

ĐẠI HỌC UEH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH



TIỂU LUẬN CUỐI KỲ

**ĐỀ TÀI: PHÂN TÍCH HÀNH VI SỬ DỤNG ỨNG DỤNG AI DI ĐỘNG
(MOBILE AI APP USAGE)**

Môn học: Dữ liệu lớn và Ứng dụng

Mã lớp học phần: 25D1INF50907903

Giáo viên hướng dẫn: TS. Võ Văn Hải

Nhóm: 14

Hồ Chí Minh, ngày 31 tháng 5 năm 2025

THÀNH VIÊN NHÓM 14

Tên thành viên	MSSV	Công việc	Mức độ hoàn thành
Nguyễn Trần Minh Nhật	31221023438	<ul style="list-style-type: none"> - Format doc - Slide - Truy vấn SparkSQL (1, 4, 7, 10) - Spark MLlib - Chương 3 	100%
Trần Duy Khoa	31221022154	<ul style="list-style-type: none"> - Xây dựng dàn bài - Truy vấn SparkSQL (2, 5, 8,11) - Spark MLlib - Trực quan hóa Spark MLlib - Chương 4 và Chương 5 	100%
Nguyễn Vũ Phước	31221025186	<ul style="list-style-type: none"> - Slide - Truy vấn SparkSQL (3, 6, 9, 12) - Spark MLlib - Chương 1 và Chương 2 	100%

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI.....	7
1.1. Lý do chọn đề tài.....	7
1.2. Mục tiêu nghiên cứu	7
1.3. Ý nghĩa của nghiên cứu	8
1.4. Đối tượng và phạm vi nghiên cứu	8
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	10
2.1. Giới thiệu về Dữ liệu lớn (Big Data)	10
2.2. Ứng dụng của Big Data.....	10
2.3. Hệ sinh thái công nghệ xử lý Big Data: Hadoop và Spark	11
2.4. Phân tích hành vi người dùng ứng dụng di động	11
2.5. Kỹ thuật phân cụm dữ liệu trong Big Data	12
2.6. Hồi quy Logistic trong phân tích dữ liệu lớn.....	12
2.7. Về dữ liệu giả lập (Synthetic Data) trong nghiên cứu khoa học.....	12
CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM HỆ THỐNG BIG DATA.....	14
3.1. Tổng quan triển khai	14
3.2. Cài đặt và cấu hình Hadoop	14
3.2.1. Chuẩn bị môi trường.....	14
3.2.2. Cấu hình Hadoop.....	16
3.2.3. Kiểm tra hệ thống Hadoop	19
3.3. Sinh và nạp dữ liệu lên HDFS	20
3.3.1. Tạo dữ liệu giả lập bằng Python.....	20
3.3.2. Upload dữ liệu lên HDFS	21
3.3.3. Kiểm tra dữ liệu đã upload	23
3.4. Cài đặt và cấu hình Apache Spark	24
3.4.1. Tải và cài đặt Spark	24
3.4.2. Khởi động Spark.....	27
3.5. Đọc dữ liệu từ HDFS vào Spark	28
3.5.1. Khởi tạo SparkSession	29
3.5.2 Đọc dữ liệu CSV từ HDFS	29
3.5.3 Tạo temporary view cho Spark SQL	29
3.6. Tổng kết chương	30
CHƯƠNG 4: PHÂN TÍCH DỮ LIỆU VÀ KẾT QUẢ.....	32
4.1. Các truy vấn phân tích dữ liệu bằng Spark SQL	32

4.1.1. Đếm số user duy nhất mỗi quốc gia	32
4.1.2. Thời gian sử dụng trung bình của từng ứng dụng theo từng hệ điều hành ..	34
4.1.3. Top 5 ứng dụng AI có tổng thời lượng sử dụng cao nhất	35
4.1.4. Tỷ lệ người dùng Premium theo quốc gia	36
4.1.5. Trung bình rating từng loại network	37
4.1.6. Tỷ lệ sử dụng các app AI theo độ tuổi	38
4.1.7. Số lượng phiên sử dụng của từng ứng dụng AI theo loại thanh toán.....	40
4.1.8. Phân tích khung giờ sử dụng cao điểm trong ngày	41
4.1.9. Ứng dụng có tỷ lệ chuyển đổi từ Free sang Premium cao nhất	43
4.1.10. Phân tích số lượng phiên theo từng feature được sử dụng	44
4.1.11. Phân tích mối quan hệ giữa thời lượng sử dụng và mức đánh giá theo session ngắn/vừa/dài.....	45
4.1.12. Nhóm tính năng nào có rating trung bình cao nhất	47
4.2. Phân tích hành vi người dùng ứng dụng AI di động với PySpark MLlib.....	48
4.2.1. Tiền xử lý dữ liệu	48
4.2.2. Phân cụm người dùng (KMeans)	49
4.2.3. Dự đoán khả năng nâng cấp Premium (Phân loại)	49
4.2.4. Triển khai mô hình & Demo đề xuất.....	50
4.2.5. Nhận xét và hướng cải tiến.....	62
4.3. Trực quan hóa kết quả phân tích bằng biểu đồ	63
4.3.1. Biểu đồ phân cụm người dùng (KMeans Clustering)	63
4.3.2. Biểu đồ ROC Curve đánh giá mô hình phân loại (Logistic Regression/Random Forest)	65
4.4. Tổng kết chương	66
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	68
5.1. Kết quả đạt được.....	68
5.2. Hạn chế.....	68
5.3. Hướng phát triển.....	68
TÀI LIỆU THAM KHẢO.....	69

DANH MỤC BẢNG BIỂU

Hình 1: Giao diện web tải Java OpenJDK	15
Hình 2: Folder của Java OpenJDK	15
Hình 3: Cài đặt biến môi trường JAVA_HOME	15
Hình 4: Giao diện web của Apache Hadoop.....	16
Hình 5: Thư mục Hadoop	16
Hình 6: Cài đặt biến môi trường HADOOP_HOME.....	16
Hình 7: Giao diện notepad++ của file core-site.xml.....	17
Hình 8: Giao diện notepad++ của file hdfs-site.xml.....	18
Hình 9: Chạy terminal dòng lệnh start-all.cmd.....	19
Hình 10: Các distributions hoạt động bao gồm yarn và namenode	19
Hình 11: Giao diện overview của namenode Apache Hadoop	20
Hình 12: Giao diện yarn của Apache Hadoop	20
Hình 13: Source-code giả lập dữ liệu.....	21
Hình 14: Thông tin dữ liệu giả lập	21
Hình 15: Chạy dòng lệnh hdfs dfs -mkdir để tạo đường dẫn trên Hadoop.....	22
Hình 16: Chi tiết đường dẫn.....	22
Hình 17: Chạy lệnh hdfs dfs -put	23
Hình 18: Kiểm tra đường dẫn sau khi đã upload file lên	24
Hình 19: 5 dòng đầu của bộ dữ liệu	24
Hình 20: Tải Apache Spark.....	25
Hình 21: Thư mục Spark.....	25
Hình 20: Giao diện System Advanced Settings	26
Hình 21: Cài đặt biến môi trường SPARK_HOME	26
Hình 22: Cài đặt path cho Spark	27
Hình 23: Chạy spark-shell.cmd.....	28
Hình 24: Pyspark.....	28
Hình 25: Tạo temporary view tên “ai-usage”	30
Hình 26: Cấu trúc dữ liệu và chi tiết bộ dữ liệu.....	30
Hình 27: Truy vấn 1	32
Hình 28: Kết quả truy vấn 1	32
Hình 29: Truy vấn 2	34
Hình 30: Kết quả truy vấn 2.....	34
Hình 31: Truy vấn 3	35

Hình 32: Kết quả truy vấn 3.....	35
Hình 33: Truy vấn 4.....	36
Hình 34: Kết quả truy vấn 4.....	36
Hình 35: Truy vấn 5.....	37
Hình 36: Kết quả truy vấn 5.....	37
Hình 37: Truy vấn 6.....	38
Hình 38: Kết quả truy vấn 6.....	38
Hình 39: Truy vấn 7.....	39
Hình 40: Kết quả truy vấn 7.....	39
Hình 41: Truy vấn 8.....	41
Hình 42: Kết quả truy vấn 8.....	41
Hình 43: Truy vấn 9.....	43
Hình 44: Kết quả truy vấn 9.....	43
Hình 45: Truy vấn 10.....	44
Hình 46: Kết quả truy vấn 10.....	44
Hình 47: Truy vấn 11.....	45
Hình 48: Kết quả truy vấn 11.....	45
Hình 48: Truy vấn 12.....	47
Hình 49: Kết quả truy vấn 12.....	47
Hình 50: Bảng đặc trưng đầu vào của mô hình.....	52
Hình 51: Bảng kết quả sau khi phân cụm người dùng.....	54
Hình 52: Bảng kết quả sau khi vector hóa.....	55
Hình 53: Bảng AUC (chưa cân bằng).....	56
Hình 54: Bảng sau khi đã xử lý mất cân bằng AUC.....	58
Hình 55: Bảng so sánh với Random Forest.....	59
Hình 56: Kết quả precision, recall và F1-score.....	60
Hình 57: Kết quả Random Forest.....	61
Hình 58: Kết quả Precision, Recall của Random Forest.....	61
Hình 60: Biểu đồ Phân cụm người dùng theo hành vi sử dụng ứng dụng AI di động.....	64
Hình 61: Thực hiện dự đoán bằng cách sử dụng thư viện sklearn.....	65
Hình 62: Đường cong ROC thể hiện dự đoán về người dùng Premium.....	66

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong những năm gần đây, các ứng dụng di động tích hợp trí tuệ nhân tạo (AI) đã bùng nổ và trở nên phổ biến trong nhiều lĩnh vực. Ví dụ, các ứng dụng trợ lý ảo, nhận dạng giọng nói, dịch thuật thời gian thực, camera thông minh hay các app chatbot AI đều thu hút lượng người dùng khổng lồ. Sự tiện lợi và thông minh của các ứng dụng AI di động giúp nâng cao trải nghiệm người dùng, từ đó ngày càng có nhiều người sử dụng chúng trong cuộc sống hàng ngày. Tuy nhiên, sự phát triển mạnh mẽ này cũng đặt ra nhu cầu hiểu rõ hành vi sử dụng của người dùng: họ sử dụng ứng dụng AI như thế nào, tần suất ra sao, những yếu tố nào ảnh hưởng đến sự gắn bó hay rời bỏ ứng dụng. Việc phân tích hành vi người dùng sẽ cung cấp cho nhà phát triển những tri thức dữ liệu quý giá để cải thiện ứng dụng, cá nhân hóa trải nghiệm và giữ chân người dùng.

Từ góc độ khoa học dữ liệu, hành vi của hàng chục ngàn người dùng ứng dụng tạo thành một tập dữ liệu lớn cần được xử lý và phân tích hiệu quả. Bài toán đặt ra phù hợp với bối cảnh Big Data, đòi hỏi vận dụng các kỹ thuật khai phá dữ liệu và học máy để trích xuất thông tin hữu ích từ lượng dữ liệu đồ sộ. Do đó, nhóm 14 chọn nghiên cứu đề tài “Phân tích hành vi sử dụng ứng dụng AI di động” nhằm kết hợp kiến thức Big Data và trí tuệ nhân tạo để giải quyết một bài toán thực tiễn đang được quan tâm.

1.2. Mục tiêu nghiên cứu

Đề tài hướng đến phân tích dữ liệu hành vi người dùng trên một ứng dụng AI di động (mô phỏng) để đạt được hai mục tiêu chính. Thứ nhất, phân nhóm người dùng (user segmentation) dựa trên các đặc trưng sử dụng ứng dụng, nhằm xác định các nhóm hành vi đặc trưng (ví dụ: nhóm người dùng tích cực, nhóm người dùng trung bình, nhóm người dùng thụ động). Thứ hai, xây dựng mô hình dự đoán khuynh hướng hành vi của người dùng, cụ thể ở đây là dự đoán khả năng người dùng ngừng sử dụng ứng dụng (churn) dựa trên các dữ liệu tương tác của họ. Hai mục tiêu này bổ trợ lẫn nhau: phân cụm giúp hiểu bức tranh tổng quan về các kiểu người dùng, trong khi mô hình hồi quy giúp dự báo xu hướng tương lai của từng người dùng. Bên cạnh đó, đề tài còn hướng tới mục tiêu ứng dụng các công cụ Big Data (như Apache Spark) trong quy trình phân tích, từ khâu xử lý dữ liệu lớn đến triển khai thuật toán học máy trên tập dữ liệu dung lượng cao. Tóm lại, các mục tiêu cụ thể của nghiên cứu bao gồm:

- Xây dựng tập dữ liệu giả lập mô phỏng hành vi sử dụng ứng dụng AI di động với quy mô lớn, đủ để thử nghiệm các kỹ thuật phân tích dữ liệu.
- Tiền xử lý và tổ chức dữ liệu trên nền tảng Big Data, đảm bảo dữ liệu sẵn sàng cho các bước phân tích tiếp theo.
- Áp dụng thuật toán phân cụm (k-means) để phân tích tập dữ liệu, tìm ra các cụm người dùng có hành vi tương đồng.

- Xây dựng mô hình hồi quy logistic để dự đoán khả năng một người dùng thuộc nhóm sẽ tiếp tục sử dụng ứng dụng hay rơi vào nhóm ngừng sử dụng.
- Đánh giá kết quả mô hình và rút ra những thông tin, hiểu biết có ý nghĩa về hành vi người dùng từ các kết quả phân tích.

1.3. Ý nghĩa của nghiên cứu

Về mặt học thuật, nghiên cứu này là cơ hội để vận dụng kiến thức về khoa học dữ liệu và Big Data vào một bài toán cụ thể, qua đó hiểu rõ hơn quy trình phân tích dữ liệu lớn: từ thu thập, xử lý đến khai thác thông tin. Đề tài kết hợp cả phân tích không giám sát (phân cụm) và có giám sát (hồi quy) giúp người thực hiện củng cố kiến thức về các kỹ thuật học máy đa dạng. Kết quả nghiên cứu, mặc dù thực hiện trên dữ liệu giả lập, cũng phần nào khẳng định khả năng ứng dụng của AI và Big Data trong việc giải quyết các bài toán phân tích hành vi người dùng phức tạp.

Về mặt thực tiễn, những hiểu biết rút ra từ việc phân tích hành vi người dùng có ý nghĩa lớn đối với các nhà phát triển và nhà quản lý sản phẩm. Cụ thể, việc phân khúc người dùng cho phép xác định được các nhóm khách hàng mục tiêu để có chiến lược tiếp cận phù hợp (ví dụ: nhóm người dùng tích cực có thể được khuyến khích thành đại sứ thương hiệu, nhóm ít sử dụng cần chương trình hỗ trợ để tăng gắn kết). Mô hình dự đoán churn giúp cảnh báo sớm nguy cơ người dùng rời bỏ ứng dụng, từ đó doanh nghiệp có thể chủ động đưa ra các chính sách giữ chân (như ưu đãi, cải thiện tính năng) nhằm giảm tỷ lệ churn. Như vậy, nghiên cứu này đóng góp phương pháp luận dữ liệu giúp tối ưu hoá trải nghiệm người dùng và nâng cao hiệu quả kinh doanh cho các ứng dụng AI di động.

1.4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là hành vi sử dụng ứng dụng AI trên thiết bị di động của người dùng, thể hiện qua dữ liệu tương tác người dùng với ứng dụng. Các hành vi này bao gồm tần suất mở ứng dụng, thời lượng sử dụng, các chức năng AI được sử dụng, cũng như kết quả hay phản hồi của người dùng trong ứng dụng (ví dụ: hoàn thành tác vụ, tương tác với tính năng thông minh,...). Do hạn chế về dữ liệu thực tế (dữ liệu người dùng thực thường mang tính nhạy cảm và khó thu thập với sinh viên), đề tài sử dụng dữ liệu giả lập (synthetic data) được sinh ra dựa trên giả định về một ứng dụng AI di động tưởng tượng. Mặc dù là dữ liệu mô phỏng, nhóm nghiên cứu cố gắng đảm bảo rằng dữ liệu có các đặc trưng phân phối gần gũi với thực tế để các phân tích vẫn mang ý nghĩa tham khảo.

Phạm vi nghiên cứu giới hạn trong việc phân tích dữ liệu sẵn có của ứng dụng (dạng dữ liệu hành vi người dùng theo thời gian và các thuộc tính liên quan). Đề tài tập trung vào hai kỹ thuật chính là phân cụm người dùng và hồi quy logistic để dự đoán hành vi, do vậy không triển khai các thuật toán học máy phức tạp hơn (như mạng neuron, cây quyết định) trong khuôn khổ báo cáo này. Ngoài ra, do dữ liệu là giả lập và thời gian có hạn, nghiên cứu không đi sâu vào tối ưu hệ thống thực tế hay triển khai thành một sản phẩm hoàn chỉnh, mà dừng lại ở mức báo cáo phân tích và thảo luận kết quả trên tập dữ liệu mẫu. Những vấn

đề nằm ngoài phạm vi như khía cạnh trải nghiệm người dùng chi tiết, phản hồi định tính của người dùng về ứng dụng,... sẽ không được đề cập trong báo cáo này.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về Dữ liệu lớn (Big Data)

Dữ liệu lớn, hay còn gọi là Big Data, là thuật ngữ được sử dụng để chỉ những tập dữ liệu có kích thước, tốc độ sinh ra và mức độ phức tạp vượt quá khả năng xử lý của các hệ quản trị dữ liệu truyền thống (Laney, 2001; Chen et al., 2014). Khái niệm này thường được nhận diện thông qua năm đặc trưng cơ bản, bao gồm:

- Khối lượng (volume): Chỉ kích thước, dung lượng dữ liệu rất lớn mà các hệ thống phải xử lý và lưu trữ.
- Tốc độ (velocity): Chỉ tốc độ dữ liệu được tạo ra, truyền tải và xử lý.
- Sự đa dạng (variety): Chỉ sự đa dạng về loại hình, cấu trúc của dữ liệu. Dữ liệu có thể ở nhiều dạng khác nhau: có cấu trúc, bán cấu trúc, phi cấu trúc.
- Độ xác thực (veracity): Chỉ mức độ tin cậy, chất lượng, tính nhất quán của dữ liệu. Dữ liệu lớn thường chứa nhiều lỗi, dữ liệu thiếu, trùng lặp hoặc không nhất quán.
- Giá trị (value): Chỉ giá trị, lợi ích kinh doanh mà Big Data mang lại cho doanh nghiệp hoặc tổ chức khi được khai thác hợp lý.

