

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
VIỆN TRÍ TUỆ NHÂN TẠO

-----***-----



BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

ĐỀ TÀI

Classifying in KNIME to identify big spenders in Catch the Pink Flamingo

Nhóm sinh viên thực hiện:

1. Nguyễn Thị Ngọc Lan - 23020390
2. Tạ Giang Thùy Loan - 23020397
3. Hoàng Ngọc Điệp - 23020357

Giảng viên hướng dẫn: TS. Trần Hồng Việt

ThS.Ngô Minh Hương

HÀ NỘI, 5/2025

MỞ ĐẦU

Công nghệ Big data đã đạt đến đỉnh cao trong việc thực hiện các chức năng của nó. Trong tháng 8/2015 Big data đã vượt ra khỏi bảng xếp hạng những công nghệ mới nổi Cycle Hype của Gartner và tạo ra một tiếng vang lớn cho *xu hướng công nghệ* của thế giới. Big data chứa trong mình rất nhiều thông tin quý giá mà nếu mà trích xuất thành công, nó sẽ giúp rất nhiều trong nhiều lĩnh vực như y tế, giao thông, giáo dục, ...

Chính vì thế những framework giúp việc xử lý BIGDATA cũng đang ngày càng được xử lý và phát triển mạnh. Một trong những công nghệ cốt lõi cho việc lưu trữ và truy cập số lượng lớn dữ liệu là Hadoop - một framework giúp lưu trữ và xử lý Big data áp dụng MapReduce.

Từ đó, chúng em đã chọn đề tài: **"Classifying in KNIME to identify big spenders in Catch the Pink Flamingo"** để làm báo kết thúc môn học của mình.

Báo cáo gồm 3 chương:

Chương 1: Tổng quan kiến thức .

Chương 2: Classifying in KNIME to identify big spenders in Catch the Pink Flamingo.

Chương 3: Kết luận và hướng phát triển.

MỤC LỤC	
MỞ ĐẦU	2
CHƯƠNG 1: TỔNG QUAN VỀ DỮ LIỆU LỚN	4
1.1 Định nghĩa.	4
1.2 Đặc trưng cơ bản của dữ liệu lớn.	4
1.3 Tổng quan apache Spark.	4
1.5 Tổng quan về Hadoop.	5
CHƯƠNG 2: Classifying in KNIME to identify big spenders in Catch the Pink Flamingo	6
2.1 Bài toán.	6
2.2 Mô tả dữ liệu.	7
2.3 Gắn nhãn và trực quan hóa dữ liệu.	7
2.4 Phân tích dữ liệu.	7
2.5 Tiền xử lý dữ liệu.	7
2.6 Huấn luyện mô hình.	7
2.7 Đánh giá mô hình.	7
CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	11
3.1 Kết luận.	11
3.2 Hướng phát triển.	11
NHIỆM VỤ CỦA CÁC THÀNH VIÊN	18

CHƯƠNG 1: TỔNG QUAN VỀ DỮ LIỆU LỚN

1.1 Định nghĩa.

Theo wikipedia: Dữ liệu lớn là một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này.

Theo Gartner : Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được phải đòi hỏi phải có hình thức mới để đưa ra quyết định khám phá và tối ưu hóa quy trình.

Dữ liệu đến từ rất nhiều nguồn khác nhau:



Hình 1. Minh họa nguồn gốc của dữ liệu.

Một số lợi ích có thể mang lại như: Cắt giảm chi phí, tiết kiệm thời gian và giúp tối ưu hóa sản phẩm, hỗ trợ con người đưa ra những quyết định đúng và hợp lý hơn.

1.2 Đặc trưng cơ bản của dữ liệu lớn.

