truy vấn dữ liệu với công cụ Spark SQL.

* Bước 1: Chuẩn bị Python 3.13, Visual Studio Code và thư viện PySpark (một giao diện lập trình ứng dụng (API) Python cho Apache Spark).
* Bước 2: Khởi động pyspark thông qua Command Prompt



Hình : PySpark

* Bước 3: Mở Visual Studio Code lên, tạo file với đuôi .py (Python) bất kỳ, và bắt đầu nhập từng dòng code lên.
* Đầu tiên ta sẽ khai báo thư viện

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

Hình : Các thư viện được khai báo

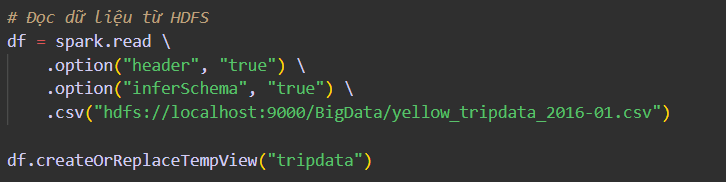
* findspark giúp Python “tìm” được thư mục cài đặt Apache Spark trên máy local. Khi chúng ta cài Spark (ví dụ gói Spark binary), chúng ta thường cần thiết lập biến môi trường SPARK\_HOME và thêm PYTHONPATH để Python có thể import được module pyspark. findspark.init() sẽ tự động dò đường dẫn đến Spark (dựa vào các biến môi trường hoặc đường dẫn mặc định) rồi thêm vào sys.path, nhờ đó chúng ta không phải cấu hình thủ công mỗi lần chạy script. Nếu chạy PySpark script ngoài cluster (ví dụ trên máy local, notebook Jupyter, VSCode…), và chưa cài đặt PySpark qua pip hoặc chưa thiết lập biến môi trường, thì findspark rất hữu dụng để khởi tạo nhanh.
* pyspark: cung cấp API tương tác với Spark (RDD, DataFrame, SQL…). SparkConf sẽ chứa cấu hình cho ứng dụng Spark (tên app, master URL, các tham số tuning…). Ví dụ chúng ta có thể set appName("MyApp"), setMaster("local[\*]")… Còn SparkContext là đối tượng cốt lõi để giao tiếp với cluster Spark. Mỗi ứng dụng Spark khởi tạo một SparkContext, dựa trên các thông số trong SparkConf. SparkSession (API từ Spark 2.x trở đi) là điểm khởi đầu cho mọi thao tác với DataFrame và SQL trong Spark. Bên trong nó đã tích hợp SparkContext và các builder để cấu hình thêm (ví dụ .builder.appName(...).getOrCreate()). Nếu chúng ta chỉ làm việc với RDD cũ, có thể dùng trực tiếp SparkContext; nhưng hầu hết ứng dụng hiện nay đều khởi tạo và dùng SparkSession.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

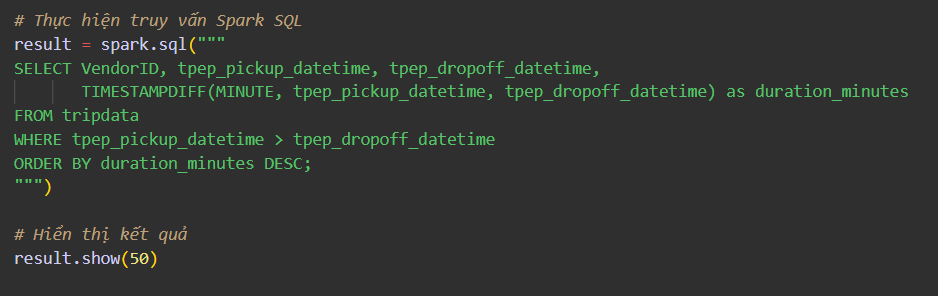
Hình : Bắt đầu thiết lập code

* spark.executor.memory - Quy định lượng RAM cấp cho mỗi Executor (quá trình làm việc chính trong cluster). Ở đây chúng ta đặt mỗi executor được cấp 2 GB.
* spark.driver.memory - Quy định lượng RAM cấp cho Driver — process điều phối toàn bộ workflow Spark. Ở đây chúng ta cấp cho driver 2 GB.
* Nếu không cấu hình, Spark sẽ dùng mặc định (thường rất nhỏ, ví dụ 1 GB), dễ bị tràn bộ nhớ khi xử lý dữ liệu lớn. Đặt tăng lên giúp chúng ta thử nghiệm với tập dữ liệu từ vừa phải đến lớn trên máy local an toàn hơn, tránh OOM. Phải gọi trước khi tạo bất kỳ SparkContext/SparkSession nào để cấu hình có hiệu lực.
* Ta sẽ khởi tạo SparkSession với .builder - bắt đầu quá trình xây dựng (builder pattern) một SparkSession.
* .appName("TripDataAnalysis") - đặt tên cho ứng dụng, sẽ hiển thị trong UI (Web UI, logs…) để bạn dễ nhận diện.
* .master("local[\*]") - chạy Spark ở chế độ local (trên máy cá nhân), và [\*] nghĩa là dùng tất cả các core CPU khả dụng.
* .getOrCreate() - nếu đã có SparkSession cùng tên/master đang chạy thì tái sử dụng, nếu chưa thì tạo mới.
* Dùng SparkSession (thay cho SparkContext + SQLContext cũ) là cách chuẩn từ Spark 2.x trở đi, hỗ trợ cả DataFrame và SQL API. Chạy local giúp chúng ta phát triển, debug, thử nghiệm ngay trên máy mà không cần cluster. getOrCreate() đảm bảo không khởi tạo trùng lặp nhiều session, tránh lỗi “only one SparkContext may be running”.



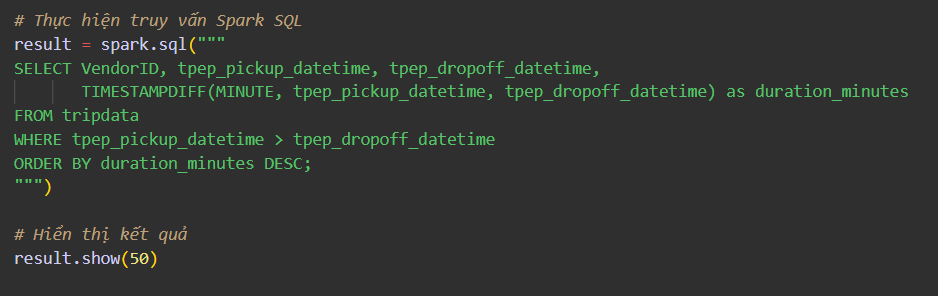
Hình : Đọc dữ liệu và tạo view với dữ liệu

* Trước hết ta sẽ đọc dữ liệu từ HDFS vào DataFrame với spark.read, sẽ tạo một DataFrameReader – đối tượng chuyên để load dữ liệu vào Spark DataFrame.
* .option("header", "true") - cho Spark biết tệp CSV có dòng đầu tiên là header (tên các cột), từ đó sẽ không coi header là bản ghi dữ liệu.
* .option("inferSchema", "true") - bật tự động suy diễn kiểu dữ liệu của từng cột (ví dụ integer, double, timestamp…), thay vì mặc định mọi cột đều là string.
* .csv("hdfs://...") - gọi phương thức đọc CSV, truyền vào đường dẫn HDFS URI: hdfs://localhost:9000/BigData/yellow\_tripdata\_2016-01.csv - localhost:9000 là địa chỉ NameNode HDFS đang chạy trên máy local. Spark sẽ dispatch các task để đọc file này từ HDFS, phân thành partitions tự động theo kích thước block.
* Sử dụng HDFS để lưu trữ và phân phối dữ liệu lớn; Spark kết nối trực tiếp vào HDFS để tận dụng khả năng đọc song song. Tự động infer schema giúp chúng ta thao tác với các cột đúng kiểu (ví dụ tính toán, so sánh ngày tháng) mà không phải tự define schema thủ công.
* Ta sẽ tạo bảng tạm (Temp View) để chạy SQL với createOrReplaceTempView("tripdata") - chuyển DataFrame df thành một view tạm thời có tên "tripdata" trong catalog của SparkSession.
* Lệnh này sẽ cho phép phân tích dữ liệu bằng Spark SQL, tận dụng khả năng tối ưu query (Catalyst optimizer) và API quen thuộc của SQL. Hữu ích khi bạn có nhiều truy vấn, báo cáo, hoặc muốn kết hợp DataFrame API với SQL API.
* Sau khi thiết lập mọi thứ xong xuôi, chúng ta sẽ bắt đầu code truy vấn dữ liệu Spark SQL.

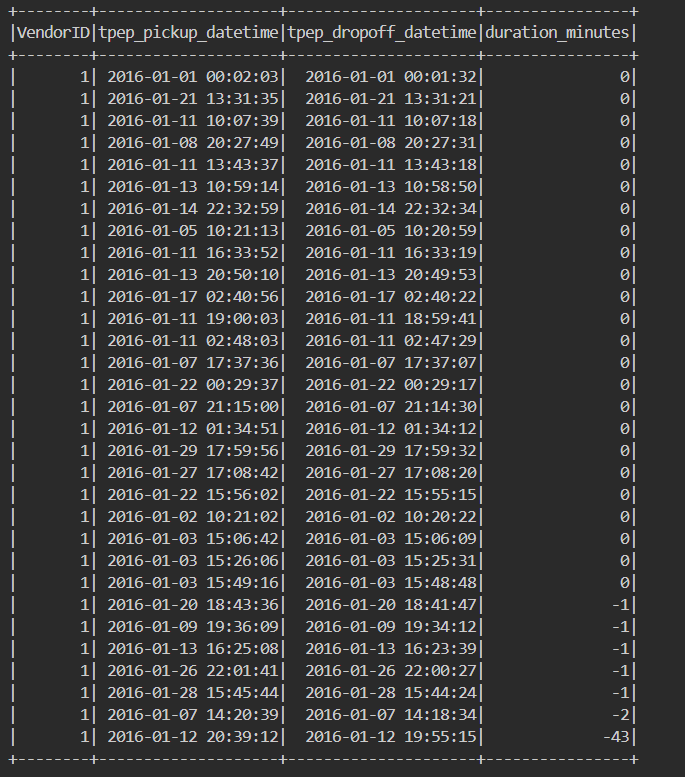


Hình : Thực hiện truy vấn Spark SQL

* spark.sql("""…""") - chạy câu lệnh SQL thuần túy trên catalog Spark (bao gồm các bảng/view đã đăng ký). Trả về một DataFrame mới (result) chứa kết quả của truy vấn.
* .show(50): in ra 50 dòng đầu của DataFrame result lên console (có header, các cột được căn bảng).
* Code truy vấn dữ liệu đầu tiên, ta sẽ tìm các chuyến đi có thời gian bất thường (thời gian đón khách sau thời gian trả khách), để phát hiện dữ liệu bất thường - cụ thể là tìm các chuyến đi taxi có thời gian đón khách muộn hơn thời gian trả khách, điều này về mặt logic là không thể xảy ra.



Hình : Lệnh truy vấn đầu tiên



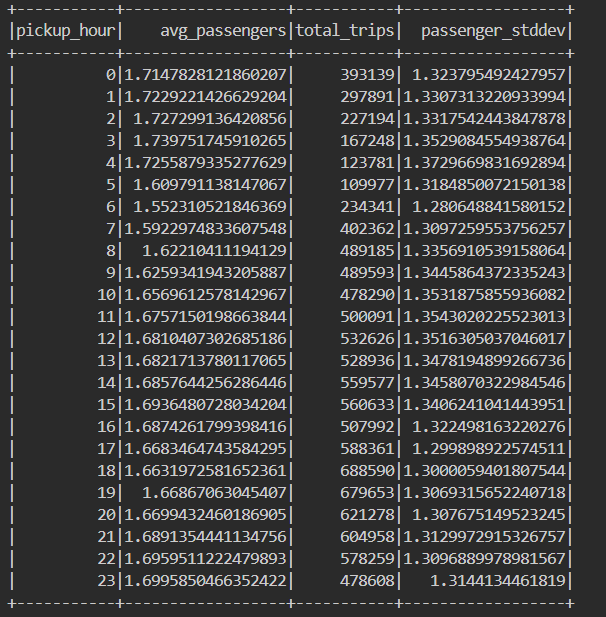
Hình : Kết quả truy vấn đầu tiên

* Hầu hết các chuyến xe trong bảng này có duration\_minutes = 0, nghĩa là thời gian đón khách và trả khách giống nhau hoặc chênh lệch rất nhỏ (có thể do lỗi nhập liệu hoặc đồng hồ hệ thống).
* Một số dòng có duration\_minutes âm (ví dụ: -1, -2, thậm chí -43), nghĩa là thời gian đón khách ghi nhận muộn hơn thời gian trả khách. Điều này là bất hợp lý về mặt thực tế, chỉ có thể do lỗi nhập dữ liệu hoặc lỗi hệ thống.
* Các trường hợp này đều là dữ liệu bất thường, cần được kiểm tra lại hoặc loại bỏ khi phân tích dữ liệu thực tế vì không phản ánh đúng logic của một chuyến xe thông thường.
* Với lệnh truy vấn thứ hai, phân tích xu hướng số lượng hành khách theo từng giờ trong ngày và loại bỏ dữ liệu bất thường.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ hai