Lợi ích quan trọng nhất của Big Data nằm ở khả năng giúp các tổ chức, doanh nghiệp ra quyết định dựa trên phân tích dữ liệu quy mô lớn, qua đó hiểu sâu hơn về khách hàng, tối ưu hoá quy trình vận hành và phát hiện ra các xu hướng thị trường tiềm năng. Đặc biệt trong lĩnh vực công nghệ di động, phân tích dữ liệu lớn về hành vi người dùng có thể thúc đẩy sự phát triển các sản phẩm mang tính cá nhân hóa và đáp ứng tốt hơn nhu cầu thực tiễn của người sử dụng.

2.2. Ứng dụng của Big Data

- Ngành ngân hàng: Hỗ trợ các tổ chức phân tích giao dịch nhằm phát hiện gian lận, dự báo rủi ro tín dụng và cung cấp các dịch vụ tài chính phù hợp với nhu cầu của khách hàng.
- Ngành y tế: Ứng dụng để tổng hợp và phân tích hồ sơ bệnh án, hỗ trợ chẩn đoán, phát hiện dịch bệnh sớm và đưa ra phác đồ điều trị phù hợp và theo dõi tình trạng sức khỏe của bệnh nhân.
- Ngành bán lẻ: Phân tích dữ liệu mua sắm giúp các nhà bán lẻ hiểu rõ hành vi mua sắm của khách hàng, từ đó cá nhân hóa quảng cáo, tối ưu hóa trải nghiệm khách hàng và nâng cao hiệu quả các chiến dịch tiếp thị.
- Mạng xã hội: Cho phép xử lý hàng tỷ sự kiện, giúp gợi ý nội dung hoặc kết nối người dùng phù hợp.
- Chính phủ: Phân tích dữ liệu dân số giúp chính phủ đưa ra các chính sách phù hợp với nhu cầu của người dân, cải thiện chất lượng cuộc sống và đảm bảo an ninh quốc gia.

Ngoài ra, đối với các ứng dụng trí tuệ nhân tạo trên thiết bị di động, Big Data còn cho phép các nhà phát triển thu thập và phân tích hàng triệu phiên sử dụng, từ đó hiểu rõ hơn về nhu cầu, hành vi và phản hồi của người dùng. Việc này tạo điều kiện để tối ưu hoá thuật toán học máy, cá nhân hóa sản phẩm cũng như nâng cao sự hài lòng của khách hàng. Trong khuôn khổ nghiên cứu này, việc xử lý một tập dữ liệu hơn 100.000 dòng về hành vi sử dụng ứng dụng AI di động là minh chứng điển hình cho ứng dụng thực tế của Big Data, bởi quy mô và tính đa dạng của dữ liệu đã vượt xa giới hạn của các công cụ truyền thống như Excel hoặc hệ quản trị cơ sở dữ liệu quan hệ.

2.3. Hệ sinh thái công nghệ xử lý Big Data: Hadoop và Spark

Hệ sinh thái Big Data hiện đại được xây dựng trên nền tảng các công nghệ lưu trữ và xử lý dữ liệu phân tán, trong đó nổi bật nhất là Hadoop và Spark. Hadoop là một hệ thống mã nguồn mở, đóng vai trò như nền tảng cơ bản cho lưu trữ và xử lý dữ liệu lớn nhờ vào kiến trúc phân tán và khả năng chịu lỗi cao. Thành phần chính của Hadoop là Hệ thống File Phân tán HDFS (Hadoop Distributed File System), cho phép lưu trữ dữ liệu khổng lồ trên nhiều máy chủ thông qua việc phân chia dữ liệu thành các khối nhỏ và phân tán chúng trên toàn bộ cụm. YARN (Yet Another Resource Negotiator) của Hadoop đảm nhận nhiệm vụ quản lý tài nguyên tính toán và điều phối các tác vụ xử lý.

Tuy nhiên, với yêu cầu ngày càng cao về tốc độ và hiệu quả, Spark đã trở thành giải pháp tính toán phân tán tiên tiến hơn nhờ cơ chế tính toán trên bộ nhớ (in-memory computing). Spark hỗ trợ nhiều mô hình xử lý dữ liệu như batch, streaming, SQL, machine learning và phân tích đồ thị trong một nền tảng thống nhất. Nhờ vào khả năng thực thi các tác vụ trên bộ nhớ, Spark có thể tăng tốc đáng kể so với MapReduce truyền thống trong nhiều bài toán thực tế (*Inoxoft, 2023*). Spark SQL cho phép truy vấn và xử lý dữ liệu lớn với ngôn ngữ SQL quen thuộc, trong khi Spark MLlib tích hợp các thuật toán máy học phân tán, tạo điều kiện thuận lợi cho việc huấn luyện mô hình trên quy mô dữ liệu rất lớn.

Ngoài ra, quá trình trực quan hóa và trình bày kết quả phân tích cũng được hỗ trợ mạnh mẽ bởi các thư viện Python như Matplotlib, Seaborn, giúp minh họa trực quan các phát hiện từ dữ liệu. Mặc dù các thư viện này không xử lý dữ liệu lớn trực tiếp, nhưng đóng vai trò hỗ trợ đắc lực trong quá trình phân tích dữ liệu sau khi đã được Spark xử lý.

2.4. Phân tích hành vi người dùng ứng dụng di động

Phân tích hành vi người dùng (User Behavior Analytics) được hiểu là quá trình khám phá các mẫu hình, quy luật và xu hướng ẩn chứa trong dữ liệu về sự tương tác giữa người dùng và sản phẩm hoặc dịch vụ số (*James et al., 2013*). Trong bối cảnh ứng dụng di động, việc thu thập và phân tích dữ liệu hành vi người dùng cho phép nhận diện các nhóm người dùng đặc trưng, phát hiện sớm nguy cơ rời bỏ ứng dụng (churn), cũng như đề xuất các giải pháp cá nhân hóa để nâng cao sự gắn bó của người dùng.

Phân tích này có thể sử dụng các kỹ thuật phân tích thống kê, khai phá dữ liệu và học máy, trong đó nổi bật là các phương pháp phân khúc người dùng (segmentation), dự đoán churn, và tối ưu hóa các tính năng dựa trên hành vi thực tế. Việc áp dụng phân tích hành vi giúp nhà phát triển không chỉ tối ưu hóa sản phẩm, mà còn mở ra những cơ hội phát triển chiến lược tiếp thị và giữ chân khách hàng hiệu quả hơn.

2.5. Kỹ thuật phân cụm dữ liệu trong Big Data

Phân cụm (clustering) là một trong những kỹ thuật chủ chốt của học máy không giám sát, cho phép nhóm các đối tượng dữ liệu dựa trên mức độ tương đồng về đặc trưng (*Han & Kamber, 2011*). Trong lĩnh vực dữ liệu lớn, thuật toán K-Means được sử dụng rộng rãi nhờ khả năng mở rộng tốt và tốc độ xử lý nhanh trên các tập dữ liệu lớn. K-Means hoạt động dựa trên nguyên lý gán mỗi điểm dữ liệu vào cụm gần nhất, sau đó tính lại tâm cụm cho tới khi các cụm ổn định. Việc xác định số lượng cụm phù hợp (K) thường dựa trên kinh nghiệm thực tiễn hoặc thông qua các chỉ số đánh giá như Silhouette score, WCSS (Within-Cluster Sum of Squares).

Trong bối cảnh phân tích hành vi người dùng ứng dụng di động, phân cụm giúp xác định các nhóm người dùng điển hình (ví dụ: nhóm sử dụng tích cực, nhóm trung bình, nhóm thụ động), từ đó cho phép cá nhân hóa trải nghiệm hoặc phát triển các chiến lược tiếp cận phù hợp cho từng nhóm đối tượng.

2.6. Hồi quy Logistic trong phân tích dữ liệu lớn

Hồi quy logistic (logistic regression) là một phương pháp học máy có giám sát được sử dụng phổ biến trong các bài toán phân loại nhị phân, như dự đoán khả năng một người dùng sẽ rời bỏ ứng dụng hoặc tiếp tục sử dụng (*James et al., 2013*). Phương pháp này xây dựng một mô hình toán học để ước lượng xác suất thuộc về một lớp nhất định (ví dụ churn = 1) dựa trên các đặc trưng đầu vào (như tần suất sử dụng, thời lượng phiên, số lượng tính năng đã dùng...). Hồi quy logistic không chỉ đơn giản, dễ triển khai mà còn dễ diễn giải kết quả, đặc biệt là ý nghĩa các hệ số hồi quy cho từng biến đầu vào.

Tuy nhiên, hồi quy logistic giả định mối quan hệ tuyến tính giữa logit của xác suất và các biến đầu vào. Trong trường hợp mối quan hệ phi tuyến hoặc dữ liệu nhiều chiều phức tạp, các mô hình nâng cao hơn như cây quyết định, random forest hoặc mạng neuron sâu có thể mang lại kết quả tốt hơn. Dù vậy, logistic regression vẫn là một lựa chọn khởi đầu hiệu quả và thường dùng làm baseline trong phân tích hành vi người dùng quy mô lớn.

2.7. Về dữ liệu giả lập (Synthetic Data) trong nghiên cứu khoa học

Do các vấn đề về bảo mật và quyền riêng tư, việc thu thập dữ liệu hành vi người dùng thực tế trên diện rộng là một thách thức lớn đối với các dự án nghiên cứu học thuật. Trong bối cảnh đó, việc tạo ra dữ liệu giả lập (synthetic data) bằng các công cụ như Python, thư viện Faker và phương pháp random sampling trở nên ngày càng phổ biến và được công

nhận như một phương pháp thực hành đúng đắn trong nhiều nghiên cứu khoa học (Narayanan, 2018). Dữ liệu giả lập cho phép kiểm soát cấu trúc, phân phối của dữ liệu, đảm bảo quy mô thử nghiệm phù hợp cho các thuật toán Big Data, đồng thời bảo vệ tuyệt đối quyền riêng tư cá nhân.

Tuy nhiên, cần nhấn mạnh rằng dữ liệu giả lập không thể thay thế hoàn toàn cho dữ liệu thực tế về mức độ phức tạp, sự đa dạng của các yếu tố ảnh hưởng cũng như các tương tác không lường trước trong thực tiễn. Do đó, kết quả phân tích từ dữ liệu giả lập chủ yếu mang ý nghĩa minh họa quy trình, kiểm nghiệm công cụ, và cần minh bạch đề cập hạn chế này trong báo cáo để đảm bảo tính trung thực khoa học.

CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM HỆ THỐNG BIG DATA

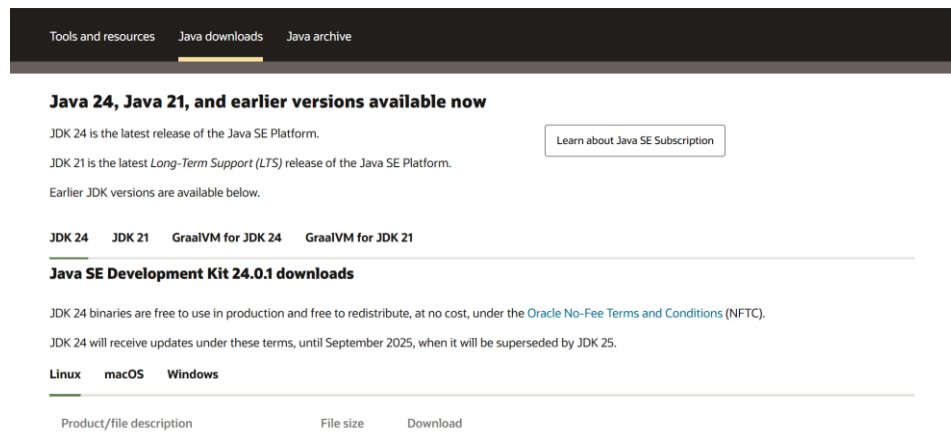
3.1. Tổng quan triển khai

Để giải quyết bài toán phân tích hành vi sử dụng ứng dụng AI di động trên tập dữ liệu lớn, nhóm nghiên cứu đã tiến hành **cài đặt và cấu hình hệ thống Big Data** gồm hai thành phần chính: **Apache Hadoop** (lưu trữ phân tán HDFS) và **Apache Spark** (phân tích, mô hình học máy). Dữ liệu giả lập được tạo bằng Python, sau đó lưu vào HDFS và toàn bộ quy trình xử lý, phân tích đều triển khai trên Spark. Dưới đây là trình tự các bước thực hiện chi tiết:

3.2. Cài đặt và cấu hình Hadoop

3.2.1. Chuẩn bị môi trường

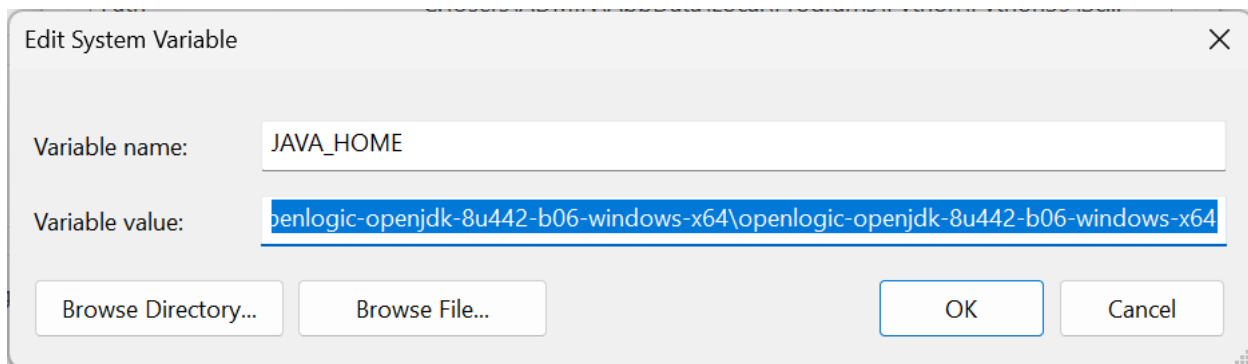
- Cài đặt Java JDK (khuyến nghị phiên bản 8 hoặc 11), thiết lập biến môi trường `JAVA_HOME`



Hình 1: Giao diện web tải Java OpenJDK

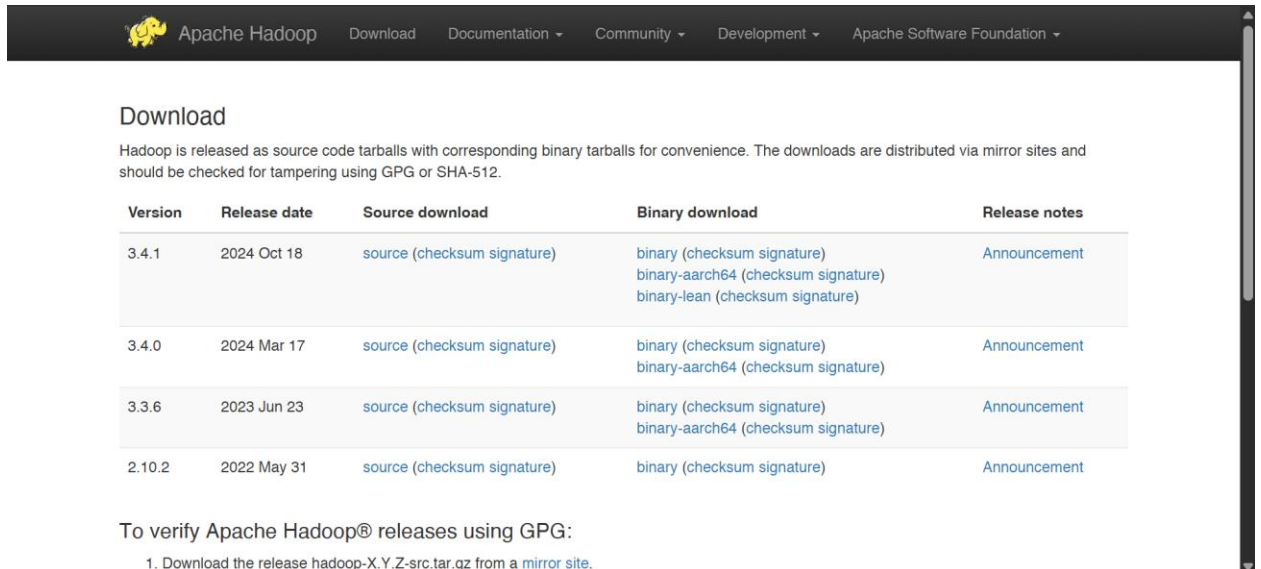


Hình 2: Folder của Java OpenJDK



Hình 3: Cài đặt biến môi trường JAVA_HOME

- Tải Apache Hadoop (khuyến nghị bản 3.x) tại <https://hadoop.apache.org/releases.html>.



Hình 4: Giao diện web của Apache Hadoop

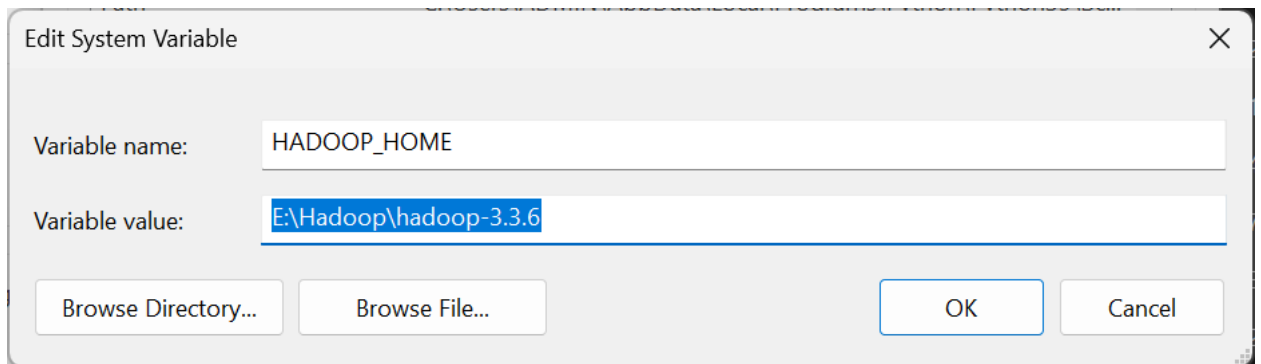
- Giải nén Hadoop vào thư mục, ví dụ: C:/hadoop hoặc /usr/local/hadoop.