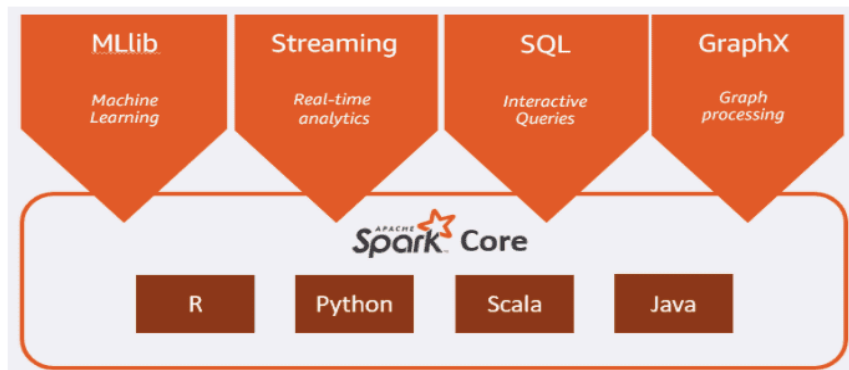
- (1) **Khối lượng lớn (Volume)**: Khối lượng dữ liệu rất lớn và đang ngày càng tăng lên, tính đến 2014 thì có thể trong khoảng vài trăm terabyte.
- (2) **Tốc độ (Velocity)**: Khối lượng dữ liệu gia tăng rất nhanh.
- (3) **Đa dạng (Variety)**: Ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc(tài liệu, blog, hình ảnh,...)
- (4) **Độ tin cậy/chính xác (Veracity)**: Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là tính chất quan trọng của bigdata.
- (5) **Giá trị (Value)**: Giá trị thông tin mang lại.

1.3 Tổng quan về Spark.



(Biểu tượng của Spark)

- + Apache Spark: là một open source cluster computing framework phát triển vào năm 2009 bởi AMPLab tại đại học California. Spark được Apache Software Foundation vào năm 2013.
- + 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.
- + Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL,



Apache Spark framework (source: AWS)

MLlib và GraphX, trong đó:

- (1) **Spark Core:** Là nền tảng cốt lõi của Spark, chịu trách nhiệm thực hiện tính toán, quản lý bộ nhớ và tương tác với hệ thống lưu trữ bên ngoài. Tất cả các thành phần khác đều chạy dựa trên Spark Core.
- (2) **Spark SQL:** Cung cấp abstraction dữ liệu mới là SchemaRDD (**DataFrame**), hỗ trợ dữ liệu có cấu trúc và bán cấu trúc. Cho phép truy vấn bằng DSL (Scala, Java, Python) và SQL.
- (3) **Spark Streaming:** Xử lý dữ liệu thời gian thực bằng cách chia stream thành các

mini-batches, rồi áp dụng các phép biến đổi RDD. Dễ tích hợp vào kiến trúc Lambda nhưng có độ trễ tương ứng với kích thước mini-batch.

(4) **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.

(5) **GraphX**: Nền tảng xử lý đồ thị, cung cấp API cho các phép tính trên đồ thị dựa trên mô hình Pregel.

1.4 Tổng quan về KNIME.



KNIME (Konstanz Information Miner) là một nền tảng mã nguồn mở được phát triển từ năm 2004 tại Đại học Konstanz, Đức, nhằm hỗ trợ phân tích dữ liệu, khai phá dữ liệu, và học máy thông qua giao diện kéo-thả trực quan.

KNIME cho phép người dùng xây dựng các quy trình xử lý dữ liệu (workflow) mà không cần viết mã, tuy nhiên cũng hỗ trợ tích hợp mã Python, R, Java nếu cần. KNIME phù hợp với cả người mới bắt đầu lẫn các chuyên gia phân tích dữ liệu.

Các đặc điểm chính:

(1) **Giao diện trực quan:** KNIME sử dụng hệ thống các node kéo-thả, mỗi node đại diện cho một bước xử lý dữ liệu như nhập, lọc, xử lý, mô hình hóa, trực quan hóa, v.v.

(2) **Mã nguồn mở, mở rộng tốt:** Người dùng có thể mở rộng KNIME thông qua các extension (tiện ích mở rộng) như KNIME Text Processing, KNIME Deep Learning, KNIME Big Data, v.v.

(3) **Hỗ trợ mạnh mẽ Machine Learning và Data Mining:** KNIME tích hợp các thuật toán học máy phổ biến (classification, regression, clustering, v.v.) cũng như các công cụ khai phá dữ liệu.