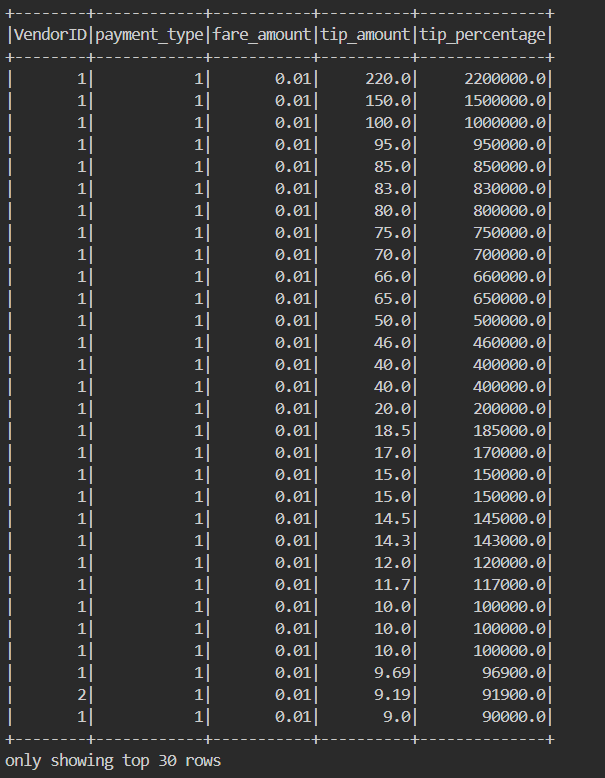
Hình : Kết quả truy vấn thứ hai

* Số chuyến xe theo giờ (total\_trips):
* Số chuyến xe thấp nhất vào các khung giờ đêm và sáng sớm (0h–5h), ví dụ 0h có 393,139 chuyến.
* Số chuyến xe tăng dần từ sáng và cao nhất vào buổi chiều, tối (đỉnh điểm là từ 17h–19h với hơn 670,000 chuyến mỗi giờ).
* Sau 20h, số chuyến bắt đầu giảm dần.
* Số hành khách trung bình mỗi chuyến (avg\_passengers):
* Trung bình mỗi chuyến dao động quanh mức 1.6–1.7 hành khách/chuyến.
* Thấp nhất vào 5h và 6h sáng (~1.61–1.55), có thể là thời điểm ít người đi ghép xe.
* Thường cao hơn vào buổi tối (19h–23h), đạt khoảng 1.69–1.70 hành khách/chuyến.
* Độ lệch chuẩn số hành khách (passenger\_stddev):
* Độ lệch chuẩn dao động từ khoảng 1.28 đến 1.37, cho thấy sự phân tán số hành khách không quá lớn.
* Các giờ buổi sáng sớm và khuya độ lệch chuẩn thấp hơn, tức là số hành khách mỗi chuyến ổn định hơn.
* Vào các khung giờ đông khách (chiều tối), độ lệch chuẩn có xu hướng tăng nhẹ.
* Với code truy vấn thứ ba, ta sẽ tìm các chuyến đi có tỷ lệ tip/fare bất thường (quá cao hoặc quá thấp)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ ba



Hình : Kết quả code truy vấn thứ ba

* Các dòng kết quả đều có fare\_amount rất nhỏ, chỉ 0.01 (có thể là do lỗi nhập liệu hoặc dữ liệu bất thường).
* Tip\_amount lại rất lớn so với fare\_amount, ví dụ: chuyến đầu tiên, fare\_amount chỉ 0.01 nhưng tip\_amount lên tới 220.0, dẫn tới tỷ lệ tip\_percentage là 2,200,000% (tức là tiền boa gấp 22,000 lần tiền cước).
* Các dòng tiếp theo cũng tương tự, tip\_percentage đều ở mức cực kỳ cao, từ 90,000% đến 2,200,000%.
* Tất cả các dòng này về bản chất đều có dấu hiệu dữ liệu bất thường hoặc lỗi nhập liệu (giao dịch tip quá lớn so với cước phí thực tế rất nhỏ).
* Code truy vấn thứ tư, ta sẽ so sánh hiệu suất hoạt động giữa các nhà cung cấp taxi (Vendor) và loại bỏ dữ liệu bất thường để đảm bảo phân tích chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ tư



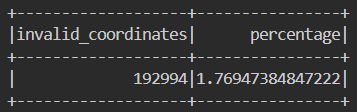
Hình : Kết quả truy vấn thứ tư

* VendorID 1:
* Có hơn 5 triệu chuyến được ghi nhận.
* Quãng đường trung bình mỗi chuyến là khoảng 2.85 dặm.
* Tổng số tiền trung bình mỗi chuyến là khoảng 15.40 USD.
* Thời gian trung bình mỗi chuyến là khoảng 12.67 phút.
* VendorID 2:
* Có gần 5.8 triệu chuyến, nhiều hơn VendorID 1.
* Quãng đường trung bình mỗi chuyến là khoảng 2.98 dặm, nhỉnh hơn VendorID 1.
* Tổng số tiền trung bình mỗi chuyến là khoảng 15.74 USD, cũng cao hơn một chút.
* Thời gian trung bình mỗi chuyến là khoảng 12.88 phút, dài hơn một chút so với VendorID 1.
* Tiếp theo, với code truy vấn thứ 5, mục đích là phát hiện và đo lường tỷ lệ dữ liệu tọa độ bất thường (ngoài khu vực New York City) để đánh giá chất lượng dữ liệu GPS.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ năm



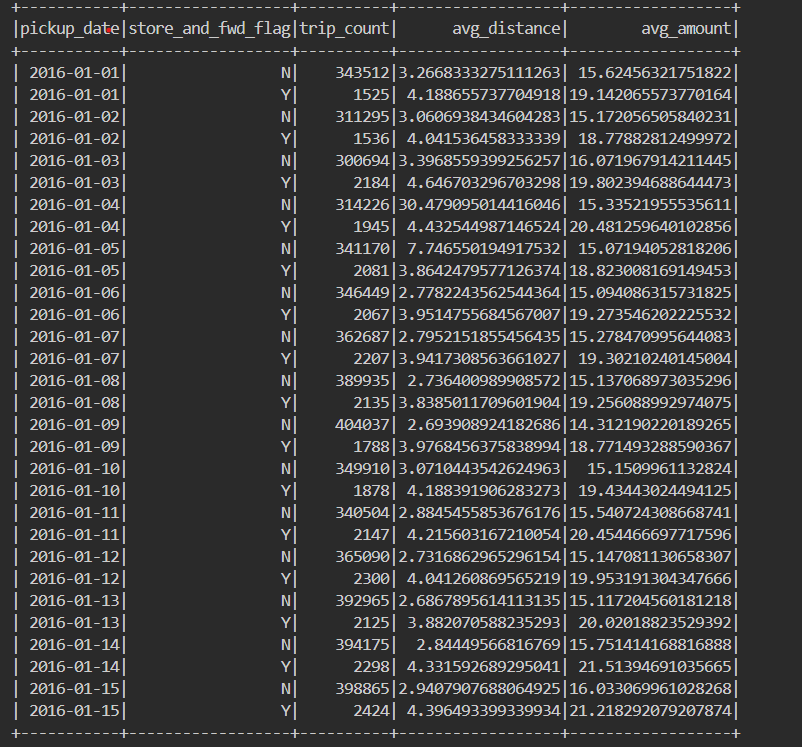
Hình : Kết quả truy vấn thứ năm

* Có 192,994 chuyến đi có tọa độ không hợp lệ (invalid\_coordinates).
* Tỷ lệ chuyến đi có tọa độ không hợp lệ chiếm khoảng 1.77% tổng số chuyến đi trong bảng dữ liệu.
* Đây là tỷ lệ không quá lớn nhưng cần lưu ý và xử lý loại bỏ các bản ghi này khi phân tích các bài toán liên quan đến vị trí, bản đồ hoặc đường đi.
* Code truy vấn thứ sáu, ta sẽ phân tích chuyến đi có Store\_and\_fwd\_flag = 'Y' (dữ liệu offline)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ sáu



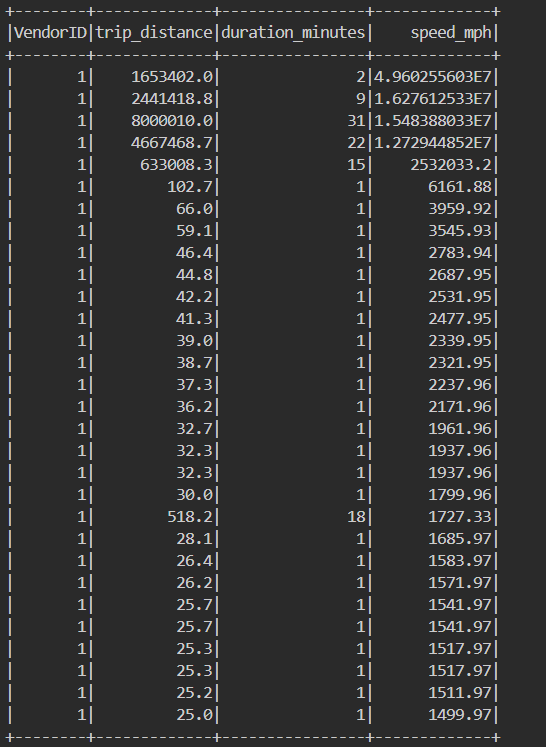
Hình : Kết quả truy vấn thứ sáu

* Số lượng chuyến (trip\_count):
* Số chuyến có Store\_and\_fwd\_flag = 'N' (dữ liệu online) luôn lớn vượt trội, thường trên 300.000 chuyến/ngày.
* Số chuyến có Store\_and\_fwd\_flag = 'Y' (dữ liệu offline) khá nhỏ, chỉ khoảng 1.500–2.400 chuyến/ngày.
* Quãng đường trung bình (avg\_distance):
* Các chuyến offline ('Y') có quãng đường trung bình cao hơn rõ rệt so với các chuyến online ('N').
* Ví dụ: Ngày 2016-01-01, 'N': 3.27 dặm, 'Y': 4.19 dặm.
* Xu hướng này lặp lại ở các ngày khác (thường cao hơn khoảng 1 dặm hoặc hơn).
* Tổng tiền trung bình (avg\_amount):
* Các chuyến offline ('Y') cũng có tổng tiền trung bình cao hơn các chuyến online ('N').
* Ví dụ: Ngày 2016-01-01, 'N': 15.62 USD, 'Y': 19.42 USD.
* Các ngày khác, mức chênh lệch này vẫn duy trì.
* Có thể lý giải rằng các chuyến offline thường đi xa (ví dụ đi ra ngoài vùng phủ sóng), hoặc gặp sự cố mạng trên những lộ trình dài.
* Điều này cần được lưu ý khi phân tích sâu hơn về hành vi khách hàng hoặc chất lượng dịch vụ của từng trạng thái truyền dữ liệu.
* Với code truy vấn thứ bảy, ta sẽ tìm các chuyến đi có tốc độ bất thường (quá nhanh hoặc quá chậm)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ bảy



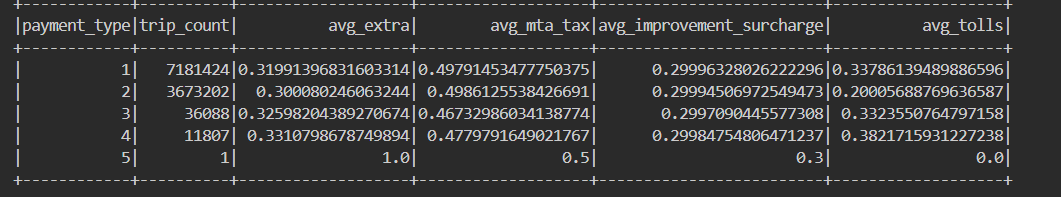
Hình : Kết quả truy vấn thứ bảy

* Kết quả trả về nhiều chuyến có tốc độ trung bình cực kỳ lớn, vượt xa mức hợp lý cho taxi (ví dụ: 2532033.2 mph hoặc thậm chí 4.96 x 10^6 mph).
* Một số chuyến có quãng đường cực lớn (hàng trăm nghìn đến vài triệu dặm) chỉ đi trong thời gian rất ngắn (chỉ 1 đến vài phút). Ví dụ:
* Chuyến đầu tiên: 1,653,402 dặm trong 2 phút → 4,960,256,037 mph (quá phi lý).
* Các chuyến tiếp theo cũng có tốc độ từ hàng nghìn đến hàng triệu mph.
* Nhiều chuyến đi chỉ có duration\_minutes = 1, nhưng quãng đường lên tới 30–100 dặm, dẫn đến tốc độ trung bình 1,500–6,000 mph (vẫn bất thường).
* Tất cả các chuyến đi trong bảng đều có tốc độ trung bình vượt xa giới hạn thực tế, đây là dấu hiệu dữ liệu bị lỗi hoặc nhập sai (có thể do sai lệch về đơn vị, nhập liệu nhầm lẫn, hoặc lỗi hệ thống).
* Với code truy vấn thứ tám, ta sẽ phân tích các khoản phí bổ sung trong taxi NYC theo từng phương thức thanh toán và phát hiện dữ liệu bất thường (anomalies) trong cấu trúc giá.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ tám