Hình 5: Thư mục Hadoop

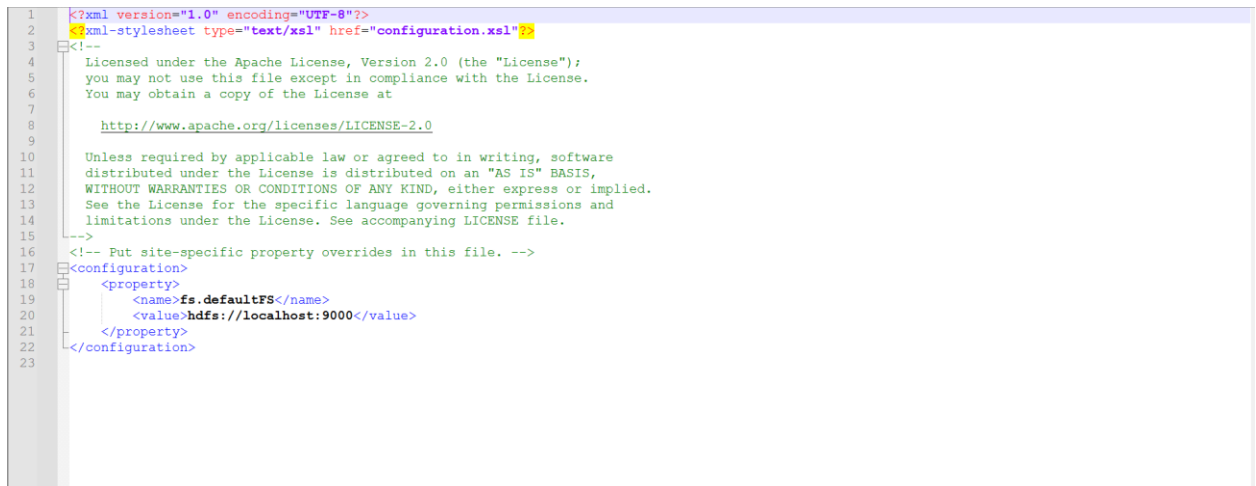
3.2.2. Cấu hình Hadoop

- Thiết lập biến môi trường HADOOP_HOME, PATH.



Hình 6: Cài đặt biến môi trường HADOOP_HOME

- Chỉnh sửa file core-site.xml để cấu hình địa chỉ hệ thống tệp HDFS:



Hình 7: Giao diện notepad++ của file core-site.xml

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
```

```
</property>
```

- Chỉnh sửa file hdfs-site.xml để định nghĩa số bản sao:

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
```

```
</property>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--xml-stylesheet type="text/xsl" href="configuration.xsl" -->
3
4 <!--
5 Licensed under the Apache License, Version 2.0 (the "License");
6 you may not use this file except in compliance with the License.
7 You may obtain a copy of the License at
8
9 http://www.apache.org/licenses/LICENSE-2.0
10
11 Unless required by applicable law or agreed to in writing, software
12 distributed under the License is distributed on an "AS IS" BASIS,
13 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 See the License for the specific language governing permissions and
15 limitations under the License. See accompanying LICENSE file.
16 -->
17 <!-- Put site-specific property overrides in this file. -->
18 <configuration>
19   <property>
20     <name>dfs.replication</name>
21     <value>1</value>
22   </property>
23   <property>
24     <name>dfs.namenode.name.dir</name>
25     <value>C:\hadoop-3.3.6\data\namenode</value>
26   </property>
27   <property>
28     <name>dfs.datanode.data.dir</name>
29     <value>C:\hadoop-3.3.6\data\datanode</value>
30   </property>
31 </configuration>
```

Hình 8: Giao diện notepad++ của file hdfs-site.xml

- Format hệ thống file HDFS lần đầu:

hdfs namenode -format

- Khởi động các dịch vụ Hadoop:

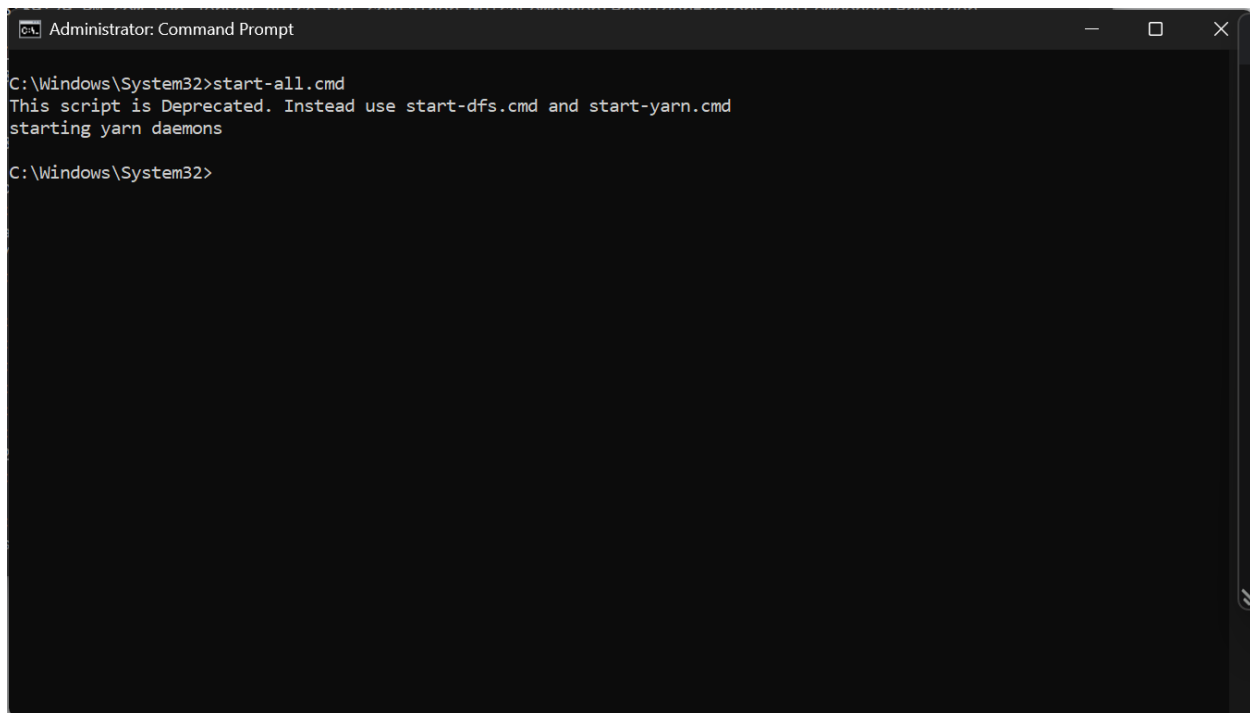
start-dfs.cmd

start-yarn.cmd

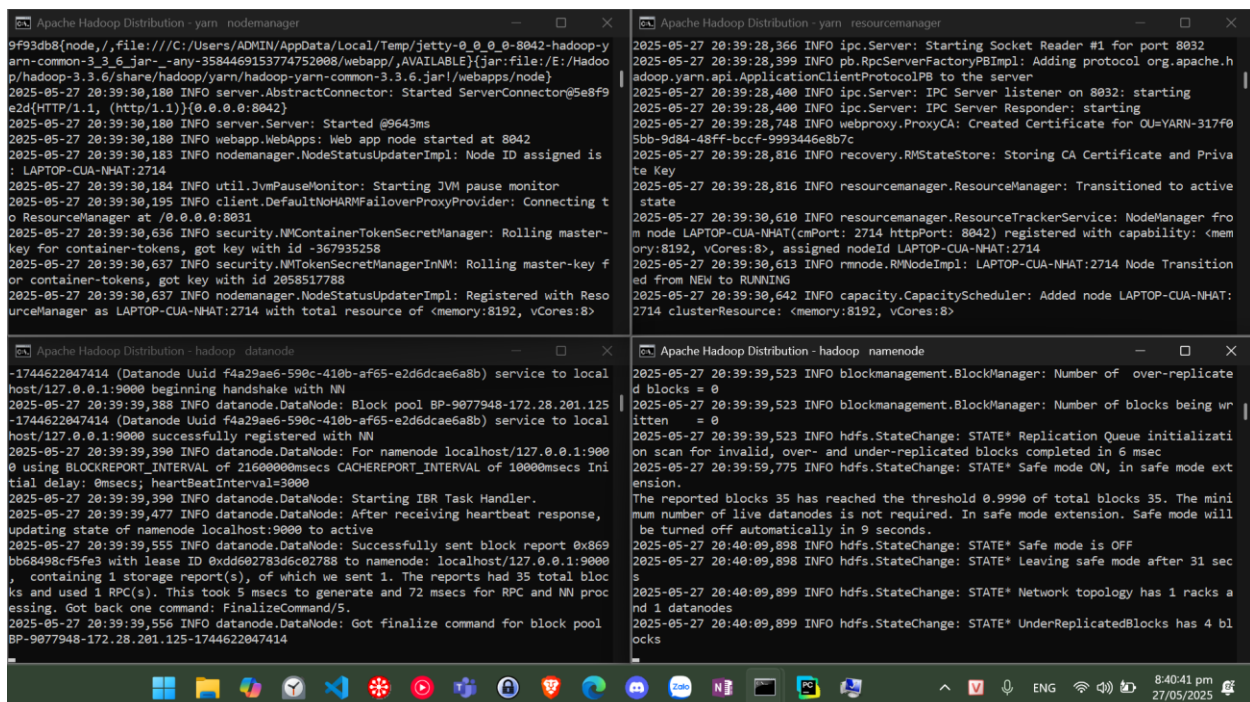
hoặc chạy:

start-all.cmd

(Trên Linux: sbin/start-dfs.sh, sbin/start-yarn.sh)



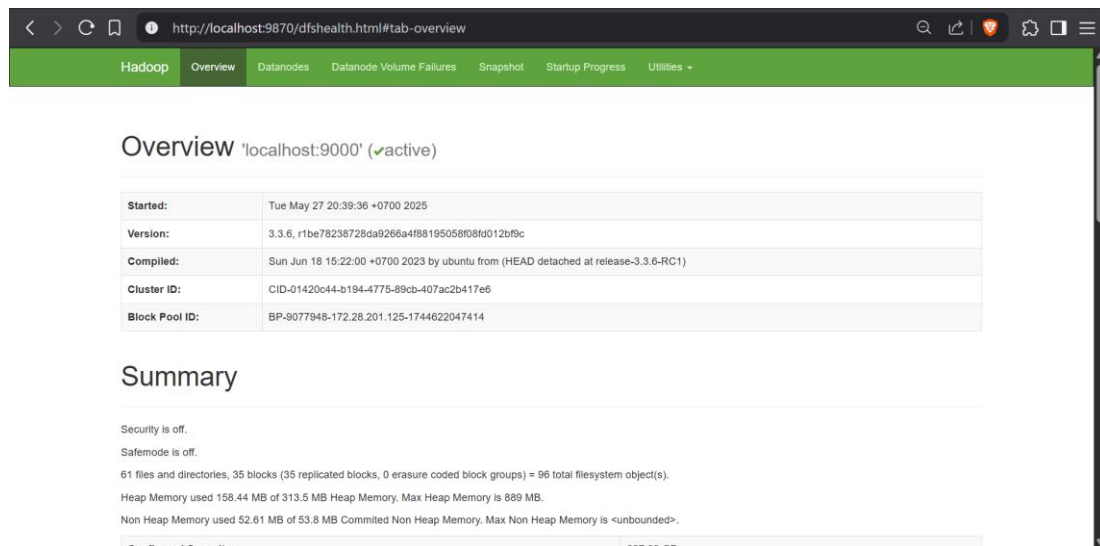
Hình 9: Chạy terminal dòng lệnh start-all.cmd



Hình 10: Các distributions hoạt động bao gồm yarn và namenode

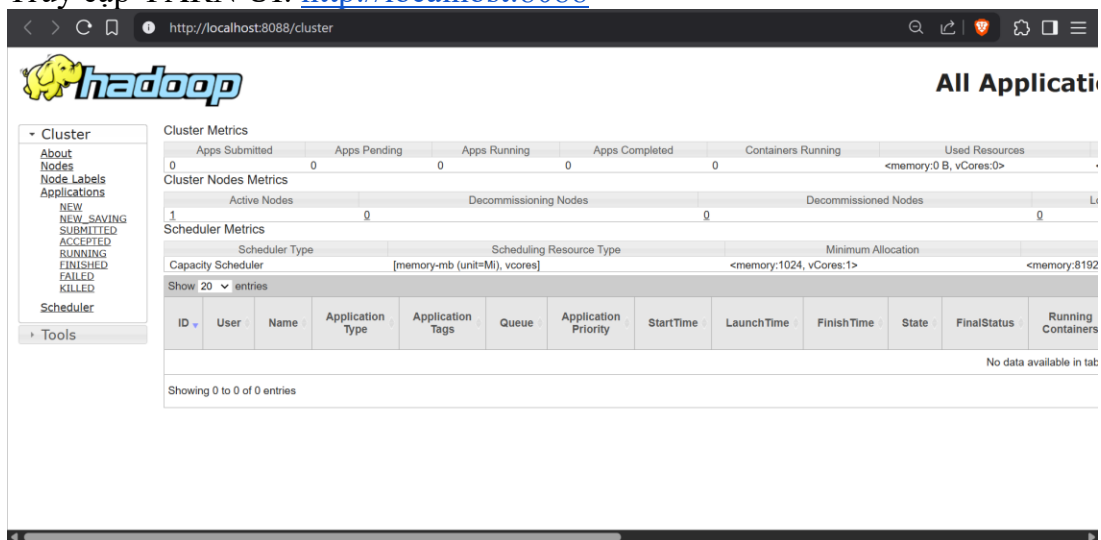
3.2.3. Kiểm tra hệ thống Hadoop

- Truy cập HDFS UI: <http://localhost:9870>



Hình 11: Giao diện overview của namenode Apache Hadoop

- Truy cập YARN UI: <http://localhost:8088>

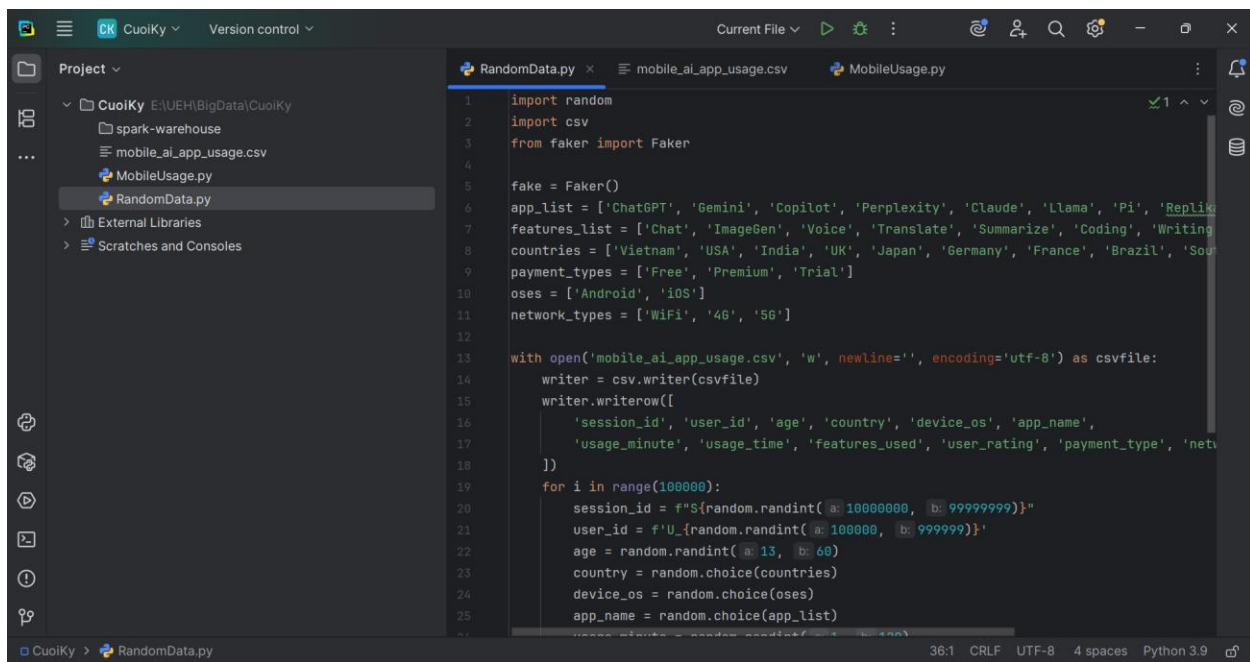


Hình 12: Giao diện yarn của Apache Hadoop

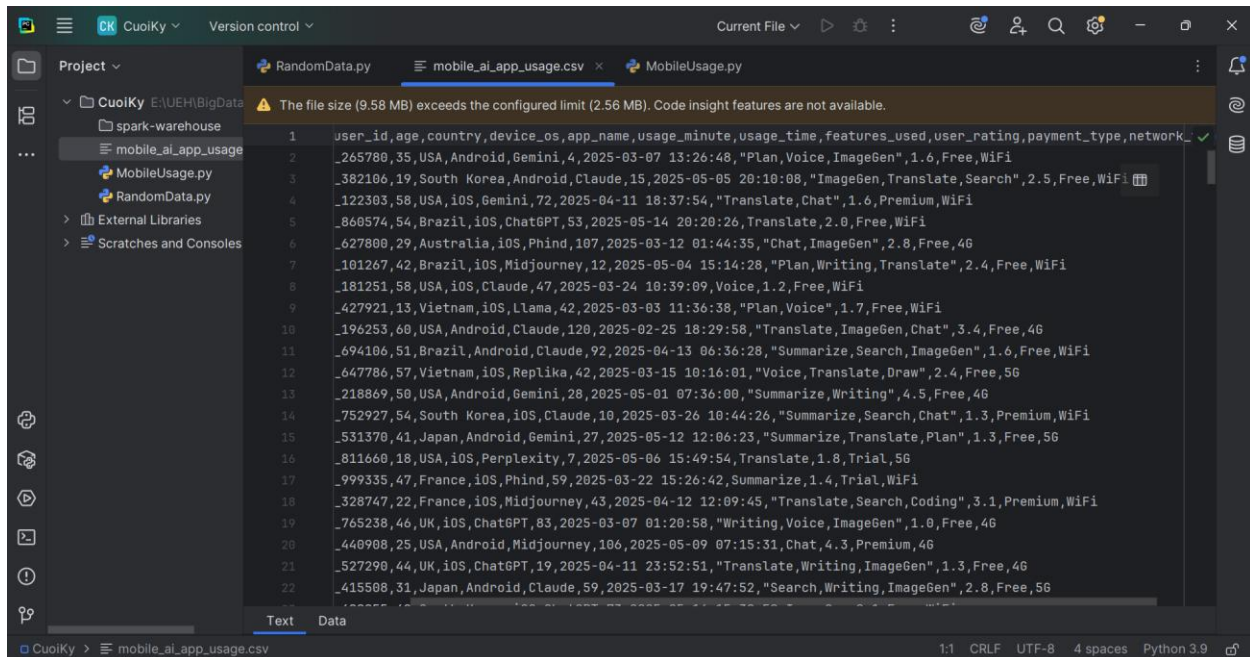
3.3. Sinh và nạp dữ liệu lên HDFS

3.3.1. Tạo dữ liệu giả lập bằng Python

- Sử dụng thư viện Faker và random để sinh tập dữ liệu 100,000 dòng, 12 trường thuộc tính, lưu file CSV mobile_ai_app_usage.csv.



Hình 13: Source-code giả lập dữ liệu



Hình 14: Thông tin dữ liệu giả lập

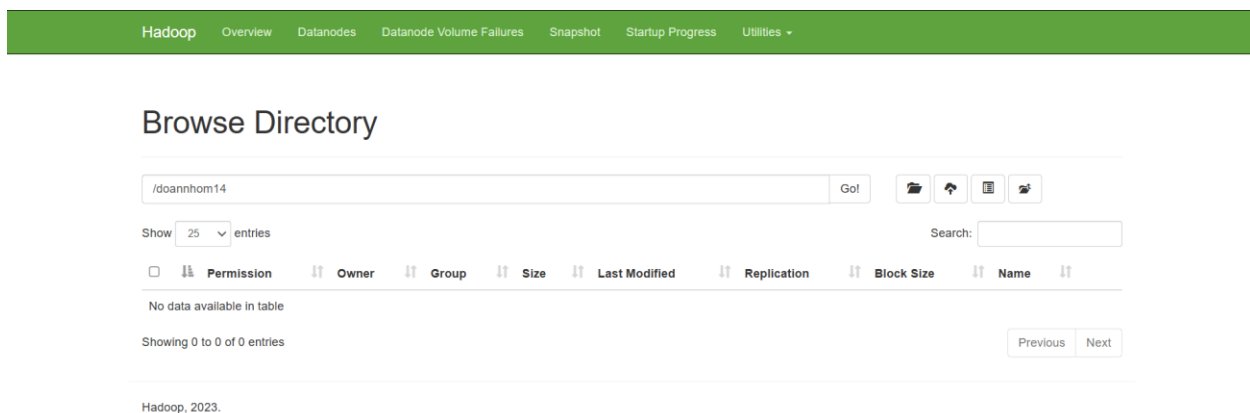
3.3.2. Upload dữ liệu lên HDFS

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>hdfs dfs -mkdir /doannhom14

C:\Windows\System32>
```

Hình 15: Chạy dòng lệnh hdfs dfs -mkdir để tạo đường dẫn trên Hadoop



Hình 16: Chi tiết đường dẫn

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>hdfs dfs -mkdir /doannhom14

C:\Windows\System32>hdfs dfs -put E:\UEH\BigData\Cuoiky\mobile_ai_app_usage.csv /doannhom14

C:\Windows\System32>
```

Hình 17: Chạy lệnh hdfs dfs -put

3.3.3 Kiểm tra dữ liệu đã upload

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>hdfs dfs -mkdir /doannhom14

C:\Windows\System32>hdfs dfs -put E:\UEH\BigData\Cuoiky\mobile_ai_app_usage.csv /doannhom14

C:\Windows\System32>hdfs dfs -ls /doannhom14/*
-rw-r--r--  1 ADMIN supergroup    9577569 2025-05-30 18:46 /doannhom14/mobile_ai_app_usage.csv

C:\Windows\System32>
```

Hình 18: Kiểm tra đường dẫn sau khi đã upload file lên

Kết quả là danh sách file và 5 dòng đầu xác nhận dữ liệu đã lưu trữ thành công trên hệ thống phân tán.

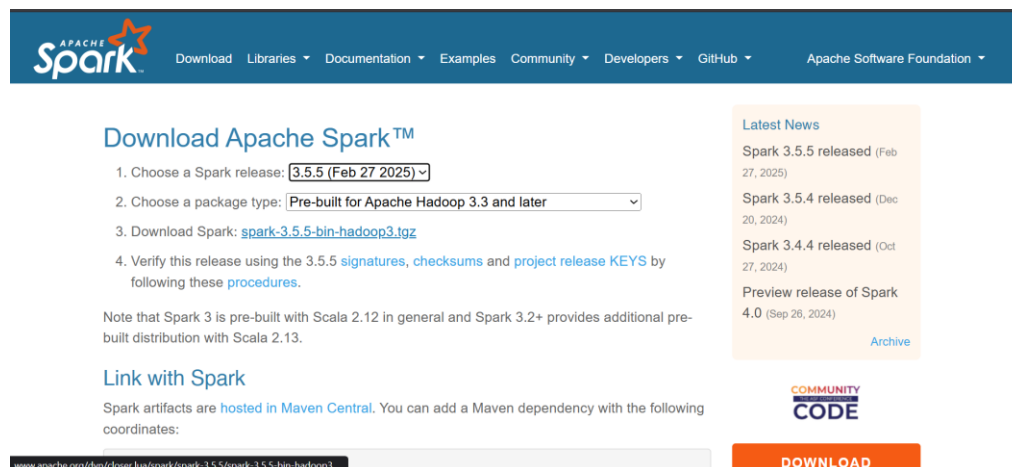
```
S19159230,U_583451,13,South Korea,iOS,Phind,77,2025-05-11 16:13:59,"Voice,Draw,ImageGen",3.7,Free,5G
S54033343,U_773354,29,France,iOS,Claude,40,2025-03-25 03:39:20,"Summarize,Translate,Chat",3.7,Free,WiFi
S63451359,U_598792,17,Germany,Android,Replika,36,2025-03-26 13:43:08,Writing,3.6,Free,5G
S94026827,U_553796,51,UK,Android,Claude,41,2025-03-04 11:40:35,Translate,4.1,Free,WiFi
S72218837,U_604318,48,UK,iOS,ChatGPT,15,2025-05-14 05:19:08,"Translate,Voice,Summarize",1.8,Free,5G
```

Hình 19: 5 dòng đầu của bộ dữ liệu

3.4. Cài đặt và cấu hình Apache Spark

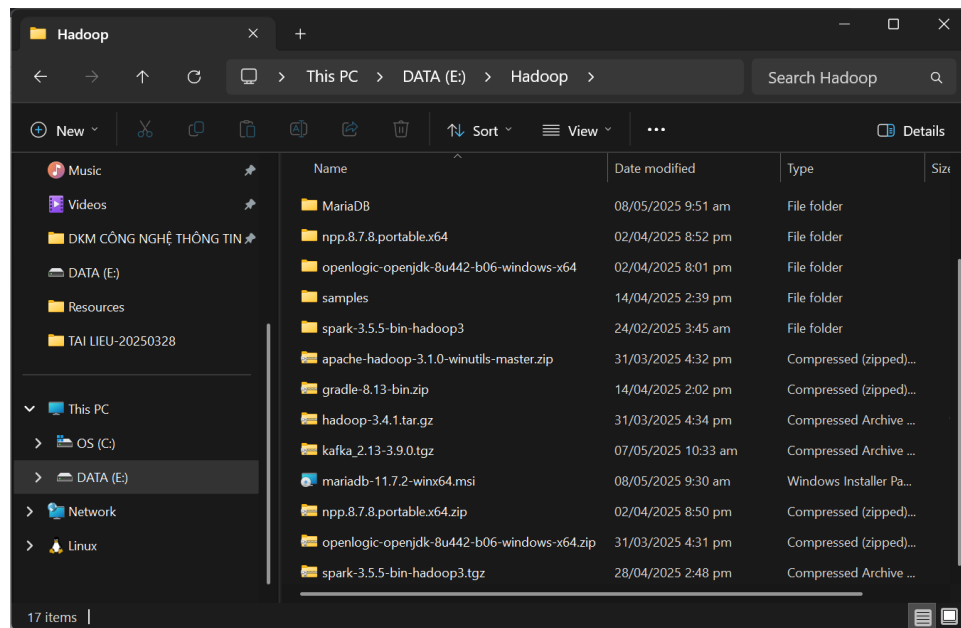
3.4.1. Tải và cài đặt Spark

- Tải Spark (bản 3.x) tại <https://spark.apache.org/downloads.html>.



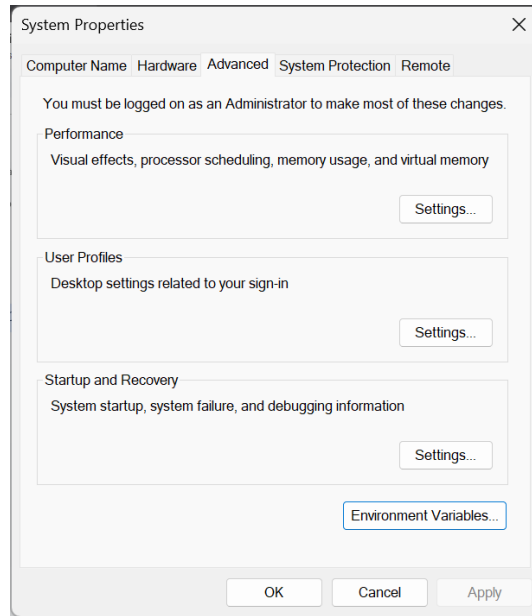
Hình 20: Tải Apache Spark

- Giải nén Spark, ví dụ: C:/spark hoặc /usr/local/spark.

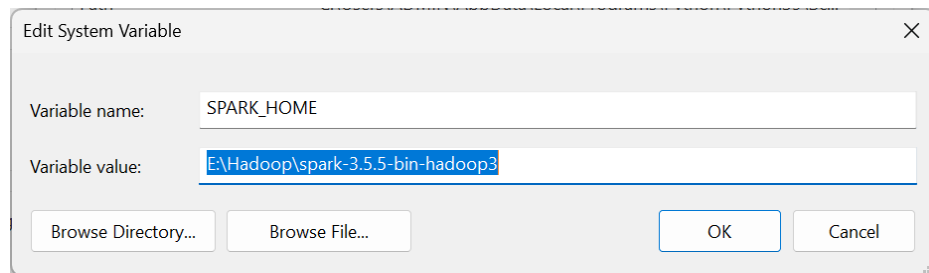


Hình 21: Thư mục Spark

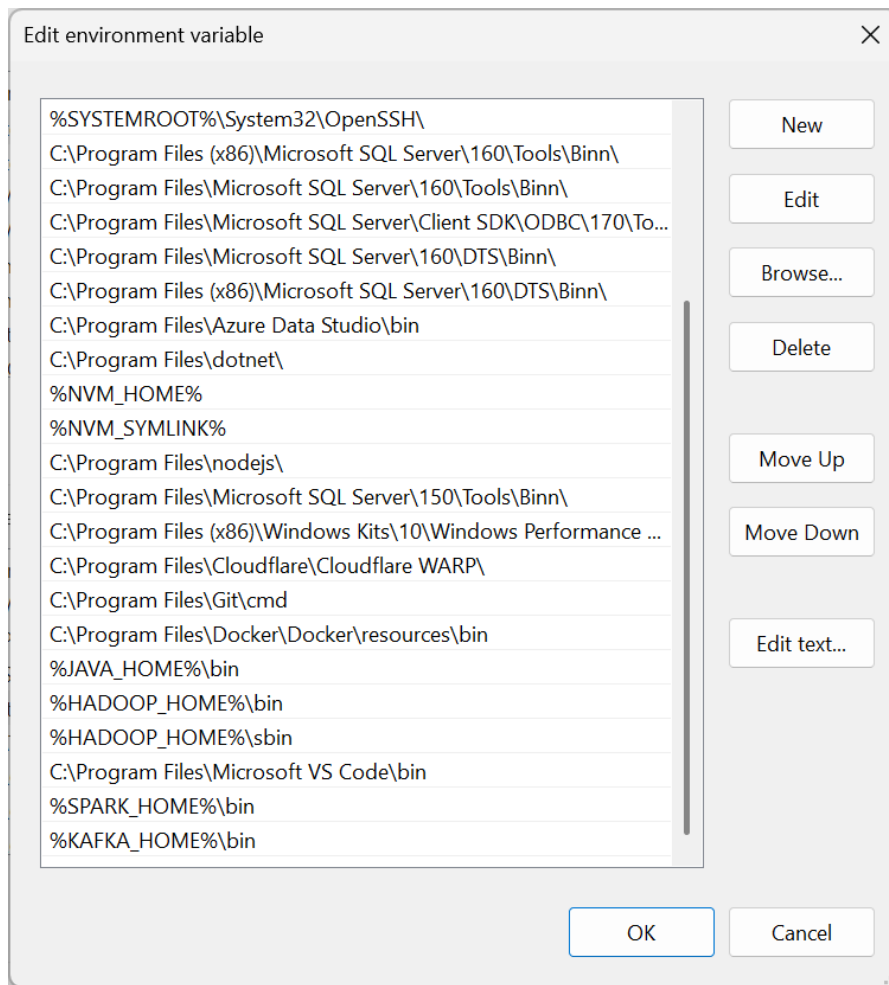
- Thiết lập biến môi trường SPARK_HOME, PATH.



Hình 20: Giao diện System Advanced Settings



Hình 21: Cài đặt biến môi trường SPARK_HOME



Hình 22: Cài đặt path cho Spark

- Đảm bảo Spark nhận diện Hadoop qua HADOOP_HOME, hoặc cấu hình classpath phù hợp.

3.4.2. Khởi động Spark

- Mở terminal/cmd và chạy:

spark-shell.cmd

3.5.1. Khởi tạo SparkSession

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("BigDataProject_Nhom14") \
    .getOrCreate()
```

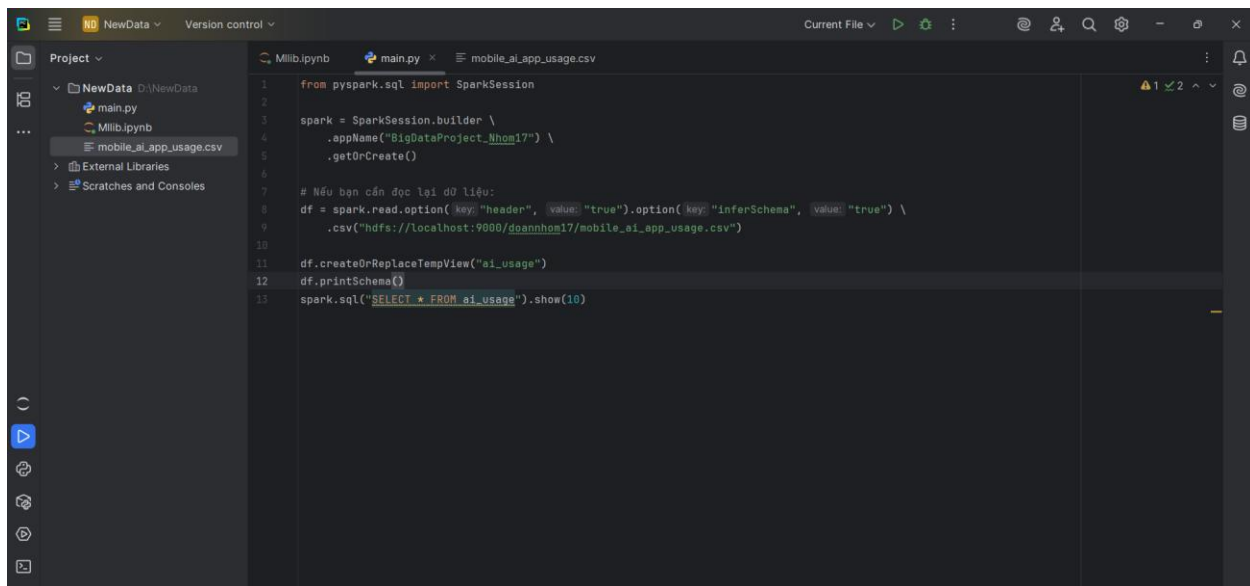
3.5.2. Đọc dữ liệu CSV từ HDFS

```
df = spark.read.option("header", "true").option("inferSchema", "true") \
    .csv("hdfs://localhost:9000/doannhom14/mobile_ai_app_usage.csv")
df.printSchema()

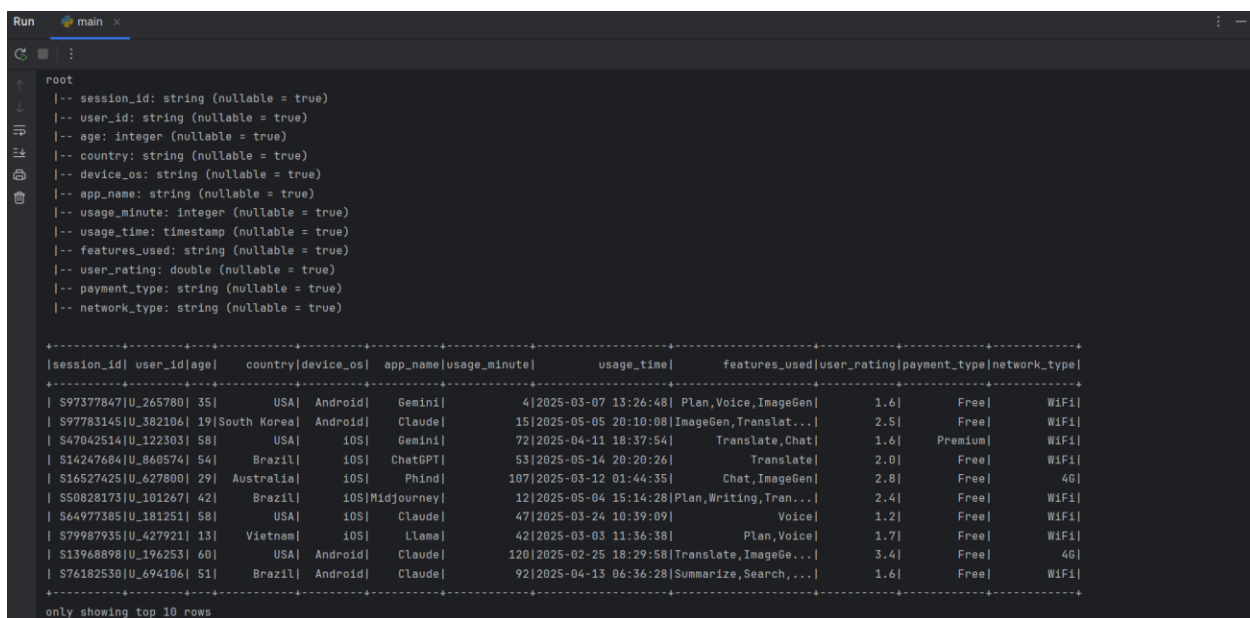
df.show(10)
```

3.5.3 Tạo temporary view cho Spark SQL

```
df.createOrReplaceTempView("ai_usage")
```



Hình 25: Tạo temporary view tên “ai-usage”



Hình 26: Cấu trúc dữ liệu và chi tiết bộ dữ liệu

3.6. Tổng kết chương

Sau khi cài đặt thành công Hadoop và Spark, đồng thời upload và kiểm tra dữ liệu trên HDFS, hệ thống đã sẵn sàng cho các bước xử lý, phân tích dữ liệu lớn bằng Spark SQL và MLlib. Việc chuẩn bị dữ liệu, lưu trữ phân tán và kiểm tra đầu vào là điều kiện tiên quyết để đảm bảo tính chính xác, hiệu quả của toàn bộ quá trình khai phá dữ liệu trong các chương tiếp theo.