Hình : Kết quả truy vấn thứ tám

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **payment\_type** | **trip\_count** | **avg\_extra** | **avg\_mta\_tax** | **avg\_improvement\_surcharge** | **avg\_tolls** |
| 1 (Thẻ tín dụng) | 7,181,424 | 0.32 | 0.50 | 0.30 | 0.34 |
| 2 (Tiền mặt) | 3,673,202 | 0.30 | 0.50 | 0.30 | 0.20 |
| 3 (Không tính phí) | 36,088 | 0.33 | 0.47 | 0.30 | 0.33 |
| 4 (Khiếu nại) | 11,807 | 0.33 | 0.48 | 0.30 | 0.38 |
| 5 (Không rõ) | 1 | 1.00 | 0.50 | 0.30 | 0.00 |

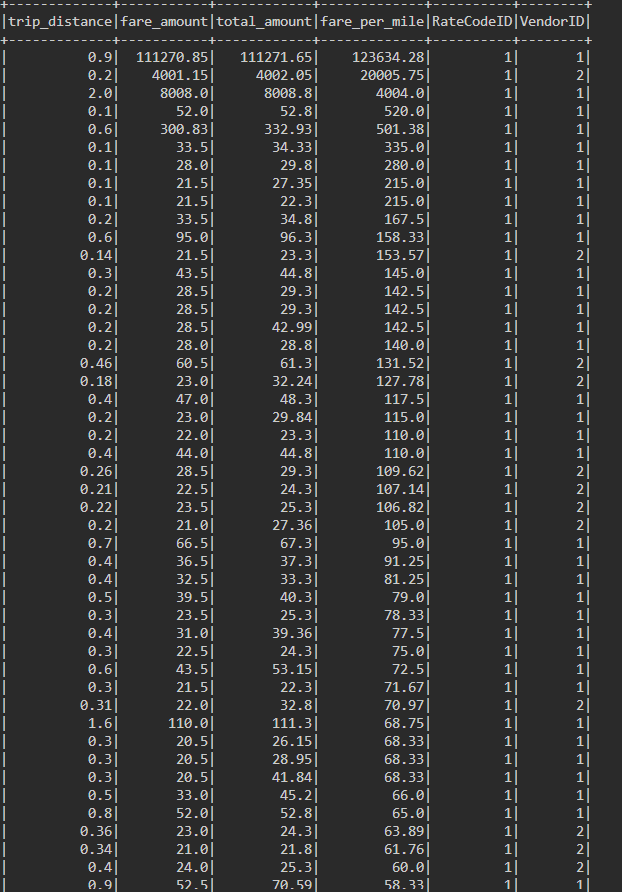
Bảng 4: Kết quả truy vấn thứ tám

* Các loại hình thanh toán phổ biến (1 & 2: thẻ tín dụng và tiền mặt) có các khoản phí bổ sung trung bình khá tương đồng, không có sự khác biệt lớn.
* avg\_extra, avg\_mta\_tax, avg\_improvement\_surcharge đều gần bằng nhau.
* Riêng phí cầu đường (avg\_tolls), thẻ tín dụng (0.34) cao hơn tiền mặt (0.20), có thể do các chuyến qua cầu/đường thu phí thường thanh toán bằng thẻ.
* payment\_type = 3 và 4 (ít phổ biến hơn) cũng có mức phí tương tự, ngoại trừ avg\_tolls của type 4 cao hơn một chút (0.38).
* payment\_type = 5 chỉ có 1 chuyến, các chỉ số không có ý nghĩa thống kê (avg\_extra = 1.00, avg\_tolls = 0).
* Dữ liệu chủ yếu ổn định, không có dấu hiệu bất thường lớn trong các khoản phí bổ sung giữa các nhóm thanh toán trên.
* Với lệnh truy vấn thứ chín, ta sẽ tìm các chuyến đi có khoảng cách ngắn nhưng giá cao (có thể gian lận)

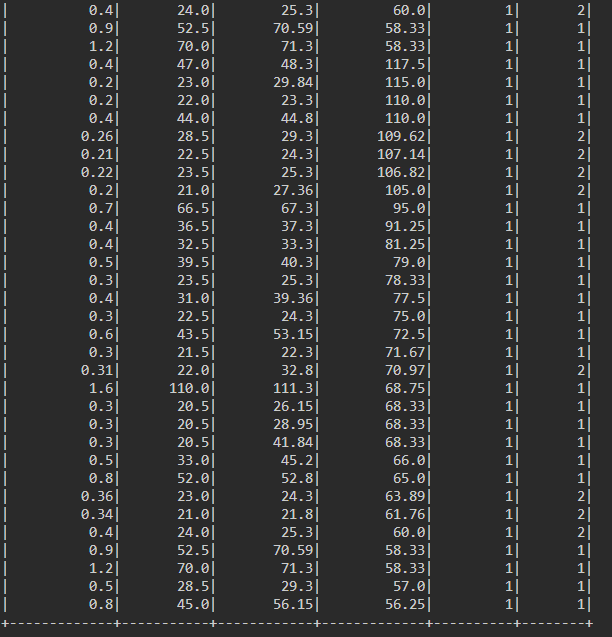
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Lệnh truy vấn thứ chin



Hình : Kết quả truy vấn thứ chin (phần 1)



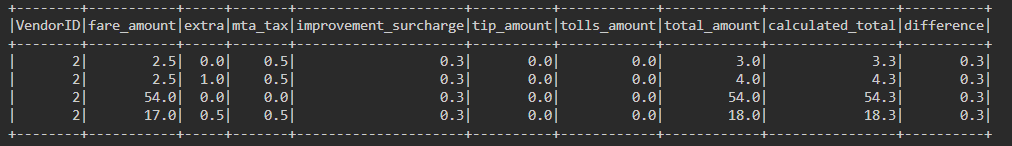
Hình : Kết quả truy vấn thứ chin (phần 2)

* Đơn giá mỗi dặm cực kỳ cao. Rất nhiều chuyến đi ngắn (dưới 1 dặm, thậm chí 0.1–0.9 dặm) nhưng Fare\_amount lên đến hàng nghìn, thậm chí hàng chục nghìn USD, khiến fare\_per\_mile lên tới hàng trăm, hàng nghìn USD/dặm. Ví dụ: 0.9 dặm với Fare\_amount 111,270.85 USD ⇒ fare\_per\_mile 123,634.28 USD/dặm.
* Xuất hiện nhiều giá trị bất thường (outlier) - những giá trị này vượt quá thực tế thông thường của dịch vụ taxi, có thể do lỗi nhập liệu, phụ phí đặc biệt, hoặc tính nhầm/phạt.
* RateCodeID và VendorID - tất cả các chuyến đều có RateCodeID = 1 (giá tiêu chuẩn). VendorID chủ yếu là 1 hoặc 2, đại diện cho hai nhà cung cấp lớn.
* Truy vấn này giúp nhanh chóng phát hiện các trường hợp bất thường, lỗi nhập dữ liệu, hoặc các giao dịch khả nghi. Các giá trị fare\_per\_mile quá cao thường là dấu hiệu dữ liệu sai lệch hoặc cần được xác minh lại. Những chuyến đi này nên được kiểm tra kỹ và có thể loại khỏi phân tích nếu muốn đánh giá trung thực về hoạt động, định giá, hoặc hành vi khách hàng.
* Với code truy vấn thứ mười, ta kiểm tra tính toán chính xác của tổng tiền trong dữ liệu taxi, tìm ra những chuyến đi có sự chênh lệch giữa tổng tiền được ghi nhận và tổng tiền tính toán lại.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ 10



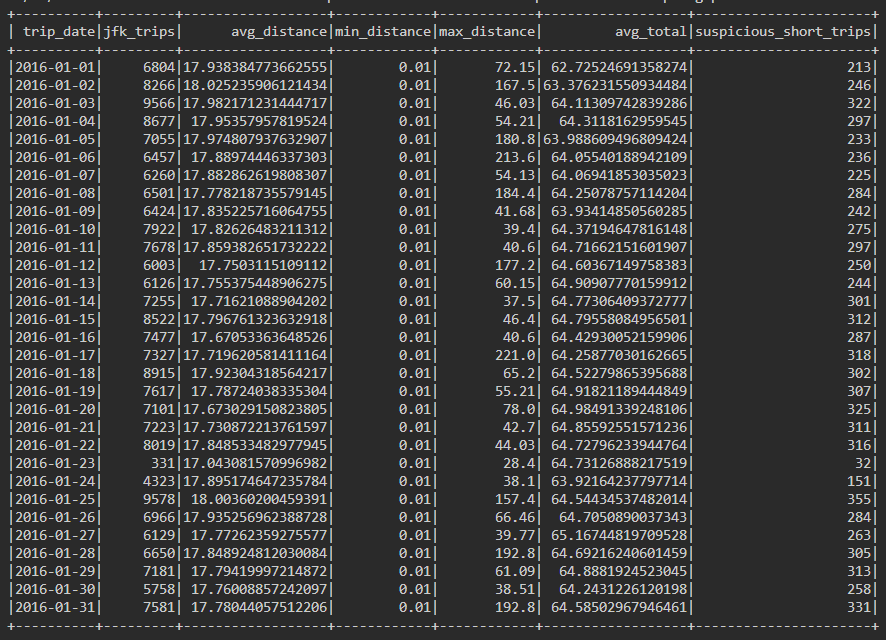
Hình : Kết quả truy vấn thứ 10

* Bảng kết quả gồm các chuyến có difference = 0.3, tức là tổng các khoản phí thành phần và tổng số tiền ghi nhận chênh nhau 0.3 USD.
* Ví dụ:
* Chuyến đầu tiên: Tổng các khoản = 3.3 USD, nhưng total\_amount chỉ là 3.0 USD, lệch 0.3 USD.
* Các dòng còn lại đều gặp tình trạng tương tự: calculated\_total lớn hơn total\_amount đúng 0.3 USD.
* Điều này cho thấy có thể có một khoản điều chỉnh, làm tròn, hoặc lỗi ghi nhận tổng tiền cuối cùng.
* Với code truy vấn thứ 11, ta sẽ phân tích các chuyến taxi áp dung mã giá cuối cùng được áp dụng tại thời điểm kết thúc chuyến đi lúc di chuyển đến sân bay JFK theo từng ngày trong tháng 1/2016, tập trung vào việc phát hiện các chuyến đi bất thường hoặc đáng ngờ.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ 11



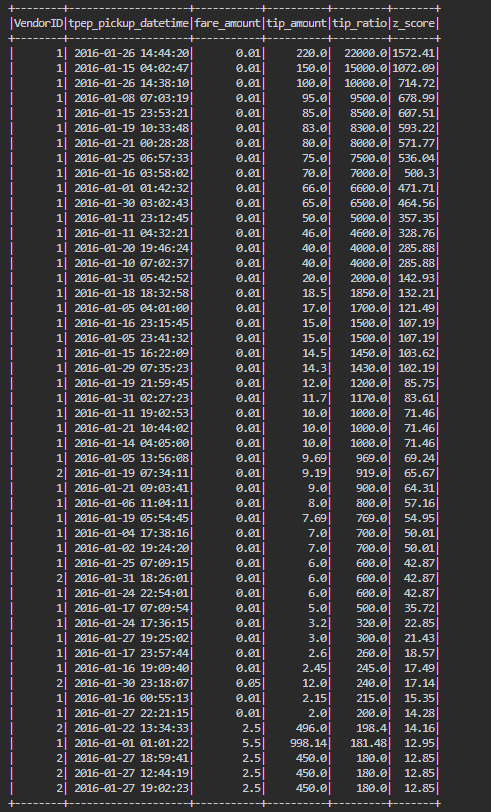
Hình : Kết quả truy vấn thứ 11

* Số chuyến JFK mỗi ngày (jfk\_trips):
* Mỗi ngày có từ hơn 6.000 đến gần 10.000 chuyến đi theo giá JFK.
* Giá trị này khá ổn định trong cả tháng.
* Quãng đường trung bình (avg\_distance):
* Quãng đường trung bình của các chuyến JFK luôn dao động quanh mức 17.7–17.9 dặm mỗi ngày, phù hợp với khoảng cách thực tế giữa Manhattan và JFK.
* Quãng đường ngắn nhất (min\_distance):
* Giá trị này luôn là 0.01 dặm, rất nhỏ, rõ ràng là dữ liệu bất thường vì không ai đi JFK mà chỉ đi 0.01 dặm.
* Quãng đường dài nhất (max\_distance):
* Có sự dao động lớn, từ 28.4 đến hơn 200 dặm tùy ngày. Một số chuyến có thể bị nhập sai hoặc thực sự có hành trình đặc biệt xa.
* Tổng số tiền trung bình (avg\_total):
* Dao động quanh mức 62–65 USD cho mỗi chuyến JFK, phù hợp vớigiá sàn taxi giữa Manhattan và JFK.
* Số chuyến nghi ngờ bất thường (suspicious\_short\_trips):
* Mỗi ngày có vài trăm chuyến JFK có quãng đường dưới 10 dặm (ví dụ ngày 1/1 là 213 chuyến, ngày 2/1 là 322 chuyến, v.v.).
* Đây là dấu hiệu bất thường vì đa số chuyến đi JFK phải trên 10 dặm, các chuyến ngắn như vậy có thể là lỗi nhập liệu, gian lận hoặc gán nhầm mã giá JFK.
* Với code truy vấn thứ 12, ta sẽ tìm các chuyến đi có tip cao bất thường so với fare

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

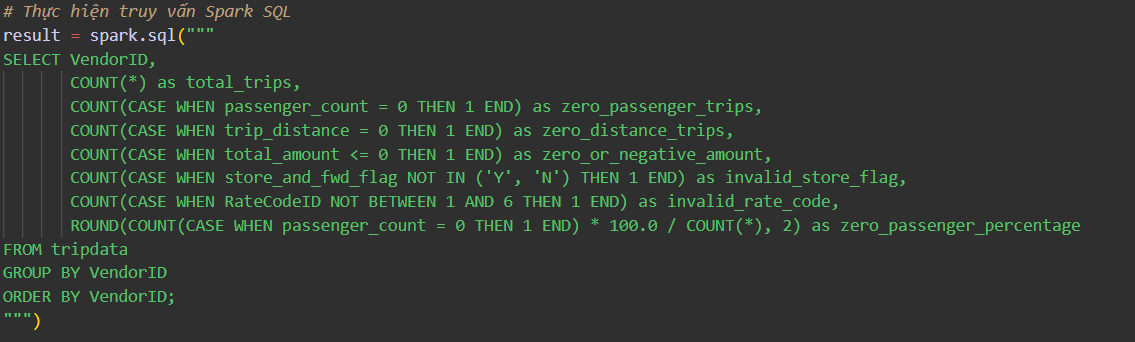
Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ 12

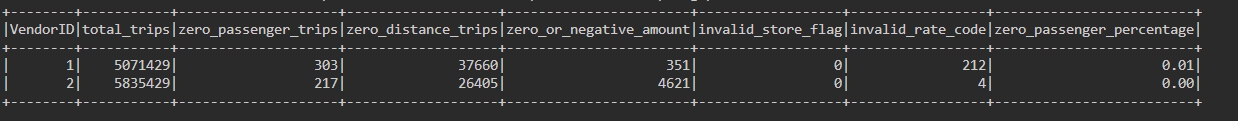


Hình : Kết quả truy vấn thứ 12

* Hầu hết các dòng dữ liệu đều có fare\_amount rất nhỏ (0.01), trong khi tip\_amount lại rất lớn (từ 10 đến 220 USD), dẫn đến tip\_ratio cực kỳ cao (từ 1,000 đến 22,000) và z-score cũng cực lớn (lên đến hơn 1,500).
* Ví dụ: chuyến đầu tiên tip 220 USD cho chuyến giá chỉ 0.01 USD, tip\_ratio tới 22,000, z\_score 1,572.41 (rất bất thường).
* Các trường hợp khác cũng tương tự: tip\_amount gấp hàng trăm, hàng nghìn lần fare\_amount.
* Một số dòng cuối có fare\_amount cao hơn (2.5–5.5 USD), tip\_amount lên đến 998.14 USD (vẫn rất lớn so với giá cước), tip\_ratio > 180, z-score > 12.
* Tất cả các chuyến này đều bị flag là anomaly (bất thường) vì vượt xa mức tip thông thường trong bộ dữ liệu.
* Với code truy vấn thứ 13, phân tích dữ liệu theo vendor và tìm sự không nhất quán và bất thường dữ liệu trong bộ dữ liệu



Hình : Code truy vấn dữ liệu 13



Hình : Kết quả truy vấn 13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **VendorID** | **total\_trips** | **zero\_passenger\_trips** | **zero\_distance\_trips** | **zero\_or\_negative\_amount** | **invalid\_store\_flag** | **invalid\_rate\_code** | **zero\_passenger\_percentage** |
| 1 | 5,071,429 | 303 | 37,660 | 35,1 | 0 | 212 | 0.01% |
| 2 | 5,835,429 | 217 | 26,405 | 4,621 | 0 | 4 | 0.00% |

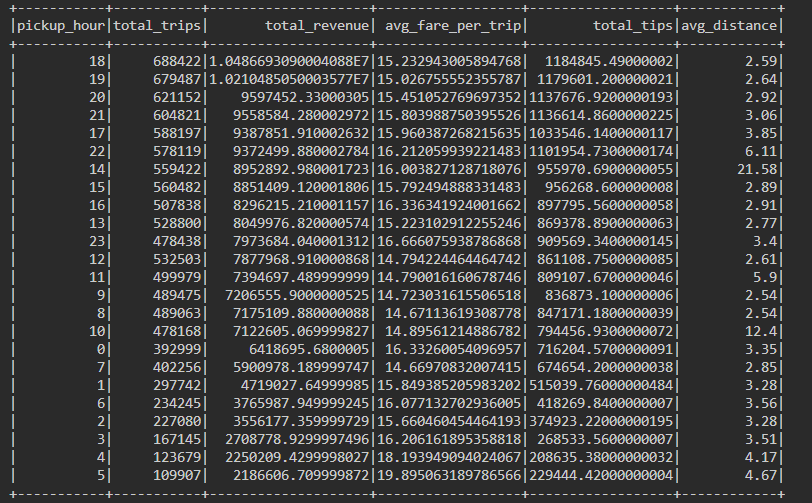
Bảng 5: Kết quả truy vấn 13

* Số chuyến có số hành khách bằng 0 (zero\_passenger\_trips) cực kỳ nhỏ, chỉ 0.01% hoặc ít hơn trên tổng số chuyến.
* Số chuyến có quãng đường bằng 0 (zero\_distance\_trips) cũng khá thấp so với tổng số chuyến, nhưng vẫn có hàng chục nghìn bản ghi (37,660 và 26,405).
* Số chuyến có tổng tiền <= 0 (zero\_or\_negative\_amount) có 351 ở VendorID 1 và 4,621 ở VendorID 2, đây là các bản ghi có thể cần loại bỏ khi phân tích doanh thu.
* Không có bản ghi nào có Store\_and\_fwd\_flag không hợp lệ.
* Số bản ghi có RateCodeID không hợp lệ rất nhỏ (212 với VendorID 1, chỉ 4 với VendorID 2).
* Tỷ lệ chuyến có số hành khách bằng 0 gần như bằng 0.
* Với code truy vấn thứ 14, ta sẽ phân tích doanh thu theo giờ và tìm ra giờ mà mang lại doanh thu nhiều nhất

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn thứ 14



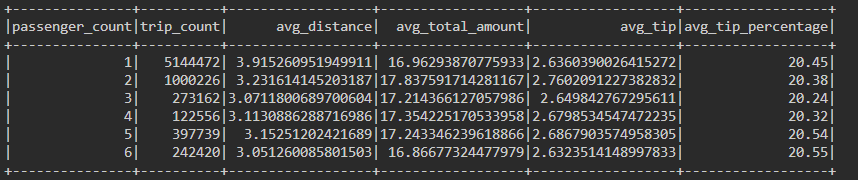
Hình : Kết quả truy vấn thứ 14

* Khung giờ cao điểm doanh thu và số chuyến:
* Các giờ có tổng doanh thu và số chuyến cao nhất là từ 18h đến 21h (tối). Cụ thể, 18h có nhiều chuyến nhất (688.422 chuyến, doanh thu hơn 10 triệu USD), tiếp theo là 19h, 20h, 21h.
* Đây là các khung giờ tan tầm, nhu cầu taxi tăng cao.
* Tiền tip và doanh thu trung bình/chuyến:
* Tổng tiền tip cũng tập trung cao vào buổi tối (18h–21h), mỗi giờ đều trên 1 triệu USD tip.
* Doanh thu trung bình mỗi chuyến dao động khoảng 15–16 USD vào khung giờ cao điểm, tăng lên hơn 19 USD vào lúc 4–5h sáng (lúc này số chuyến rất ít).
* Quãng đường trung bình:
* Phần lớn các giờ trong ngày, quãng đường trung bình mỗi chuyến khoảng 2.5–3.5 dặm.
* Một số khung giờ như 22h (6.11 dặm), 14h (21.85 dặm), 5h (4.67 dặm) có giá trị cao bất thường, có thể do số chuyến đặc biệt dài hoặc có dữ liệu bất thường.
* Các khung giờ ít hoạt động:
* Rạng sáng (3–5h) là thời điểm số chuyến taxi ít nhất, tổng doanh thu và tip cũng thấp nhất.
* Tuy nhiên, doanh thu trung bình mỗi chuyến lại cao hơn (có thể do một số chuyến đi dài ra sân bay hoặc ngoại thành).
* Với lệnh truy vấn cuối cùng, mình sẽ phân tích hành vi khách hàng dựa theo số lượng hành khách trong xe

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Code truy vấn cuối cùng



Hình : Kết quả truy vấn cuối cùng

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Passenger\_count** | **trip\_count** | **avg\_distance** | **avg\_total\_amount** | **avg\_tip** | **avg\_tip\_percentage** |
| 1 | 5,144,472 | 3.92 | 16.96 | 2.64 | 20.45% |
| 2 | 1,000,226 | 3.23 | 17.84 | 2.76 | 20.38% |
| 3 | 27,162 | 3.07 | 17.21 | 2.65 | 20.24% |
| 4 | 122,556 | 3.11 | 17.35 | 2.68 | 20.32% |
| 5 | 397,739 | 3.15 | 17.24 | 2.69 | 20.54% |
| 6 | 242,420 | 3.05 | 16.87 | 2.63 | 20.55% |

Bảng : Kết quả truy vấn cuối cùng

* Số chuyến đi có 1 hành khách là lớn nhất, giảm dần khi số hành khách tăng lên.
* Quãng đường trung bình mỗi chuyến dao động quanh 3–4 dặm, không khác biệt nhiều giữa các nhóm.
* Tổng số tiền trung bình mỗi chuyến lớn nhất cho nhóm 2 hành khách (~17.84 USD), còn lại quanh 17 USD.
* Tiền tip trung bình mỗi chuyến khoảng 2.6–2.7 USD cho mọi nhóm.
* Tỉ lệ tip trung bình so với tiền cước luôn ổn định quanh mức 20.2%–20.6% cho mọi nhóm hành khách.

1. **Tổng kết**

Sau khi triển khai và thực thi thành công 15 câu truy vấn Spark SQL trên bộ dữ liệu NYC Yellow Taxi Trip Data, quá trình phân tích đã mang lại nhiều thông tin giá trị như sau:

* ***Chất lượng dữ liệu được kiểm soát tốt:*** Nhờ các truy vấn kiểm tra dữ liệu bất thường (ví dụ như thời gian đón lớn hơn thời gian trả khách, hoặc tọa độ ngoài phạm vi hợp lý), các bản ghi lỗi đã được phát hiện để loại bỏ, đảm bảo dữ liệu đầu vào cho phân tích là tin cậy.
* ***Phân tích theo thời gian và không gian:*** Các truy vấn đã giúp xác định được các khung giờ cao điểm, các tuyến đường phổ biến nhất, và sự khác biệt về nhu cầu di chuyển giữa các ngày, những khung giờ. Nhờ đó, có thể thấy rõ mô hình di chuyển và nhu cầu sử dụng taxi tại New York.
* ***Hiệu quả kinh doanh của các nhà cung cấp:*** So sánh giữa các VendorID cho thấy sự khác biệt về số lượng chuyến đi, doanh thu trung bình, tỉ lệ thanh toán qua thẻ tín dụng, cũng như thời lượng chuyến đi.
* ***Hành vi khách hàng:*** Thông qua phân tích số lượng hành khách, khoảng cách chuyến đi, và xu hướng tip theo từng nhóm khách, đã phát hiện những nhóm khách hàng thường xuyên sử dụng dịch vụ, mức tip trung bình, và những trường hợp tip cao bất thường.
* ***Phân loại chuyến đi và hiệu suất:*** Các truy vấn đã phân loại chuyến đi theo độ dài, khoảng cách, và loại mã cước (RateCodeID), giúp phát hiện các nhóm chuyến đi đặc biệt như đi sân bay JFK, Newark, hoặc các khu vực lân cận.
* ***Các yếu tố ảnh hưởng đến doanh thu:*** Qua việc phân tích sự đóng góp của các khoản phụ phí, thuế, tip, và phí cầu đường, có thể xác định các yếu tố chính ảnh hưởng đến tổng doanh thu của từng chuyến đi.
* ***Xu hướng thanh toán:*** Việc so sánh giữa các hình thức thanh toán giúp nhận diện sự chuyển dịch từ tiền mặt sang thẻ tín dụng, đồng thời phân tích mối liên hệ giữa hình thức thanh toán và hành vi tip của khách.
* ***Phát hiện các mẫu bất thường:*** Một số truy vấn còn giúp phát hiện các trường hợp có thể là gian lận hoặc nhập sai dữ liệu, như các chuyến đi cự ly quá ngắn nhưng tiền cước cao, hoặc tip vượt quá mức trung bình nhiều lần.