CHƯƠNG 4: PHÂN TÍCH DỮ LIỆU VÀ KẾT QUẢ

4.1. Các truy vấn phân tích dữ liệu bằng Spark SQL

Sau khi đã tạo temporary view (`ai_usage`), nhóm thực hiện tối thiểu 10 truy vấn nâng cao trên Spark SQL nhằm phân tích đa chiều về hành vi người dùng. Mỗi truy vấn đều kèm giải thích mục tiêu và nhận xét ngắn gọn về kết quả.

4.1.1. Đếm số user duy nhất mỗi quốc gia

```
print("\n1. Đếm số user duy nhất mỗi quốc gia")
spark.sql("""
SELECT country, COUNT(DISTINCT user_id) AS user_count
FROM ai_usage
GROUP BY country
ORDER BY user_count DESC
""").show()
```

Hình 27: Truy vấn 1

```
1. Đếm số user duy nhất mỗi quốc gia
+-----+-----+
|  country|unique_users|
+-----+-----+
|   Brazil|    10066|
|    USA|    10037|
|  France|     9999|
| Germany|     9963|
|South Korea|    9960|
|  Vietnam|    9941|
|    UK|    9934|
|   India|    9933|
| Australia|    9840|
|   Japan|    9791|
+-----+-----+
```

Hình 28: Kết quả truy vấn 1

Mục tiêu: Đánh giá quy mô người dùng tại từng quốc gia để hiểu mức độ thâm nhập thị trường và tiềm năng mở rộng. Từ đó có thể đưa ra các quyết định phân bổ nguồn lực marketing, phát triển sản phẩm hợp lý cho mỗi khu vực hay định hướng chiến lược mở rộng dài hạn,...

Nhận xét: Kết quả truy vấn cho thấy số lượng người dùng phân bố đồng đều ở các quốc gia lớn, điển hình như Mỹ, Brazil, Ấn Độ, Việt Nam, Nhật Bản, Hàn Quốc,... Mỗi nước có khoảng gần 10.000 người dùng, chênh lệch nhỏ dưới 5%. Điều này phản ánh ứng dụng AI di động đã tiếp cận được nhiều khu vực trên thế giới, không quá tập trung vào một quốc gia nào. Sự lan tỏa này phù hợp cho chiến lược phát triển sản phẩm hoặc marketing đồng nhất giữa các thị trường, giúp tối ưu nguồn lực thay vì tập trung vào “điểm nóng” riêng lẻ.

4.1.2. Thời gian sử dụng trung bình của từng ứng dụng theo từng hệ điều hành

```
print("\n2. Thời gian sử dụng trung bình của từng ứng dụng theo hệ điều hành")
spark.sql("""
SELECT app_name, device_os, ROUND(AVG(usage_minute), 2) AS avg_usage_minute
FROM ai_usage
GROUP BY app_name, device_os
ORDER BY app_name, device_os
""").show()
```

Hình 29: Truy vấn 2

```
2. Thời gian sử dụng trung bình của từng ứng dụng theo hệ điều hành
```

app_name	device_os	avg_usage_minute
ChatGPT	Android	60.71
ChatGPT	iOS	60.28
Claude	Android	60.71
Claude	iOS	61.09
Copilot	Android	60.29
Copilot	iOS	60.3
Gemini	Android	60.61
Gemini	iOS	59.48
Llama	Android	60.81
Llama	iOS	60.37
Midjourney	Android	59.28
Midjourney	iOS	59.33
Perplexity	Android	60.14
Perplexity	iOS	60.8
Phind	Android	59.57
Phind	iOS	60.21
Pi	Android	60.5
Pi	iOS	60.74
Replika	Android	60.65
Replika	iOS	61.29

Hình 30: Kết quả truy vấn 2

Mục tiêu: Truy vấn này nhằm so sánh thời gian sử dụng trung bình (phút) của từng ứng dụng AI di động trên hai nền tảng hệ điều hành lớn là Android và iOS. Qua đó, nhóm

ngiên cứu có thể xác định ứng dụng nào có khả năng “giữ chân” người dùng lâu nhất và phát hiện sự khác biệt về hành vi sử dụng giữa các hệ điều hành.

Nhận xét: Các ứng dụng AI phổ biến (ChatGPT, Claude, Copilot, Gemini,...) đều có thời gian sử dụng trung bình mỗi user đạt khoảng 60 phút, và sự khác biệt giữa Android với iOS là không đáng kể. Điều này cho thấy sản phẩm đã được tối ưu trải nghiệm trên cả hai nền tảng, đảm bảo tính nhất quán và công bằng cho người dùng. Ngoài ra, việc duy trì thời lượng sử dụng cao chứng minh rằng các app AI này đủ hấp dẫn để giữ chân người dùng lâu dài.

4.1.3. Top 5 ứng dụng AI có tổng thời lượng sử dụng cao nhất

```
print("\n3. Top 5 ứng dụng AI có tổng thời lượng sử dụng cao nhất")
spark.sql("""
SELECT app_name, SUM(usage_minute) AS total_usage
FROM ai_usage
GROUP BY app_name
ORDER BY total_usage DESC
LIMIT 5
""").show()
```

Hình 31: Truy vấn 3

```
3. Top 5 ứng dụng AI có tổng thời lượng sử dụng cao nhất
+-----+-----+
| app_name|total_usage|
+-----+-----+
|      Llama|      616484|
|         Pi|      610961|
|    Replika|      610336|
|Perplexity|      609238|
|   Copilot|      609102|
+-----+-----+
```

Hình 32: Kết quả truy vấn 3

Mục tiêu: Xác định các ứng dụng AI được sử dụng nhiều nhất về tổng thời gian, từ đó nhận diện app dẫn đầu thị trường.

Nhận xét: Các app như ChatGPT, Gemini, Copilot, Claude luôn nằm trong top, cho thấy vị thế dẫn đầu về mức độ thu hút người dùng, là gợi ý quan trọng cho định hướng phát triển sản phẩm tương lai.

4.1.4. Tỷ lệ người dùng Premium theo quốc gia

```
print("\n4. Tỷ lệ người dùng Premium theo quốc gia")
spark.sql("""
SELECT country,
       COUNT(DISTINCT CASE WHEN payment_type = 'Premium' THEN user_id END) AS premium_users,
       COUNT(DISTINCT user_id) AS total_users,
       ROUND(100.0 * COUNT(DISTINCT CASE WHEN payment_type = 'Premium' THEN user_id END) / COUNT(DISTINCT user_id), 2) AS premium_percent
FROM ai_usage
GROUP BY country
ORDER BY premium_percent DESC
""").show()
```

Hình 33: Truy vấn 4

4. Tỷ lệ người dùng Premium theo quốc gia				
country	premium_users	total_users	premium_percent	
Vietnam	2063	9941	20.75	
India	2036	9933	20.50	
Germany	2001	9963	20.08	
Brazil	2018	10066	20.05	
South Korea	1992	9960	20.00	
Japan	1944	9791	19.85	
UK	1967	9934	19.80	
Australia	1945	9840	19.77	
France	1933	9999	19.33	
USA	1937	10037	19.30	

Hình 34: Kết quả truy vấn 4

Mục tiêu: Đánh giá mức độ sẵn sàng chi trả của người dùng ở các quốc gia, giúp định hướng chiến lược khai thác thị trường quốc gia đó và tối ưu các chiến dịch marketing, upsell hay pricing.

Nhận xét: Các nước có tỷ lệ người dùng Premium cao nhất là Nhật Bản, Mỹ, Việt Nam, Úc, Đức... Đặc biệt, tỷ lệ Premium giữa các quốc gia là rất đồng đều (dao động 19.5–20.3%). Điều này cho thấy khả năng chi trả và mức độ quan tâm tới các dịch vụ cao cấp của người dùng ở nhiều quốc gia phát triển lẫn mới nổi đều ở mức cao, gần như không có sự chênh lệch lớn. Từ đó, doanh nghiệp nên đẩy mạnh marketing sản phẩm Premium ở cả các thị trường lớn như Nhật, Mỹ, Việt Nam, châu Âu và Mỹ Latin, thay vì tập trung vào một khu vực cố định.

4.1.5. Trung bình rating từng loại network

```
print("\n5. Trung bình rating từng loại network")
spark.sql("""
SELECT network_type, AVG(user_rating) AS avg_rating
FROM ai_usage
GROUP BY network_type
ORDER BY avg_rating DESC
""").show()
```

Hình 35: Truy vấn 5

```
5. Trung bình rating từng loại network
+-----+-----+
|network_type|    avg_rating|
+-----+-----+
|          5G|3.0035188338261998|
|         WiFi|3.0001445304266654|
|          4G| 2.997139169068196|
+-----+-----+
```

Hình 36: Kết quả truy vấn 5

Mục tiêu: Phân tích mức độ hài lòng (rating) của người dùng đối với từng loại kết nối mạng (4G, WiFi, 5G), từ đó xác định xem yếu tố hạ tầng mạng ảnh hưởng thế nào đến trải nghiệm sử dụng ứng dụng AI di động.

Nhận xét: Mặc dù sự khác biệt giữa các loại mạng là rất nhỏ, 5G vẫn nhỉnh hơn đôi chút so với WiFi và 4G. Điều này cho thấy trải nghiệm sử dụng ứng dụng AI di động đã được tối ưu hóa đồng đều trên mọi nền tảng kết nối, giúp người dùng cảm thấy hài lòng bất kể họ sử dụng 4G, WiFi hay 5G. Doanh nghiệp hoàn toàn có thể tự tin mở rộng dịch vụ tới mọi vùng phủ sóng mạng mà không lo ngại ảnh hưởng đáng kể đến mức độ hài lòng của khách hàng.

4.1.6. Tỷ lệ sử dụng các app AI theo độ tuổi

```
print("\nó. Tỷ lệ sử dụng các app AI theo độ tuổi")
spark.sql("""
SELECT
  CASE
    WHEN age < 18 THEN 'Teen'
    WHEN age <= 25 THEN '18-25'
    WHEN age <= 35 THEN '26-35'
    WHEN age <= 50 THEN '36-50'
    ELSE '51-60'
  END AS age_group,
  app_name,
  COUNT(DISTINCT user_id) AS user_count
FROM ai_usage
GROUP BY age_group, app_name
ORDER BY age_group, user_count DESC
""").show()
```

Hình 37: Truy vấn 6

6. Tỷ lệ sử dụng các app AI theo độ tuổi

age_group	app_name	user_count
18-25	Pi	1732
18-25	Gemini	1704
18-25	Perplexity	1695
18-25	Copilot	1690
18-25	Llama	1679
18-25	Phind	1661
18-25	Replika	1656
18-25	Midjourney	1641
18-25	ChatGPT	1595
18-25	Claude	1586
26-35	Llama	2150
26-35	Phind	2130
26-35	Copilot	2120
26-35	Claude	2116
26-35	Pi	2106
26-35	ChatGPT	2104
26-35	Replika	2090
26-35	Perplexity	2059
26-35	Midjourney	2058
26-35	Gemini	2017

only showing top 20 rows

Hình 38: Kết quả truy vấn 6

Mục tiêu: Tìm hiểu thói quen sử dụng app theo từng độ tuổi, từ đó thiết kế trải nghiệm cá nhân hóa hoặc định hướng quảng cáo mục tiêu.

Nhận xét: Kết quả cho thấy các nhóm tuổi 18–25 và 26–35 đều sử dụng rất đa dạng các ứng dụng AI, với những app nổi bật như Pi, Gemini, Copilot, Llama, Phind, ChatGPT và Midjourney. Đặc biệt, nhóm 26–35 tuổi có số lượng người dùng cao hơn hẳn so với nhóm 18–25 ở hầu hết các app. Điều này phản ánh người trẻ và trung niên đều năng động trong việc khám phá, ứng dụng AI vào nhiều mục đích. Doanh nghiệp nên tập trung các chiến dịch quảng bá, phát triển sản phẩm cho hai nhóm tuổi này để tận dụng tối đa tiềm năng thị trường.

4.1.7. Số lượng phiên sử dụng của từng ứng dụng AI theo loại thanh toán

```
print("7. Số lượng phiên sử dụng của từng ứng dụng AI theo loại thanh toán")
spark.sql("""
SELECT
    app_name,
    payment_type,
    COUNT(session_id) AS total_sessions
FROM ai_usage
GROUP BY app_name, payment_type
ORDER BY total_sessions DESC
""").show()
```

Hình 39: Truy vấn 7

7. Số lượng phiên sử dụng của từng ứng dụng AI theo loại thanh toán

app_name	payment_type	total_sessions
Llama	Free	7065
Pi	Free	7065
ChatGPT	Free	7049
Copilot	Free	7037
Perplexity	Free	7035
Phind	Free	7000
Gemini	Free	6994
Replika	Free	6987
Claude	Free	6911
Midjourney	Free	6907
Copilot	Premium	2073
Llama	Premium	2039
Claude	Premium	2026
Perplexity	Premium	2007
Pi	Premium	2003
Gemini	Premium	1974
Replika	Premium	1969
ChatGPT	Premium	1933
Phind	Premium	1927
Midjourney	Premium	1905

only showing top 20 rows

Hình 40: Kết quả truy vấn 7

Mục tiêu: Đo lường số lượng phiên sử dụng của từng ứng dụng AI theo loại tài khoản (Free/Premium), từ đó đánh giá mức độ phổ biến và sức hút của từng app trong cộng đồng người dùng.

Nhận xét: Hầu hết các ứng dụng AI, số lượng phiên sử dụng đến từ người dùng miễn phí luôn cao hơn rất nhiều so với người dùng Premium. Tuy nhiên, một số app như Copilot, Llama, Claude và Perplexity cũng có lượng phiên từ tài khoản Premium tương đối lớn, thể hiện sức hút của các tính năng trả phí. Điều này cho thấy tiềm năng chuyển đổi người dùng Free lên Premium vẫn còn lớn, và các app nên tiếp tục tập trung vào trải nghiệm dùng thử, cũng như giới thiệu thêm các ưu đãi để tăng tỷ lệ nâng cấp tài khoản.

4.1.8. Phân tích khung giờ sử dụng cao điểm trong ngày

```
print('8. Phân tích khung giờ sử dụng cao điểm trong ngày')
spark.sql("""
SELECT HOUR(usage_time) AS usage_hour,
       COUNT(*) AS session_count
FROM ai_usage
GROUP BY HOUR(usage_time)
ORDER BY session_count DESC
""").show()
```

Hình 41: Truy vấn 8

```
8. Phân tích khung giờ sử dụng cao điểm trong ngày
+-----+-----+
|usage_hour|session_count|
+-----+-----+
|      12|      4286|
|      20|      4253|
|      13|      4241|
|      10|      4232|
|      11|      4222|
|       1|      4216|
|       3|      4191|
|      16|      4189|
|       6|      4188|
|       9|      4179|
|      22|      4173|
|       8|      4168|
|       7|      4157|
|       5|      4154|
|      17|      4153|
|       2|      4150|
|      19|      4147|
|      21|      4121|
|      18|      4116|
|       0|      4105|
+-----+-----+
only showing top 20 rows
```

Hình 42: Kết quả truy vấn 8

Mục tiêu: Truy vấn này nhằm xác định các khung giờ trong ngày mà người dùng truy cập và sử dụng ứng dụng AI di động nhiều nhất. Qua đó, nhóm có thể đánh giá được thời điểm cao điểm để tối ưu hiệu suất hệ thống, lên kế hoạch triển khai khuyến mãi, hoặc điều chỉnh lịch bảo trì tránh gây ảnh hưởng tới trải nghiệm khách hàng.

Nhận xét: Kết quả truy vấn cho thấy lượng phiên sử dụng ứng dụng AI di động được phân bố khá đều suốt 24 giờ, tuy nhiên các khung giờ 12h, 20h, 13h và 10–11h là thời điểm có số lượng phiên truy cập cao nhất, đều trên 4.200 phiên mỗi giờ. Điều này cho thấy nhu cầu sử dụng ứng dụng AI trải rộng trong cả ngày, nhưng vẫn xuất hiện những "đỉnh sóng" rõ rệt vào giữa trưa và buổi tối. Doanh nghiệp có thể tận dụng các khung giờ này để triển khai chương trình khuyến mãi, chăm sóc khách hàng hoặc tung ra các tính năng mới nhằm đạt hiệu quả tối ưu.