Thông qua các truy vấn Spark SQL này, quá trình phân tích bộ dữ liệu NYC Yellow Taxi Trip Data không chỉ dừng lại ở việc kiểm soát chất lượng dữ liệu mà còn khai thác được nhiều khía cạnh về vận hành, hành vi khách hàng, hiệu quả kinh doanh và các yếu tố ảnh hưởng đến doanh thu. Đây là nền tảng vững chắc để đưa ra các quyết định quản lý, tối ưu hóa dịch vụ và định hướng phát triển cho doanh nghiệp taxi tại thành phố New York. Bây giờ ta sẽ bắt đầu tiến tới xây dựng mô hình học máy với bộ dữ liệu trên qua công cụ SparkMLlib

1. **Xây dựng mô hình Machine Learning qua SparkMLlib**
2. Tổng quan về SparkMLlib

MLlib là một thư viện machine learning được tích hợp sẵn trong Apache Spark, cung cấp một loạt các công cụ và thuật toán machine learning để xây dựng và triển khai các mô hình trên dữ liệu phân tán. Dưới đây là một số điểm quan trọng cần hiểu về MLlib:

* *Tích hợp với Apache Spark:* MLlib được tích hợp sâu vào Apache Spark, cho phép bạn xây dựng và triển khai các mô hình machine learning trực tiếp trên dữ liệu phân tán mà không cần di chuyển dữ liệu ra khỏi Spark. Bằng cách tích hợp MLlib với Apache Spark, chúng ta có thể thực hiện các công việc phức tạp như huấn luyện mô hình machine learning trên dữ liệu lớn một cách dễ dàng và hiệu quả.
* *Thuật toán machine learning đa dạng:* MLlib cung cấp một loạt các thuật toán machine learning phổ biến như hồi quy tuyến tính, phân loại, gom cụm, học tăng cường, phân tích chuỗi thời gian, và nhiều hơn nữa. Điều này giúp bạn áp dụng các phương pháp machine learning phù hợp với nhu cầu của mình.
* *Tối ưu hóa hiệu suất:* MLlib được thiết kế để tận dụng các tính năng in-memory và xử lý song song của Apache Spark, giúp tăng tốc độ xử lý dữ liệu và huấn luyện mô hình trên dữ liệu lớn.
* *Tích hợp với Spark SQL và DataFrames:* MLlib tích hợp chặt chẽ với Spark SQL và DataFrames, cho phép bạn sử dụng dữ liệu đã được xử lý trong Spark SQL và DataFrames để huấn luyện và đánh giá mô hình machine learning.
* *Hỗ trợ nhiều ngôn ngữ lập trình*: MLlib hỗ trợ nhiều ngôn ngữ lập trình như Scala, Java, Python và R, giúp bạn làm việc với MLlib trong môi trường phát triển mà bạn thoải mái nhất.
* *Tích hợp với công cụ phân tích dữ liệu khác:* MLlib tích hợp chặt chẽ với các công cụ và thư viện phân tích dữ liệu khác trong hệ sinh thái Spark như Spark SQL, GraphX, và Streaming, giúp bạn kết hợp các phương tiện phân tích dữ liệu khác nhau để thực hiện các nhiệm vụ phức tạp trong lĩnh vực Big Data.

Tóm lại, MLlib là một thư viện machine learning mạnh mẽ trong Apache Spark, cung cấp các công cụ và thuật toán machine learning để xây dựng và triển khai các mô hình trên dữ liệu phân tán. Điều này làm cho việc phát triển và triển khai các ứng dụng machine learning trở nên dễ dàng và hiệu quả hơn trong lĩnh vực Big Data.

1. Thực nghiệm

Từ bộ dữ liệu NYC Yellow Taxi Trip Data, ta sẽ bắt đầu xây dựng mô hình bài toàn Học máy trên tập dữ liệu đấy, qua công cụ thư viện Mllib của Apache Spark

* Bài toán đầu tiên nhóm muốn giới thiệu là dự đoán tiền boa (sử dụng Supervised Learning)
* Mục tiêu: Xây dựng mô hình dự đoán số tiền boa mà khách hàng sẽ trả dựa trên các đặc trưng của chuyến đi.
* Thuộc tính sử dụng:
* Trip\_distance
* Fare\_amount
* Passenger\_count
* tip\_amount – biến mục tiêu
* Thuật toán sử dụng: Linear Regression
* Về Linear Regression, thuật toán này dễ triển khai, dễ giải thích kết quả. Có thể nhanh chóng kiểm tra xem các đặc trưng (feature) có mối quan hệ tuyến tính với tiền boa hay không. Hiệu quả với dữ liệu tuyến tính và cho kết quả tham chiếu (baseline) để so sánh với các mô hình phức tạp hơn.
* Bước 1: Khai báo những thư viện sau
* findspark
* pyspark
* SparkContext, SparkConf
* SparkSession
* sql.functions: Cung cấp các hàm xử lý dữ liệu như group, filter, aggregate... Dùng để thao tác, biến đổi dữ liệu trong DataFrame.
* ml.feature.VectorAssembler: Dùng để kết hợp nhiều cột thành một vector đặc trưng, chuẩn bị dữ liệu cho mô hình học máy.
* ml.regression.LinearRegression: Thuật toán hồi quy tuyến tính, dùng để xây dựng mô hình dự đoán giá trị số.
* ml.evaluation.RegressionEvaluator: Dùng để đánh giá hiệu quả mô hình hồi quy (sử dụng các chỉ số như RMSE, MAE...).

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Khai báo thư viện cho bài toán dự đoán tiền boa

* Bước 2: ta nâng cấp RAM được dùng trong Spark, và tiến hành khởi tạo SparkSession và đọc dữ liệu từ HDFS
* Bước 3: ta sẽ làm sạch dữ liệu, thống kê mô tả và tạo features

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình : Làm sạch dữ liệu, thống kê mô tả và tạo features cho bài toán dự đoán tiền boa

* Làm sạch dữ liệu (Data Cleaning): df\_clean = df.filter(...).select(...) - lọc ra các dòng dữ liệu hợp lệ, loại bỏ các giá trị nhỏ hơn 0, khoảng cách <= 0, chỉ lấy payment\_type = 1. Sau đó chỉ giữ lại các cột cần thiết để máy tính học vào. Đảm bảo dữ liệu đầu vào là hợp lệ, loại bỏ outlier và nhiễu, tránh làm sai lệch kết quả mô hình.
* Thống kê mô tả (Descriptive Statistics): df\_clean.describe().show() - tính toán các chỉ số thống kê như mean, stddev, min, max... cho từng cột dữ liệu đã làm sạch. Giúp kiểm tra nhanh dữ liệu, phát hiện bất thường, hiểu sơ bộ về dữ liệu trước khi modeling.
* Tạo features (Feature Engineering): VectorAssembler(...).transform(df\_clean) - kết hợp các cột số thành một vector features. Chuẩn bị dữ liệu đúng định dạng cho các thuật toán học máy của PySpark (yêu cầu 1 cột kiểu vector).
* Bước 4: Bắt đầu huấn luyện mô hình

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Huấn luyện mô hình hồi quy dự đoán tiền boa

* Chia train/test: train\_df, test\_df = df\_features.randomSplit([0.8, 0.2], seed=42) - dữ liệu được chia thành 2 phần: 80% để huấn luyện (train\_df) và 20% để kiểm tra (test\_df), đảm bảo ngẫu nhiên với seed=42 để tái lập kết quả. Đảm bảo mô hình học trên dữ liệu huấn luyện và đánh giá khách quan trên dữ liệu chưa từng thấy (test) và tránh overfitting (quá khớp).
* Huấn luyện mô hình: lr = LinearRegression(featuresCol="features", labelCol="tip\_amount"), model = lr.fit(train\_df) - tạo một mô hình hồi quy tuyến tính, lấy cột “features” làm đặc trưng đầu vào và “tip\_amount” làm biến mục tiêu và huấn luyện mô hình trên tập train\_df. LinearRegression là thuật toán dự đoán giá trị số,còn fit(train\_df) giúp mô hình học được mối quan hệ giữa đặc trưng và nhãn từ dữ liệu thực tế.
* Dự đoán: predictions = model.transform(test\_df) - áp dụng mô hình vừa huấn luyện để dự đoán kết quả trên dữ liệu kiểm tra (test\_df). Ta sẽ kiểm tra khả năng tổng quát hóa của mô hình trên dữ liệu mới và đánh giá chất lượng mô hình trước khi đưa vào sử dụng thực tế.
* Bước 5: Đánh giá mô hình

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Đánh giá mô hình dự đoán tiền boa

* Đánh giá mô hình: Tạo một đối tượng RegressionEvaluator để đánh giá mô hình hồi quy dựa trên nhãn thực tế (tip\_amount) và giá trị dự đoán (prediction). Ta sẽ tính toán hai chỉ số quan trọng như sau: RMSE (Root Mean Squared Error) - Đánh giá độ lệch trung bình giữa giá trị thực tế và dự đoán, giá trị càng thấp, mô hình càng tốt. R² (R squared) - đo lường mức độ giải thích phương sai của dữ liệu bởi mô hình (giá trị gần 1 là tốt). Đây là 2 chỉ số thông dụng nhất để đánh giá chất lượng của mô hình hồi quy. Nếu ta không đo lường, bạn sẽ không biết mô hình dự đoán tốt hay dở.
* Ta sẽ in kết quả ra màn hình
* Hiển thị một số dự đoán mẫu: Hiển thị 10 dòng dự đoán mẫu gồm giá trị thực tế, giá trị dự đoán và các đặc trưng liên quan, giúp bạn kiểm tra trực tiếp output của model. Ta sẽ kiểm tra xem mô hình dự đoán có hợp lý không, phát hiện lỗi hoặc bất thường trong dự đoán.
* spark.stop() - Kết thúc phiên làm việc với Spark, giải phóng tài nguyên.
* Về kết quả của bài toán này, nhóm sẽ đánh giá ở phần sau: Đánh giá và tổng kết.
* Bài toán tiếp theo là bài toán phân cụm khu vực đón khách (sử dụng Unsupervised Learning)
* Mục tiêu: Sử dụng K-Means để phân cụm các điểm đón khách thành các khu vực có đặc điểm tương tự.
* Thuộc tính sử dụng:
* Pickup\_longitude, Pickup\_latitude
* Thuật toán sử dụng: K-means
* K-Means là thuật toán phân cụm phổ biến nhất, giúp chia các điểm dữ liệu (ở đây là các điểm đón khách) thành các nhóm (clusters) sao cho các điểm trong cùng một nhóm gần nhau nhất (theo khoảng cách Euclidean).
* Đặc trưng Pickup\_longitude, Pickup\_latitude là tọa độ không gian hai chiều (2D), rất phù hợp cho K-Means vì thuật toán này hoạt động hiệu quả với dữ liệu tọa độ/phân bố không gian.
* Bước 1: Khai báo những thư viện sau:
* findspark
* pyspark
* SparkContext, SparkConf
* SparkSession
* sql.functions
* ml.feature.VectorAssembler
* ml.clustering.KMeans - thuật toán phân cụm KMeans, dùng để nhóm các điểm dữ liệu thành các cụm (clusters) dựa trên độ tương đồng.
* ml.evaluation.ClusteringEvaluator - đánh giá hiệu quả của mô hình phân cụm (cluster) bằng các chỉ số như silhouette score.



Hình : Khai báo thư viện cho bài toán phân cụm

* Bước 2: ta nâng cấp RAM được dùng trong Spark, và tiến hành khởi tạo SparkSession và đọc dữ liệu từ HDFS
* Bước 3: Lọc dữ liệu, lấy mẫu dữ liệu và tạo features

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Lọc dữ liệu, lấy mẫu và tạo features cho bài toán phân cụm

* Lọc dữ liệu vị trí hợp lệ (trong NYC): Lọc các điểm đón khách chỉ nằm trong khoảng vĩ độ, kinh độ tương ứng với khu vực thành phố New York (NYC). Sau đó chỉ giữ lại hai cột vị trí là pickup\_longitude và pickup\_latitude. Ta sẽ loại bỏ các điểm dữ liệu bất thường hoặc nằm ngoài phạm vi nghiên cứu (NYC), giúp dữ liệu "sạch" và phân tích chính xác hơn, tránh nhiễu khi mô hình hóa hoặc phân tích không liên quan.
* Lấy mẫu để tăng tốc độ xử lý: df\_sample = df\_location.sample(0.01, seed=42) - lấy ngẫu nhiên 1% dữ liệu từ tập dữ liệu vị trí, giúp giảm lượng dữ liệu cần xử lý. Khi dữ liệu quá lớn, việc lấy mẫu giúp giảm thời gian xử lý, thử nghiệm mô hình nhanh chóng mà vẫn giữ được tính đại diện. Ta có thể đặt seed để kết quả lấy mẫu có thể lặp lại (reproducible).
* Tạo vector features cho học máy: Kết hợp hai cột longitude và latitude thành một vector features dùng để đưa vào thuật toán học máy (ví dụ: phân cụm, phân loại...).
* Bước 4: Triển khai thuật toán K-means

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Triển khai thuật toán K-means

* Thử nghiệm các số cụm khác nhau (k): Dùng thuật toán K-Means để thử nghiệm phân cụm dữ liệu với các giá trị k khác nhau (số lượng cụm: 5, 10, 15, 20). Việc thử nhiều giá trị k giúp tìm ra số cụm phù hợp nhất cho dữ liệu, tránh chia quá ít hoặc quá nhiều cụm.
* Đánh giá bằng Silhouette Score: Sử dụng ClusteringEvaluator để tính Silhouette Score, một chỉ số cho biết mức độ phân biệt giữa các cụm (càng cao càng tốt). Lưu lại k và điểm silhouette để chọn phương án tối ưu. Silhouette Score là tiêu chí chuẩn để đánh giá chất lượng phân cụm.
* Chọn k tốt nhất dùng Python builtins: Dùng hàm max trong thư viện builtins để chọn ra giá trị k có Silhouette Score cao nhất. Nó sẽ xuất ra số cụm tốt nhất và điểm số tương ứng. Phải đảm bảo tự động chọn ra k tối ưu nhất, không cần kiểm tra thủ công. Ta sẽ bổ sung thêm thư viện builtins - thư viện chuẩn Python, cung cấp hàm max.
* Bước 5: Huấn luyện mô hình và đánh giá mô hình

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Huấn luyện mô hình phân cụm

* Huấn luyện mô hình cuối cùng với số cụm tối ưu: Tạo và huấn luyện mô hình phân cụm KMeans với số cụm tốt nhất (best\_k) vừa tìm được, trên toàn bộ dữ liệu đặc trưng và dự đoán cụm cho từng điểm dữ liệu. Đảm bảo mô hình cuối cùng sử dụng đúng số cụm tối ưu, cho kết quả phân cụm chính xác nhất trên toàn bộ dữ liệu. final\_predictions chứa thông tin mỗi điểm thuộc về cụm nào, dùng cho phân tích tiếp theo.
* Phân tích các cụm: Gom nhóm dữ liệu theo nhãn cụm (prediction), tính số lượng điểm trong mỗi cụm và tính trung bình vị trí (longitude, latitude) của từng cụm. Xong rồi hiển thị bảng thống kê cho từng cụm. Ta phải hiểu rõ đặc trưng và quy mô của từng cụm sau khi phân cụm, kiểm tra xem các cụm có hợp lý, cân bằng hoặc trùng lặp không.
* Hiển thị tâm các cụm: Lấy tọa độ tâm của mỗi cụm (cluster center) và in ra vị trí của từng tâm. Giúp hình dung vị trí đại diện của từng cụm trên bản đồ/thực tế và phục vụ cho việc trực quan hóa hoặc ứng dụng thực tế (ví dụ: xác định các vùng tập trung đón khách).
* Dừng Spark: spark.stop()
* Về kết quả của bài toán này, nhóm sẽ đánh giá ở phần sau: Đánh giá và tổng kết.
* Bài toán cuối cùng nhóm muốn giới thiệu là phân loại hình thức thanh toán
* Mục tiêu: Dự đoán khách hàng sẽ thanh toán bằng thẻ tín dụng hay tiền mặt.
* Đặc trưng sử dụng:
* Fare\_amount
* Trip\_distance
* Tip\_amount
* passenger\_count
* Thuật toán sử dụng: Logistic Regression
* Logistic Regression là một trong những thuật toán kinh điển và hiệu quả nhất cho các bài toán phân loại nhị phân (binary classification), bài toán lần này là dự đoán khách hàng sẽ thanh toán bằng thẻ tín dụng hay tiền mặt là sẽ chỉ dùng 2 giá trị nhị phân 0 và 1, rất phù hợp với Logistic Regression.
* Bước 1: Khai báo những thư viện sau:
* findspark
* pyspark
* SparkContext, SparkConf
* SparkSession
* sql.functions
* ml.feature.VectorAssembler
* pyspark.ml.classification.LogisticRegression - thuật toán hồi quy logistic để phân loại nhị phân hoặc đa lớp. Phù hợp cho các bài toán phân loại (ví dụ: dự báo khách hàng có mua hàng hay không, email spam hay không spam, v.v).
* pyspark.ml.evaluation BinaryClassificationEvaluator - đánh giá mô hình phân loại nhị phân (ví dụ: accuracy, AUC, F1...), MulticlassClassificationEvaluator - đánh giá mô hình phân loại đa lớp (ví dụ: accuracy, F1, weightedPrecision...). Giúp kiểm tra chất lượng mô hình phân loại, xác định mô hình có dự báo tốt hay không.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Khai báo thư viện cho bài toán dự đoán hình thức thanh toán

* Bước 2: ta nâng cấp RAM được dùng trong Spark, và tiến hành khởi tạo SparkSession và đọc dữ liệu từ HDFS
* Bước 3: Tiền xử lý dữ liệu và tạo features

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Tiền xử lý dữ liệu và tạo features cho bài toán dự đoán hình thức thanh toán

* Chuẩn bị dữ liệu – chỉ lấy thẻ tín dụng (1) và tiền mặt (2): Lọc dữ liệu chỉ giữ lại các chuyến đi thanh toán bằng thẻ tín dụng hoặc tiền mặt, đồng thời loại bỏ các dòng có giá trị không hợp lệ (fare\_amount hoặc trip\_distance nhỏ hơn hoặc bằng 0). Chỉ lấy những cột cần thiết cho bài toán, giúp mô hình dễ học và tăng độ chính xác.
* Chuyển payment\_type thành binary (1: thẻ tín dụng, 0: tiền mặt): Tạo một cột mới là is\_credit\_card, gán giá trị 1 nếu thanh toán bằng thẻ tín dụng, 0 nếu là tiền mặt. Ta sẽ đưa bài toán về dạng phân loại nhị phân (binary classification), phù hợp với các thuật toán học máy phân loại.
* Thống kê phân bố: Đếm số lượng giao dịch theo từng loại thanh toán (thẻ tín dụng hoặc tiền mặt), từ đó kiểm tra dữ liệu có cân bằng giữa hai loại thanh toán không, giúp đánh giá chất lượng dữ liệu đầu vào cho mô hình học máy.
* Tạo features
* Bước 4: Huấn luyện mô hình và đánh giá mô hình

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình : Huấn luyện và đánh giá mô hình bài toán dự đoán hình thức thanh toán

* Chia train/test: train\_df, test\_df = df\_features.randomSplit([0.8, 0.2], seed=42) - chia dữ liệu thành 80% để huấn luyện (train) và 20% để kiểm tra (test), đảm bảo tính ngẫu nhiên với seed cố định để tái lập kết quả.Phải đảm bảo mô hình học trên tập train và đánh giá trên tập test, tránh overfitting và kiểm tra được khả năng tổng quát hóa.
* Huấn luyện mô hình Logistic Regression: lr = LogisticRegression(featuresCol="features", labelCol="is\_credit\_card"), model = lr.fit(train\_df) - tạo mô hình hồi quy logistic dựa trên các đặc trưng (features) để dự đoán nhãn nhị phân is\_credit\_card, xong rồi ta huấn luyện mô hình trên dữ liệu train. fit(train\_df) giúp mô hình học được mối quan hệ giữa đặc trưng và nhãn.
* Dự đoán trên tập test: predictions = model.transform(test\_df) - áp dụng mô hình vừa huấn luyện để dự đoán nhãn trên tập test, từ đó đánh giá khả năng dự đoán của mô hình trên dữ liệu chưa từng thấy.
* Đánh giá mô hình: Với AUC Score, ta sẽ đánh giá mô hình bằng chỉ số AUC (Area Under ROC Curve) – thước đo chất lượng phân loại nhị phân, càng gần 1 càng tốt. AUC là chỉ số chuẩn để đánh giá mô hình phân loại nhị phân, đặc biệt khi dữ liệu không cân bằng. Còn Accuracy là đo độ chính xác (accuracy) của mô hình: tỷ lệ dự đoán đúng trên tổng số mẫu. Đây là chỉ số đơn giản, trực quan để đánh giá mô hình phân loại.
* In kết quả
* Bước 5: Phân tích hệ số và hiển thị mẫu dự đoán

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình : Phân tích hệ số và hiển thị mẫu dự đoán cho bài toán dự đoán hình thức thanh toán

* Phân tích hệ số mô hình: Lấy các hệ số (coefficients) của mô hình Logistic Regression vừa huấn luyện, xong rồi gán tên các đặc trưng (feature\_names) tương ứng với từng hệ số, cuối cùng in ra giá trị hệ số của từng đặc trưng. Hệ số (coefficient) cho biết mức độ ảnh hưởng của từng đặc trưng lên xác suất dự đoán kết quả (ở đây là xác suất thanh toán bằng thẻ tín dụng). Từ đó giúp giải thích mô hình, hiểu được yếu tố nào quan trọng hơn (feature importance), hỗ trợ giải thích và kiểm chứng mô hình.
* Hiển thị mẫu dự đoán: Hiển thị 10 dòng đầu tiên gồm nhãn thực tế, dự đoán của mô hình, xác suất dự đoán, và các thông tin liên quan của giao dịch. Ta sẽ kiểm tra và so sánh giữa giá trị thực tế và dự đoán của mô hình, từ đó rà soát xem mô hình hoạt động có hợp lý không, phát hiện sớm các trường hợp bất thường.
* spark.stop(): Kết thúc phiên làm việc với Spark, giải phóng tài nguyên.
* Về kết quả của bài toán này, nhóm sẽ đánh giá ở phần sau: Đánh giá và tổng kết.

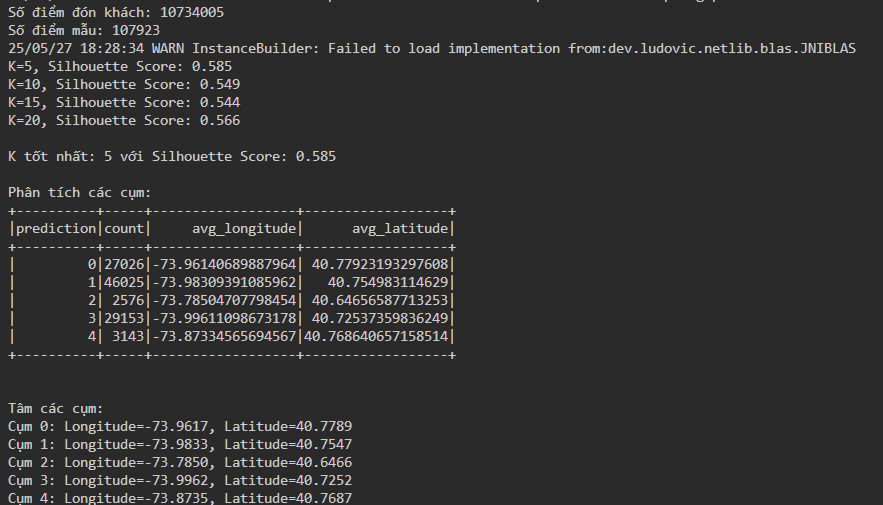
1. **Đánh giá và tổng kết**

* Về bài toán dự đoán tiền boa, sau khi huấn luyện xong nhóm có kết quả như sau:



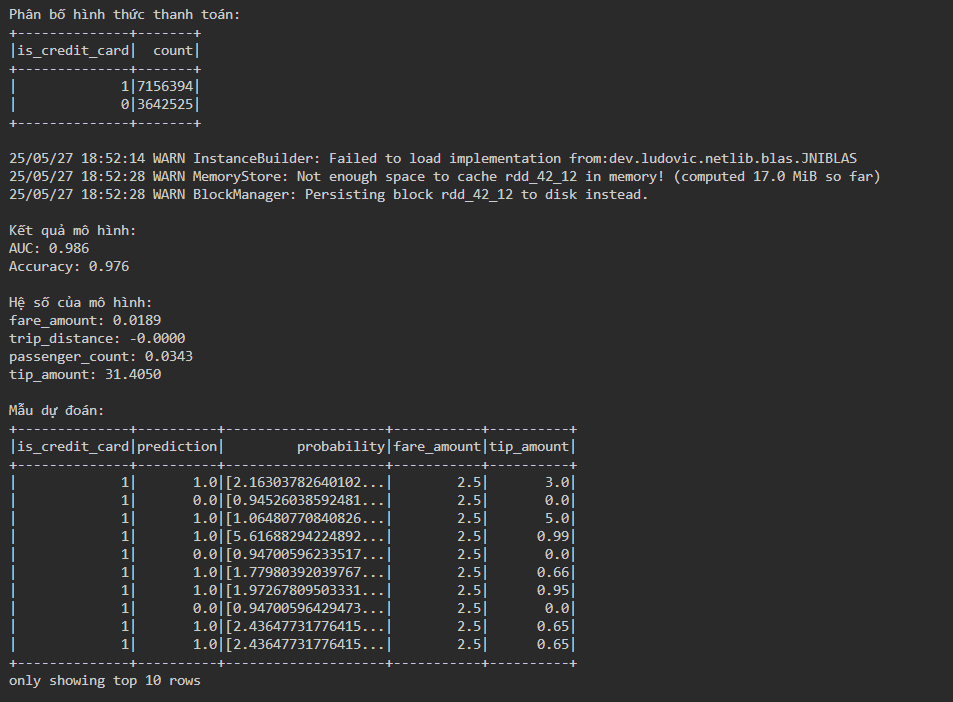
Hình : Kết quả huấn luyện bài toán dự đoán tiền boa