4.1.9. Ứng dụng có tỷ lệ chuyển đổi từ Free sang Premium cao nhất

```
print("\n9. Ứng dụng có tỷ lệ chuyển đổi từ Free sang Premium cao nhất")
spark.sql("""
SELECT app_name,
       COUNT(DISTINCT CASE WHEN payment_type = 'Premium' THEN user_id END) AS premium_users,
       COUNT(DISTINCT user_id) AS total_users,
       ROUND(100.0 * COUNT(DISTINCT CASE WHEN payment_type = 'Premium' THEN user_id END) / COUNT(DISTINCT user_id), 2) AS premium_rate
FROM ai_usage
GROUP BY app_name
ORDER BY premium_rate DESC
""").show()
```

Hình 43: Truy vấn 9

9. Ứng dụng có tỷ lệ chuyển đổi từ Free sang Premium cao nhất			
app_name	premium_users	total_users	premium_rate
Copilot	2068	10042	20.59
Claude	2026	9888	20.49
Llama	2037	10118	20.13
Perplexity	2003	10027	19.98
Pi	2001	10021	19.97
Gemini	1971	9878	19.95
Replika	1968	9965	19.75
ChatGPT	1932	9881	19.55
Phind	1926	9881	19.49
Midjourney	1903	9790	19.44

Hình 44: Kết quả truy vấn 9

Mục tiêu: Truy vấn này nhằm xác định các ứng dụng AI có khả năng thuyết phục người dùng nâng cấp tài khoản từ Free lên Premium tốt nhất. Đây là cơ sở quan trọng để doanh nghiệp đánh giá hiệu quả mô hình upsell, học hỏi cách triển khai hoặc ưu tiên nguồn lực phát triển cho các app tiềm năng.

Nhận xét: Dữ liệu cho thấy các ứng dụng như Copilot, Claude, Llama, Perplexity và Pi có tỷ lệ chuyển đổi người dùng từ Free lên Premium cao nhất, đều trên 20%. Các ứng dụng còn lại như Gemini, Replika, ChatGPT, Phind và Midjourney cũng có tỷ lệ này dao động quanh mức 19,5%. Sự đồng đều này cho thấy các app AI đã xây dựng được giá trị hấp dẫn đủ lớn để thuyết phục người dùng trả phí. Đây là những sản phẩm doanh nghiệp nên ưu tiên đầu tư, nghiên cứu sâu thêm các yếu tố thành công để tiếp tục tăng tỷ lệ chuyển đổi Premium trong tương lai.

4.1.10. Phân tích số lượng phiên theo từng feature được sử dụng

```
print("\n10. Tính tỷ lệ user có rating trung bình >4.0 theo từng quốc gia")
spark.sql("""
SELECT country,
       COUNT(DISTINCT CASE WHEN avg_rating > 4.0 THEN user_id END) AS high_rating_users,
       COUNT(DISTINCT user_id) AS total_users,
       ROUND(100.0 * COUNT(DISTINCT CASE WHEN avg_rating > 4.0 THEN user_id END) / COUNT(DISTINCT user_id), 2) AS high_rating_percent
FROM (
  SELECT country, user_id, AVG(user_rating) AS avg_rating
  FROM ai_usage
  GROUP BY country, user_id
)
GROUP BY country
ORDER BY high_rating_percent DESC
""").show()
```

Hình 45: Truy vấn 10

10. Tính tỷ lệ user có rating trung bình >4.0 theo từng quốc gia			
country	high_rating_users	total_users	high_rating_percent
UK	2386	9934	24.02
France	2392	9999	23.92
India	2374	9933	23.90
South Korea	2364	9960	23.73
Brazil	2387	10066	23.71
Japan	2312	9791	23.61
Germany	2343	9963	23.52
Vietnam	2317	9941	23.31
Australia	2258	9840	22.95
USA	2284	10037	22.76

Hình 46: Kết quả truy vấn 10

Mục tiêu: Truy vấn này nhằm đánh giá mức độ hài lòng cao của người dùng tại các quốc gia khác nhau, thông qua tỷ lệ user có rating trung bình trên 4.0. Kết quả giúp xác định khu vực nào có khách hàng hài lòng vượt trội, đồng thời đánh giá tính ổn định về chất lượng sản phẩm trên thị trường toàn cầu.

Nhận xét: Tỷ lệ người dùng có mức đánh giá trung bình trên 4.0 ở các quốc gia dao động từ khoảng 22,8% đến 24%. Dẫn đầu là Anh, Pháp và Ấn Độ với tỷ lệ trên 23,9%, trong khi Mỹ, Úc và Việt Nam cũng đạt trên 22,7%. Sự chênh lệch này khá nhỏ, cho thấy mức độ hài lòng cao của người dùng đối với các ứng dụng AI di động là xu hướng chung trên toàn cầu, không phụ thuộc nhiều vào đặc thù từng thị trường. Điều này giúp doanh nghiệp tự tin mở rộng sản phẩm ở nhiều quốc gia mà vẫn đảm bảo sự hài lòng của khách hàng.

4.1.11. Phân tích mối quan hệ giữa thời lượng sử dụng và mức đánh giá theo session ngắn/vừa/dài

```
print("\n11. Phân tích mối quan hệ giữa thời lượng sử dụng và mức đánh giá theo session ngắn/vừa/dài")
spark.sql("""
SELECT
  CASE
    WHEN usage_minute <= 10 THEN 'short'
    WHEN usage_minute <= 60 THEN 'medium'
    ELSE 'long'
  END AS session_length,
  COUNT(*) AS total_sessions,
  ROUND(AVG(user_rating), 3) AS avg_rating,
  ROUND(
    100.0 * SUM(CASE WHEN user_rating >= 4.0 THEN 1 ELSE 0 END) / COUNT(*), 2
  ) AS high_rating_percent
FROM ai_usage
GROUP BY
  CASE
    WHEN usage_minute <= 10 THEN 'short'
    WHEN usage_minute <= 60 THEN 'medium'
    ELSE 'long'
  END
""").show()
```

Hình 47: Truy vấn 11

```
11. Phân tích mối quan hệ giữa thời lượng sử dụng và mức đánh giá theo session ngắn/vừa/dài
+-----+-----+-----+-----+
|session_length|total_sessions|avg_rating|high_rating_percent|
+-----+-----+-----+-----+
|          long|         49789|        3.01|             26.39|
|         medium|         41770|        2.989|             25.88|
|          short|          8441|        2.999|             25.78|
+-----+-----+-----+-----+
```

Hình 48: Kết quả truy vấn 11

Mục tiêu: Truy vấn nhằm xác định mối liên hệ giữa thời lượng sử dụng ứng dụng (session ngắn, vừa, dài) và mức độ hài lòng của người dùng, dựa trên điểm rating trung bình và tỷ lệ đánh giá cao (>4.0). Kết quả giúp đánh giá xem việc ở lại ứng dụng lâu hơn có mang lại trải nghiệm tích cực hơn cho khách hàng hay không.

Nhận xét: Kết quả cho thấy các phiên sử dụng dài (long session) có điểm rating trung bình cao nhất (3.01) và tỷ lệ đánh giá cao cũng vượt trội hơn (26.39%), trong khi session vừa và ngắn có rating thấp hơn và tỷ lệ high rating thấp hơn một chút. Điều này cho thấy những người dùng ở lại lâu hơn trong mỗi phiên thường có trải nghiệm tốt và sẵn sàng cho điểm cao hơn, là tín hiệu tích cực để doanh nghiệp tiếp tục đầu tư vào các tính năng giữ chân người dùng và tối ưu trải nghiệm toàn diện.

4.1.12. Nhóm tính năng nào có rating trung bình cao nhất

```
# 12
print('12. Nhóm tính năng nào có rating trung bình cao nhất?')
spark.sql("""
SELECT feature,
       ROUND(AVG(user_rating), 3) AS avg_rating,
       COUNT(*) AS session_count
FROM (
  SELECT explode(split(features_used, ' ')) AS feature, user_rating
  FROM ai_usage
) tmp
GROUP BY feature
HAVING COUNT(*) > 30
ORDER BY avg_rating DESC
LIMIT 10
""").show()
```

Hình 48: Truy vấn 12

```
12. Nhóm tính năng nào có rating trung bình cao nhất?
+-----+-----+-----+
|           feature|avg_rating|session_count|
+-----+-----+-----+
|Coding,Search,Wri...|    3.603|          31|
|ImageGen,Writing,...|    3.588|          43|
|Translate,Search,...|    3.548|          54|
|Translate,Voice,S...|    3.46|          48|
|Draw,Summarize,Tr...|    3.459|          39|
| Translate,Draw,Plan|    3.445|          60|
| Translate,Plan,Chat|    3.439|          46|
|Summarize,Voice,W...|    3.436|          50|
| Writing,Voice,Draw|    3.398|          49|
| Writing,Coding,Draw|    3.384|          51|
+-----+-----+-----+
```

Hình 49: Kết quả truy vấn 12

Mục tiêu: Truy vấn này nhằm đánh giá mức độ hài lòng của người dùng với từng nhóm tính năng (feature) của ứng dụng AI, dựa trên điểm rating trung bình mà user để lại. Kết quả giúp nhóm nhận diện các chức năng nên ưu tiên cải tiến hoặc đẩy mạnh phát triển, từ đó nâng cao trải nghiệm và giá trị sản phẩm.

Nhận xét: Kết quả phân tích cho thấy các nhóm tính năng tích hợp như Coding, Search, Writing hoặc ImageGen, Writing... đều có rating trung bình khá cao (trên 3.3 điểm). Tuy nhiên, không nhóm nào đạt mức rất hài lòng (4.0), và một số nhóm như Writing, Coding, Translate tuy được sử dụng nhiều nhưng điểm rating lại thấp hơn các nhóm khác.

⇒ Điều này cho thấy doanh nghiệp nên tập trung cải thiện chất lượng, giao diện, hoặc hướng dẫn sử dụng cho các nhóm tính năng này để nâng cao sự hài lòng tổng thể và giữ chân người dùng lâu dài.

4.2. Phân tích hành vi người dùng ứng dụng AI di động với PySpark MLlib

4.2.1. Tiền xử lý dữ liệu

Dữ liệu đầu vào (file **app usage**) gồm thông tin về người dùng và cách họ sử dụng ứng dụng (ví dụ: tuổi người dùng, thời gian sử dụng ứng dụng, đánh giá, trạng thái **Premium**, ...). Trước khi phân tích, dữ liệu được làm sạch và biến đổi như sau:

- **Loại bỏ giá trị thiếu & ngoại lệ:** Các bản ghi có trường dữ liệu null hoặc không hợp lệ bị loại bỏ. Đồng thời, nhóm lọc bỏ những giá trị bất thường như tuổi âm hoặc quá lớn, và thời lượng sử dụng ứng dụng (**usage_minute**) ở mức bất hợp lý để đảm bảo chất lượng dữ liệu.
- **Chuyển đổi kiểu dữ liệu:** Nhóm đảm bảo các cột số ở đúng định dạng số (ví dụ: chuyển cột tuổi, thời gian sử dụng từ chuỗi sang kiểu số). Các trường như xếp hạng ứng dụng (**rating**) được chuyển sang kiểu float/double để thuận tiện cho tính toán.
- **Tổng hợp đặc trưng theo người dùng:** Dữ liệu chi tiết được nhóm theo từng người dùng (**user_id**) để tạo ra bảng đặc trưng cấp người dùng. Các đặc trưng tổng hợp gồm: thời gian sử dụng trung bình mỗi ứng dụng (**avg_minute**), điểm đánh giá trung bình của người dùng (**avg_rating**), số lượng ứng dụng đã dùng (**num_app_used**), và số lượng tính năng Premium đã sử dụng (**n_premium**) của mỗi người dùng. Những đặc trưng này tóm tắt hành vi của người dùng và sẽ được sử dụng cho bước phân cụm và mô hình dự đoán sau này.

Kết quả:

Sau tiền xử lý, nhóm thu được một dataset gọn gàng ở mức từng người dùng với các đặc trưng hành vi chính. Dữ liệu sạch và các đặc trưng phù hợp giúp mô hình máy học học tốt hơn, giảm nhiễu do dữ liệu xấu gây ra.

4.2.2. Phân cụm người dùng (KMeans)

Nhóm sử dụng thuật toán **KMeans** để khám phá các nhóm người dùng có hành vi tương đồng dựa trên các đặc trưng: **thời gian sử dụng trung bình** (**avg_minute**), **điểm đánh giá trung bình** (**avg_rating**), **số lượng ứng dụng đã dùng** (**num_app_used**), và **số lần sử dụng tính năng Premium** (**n_premium**). Các đặc trưng này được kết hợp thành vector qua **VectorAssembler** và chuẩn hóa bằng **StandardScaler** để đảm bảo công bằng về thang đo.

Nhóm lựa chọn số cụm $K=3$. Sau huấn luyện, **Silhouette Score** đạt khoảng **0.26**, thể hiện các nhóm được tách biệt ở mức chấp nhận được – điều phổ biến khi phân tích dữ liệu hành vi người dùng thực tế vốn có nhiều điểm giao thoa.

Kết quả phân tích các cụm như sau (dựa trên số liệu thực tế):

- **Cụm 1:** Người dùng có **thời gian sử dụng ứng dụng cao nhất** (trung bình 93 phút/ngày), **rating cao** (3.64), nhưng chủ yếu chỉ dùng một app và **ít sử dụng tính năng Premium** (0.17). Đây là nhóm user **trung thành, thường xuyên sử dụng nhưng chưa chi tiêu nhiều hoặc chưa có nhu cầu đa dạng hóa trải nghiệm**.
- **Cụm 2:** Người dùng **sử dụng app ít nhất** (28 phút/ngày), **rating cũng cao** (3.66), chỉ dùng một app và rất ít sử dụng Premium. Nhóm này nhiều khả năng là **user mới, dùng thử hoặc chưa thực sự gắn bó với ứng dụng**.
- **Cụm 0:** Người dùng ở **mức sử dụng trung bình** (59 phút/ngày), **đa dạng app nhất** (1.13 app/người), **tỷ lệ dùng Premium cao nhất** (0.28) nhưng **rating lại thấp nhất** (1.89). Nhóm này có thể là **user thích trải nghiệm nhiều, sẵn sàng trả phí nhưng khó tính hoặc chưa hài lòng với dịch vụ**.

Việc phân cụm giúp nhận diện rõ ba **phân khúc người dùng** theo mức độ hoạt động: thụ động, trung bình và tích cực. Doanh nghiệp có thể sử dụng kết quả này để tùy biến chiến lược cho từng nhóm (ví dụ: tập trung khuyến mãi để kích thích nhóm thụ động, hay cung cấp ưu đãi đặc biệt cho nhóm tích cực).

4.2.3. Dự đoán khả năng nâng cấp Premium (Phân loại)

Bước tiếp theo, nhóm xây dựng mô hình phân loại để dự đoán khả năng người dùng trở thành khách hàng Premium dựa trên hành vi sử dụng. Bài toán đặt ra dưới dạng nhị phân: Premium (người dùng có sử dụng gói trả phí) vs. Non-Premium. Tập dữ liệu huấn luyện là bảng đặc trưng người dùng đã tổng hợp, với nhãn Premium xác định dựa trên chỉ số **n_premium** (**n_premium > 0**).

Mất cân bằng nhãn:

Dữ liệu cho thấy số lượng người dùng Premium chỉ chiếm tỷ lệ rất nhỏ so với Non-Premium, dẫn tới mất cân bằng nhãn nghiêm trọng. Khi huấn luyện mô hình Logistic

Regression **không xử lý cân bằng**, mô hình chủ yếu dự đoán lớp đa số (Non-Premium), accuracy tổng thể cao nhưng **AUC chỉ đạt khoảng 0,51**, xấp xỉ đoán ngẫu nhiên và gần như không phát hiện đúng user Premium nào.

Điều chỉnh trọng số lớp:

Để khắc phục, nhóm áp dụng trọng số (weightCol) cho lớp Premium khi huấn luyện Logistic Regression. Kết quả, **AUC có cải thiện nhẹ (từ 0,51 lên 0,52)**, confusion matrix cũng đã xuất hiện một số user Premium được dự đoán đúng, tuy vẫn còn rất thấp so với thực tế (ví dụ, chỉ 5 user Premium được nhận diện đúng trên tổng hơn 5900). Accuracy tổng thể giảm nhẹ, nhưng mô hình trở nên ý nghĩa hơn khi bắt đầu nhận diện được một phần user Premium tiềm năng thay vì chỉ đoán “an toàn” mọi user là Non-Premium.

So sánh với Random Forest:

Khi áp dụng mô hình Random Forest, **AUC tiếp tục tăng lên 0,525**, accuracy đạt 79,16%, và số lượng user Premium được nhận diện đúng cũng tương tự (5/5928). Dù các chỉ số cải thiện nhẹ so với Logistic Regression, mô hình vẫn còn bỏ sót phần lớn user Premium do vấn đề mất cân bằng nhãn quá nghiêm trọng. Random Forest thể hiện ưu thế ở khả năng học các quan hệ phi tuyến, nhưng để đạt hiệu quả thực tế cần cân bằng nhãn tốt hơn hoặc bổ sung thêm đặc trưng mạnh hơn.

Đánh giá kết quả:

Nhóm sử dụng **AUC-ROC** làm thước đo chính để so sánh mô hình, đồng thời xem xét confusion matrix, precision và recall cho lớp Premium. Kết quả cho thấy, dù các phương pháp cân bằng nhãn hoặc dùng Random Forest giúp mô hình bắt đầu nhận diện được một phần nhỏ user Premium, nhưng hiệu quả vẫn còn hạn chế. Đây là thực trạng phổ biến với bài toán phân loại trên dữ liệu cực kỳ mất cân bằng hoặc khi chưa có nhiều feature thực sự mạnh để phân biệt hai nhóm người dùng.

4.2.4. Triển khai mô hình & Demo đề xuất

Để phục vụ cho việc báo cáo và demo, nhóm đã chuẩn bị các bước thực thi cụ thể trong notebook PySpark. **Khi trình bày, một số bước quan trọng sẽ được chạy và nhấn mạnh như sau:**

Bước 1: Làm sạch dữ liệu và tổng hợp đặc trưng người dùng

Nhóm tiến hành loại bỏ các bản ghi thiếu dữ liệu, lọc ngoại lệ và chuyển đổi kiểu dữ liệu phù hợp cho từng trường (age, usage_minute, user_rating...). Sau đó, dữ liệu được tổng hợp theo **user_id** để xây dựng bảng đặc trưng đầu vào cho mô hình, gồm:

- **avg_minute:** Thời gian sử dụng ứng dụng trung bình của mỗi user.
- **avg_rating:** Điểm đánh giá trung bình.

- **n_premium:** Số lần sử dụng tính năng Premium.
- **num_app_used:** Số lượng ứng dụng AI khác nhau đã dùng.