* RMSE (Root Mean Squared Error) = 1.795: Giá trị này cho thấy sai số trung bình giữa giá trị dự đoán và thực tế là khoảng 1.8. Với các giá trị tip\_amount thường nhỏ (phần lớn là 0), đây là mức sai số khá lớn.
* R² = 0.552: Mô hình chỉ giải thích được khoảng 55.2% phương sai của dữ liệu thực. Như vậy, mô hình còn khá nhiều hạn chế trong việc dự đoán tiền boa.
* Dự đoán gần như đồng nhất: Nhìn vào bảng mẫu, giá trị tip\_amount thực tế đều là 0, nhưng mô hình dự đoán đều trong khoảng 0.72–0.81. Điều này cho thấy mô hình linear regression đang gặp hiện tượng “underfitting”, dự đoán các giá trị khá giống nhau và không bắt được sự biến động thực tế.
* Về mặt hạn chế thì độ chính xác chưa cao: RMSE lớn và R² chỉ ở mức trung bình.
* Không phản ánh được các giá trị tiền boa thực tế: Khi tiền boa thực tế là 0, nhưng mô hình vẫn dự đoán giá trị dương.
* Thiếu các biến đầu vào quan trọng: Mô hình chỉ dùng fare\_amount, trip\_distance, passenger\_count. Trong thực tế, tip phụ thuộc vào nhiều yếu tố khác như thời gian, địa điểm, ngày trong tuần, dịch vụ, thời tiết, v.v.
* Dữ liệu mất cân bằng: Phần lớn tiền boa là 0, mô hình khó học được quy luật đúng.
* Về những giải pháp trong tương lai, ta sẽ thêm nhiều đặc trưng (feature engineering): Bổ sung các thuộc tính như thời gian bắt đầu/kết thúc chuyến đi, vị trí đón/trả khách, loại xe, ngày trong tuần, v.v.
* Thử các mô hình phức tạp hơn: Sử dụng Random Forest, Gradient Boosting, hoặc các mô hình phi tuyến khác để bắt được quan hệ phức tạp hơn giữa các biến.
* Xử lý dữ liệu mất cân bằng: Áp dụng các kỹ thuật như oversampling, undersampling, hoặc chuyển sang bài toán phân loại (dự đoán khách có tip hay không).
* Chuẩn hóa/biến đổi dữ liệu: Áp dụng scaling, log-transform với các biến có phân phối lệch.
* Đánh giá trên nhiều chỉ số: Không chỉ RMSE, nên xem xét thêm MAE, Precision, Recall (nếu chia tip thành có/không).
* Như vậy, mô hình hiện tại có kết quả hạn chế do thiếu biến đầu vào quan trọng và chỉ dùng Linear Regression. Để cải thiện, cần thêm nhiều đặc trưng, thử các mô hình phức tạp hơn và xử lý tốt hơn bài toán mất cân bằng dữ liệu tip.
* Còn với bài toán phân cụm khu vực đón khách, ta có kết quả như sau



Hình : Kết quả huấn luyện bài toán phân cụm khu vực đón khách

* Số lượng điểm đón rất lớn (10,734,005), mẫu lấy ra là 107,923 điểm (~1%).
* Đã thử nghiệm các giá trị K: 5, 10, 15, 20.
* Silhouette Score cao nhất đạt được là 0.585 với K=5, các K khác đều thấp hơn (chênh lệch không nhiều).
* Số lượng điểm trong các cụm phân bố không đồng đều, nhưng không quá chênh lệch.
* Tâm cụm được xác định rõ ràng, phản ánh các khu vực tập trung điểm đón khách (có thể là sân bay, trung tâm thương mại, nhà ga...).
* Về điểm hạn chế, Silhouette Score chưa cao (chỉ ở mức trung bình = 0.59), cho thấy phân cụm chưa thực sự tách biệt, có thể các cụm còn chồng lấn lên nhau.
* Chỉ dùng hai thuộc tính vị trí (kinh độ, vĩ độ), bỏ qua các yếu tố khác như thời gian, ngày trong tuần, điều kiện thời tiết, loại phương tiện...
* Không kiểm tra trực quan trên bản đồ để đánh giá ý nghĩa thực tiễn của các cụm.
* Không đánh giá ý nghĩa thực tế của từng cụm (ví dụ: cụm nào là sân bay, cụm nào là Manhattan,…).
* Việc lấy mẫu ngẫu nhiên 1% có thể làm mất đi các đặc trưng nhỏ nhưng quan trọng (outlier hoặc cụm nhỏ).
* Chưa thử nghiệm các thuật toán phân cụm khác ngoài KMeans (ví dụ: DBSCAN, Gaussian Mixture…).
* Về phương án cải thiện trong tương lai, bổ sung thêm thuộc tính: Thêm thời gian đón, loại phương tiện, ngày trong tuần để phân cụm đa chiều hơn (spatio-temporal clustering).
* Trực quan hóa trên bản đồ: Visualize các cụm trên bản đồ thực tế để kiểm tra ý nghĩa địa lý của từng cluster.
* Thử nghiệm nhiều thuật toán khác: Áp dụng DBSCAN (phân cụm mật độ), Gaussian Mixture Model để so sánh hiệu quả với KMeans.
* Phân tích chi tiết hơn trong từng cụm: Với mỗi cụm, phân tích sâu các đặc trưng như thời gian đón phổ biến, số lượng khách, v.v.
* Tối ưu hóa chọn K: Thử thêm các giá trị K, hoặc sử dụng phương pháp Elbow để chọn K tối ưu hơn.
* Làm sạch dữ liệu kỹ hơn: Loại bỏ các điểm đón bất thường, vị trí lạ, hoặc kiểm tra kỹ điều kiện lọc.
* Phân tích outlier: Kiểm tra các điểm không thuộc cụm nào rõ ràng để phát hiện điểm bất thường.
* Như vậy, mô hình đã cho ra các cụm hợp lý với K=5, nhưng mức phân tách chưa thực sự tốt (Silhouette Score = 0.59), cần bổ sung thêm đặc trưng, trực quan hóa và thử nhiều phương pháp khác nhằm nâng cao chất lượng phân cụm và giá trị khai thác thực tế.
* Với bài toán cuối cùng là phân loại hình thức thanh toán, nhóm có kết quả huấn luyện như sau:



Hình : Kết quả huấn luyện phân loại hình thức thanh toán

* Thanh toán bằng thẻ tín dụng chiếm đa số: 71,563,941 (tương ứng với 1),
* Tiền mặt: 36,425,251 (tương ứng với 0).
* Điều này cho thấy dữ liệu có phần mất cân bằng nhưng chưa quá nghiêm trọng.
* AUC = 0.986 (gần 1): mô hình phân biệt rất tốt hai hình thức thanh toán.
* Accuracy = 0.976 (97.6%): độ chính xác rất cao.
* Mẫu dự đoán đều rất sát với giá trị thực tế (is\_credit\_card = 1 → prediction = 1).
* fare\_amount: 0.0189 (ảnh hưởng nhỏ, dương)
* trip\_distance: gần như 0 (không ảnh hưởng)
* passenger\_count: 0.0343 (ảnh hưởng nhỏ)
* tip\_amount: 31.4050 (ảnh hưởng rất lớn, dương). Điều này cho thấy tip\_amount là đặc trưng mạnh nhất quyết định hành vi trả bằng thẻ tín dụng (vì tip thường chỉ có khi thanh toán bằng thẻ).
* Mẫu dự đoán minh họa hầu hết các dòng có prediction là 1, xác suất rất cao, phù hợp với biến đầu ra thực tế.
* Về những điểm hạn chế, đặc trưng tip\_amount quá mạnh: tip\_amount có thể “leak” thông tin nhãn (data leakage), vì ở phần lớn các chuyến đi trả tiền mặt, tip\_amount có thể là 0, còn trả bằng thẻ thì thường có tip. Việc này khiến mô hình dễ đạt accuracy/AUC cao nhưng khả năng áp dụng thực tế thấp: nếu biết tip\_amount thì việc nhận diện hình thức thanh toán gần như là hiển nhiên.
* Một số feature khác chưa khai thác: trip\_distance và passenger\_count có hệ số gần 0, ảnh hưởng yếu. Chưa thử các thuộc tính như thời gian, ngày trong tuần, vị trí đón/trả khách,...
* Vấn đề mất cân bằng dữ liệu: Dù chưa nghiêm trọng, nhưng số lượng chuyến trả bằng thẻ tín dụng vẫn gấp đôi tiền mặt, có thể gây bias cho mô hình.
* Chưa đánh giá các chỉ số khác: Chưa có confusion matrix, precision, recall, F1-score để đánh giá tổng thể hơn.
* Về những giải pháp cải tiến trong tương lai: Loại bỏ hoặc điều chỉnh tip\_amount: Nếu muốn mô hình thực sự dự đoán hình thức thanh toán, không nên dùng tip\_amount (trừ khi mục tiêu là kiểm tra chất lượng học máy, không phải ứng dụng thực tế). Nếu vẫn dùng tip\_amount, nên làm rõ mục tiêu mô hình là gì.
* Bổ sung đặc trưng: Thêm các đặc trưng về thời gian, địa điểm, loại xe, ngày trong tuần, v.v.
* Cân bằng lại dữ liệu: Áp dụng undersampling/oversampling hoặc kỹ thuật weight để giảm bias nếu cần.
* Đánh giá toàn diện hơn: Thêm confusion matrix, precision, recall, F1-score, ROC curve.
* Thử nghiệm mô hình khác: Thử Random Forest, Gradient Boosting, hoặc các mô hình phi tuyến để so sánh.
* Kiểm tra và ngăn data leakage: Đảm bảo các đặc trưng đầu vào không vô tình chứa thông tin nhãn.
* Như vậy, mô hình đạt chỉ số rất cao nhờ tip\_amount là đặc trưng mạnh, nhưng đây lại là dấu hiệu của việc rò rỉ thông tin nhãn (data leakage). Để mô hình có giá trị dự báo thực tế hơn, nên loại bỏ tip\_amount hoặc kiểm soát kỹ các đặc trưng, đồng thời bổ sung thêm các yếu tố khác và đánh giá kỹ hơn trên các chỉ số đa dạng.

Như vậy, nhóm đã thực hiện sử dụng thư viện MLlib để xây dựng mô hình thuật toán để huấn luyện những bài toán khác nhau cho bộ dữ liệu trên và đã cho một số kết quả nhất định. Dù kết quả đấy vẫn chưa thể nào thỏa mãn được như mong muốn của chúng em, nhưng nhóm tin rằng, với những giải pháp cải tiến trong tương lai, kết quả huấn luyện của nhóm sẽ khả quan hơn rất nhiều. Bây giờ, nhóm xin phép sang chương cuối cùng.

1. **CHƯƠNG IV: ĐỀ XUẤT NHỮNG VẤN ĐỀ CÓ THỂ DÙNG DỮ LIỆU LỚN ĐỂ GIẢI QUYẾT VÀ KÉT LUẬN**

Xuyên suốt các chương trước, chúng ta đã cùng nhau tìm hiểu tổng quan về Dữ liệu lớn (Big Data), thực nghiệm cài đặt, cấu hình và sử dụng các công cụ Apache Hadoop để lưu trữ dữ liệu và Apache Spark để xử lý, truy vấn cũng như xây dựng các mô hình học máy cơ bản trên bộ dữ liệu "NYC Yellow Taxi Trip Data". Chương này sẽ tổng kết lại toàn bộ quá trình thực hiện tiểu luận, đánh giá những kết quả đạt được, chỉ ra những hạn chế còn tồn tại và đề xuất những vấn đề tiềm năng có thể ứng dụng Dữ liệu lớn để giải quyết, cũng như các hướng phát triển cho đề tài trong tương lai.