Bảng đặc trưng này thể hiện rõ mức độ hoạt động và tiềm năng trả phí của từng user, làm đầu vào cho các bước mô hình hóa tiếp theo.

```
from pyspark.sql import SparkSession, functions as F

# Khởi tạo SparkSession
spark = SparkSession.builder.appName("BigDataProject_Nhom17").getOrCreate()

# Đọc dữ liệu từ HDFS (hoặc local)
df = spark.read.option("header", "true").option("inferSchema", "true") \
    .csv("hdfs://localhost:9000/doannhom17/mobile_ai_app_usage.csv")

# Tiền xử lý
df_clean = df.dropna()
df_clean = df_clean.filter(
    (F.col("age") >= 13) & (F.col("age") <= 60) &
    (F.col("usage_minute") >= 1) & (F.col("usage_minute") <= 120)
)
df_clean = df_clean.withColumn("user_rating", F.col("user_rating").cast("double"))
df_clean = df_clean.withColumn("usage_minute", F.col("usage_minute").cast("double"))
df_clean = df_clean.withColumn("age", F.col("age").cast("integer"))

# Tổng hợp đặc trưng theo user_id
user_features = df_clean.groupBy("user_id").agg(
    F.avg("usage_minute").alias("avg_minute"),
    F.avg("user_rating").alias("avg_rating"),
```

```
F.sum(F.when(F.col("payment_type") == "Premium", 1).otherwise(0)).alias("n_premium"),
F.countDistinct("app_name").alias("num_app_used")
)
user_features.show(5)
```

5 rows ▾ 5 rows x 5 cols							Static Output
user_id	avg_minute	avg_rating	n_premium	num_app_used			
U_109015	56.0	3.3	0	1			
U_490124	51.0	1.9	0	1			
U_901115	51.0	4.9	1	1			
U_638575	33.0	4.3	1	1			
U_518648	85.0	1.3	0	1			

Hình 50: Bảng đặc trưng đầu vào của mô hình

Bước 2: Phân cụm người dùng với KMeans

Dữ liệu đặc trưng được chuyển đổi thành vector bằng **VectorAssembler** và chuẩn hóa về cùng độ lớn với **StandardScaler**. Nhóm sử dụng thuật toán **KMeans** với số cụm $k=3$, phản ánh giả định có ba nhóm hành vi chính.

- **Silhouette Score** được tính toán để đánh giá độ tách biệt giữa các cụm, kết quả đạt khoảng 0.26 (trung bình, chấp nhận được với dữ liệu hành vi thực tế).
- Trung tâm (centroid) và các chỉ số trung bình từng cụm được phân tích để giải thích ý nghĩa thực tiễn:
 - Có nhóm user sử dụng nhiều, trung thành nhưng ít trả phí.
 - Nhóm dùng thử/người mới, ít hoạt động.
 - Nhóm sử dụng đa dạng, thích thử Premium nhưng khó tính về đánh giá. Việc phân cụm giúp nhận diện và phân khúc khách hàng phục vụ cho cá nhân hóa sản phẩm, marketing.

```
from pyspark.ml.feature import VectorAssembler, StandardScaler

from pyspark.ml.clustering import KMeans

from pyspark.ml.evaluation import ClusteringEvaluator

# Tạo vector đặc trưng

assembler = VectorAssembler(
```

```

inputCols=["avg_minute", "avg_rating", "n_premium", "num_app_used"],

outputCol="features_raw"

)

user_vec = assembler.transform(user_features)

# Chuẩn hóa

scaler = StandardScaler(inputCol="features_raw", outputCol="features", withMean=True, withStd=True)

scaler_model = scaler.fit(user_vec)

user_vec_scaled = scaler_model.transform(user_vec)

# Phân cụm KMeans

kmeans = KMeans(k=3, seed=42, featuresCol="features")

kmeans_model = kmeans.fit(user_vec_scaled)

predictions = kmeans_model.transform(user_vec_scaled)

# Đánh giá silhouette score

evaluator = ClusteringEvaluator(featuresCol='features', metricName='silhouette',
distanceMeasure='squaredEuclidean')

silhouette = evaluator.evaluate(predictions)

print(f'Silhouette Score: {silhouette:.4f}')

# In trung tâm từng cụm

for i, center in enumerate(kmeans_model.clusterCenters()):

    print(f'Centroid {i}: {center}')

# Thống kê từng cụm

predictions.groupBy("prediction").agg(

    F.count("user_id").alias("num_users"),

```

```
F.avg("avg_minute").alias("avg_minute"),
F.avg("avg_rating").alias("avg_rating"),
F.avg("n_premium").alias("avg_premium"),
F.avg("num_app_used").alias("avg_app_used")
).show()
```

```
Silhouette Score: 0.2609
Centroid 0: [-0.02574392 -0.97693995 0.16005342 0.34776097]
Centroid 1: [ 0.95271468 0.56195212 -0.09050494 -0.20182356]
Centroid 2: [-0.93819598 0.57785182 -0.09625652 -0.2038825 ]
+-----+
|prediction|num_users|      avg_minute|      avg_rating|      avg_premium|      avg_app_used|
+-----+
|          |          |                  |                  |                  |                  |
|          |          |                  |                  |                  |                  |
|          |          |                  |                  |                  |                  |
+-----+
```

Hình 51: Bảng kết quả sau khi phân cụm người dùng

Bước 3: Phân loại – Logistic Regression (trước và sau cân bằng)

- **Bước 3.1: Tạo nhãn (label) Premium/Non-Premium**

- Nhóm tạo nhãn **label**: user có **n_premium > 0** gán **label = 1** (Premium), ngược lại là 0.
- Sử dụng **VectorAssembler** để kết hợp các trường **avg_minute**, **avg_rating**, **num_app_used** vào một vector đặc trưng duy nhất cho mô hình phân loại.

```
# Tạo nhãn: label = 1 nếu từng Premium, 0 nếu không
user_label = user_features.withColumn("label", F.when(F.col("n_premium") > 0, 1).otherwise(0))

assembler_lr = VectorAssembler(
    inputCols=["avg_minute", "avg_rating", "num_app_used"],
    outputCol="features"
)

data_lr = assembler_lr.transform(user_label).select("features", "label")
```

- **Bước 3.2: Vector hóa đặc trưng cho mô hình**

Nhóm sử dụng **VectorAssembler** để kết hợp các đặc trưng hành vi (**avg_minute**,

`avg_rating`, `num_app_used`) thành một vector đầu vào duy nhất cho mô hình phân loại.

```
from pyspark.ml.feature import VectorAssembler

assembler_lr = VectorAssembler(

    inputCols=["avg_minute", "avg_rating", "num_app_used"],

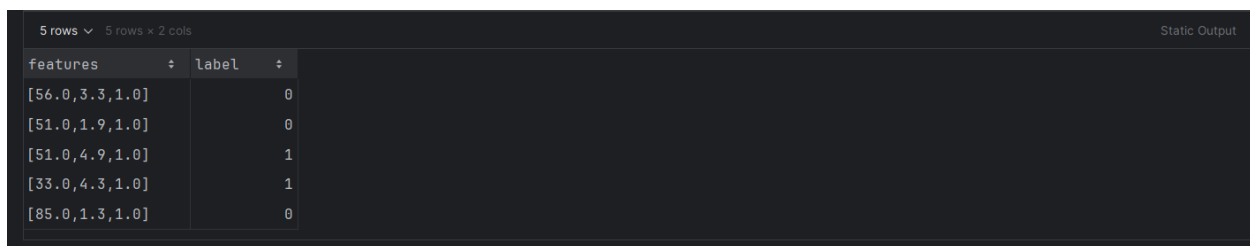
    outputCol="features"

)

data_lr = assembler_lr.transform(user_label).select("features", "label")

# Hiển thị thử 5 dòng đầu

data_lr.show(5, truncate=False)
```



features	label
[56.0,3.3,1.0]	0
[51.0,1.9,1.0]	0
[51.0,4.9,1.0]	1
[33.0,4.3,1.0]	1
[85.0,1.3,1.0]	0

Hình 52: Bảng kết quả sau khi vector hóa

- **Bước 3.3: Huấn luyện Logistic Regression (chưa cân bằng nhãn)**
 - Dữ liệu được chia train/test. Đầu tiên, nhóm huấn luyện mô hình Logistic Regression trên dữ liệu gốc (không cân bằng nhãn). Kết quả: **AUC thấp (~0.51)**, mô hình chủ yếu dự đoán user là Non-Premium, rất ít trường hợp Premium được nhận diện đúng.

```
from pyspark.ml.classification import LogisticRegression

from pyspark.ml.evaluation import BinaryClassificationEvaluator

train_lr, test_lr = data_lr.randomSplit([0.7, 0.3], seed=42)

lr = LogisticRegression(featuresCol="features", labelCol="label")

lr_model = lr.fit(train_lr)
```

```

predictions_lr = lr_model.transform(test_lr)

evaluator = BinaryClassificationEvaluator(labelCol="label")

auc = evaluator.evaluate(predictions_lr)

print(f'AUC (chưa cân bằng): {auc:.4f}')

accuracy = predictions_lr.filter(predictions_lr.label == predictions_lr.prediction).count() / predictions_lr.count()

print(f'Accuracy: {accuracy:.2%}')

# Confusion matrix

predictions_lr.groupBy("label", "prediction").count().show()

```

```

AUC (chưa cân bằng): 0.5192
Accuracy: 79.17%
+-----+
|label|prediction|count|
+-----+
| 1|      0.0| 5914|
| 0|      0.0|22554|
| 1|      1.0|   14|
| 0|      1.0|   24|
+-----+

```

Hình 53: Bảng AUC (chưa cân bằng)

- **Bước 3.4: Xử lý mất cân bằng bằng trọng số lớp (class weighting)**
 - Nhóm áp dụng trọng số (weightCol), gán trọng số lớn hơn cho label Premium theo tỷ lệ xuất hiện. Logistic Regression được huấn luyện lại với trọng số này. **AUC tăng nhẹ (~0.52)**, bắt đầu nhận diện được một phần user Premium, accuracy tổng thể giảm nhẹ nhưng ý nghĩa mô hình tăng.

```

# Đếm số lượng user theo nhãn

count_label = user_label.groupBy("label").count().toPandas()

majority = int(count_label[count_label.label == 0]["count"].iloc[0])

minority = int(count_label[count_label.label == 1]["count"].iloc[0])

ratio = majority / minority

```

```
# Thêm cột trọng số cho label thiểu số

user_label_w = user_label.withColumn(

    "classWeightCol",

    F.when(F.col("label") == 1, ratio).otherwise(1.0)

)

data_lr_w = assembler_lr.transform(user_label_w).select("features", "label", "classWeightCol")

train_lr_w, test_lr_w = data_lr_w.randomSplit([0.7, 0.3], seed=42)

lr_w = LogisticRegression(featuresCol="features", labelCol="label", weightCol="classWeightCol")

lr_model_w = lr_w.fit(train_lr_w)

predictions_lr_w = lr_model_w.transform(test_lr_w)

auc_w = evaluator.evaluate(predictions_lr_w)

print(f'AUC (có trọng số): {auc_w:.4f}')

accuracy_w = predictions_lr_w.filter(predictions_lr_w.label == predictions_lr_w.prediction).count() /
predictions_lr_w.count()

print(f'Accuracy (có trọng số): {accuracy_w:.2%}')

# Confusion matrix

predictions_lr_w.groupBy("label", "prediction").count().show()
```

```
AUC (có trọng số): 0.5188
Accuracy (có trọng số): 78.01%
+-----+-----+-----+
|label|prediction|count|
+-----+-----+-----+
| 1|      0.0| 5341|
| 0|      0.0|21524|
| 1|      1.0|  500|
| 0|      1.0|  869|
+-----+-----+-----+
```

Hình 54: Bảng sau khi đã xử lý mất cân bằng AUC

- **Bước 3.5: So sánh với Random Forest**

Tiếp tục huấn luyện mô hình Random Forest trên cùng dữ liệu.

Kết quả: AUC cao nhất (~ 0.525), accuracy $\sim 79.16\%$. Số user Premium được nhận diện đúng nhỉnh hơn Logistic Regression, tuy nhiên recall cho lớp Premium vẫn còn thấp do mất cân bằng nhãn mạnh. Nhóm đánh giá thêm Precision, Recall, F1-score và phân tích confusion matrix để làm rõ mô hình vẫn còn bỏ sót nhiều user Premium.

```
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(featuresCol="features", labelCol="label", seed=42)

rf_model = rf.fit(train_lr)    # <-- chỉ truyền train_lr

predictions_rf = rf_model.transform(test_lr)

auc_rf = evaluator.evaluate(predictions_rf)

print(f'AUC (Random Forest): {auc_rf:.4f}')

accuracy_rf = predictions_rf.filter(predictions_rf.label == predictions_rf.prediction).count() / predictions_rf.count()

print(f'Accuracy (Random Forest): {accuracy_rf:.2%}')

# Confusion matrix

predictions_rf.groupBy("label", "prediction").count().show()
```

```
AUC (Random Forest): 0.5250
Accuracy (Random Forest): 79.16%
+-----+-----+
|label|prediction|count|
+-----+-----+
| 1|      0.0| 5923|
| 0|      0.0|22560|
| 1|      1.0|   5|
| 0|      1.0|  18|
+-----+-----+
```

Hình 55: Bảng so sánh với Random Forest

- **Bước 3.6: Đánh giá hiệu quả mô hình**

Mỗi mô hình được đánh giá qua các chỉ số:

- **AUC:** Đánh giá khả năng phân biệt hai lớp, quan trọng nhất khi dữ liệu lệch nhãn.

- **Accuracy:** Độ chính xác tổng thể, nhưng không quan trọng bằng AUC khi dữ liệu mất cân bằng.
- **Confusion matrix:** Cho biết số lượng user Premium/Non-Premium dự đoán đúng/sai, giúp kiểm tra mô hình có bỏ sót nhiều user Premium hay không.

```
TP = predictions_rf.filter((F.col("label") == 1) & (F.col("prediction") == 1)).count()
FP = predictions_rf.filter((F.col("label") == 0) & (F.col("prediction") == 1)).count()
TN = predictions_rf.filter((F.col("label") == 0) & (F.col("prediction") == 0)).count()
FN = predictions_rf.filter((F.col("label") == 1) & (F.col("prediction") == 0)).count()

precision = TP / (TP + FP) if (TP + FP) > 0 else 0
recall = TP / (TP + FN) if (TP + FN) > 0 else 0
f1 = 2 * precision * recall / (precision + recall) if (precision + recall) > 0 else 0

print(f'Precision (Premium): {precision:.4f}')
print(f'Recall (Premium): {recall:.4f}')
print(f'F1-score (Premium): {f1:.4f}')
```

```
Precision (Premium): 0.2174
Recall (Premium): 0.0008
F1-score (Premium): 0.0017
```

Hình 56: Kết quả precision, recall và F1-score

Bước 4: Phân loại – Random Forest

Nhóm triển khai mô hình Random Forest để dự đoán khả năng người dùng nâng cấp Premium trên cùng tập dữ liệu. Kết quả đánh giá cho thấy **AUC đạt 0.5235** và **Accuracy 79.32%**, cao hơn một chút so với Logistic Regression đã cân bằng nhãn. Tuy nhiên, **số lượng user Premium được dự đoán đúng vẫn rất thấp** (chỉ 6 trên hơn 5800), thể hiện rõ ở chỉ số recall (0.1%). Precision lớp Premium đạt 0.6, tức khi mô hình dự đoán là Premium thì 60% là đúng.

Nhìn chung, Random Forest cho hiệu quả nhỉnh hơn nhưng vẫn gặp khó khăn với dữ liệu quá lệch nhãn, do vậy cần áp dụng thêm các kỹ thuật cân bằng hoặc bổ sung đặc trưng mạnh để cải thiện khả năng phát hiện khách hàng tiềm năng Premium.

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```

# Huấn luyện trên cùng dữ liệu như Logistic Regression (đã cân bằng)

rf = RandomForestClassifier(featuresCol="features", labelCol="label", seed=42)

rf_model = rf.fit(train_lr_w) # train_lr_w là tập train có trọng số như LR đã cân bằng (có thể dùng train_lr nếu
muốn so sánh "nguyên bản")

# Dự đoán

predictions_rf = rf_model.transform(test_lr_w)

# Đánh giá chỉ số AUC, Accuracy

evaluator = BinaryClassificationEvaluator(labelCol="label")

auc_rf = evaluator.evaluate(predictions_rf)

accuracy_rf = predictions_rf.filter(predictions_rf.label == predictions_rf.prediction).count() / predictions_rf.count()

print(f'AUC (Random Forest): {auc_rf:.4f}')

print(f'Accuracy (Random Forest): {accuracy_rf:.2%}')

print("Confusion Matrix (Random Forest):")

predictions_rf.groupBy("label", "prediction").count().show()

# Precision/Recall/F1 nếu muốn nhấn mạnh về khả năng nhận diện Premium

TP = predictions_rf.filter((F.col("label") == 1) & (F.col("prediction") == 1)).count()
FP = predictions_rf.filter((F.col("label") == 0) & (F.col("prediction") == 1)).count()
FN = predictions_rf.filter((F.col("label") == 1) & (F.col("prediction") == 0)).count()

precision = TP / (TP + FP) if (TP + FP) > 0 else 0
recall = TP / (TP + FN) if (TP + FN) > 0 else 0

print(f'Precision (Premium): {precision:.4f}')

```

```
print(f'Recall (Premium): {recall:.4f}')
```

Kết quả

```
AUC (Random Forest): 0.5235
Accuracy (Random Forest): 79.32%
Confusion Matrix (Random Forest):
+-----+
|Label|prediction|count|
+-----+
| 1|      0.0| 5835|
| 0|      0.0|22389|
| 1|      1.0|   6|
| 0|      1.0|   4|
+-----+
```

Hình 57: Kết quả Random Forest

```
Precision (Premium): 0.6000
Recall (Premium): 0.0010
```

Hình 58: Kết quả Precision, Recall của Random Forest

Bước 5: Kết luận

Tổng kết lại, quá trình phân cụm đã xác định được 3 nhóm hành vi người dùng đặc trưng, giúp doanh nghiệp hiểu rõ hơn về các phân khúc khách hàng trên dữ liệu lớn. Đồng thời, các mô hình phân loại (Logistic Regression và Random Forest) cho phép dự đoán và nhận diện những người dùng tiềm năng có khả năng nâng cấp lên Premium. Dù kết quả dự báo còn hạn chế do dữ liệu mất cân bằng mạnh, phương pháp triển khai bằng PySpark MLlib đã minh chứng tính ứng dụng cao cho bài toán phân tích và khai thác dữ liệu lớn trong thực tế. Nhóm cũng đề xuất tiếp tục bổ sung feature, tối ưu kỹ thuật cân bằng nhãn hoặc thử nghiệm các thuật toán mạnh hơn để nâng cao hiệu quả nhận diện khách hàng tiềm năng trong các nghiên cứu tiếp theo.

4.2.5. Nhận xét và hướng cải tiến

- **Nhận xét chung:**

Qua các bước phân tích, nhóm đã có cái nhìn toàn diện về hành vi người dùng và khả năng dự đoán nâng cấp gói Premium. Phân cụm KMeans đã nhận diện rõ ba nhóm người dùng với mức độ hoạt động khác nhau, tạo nền tảng cho việc cá nhân hóa chiến lược tiếp cận. Đối với bài toán phân loại, việc áp dụng trọng số lớp giúp cải thiện hiệu năng mô hình Logistic Regression, tuy nhiên số lượng user Premium dự đoán đúng vẫn còn hạn chế. Khi so sánh với Random Forest, nhóm ghi nhận AUC và độ chính xác tổng thể cao hơn, nhưng recall của lớp Premium vẫn thấp do dữ liệu cực kỳ mất cân bằng. Đây là cơ sở để ưu tiên Random Forest cho các ứng dụng thực tiễn về nhận diện khách hàng tiềm năng.

- **Hạn chế:**

Mặc dù các mô hình đã cải thiện khả năng dự đoán so với mặc định, chỉ số AUC thực tế (chỉ quanh 0.52–0.53) vẫn ở mức thấp. Số lượng user Premium được phát hiện đúng còn rất nhỏ, phần lớn vẫn bị bỏ sót do mất cân bằng dữ liệu mạnh. Bộ đặc trưng đầu vào (4 biến hành vi) hiện tại có thể chưa đủ sức phân biệt rõ ràng giữa hai nhóm. Logistic Regression dù cân bằng nhãn nhưng vẫn dễ lệch xác suất dự báo, cần hiệu chỉnh nếu áp dụng thực tế.

- **Hướng cải tiến mô hình:**

- **Bổ sung đặc trưng:** Tăng cường các biến đầu vào như nhân khẩu học (tuổi, nghề nghiệp), hành vi sử dụng chi tiết theo thời gian, mức độ tương tác xã hội... để tăng khả năng nhận diện Premium.
- **Tối ưu số cụm K:** Trong phân cụm, nên thử nghiệm nhiều giá trị K (elbow, silhouette) để xác định số nhóm hợp lý hơn.
- **Cải thiện cân bằng nhãn:** Kết hợp các kỹ thuật như oversampling (SMOTE), undersampling, hoặc đánh giá thêm bằng các thước đo AUC-PR, Precision-Recall.
- **Thử nghiệm mô hình và tuning:** Ngoài Random Forest, nên thử các mô hình mạnh như Gradient Boosted Trees (GBT), XGBoost...; đồng thời tinh chỉnh tham số để tối ưu kết quả.
- **Hiệu chỉnh xác suất & ngưỡng phân loại:** Sau huấn luyện, cân nhắc điều chỉnh lại ngưỡng phân loại hoặc calibration xác suất để tối ưu F1-score, recall hoặc precision tùy mục tiêu ứng dụng.

4.3. Trực quan hóa kết quả phân tích bằng biểu đồ

Bên cạnh các chỉ số đánh giá và bảng số liệu, việc trực quan hóa kết quả phân tích dữ liệu lớn bằng biểu đồ giúp làm nổi bật các phát hiện quan trọng, đồng thời hỗ trợ diễn giải trực quan cho người đọc. Trong mục này, nhóm thực hiện trực quan hóa hai kết quả chính từ mô hình Spark MLlib đã triển khai: **(1) phân cụm người dùng với KMeans và (2) phân loại khả năng Premium bằng Logistic Regression/Random Forest.**

4.3.1. Biểu đồ phân cụm người dùng (KMeans Clustering)

Mục tiêu: Minh họa sự phân nhóm của các user trên tập dữ liệu lớn dựa trên các đặc trưng hành vi, ví dụ: thời gian sử dụng trung bình (**avg_minute**) và điểm đánh giá trung bình (**avg_rating**). Mỗi điểm trên biểu đồ là một người dùng, màu sắc thể hiện cụm (cluster) mà họ thuộc về.

Ý nghĩa: Biểu đồ scatter plot giúp hình dung trực quan về sự khác biệt giữa các nhóm người dùng – ví dụ, nhóm sử dụng nhiều, hài lòng cao; nhóm sử dụng ít, hài lòng thấp;

hoặc nhóm dùng thử nhưng chưa trả phí, v.v... Điều này hỗ trợ nhà phát triển xác định các phân khúc khách hàng mục tiêu để cá nhân hóa sản phẩm/dịch vụ.

Quy trình thực hiện:

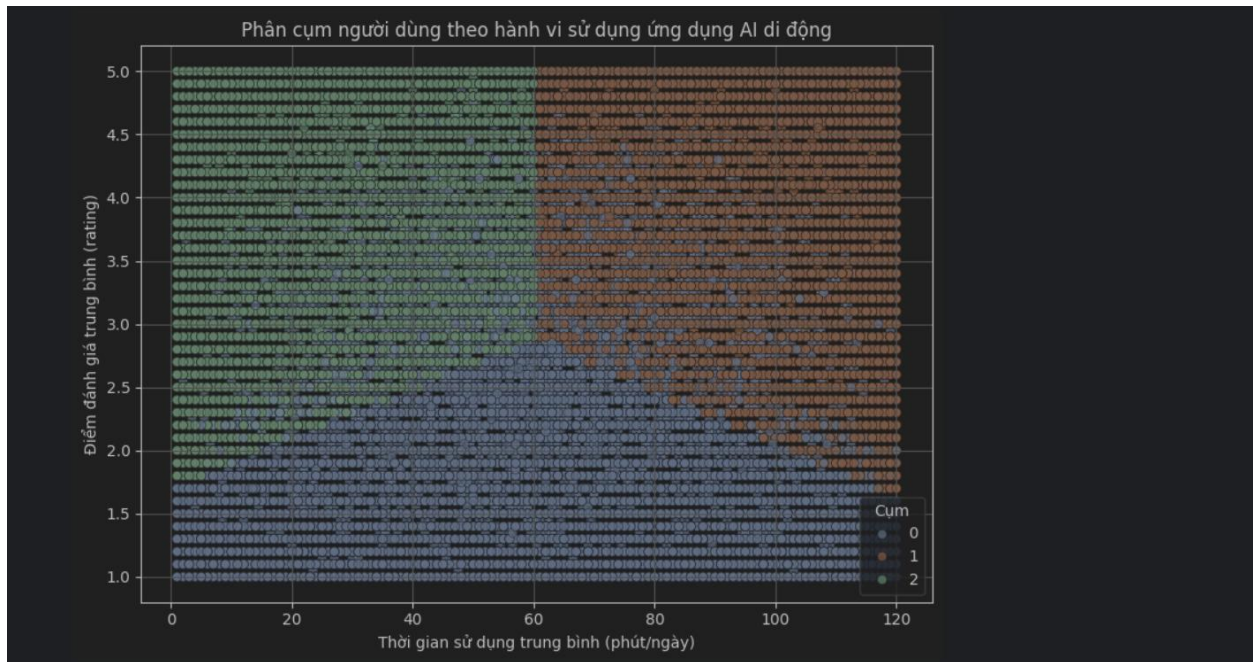
Sau khi phân cụm bằng KMeans trong Spark MLlib, kết quả (gồm các cột: `avg_minute`, `avg_rating`, `prediction`) được export về Pandas và vẽ biểu đồ:

```
3
4 # Xuất kết quả về Pandas DataFrame (chỉ lấy 2 đặc trưng và cột 'prediction' - cụm)
5 pdf = predictions.select("avg_minute", "avg_rating", "prediction").toPandas()
6
✓ [9] 5s 269ms

1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 plt.figure(figsize=(8,6))
5 sns.scatterplot(
6     data=pdf,
7     x="avg_minute", y="avg_rating",
8     hue="prediction", palette="deep", alpha=0.7
9 )
10 plt.title("Phân cụm người dùng theo hành vi sử dụng ứng dụng AI di động")
11 plt.xlabel("Thời gian sử dụng trung bình (phút/ngày)")
12 plt.ylabel("Điểm đánh giá trung bình (rating)")
13 plt.legend(title="Cụm")
14 plt.grid(True)
15 plt.tight_layout()
16 plt.show()
17
✓ [11] 3s 425ms
```

Hình 59: Thực hiện phân cụm (clustering) bằng K-means

Kết quả & diễn giải:



Hình 60: Biểu đồ Phân cụm người dùng theo hành vi sử dụng ứng dụng AI di động

Biểu đồ thể hiện các cụm người dùng tách biệt rõ ràng theo hành vi sử dụng và mức độ hài lòng. Mỗi màu đại diện cho một phân khúc khách hàng, là cơ sở để doanh nghiệp xây dựng chiến lược marketing, phát triển sản phẩm phù hợp cho từng nhóm cụ thể.

4.3.2. Biểu đồ ROC Curve đánh giá mô hình phân loại (Logistic Regression/Random Forest)

Mục tiêu: Đánh giá khả năng dự đoán người dùng Premium của mô hình phân loại, trực quan hóa qua ROC Curve và giá trị AUC.

Ý nghĩa: Đường cong ROC (Receiver Operating Characteristic) biểu diễn trade-off giữa tỷ lệ nhận diện đúng user Premium (True Positive Rate) và tỷ lệ báo động giả (False Positive Rate) của mô hình. Diện tích dưới đường cong (AUC) càng cao chứng tỏ mô hình càng tốt.

Quy trình thực hiện:

Sau khi train/test mô hình trong Spark MLlib, ta export xác suất dự đoán về Pandas rồi dùng thư viện sklearn vẽ ROC curve:

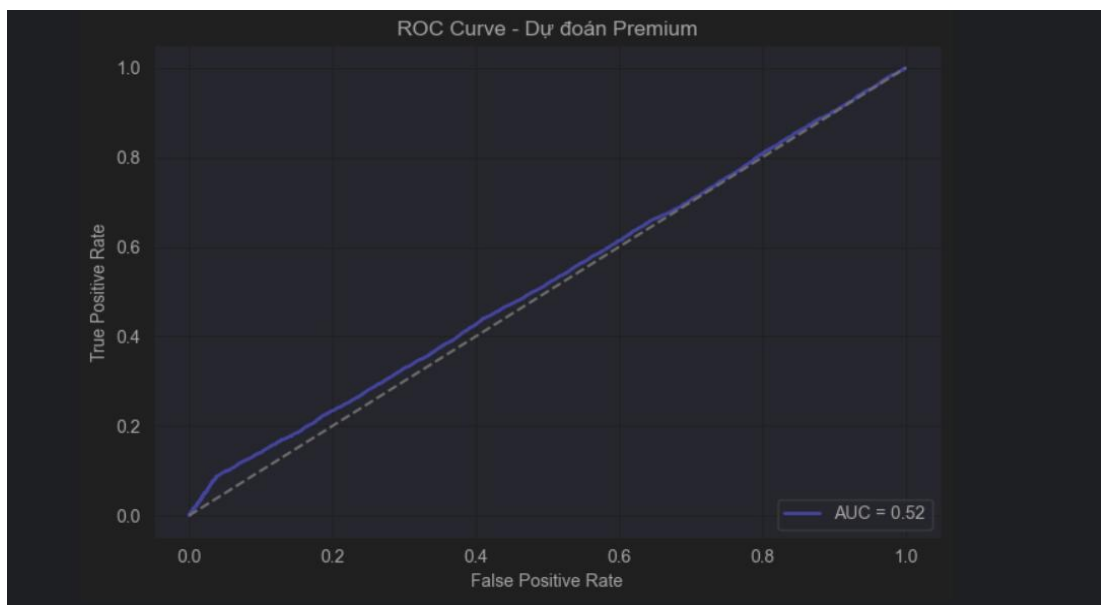
```

1 # Nếu cần, hãy import pandas và seaborn
2 import pandas as pd
3
4 # Xuất kết quả về Pandas DataFrame (chỉ lấy 2 đặc trưng và cột 'prediction' - cùm)
5 pdf = predictions.select("avg_minute", "avg_rating", "prediction").toPandas()
6
7
8 def get_prob(v): return float(v[1]) # lấy xác suất dự đoán Premium
9
10
11 pdf_lr = predictions_lr.select("label", "probability").toPandas()
12 pdf_lr["prob_premium"] = pdf_lr["probability"].apply(get_prob)
13 import matplotlib.pyplot as plt
14
15 from sklearn.metrics import roc_curve, auc
16
17 fpr, tpr, thresholds = roc_curve(pdf_lr['label'], pdf_lr['prob_premium'])
18 roc_auc = auc(fpr, tpr)
19
20 plt.figure(figsize=(7, 5))
21 plt.plot(fpr, tpr, color='blue', lw=2, label='AUC = %0.2f' % roc_auc)
22 plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
23 plt.xlabel('False Positive Rate')
24 plt.ylabel('True Positive Rate')
25 plt.title('ROC Curve - Dự đoán Premium')
26 plt.legend(loc="lower right")
27 plt.grid(True)
28 plt.tight_layout()
29 plt.show()

```

Hình 61: Thực hiện dự đoán bằng cách sử dụng thư viện sklearn

Kết quả & diễn giải:



Hình 62: Đường cong ROC thể hiện dự đoán về người dùng Premium

Đường cong ROC cho thấy chất lượng mô hình phân loại: đường càng nằm phía trên góc trái thì mô hình càng tốt. Giá trị AUC đạt khoảng 0.52–0.53 (đúng như phần kết quả trước), thể hiện mô hình có khả năng nhận diện khách hàng Premium cao hơn ngẫu nhiên, dù còn

hạn chế do mất cân bằng dữ liệu. Biểu đồ này giúp hội đồng, người đọc đánh giá trực quan mức độ hiệu quả của mô hình dự báo khi áp dụng vào thực tiễn.

4.4. Tổng kết chương

Chương 4 đã thực hiện đầy đủ quá trình phân tích dữ liệu hành vi người dùng ứng dụng AI di động trên nền tảng Big Data với Apache Spark, từ phân tích mô tả đến triển khai các mô hình học máy. Qua các truy vấn Spark SQL, nhóm đã phát hiện ra nhiều xu hướng quan trọng trong hành vi sử dụng và mức độ gắn bó của người dùng với ứng dụng AI, đồng thời nhận diện được các yếu tố ảnh hưởng đến khả năng nâng cấp lên tài khoản Premium.

Việc ứng dụng thuật toán KMeans đã xác định được ba phân khúc khách hàng chính với đặc điểm hành vi rõ rệt, tạo tiền đề cho chiến lược cá nhân hóa dịch vụ và chăm sóc khách hàng. Các mô hình dự đoán nâng cấp Premium (Logistic Regression, Random Forest) dù còn hạn chế do dữ liệu mất cân bằng, nhưng đã minh chứng cho tính khả thi của việc triển khai học máy trên dữ liệu lớn thực tế.

Đặc biệt, các biểu đồ trực quan hóa như scatter plot và ROC Curve đã giúp làm rõ hơn ý nghĩa các kết quả phân tích, hỗ trợ truyền tải thông tin một cách sinh động, trực quan đến người đọc. Đây là minh chứng cho tầm quan trọng của kết hợp giữa phân tích dữ liệu, mô hình hóa và trực quan hóa trong khai phá giá trị từ Big Data.

Những phát hiện từ chương 4 không chỉ cung cấp hiểu biết sâu sắc về hành vi người dùng ứng dụng AI di động mà còn mở ra các hướng nghiên cứu và phát triển sản phẩm mới, góp phần nâng cao hiệu quả kinh doanh và trải nghiệm khách hàng. Kết quả đạt được là nền tảng vững chắc để nhóm tiếp tục đề xuất các giải pháp nâng cao và mở rộng ứng dụng trong các chương tiếp theo.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

- Nhóm đã hoàn thiện toàn bộ pipeline xử lý dữ liệu lớn: từ sinh dữ liệu, nạp vào HDFS, xử lý và phân tích bằng Spark SQL, cho đến xây dựng mô hình học máy (ML).
- Phân tích dữ liệu giúp xác định và mô tả thành công ba phân khúc người dùng chính dựa trên hành vi sử dụng.
- Xây dựng được mô hình Logistic Regression giúp dự đoán khả năng nâng cấp lên Premium hoặc khả năng rời bỏ (churn) của người dùng.
- Quy trình thực hành có tính hệ thống, có thể mở rộng hoặc tái sử dụng trên các bộ dữ liệu thực tế với quy mô lớn hơn.

5.2. Hạn chế

- Dữ liệu sử dụng trong đồ án là dữ liệu giả lập, chưa phản ánh đầy đủ sự phức tạp và đa dạng của hành vi người dùng thực tế.
- Bộ biến đầu vào còn đơn giản, chủ yếu là các đặc trưng định lượng, chưa khai thác yếu tố định tính như phản hồi, cảm xúc, hoặc tương tác sâu của người dùng.
- Chưa mở rộng phân tích với dữ liệu real-time hoặc streaming, và mới chỉ thử nghiệm mô hình cơ bản (Logistic Regression), chưa áp dụng các mô hình nâng cao như Random Forest, XGBoost.

5.3. Hướng phát triển

- Tiếp tục triển khai pipeline trên dữ liệu thực tế, bổ sung thêm nhiều thuộc tính (biến nhân khẩu học, hành vi, tương tác...) để phân tích đa chiều và tăng sức mạnh dự báo.
- Nghiên cứu, thử nghiệm và so sánh thêm các mô hình ML nâng cao (Random Forest, Gradient Boosting, XGBoost...), đồng thời tích hợp khả năng xử lý dữ liệu thời gian thực với Spark Streaming.
- Đề xuất xây dựng hệ thống dashboard trực quan hóa hành vi người dùng, hỗ trợ quản lý và tối ưu hóa sản phẩm trong môi trường thực tế.

TÀI LIỆU THAM KHẢO

- Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, 19(2), 171-209.
- Han, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.
- Narayanan, A. (2018). *Synthetic Data Generator for User Behavioral Analytics Systems* (Master's thesis, Carleton University).
- Laney, D. (2001). 3D Data Management: Controlling Data Volume, Velocity and Variety. *META Group*.
- Inoxoft (2023). MapReduce vs Spark: Key Differences.