1. **Đề xuất những vấn đề có thể áp dụng dữ liệu lớn để giải quyết**

Sự bùng nổ của Dữ liệu lớn đã và đang mở ra vô vàn cơ hội để giải quyết các bài toán phức tạp trong nhiều lĩnh vực khác nhau của đời sống. Dựa trên những kiến thức đã được trình bày về đặc điểm (7Vs), tầm quan trọng và các ứng dụng thực tiễn của Big Data, cùng với kinh nghiệm thực hành trên bộ dữ liệu taxi, nhóm xin đề xuất một số vấn đề có thể ứng dụng hiệu quả công nghệ Dữ liệu lớn, Apache Hadoop và Apache Spark:

* ***Cải thiện chăm sóc sức khỏe và y tế cá nhân hóa:***
* Phân tích hồ sơ bệnh án điện tử, dữ liệu hình ảnh y khoa và thiết bị đeo (wearables) để phát hiện sớm bệnh lý, tối ưu phác đồ điều trị và theo dõi bệnh nhân từ xa.
* Ví dụ: Big Data Analytics hỗ trợ dự báo diễn biến bệnh và giảm thiểu biến chứng trong điều trị ung thư hay tim mạch thông qua mô hình học máy trên dữ liệu lâm sàng quy mô lớn
* ***Dự phòng bảo trì (Predictive Maintenance) trong sản xuất và công nghiệp 4.0:***
* Thu thập và phân tích tín hiệu từ cảm biến máy móc, dữ liệu vận hành qua thời gian thực để dự đoán sự cố, lập kế hoạch bảo trì chủ động, giảm thiểu thời gian chết máy (downtime) và tối ưu chi phí.
* Nghiên cứu cho thấy Big Data Analytics đã được áp dụng thành công trong việc tối ưu hóa bảo trì động cơ tuabin và dây chuyền sản xuất, giúp giảm 20–30% chi phí bảo trì
* ***Tối ưu hóa chuỗi cung ứng và logistics:***
* Phân tích dữ liệu đơn hàng, tồn kho, vận tải và điều kiện thời tiết để dự báo nhu cầu, lập lộ trình giao hàng tối ưu và giảm thiểu lãng phí.
* Ví dụ: Sử dụng mô hình học máy trên dữ liệu lớn cho phép các tập đoàn bán lẻ cắt giảm 15–20% chi phí vận hành kho bãi và giao nhận
* ***Phát hiện gian lận và an ninh tài chính:***
* Phân tích luồng giao dịch thẻ tín dụng, hành vi người dùng và dữ liệu mạng để phát hiện bất thường, ngăn chặn gian lận trong thời gian thực.
* Hệ thống AI của các ngân hàng lớn như Mastercard đã tăng tỷ lệ phát hiện giao dịch gian lận lên đến 300% và giảm đáng kể tỷ lệ từ chối giao dịch oan uổng
* ***Cá nhân hóa trải nghiệm khách hàng và tiếp thị thông minh:***
* Phân tích dữ liệu tương tác người dùng (clickstream, lịch sử mua hàng, phản hồi mạng xã hội) để đưa ra khuyến nghị sản phẩm, chiến dịch marketing theo nhóm khách hàng mục tiêu.
* Ứng dụng này giúp doanh nghiệp tăng doanh thu thêm 10–15% thông qua mô hình đề xuất (recommendation engine)
* ***Quản lý giao thông thông minh và đô thị bền vững:***
* Thu thập dữ liệu cảm biến giao thông, camera, GPS phương tiện để phân tích luồng xe, điều khiển tín hiệu đèn thông minh, giảm ùn tắc và ô nhiễm.
* Nhiều thành phố lớn đã ứng dụng nền tảng Big Data để giảm 20–25% thời gian di chuyển trung bình của người dân
* ***Giám sát và ứng phó biến đổi khí hậu:***
* Phân tích dữ liệu khí tượng, thủy văn, ô nhiễm không khí và biển để dự báo lũ lụt, hạn hán, diễn biến ô nhiễm, từ đó lập phương án ứng phó kịp thời.
* Các startup như Sinay sử dụng AI xử lý dữ liệu đại dương quy mô lớn, giúp giám sát môi trường biển và giảm thiểu tác động sinh thái
* ***Nông nghiệp chính xác (Precision Agriculture):***
* Ứng dụng dữ liệu vệ tinh, cảm biến đất đai và khí hậu để tối ưu lịch gieo trồng, tưới tiêu, bón phân, giảm lãng phí nước và phân bón, nâng cao năng suất cây trồng.
* Dự án nông nghiệp thông minh tại nhiều nước đã cho thấy tăng năng suất từ 10–30% trong khi giảm 15–20% chi phí đầu vào
* ***Phân tích học tập và giáo dục thông minh (Learning Analytics):***
* Phân tích dữ liệu tương tác học trực tuyến, kết quả bài tập, mức độ tương tác trong lớp để cá nhân hóa chương trình đào tạo, dự báo nguy cơ bỏ học và tư vấn kịp thời.
* Các trường đại học sử dụng hệ thống Big Data để nâng cao tỷ lệ tốt nghiệp, giảm 10–12% tỷ lệ bỏ học giữa chừng
* ***Quản lý rủi ro và tuân thủ quy định trong lĩnh vực bảo hiểm:***
* Phân tích dữ liệu thông tin cá nhân, lịch sử khiếu nại, dữ liệu IoT (ví dụ như thiết bị giám sát sức khỏe) để đánh giá rủi ro, định giá sản phẩm bảo hiểm chính xác và phát hiện gian lận.
* Báo cáo khoa học chỉ ra Big Data đã giúp ngành bảo hiểm cải thiện mức lợi nhuận hoạt động lên đến 5–8% nhờ tối ưu hóa đánh giá rủi ro

Những đề xuất trên chỉ là một phần nhỏ trong vô vàn những ứng dụng tiềm năng của Dữ liệu lớn. Việc triển khai thành công các giải pháp này đòi hỏi sự kết hợp giữa công nghệ mạnh mẽ như Hadoop, Spark với kiến thức chuyên môn sâu rộng trong từng lĩnh vực cụ thể.

1. **Kết luận**

Xuyên suốt tiểu luận này, nhóm đã thực hiện một hành trình khám phá toàn diện về thế giới Dữ liệu lớn, từ những khái niệm nền tảng đến ứng dụng thực tiễn. Bắt đầu từ việc tìm hiểu tổng quan về sự phát triển và tầm quan trọng của Big Data với 7 đặc điểm cốt lõi (7 "Vs"), nhóm đã nhận thức rõ những lợi ích to lớn mà công nghệ này mang lại cho nhiều lĩnh vực đời sống.

Trọng tâm của tiểu luận là quá trình thực nghiệm với bộ dữ liệu "NYC Yellow Taxi Trip Data". Nhóm đã thành công trong việc cài đặt và cấu hình Apache Hadoop, một nền tảng vững chắc cho việc lưu trữ dữ liệu lớn, và đã tiến hành đẩy toàn bộ bộ dữ liệu lên Hadoop Distributed File System (HDFS). Tiếp đó, Apache Spark đã được cài đặt và cấu hình để xử lý và phân tích dữ liệu. Sử dụng Spark SQL, nhóm đã thực hiện thành công 15 câu truy vấn đa dạng, từ đó khám phá và làm sạch dữ liệu, phát hiện các điểm bất thường, đồng thời rút ra nhiều thông tin giá trị về xu hướng hoạt động, hành vi khách hàng và hiệu quả kinh doanh của dịch vụ taxi tại New York.

Bên cạnh đó, nhóm đã bước đầu ứng dụng thư viện Spark MLlib để xây dựng ba mô hình học máy cơ bản: dự đoán tiền boa (Linear Regression), phân cụm khu vực đón khách (K-Means), và phân loại hình thức thanh toán (Logistic Regression). Mặc dù các mô hình ban đầu cho ra những kết quả nhất định và chỉ ra những hạn chế cần cải thiện, quá trình này đã mang lại kinh nghiệm quý báu về việc áp dụng học máy trên dữ liệu lớn.

Dù đã nỗ lực và đạt được những mục tiêu đề ra, bài làm chắc chắn không tránh khỏi thiếu sót. Tuy nhiên, qua quá trình thực hiện tiểu luận, nhóm đã thu được những kiến thức và kỹ năng quan trọng về công nghệ Dữ liệu lớn, đặc biệt là với Apache Hadoop và Apache Spark, làm nền tảng cho việc học tập và nghiên cứu sâu hơn trong tương lai.

# **TÀI LIỆU THAM KHẢO**

1. Mohammad Shahid Husain, Mohammad Zunnun Khan, & Siddiqui, T. (2023). Big Data Concepts, Technologies, and Applications (1st ed.). CRC Press.
2. Heitman, S. (2022, May 5). What Happens in an Internet Minute? [2022 Statistics]. LOCALiQ. <https://localiq.com/blog/what-happens-in-an-internet-minute/>
3. Statista. (2023, August 22). Data Created Worldwide 2010-2025 . Statista; Statista. <https://www.statista.com/statistics/871513/worldwide-data-created/>
4. Elcom. (2023, July 17). Big Data là gì? Lợi ích và ứng dụng quan trọng của Dữ liệu lớn. Elcom; Elcom. <https://www.elcom.com.vn/big-data-la-gi-ung-dung-du-lieu-lon-trong-thuc-tien-1648786616>
5. Elemento. (2022, May). NYC Yellow Taxi Trip Data. Kaggle. <https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data>
6. NYC Taxi & Limousine Commission (TLC). (2015). TLC Trip Record Data. NYC.gov; NYC.gov. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
7. Phan Thanh Vinh. (2023, August 27). Tìm Hiểu Về Hadoop, HDFS, Hadoop MapReduce (Lý Thuyết). Viblo; Viblo. <https://viblo.asia/p/tim-hieu-ve-hadoop-hdfs-hadoop-mapreduce-ly-thuyet-5pPLkjNZJRZ>
8. Duy Thanh CSE. (2021, January 1). Cài đặt Hadoop trên Windows. Duy Thanh CSE; Duy Thanh CSE. <https://duythanhcse.wordpress.com/2021/01/01/cai-dat-hadoop-tren-windows/>
9. Apache Hadoop. (2020, July 6). HDFS Architecture. Apache Hadoop; <https://hadoop.apache.org/docs/r3.3.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>
10. Yasar, K., Rosencrance, L., & Vaughan, J. (2024, April 19). Hadoop Distributed File System (HDFS). Search Data Management; TechTarget. <https://www.techtarget.com/searchdatamanagement/definition/Hadoop-Distributed-File-System-HDFS>
11. Phuc Ngoc Nghia. (2020, March 22). Tìm hiểu về Apache Spark. Viblo.asia; <https://viblo.asia/p/tim-hieu-ve-apache-spark-ByEZkQQW5Q0>
12. Jevtic, G. (2024, October 10). How to Install Apache Spark on Windows. PhoenixNAP. <https://phoenixnap.com/kb/install-spark-on-windows-10>
13. Thống PM. (2024, June 3). Spark SQL trong Apache Spark. Viblo. <https://viblo.asia/p/spark-sql-trong-apache-spark-pgjLN25PL32>
14. Trannguyenhan. (2021, September 2). Phân tích dữ liệu bán lẻ với Spark SQL. De Manejar. <https://demanejar.github.io/posts/retail-data-analytics-with-spark-sql/>
15. Jaiswal, A. (2024, March 21). End-to-End Beginners Guide on Spark SQL in Python. Analytics Vidhya; Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2022/04/end-to-end-beginners-guide-on-spark-sql-in-python/>
16. Thống PM [@Juu-dev]. (2024, June 3). Thư viện MLlib trong Apache Spark. Viblo. <https://viblo.asia/p/thu-vien-mllib-trong-apache-spark-3kY4g0mOJAe>
17. Sharma, S. (2023, June 20). Machine learning with Pyspark MLlib: Part 1 Regression. Medium. <https://sharmashorya1996.medium.com/machine-learning-with-pyspark-mllib-part-1-regression-e7ad4d9780af>
18. GeeksforGeeks. (2025, April 24). K-Means Clustering using PySpark Python. GeeksforGeeks. <https://www.geeksforgeeks.org/k-means-clustering-using-pyspark-python/>
19. GeeksforGeeks. (2025, April 24). Logistic Regression using PySpark Python. GeeksforGeeks. Retrieved May 28, 2025, from <https://www.geeksforgeeks.org/logistic-regression-using-pyspark-python/>
20. Batko, K., & Ślęzak, A. (2022). The use of Big Data Analytics in healthcare. Journal of Big Data, 9, Article 3. <https://doi.org/10.1186/s40537-021-00553-4>
21. Jones, J. (2024, December 21). Big Data Analytics in Industry 4.0: A Systematic Review of Use Cases, Challenges, and Future Directions [Preprint]. SSRN. <https://doi.org/10.2139/ssrn.5067663>
22. Dataforest. (2024, September 3). Top 20 Highly Effective Use Cases of Big Data Analytics for Businesses in 2024. Dataforest.ai Blog; Dataforest.ai. <https://dataforest.ai/blog/big-data-analytics-use-cases>
23. Insider Studios. (2025, May 22). From fighting fraud to fueling personalization, AI at scale is redefining how commerce works online. Business Insider; Insider Inc. <https://www.businessinsider.com/sc/how-ai-at-scale-is-shaping-the-future-of-commerce>
24. Shruti, M. (2025, May 11). 14 Big Data Examples & Applications Across Industries. Simplilearn. <https://www.simplilearn.com/tutorials/big-data-tutorial/big-data-applications>
25. Financial Times. (2024, May 16). AI start-ups take aim at climate change. Financial Times. <https://www.ft.com/content/2385e6ab-26ed-4e98-b70c-1d968246ea69>
26. ProjectPro. (2024, October 28). 6 Big Data Use Cases – How Companies Use Big Data? ProjectPro. Retrieved May 28, 2025, from <https://www.projectpro.io/article/5-big-data-use-cases-how-companies-use-big-data/155>
27. Tableau. (2025). Big Data In Action: Use Cases and Examples. Tableau; Tableau. <https://www.tableau.com/learn/articles/big-data-examples-use-cases>
28. Ellili, N. O. D., Alsaiari, L., Shanti, H., Hillebrand, B. L., Hassanain, N., Elfout, L., & Nobanee, H. (2023, November). The applications of big data in the insurance industry: A bibliometric and systematic review of relevant literature. The Journal of Finance and Data Science, 9, Article 100102. <https://doi.org/10.1016/j.jfds.2023.100102>