- (4) **Khả năng tích hợp với công cụ bên ngoài:** KNIME hỗ trợ kết nối và tích hợp với nhiều hệ thống như Python, R, Weka, TensorFlow, H2O, Apache Spark, SQL Databases, Hadoop (qua KNIME Big Data Extension), v.v.
- (5) **Xử lý dữ liệu thời gian thực và dữ liệu lớn:** KNIME Server và KNIME Hub giúp chia sẻ workflow, tự động hóa quy trình và tích hợp với môi trường doanh nghiệp.

Nền tảng	Tính năng nổi bật	Viết mã	Xử lý song song	Mục tiêu chính
Apache Spark	Tính toán phân tán trên RAM, xử lý thời gian thực.	Có (Scala, Java, Python)	Có	Tính toán nhanh trên dữ liệu lớn.
KNIME	Giao diện kéo-thả, học máy, tích hợp đa ngôn ngữ.	Không cần (nhưng hỗ trợ nếu có)	Không mặc định (có thể kết hợp Spark)	Phân tích và xử lý dữ liệu, học máy, ETL.

CHƯƠNG 2: Classifying in KNIME to identify big spenders in Catch the Pink Flamingo

2.1 Bài toán:

- ❖ Mô tả trò chơi “Catch the Pink Flamingo”
 - Đây là một trò chơi di động nhiều người chơi của công ty giả lập Eglence Inc. Mục tiêu là bắt càng nhiều hồng hạc càng tốt thông qua các nhiệm vụ theo thời gian thực.
 - Người chơi phải hoàn thành toàn bộ bản đồ (mỗi ô có ít nhất 1 điểm) để lên cấp.
 - Cấp độ đầu (Level 1): Người chơi chơi cá nhân, bản đồ 8x8, nhiệm vụ đơn giản.
 - Cấp độ tiếp theo: Bản đồ mở rộng, nhiệm vụ phức tạp hơn, người chơi chơi theo nhóm (1–30 người).
 - Điểm âm: Nếu người chơi chọn sai loại hồng hạc theo yêu cầu nhiệm vụ.
 - Người chơi có thể giao tiếp qua chat nhóm và mạng xã hội. Trò chơi không có hồi kết, luôn có cấp độ mới khó hơn để giữ chân người chơi.
 - Một số đặc điểm hệ thống
 - Xếp hạng người chơi: Theo tốc độ và độ chính xác. Có thể xem thông qua ứng dụng hoặc web.
 - Phân loại người chơi: “ngôi sao mới”, “cựu binh”, “huấn luyện viên”, “thích giao tiếp”, “hồng hạc nổi bật”.
 - Xếp hạng đội: Dựa trên tổng điểm, có thể tuyên hoặc loại thành viên bằng biểu quyết.
 - Mua hàng trong game: Kính viễn vọng, thẻ đổi điểm, khối băng, hồng hạc đặc biệt,...
 - Quản lý đội: Đội không có thành viên sẽ bị xóa khỏi hệ thống.
 - + Một người được coi là **big spender** nếu người đó chi tiêu tổng số tiền lớn hơn hoặc bằng 1 ngưỡng nhất định (Rule Engine hoặc KMeans). Khi huấn luyện mô hình, dữ liệu được dùng chỉ là 3 session đầu tiên của mỗi người dùng.
 - + Big spender ít nhưng tạo ra nhiều doanh thu cho chủ thầu vậy nên phải để mô hình cố gắng phân tích được chính xác nhiều big spender nhất có thể mặc dù dữ liệu có bị mất cân bằng để đưa ra được chiến lược tiếp thị phù hợp.
 - + Bài toán xác định big spenders trong trò Catch the Pink Flamingo cần trải qua các công đoạn chính:
 - (1) Chuẩn bị dữ liệu và gán nhãn.
 - (2) Phân tích dữ liệu và đánh giá.
 - (3) Tiền xử lý dữ liệu và chọn đặc trưng.
 - (4) Huấn luyện mô hình và đánh giá trên tập test.
- Nhãn: **big spender**, **low spender** - người dùng không chi tiêu nhiều

2.2 Mô tả dữ liệu:

Giới thiệu:

Dữ liệu được sử dụng trong dự án cuối khoá của chuyên ngành Big Data trên Coursera (Đại học UC San Diego) nhằm phát triển các đề xuất tăng doanh thu cho trò chơi giả lập “Catch the Pink Flamingo”.

Dữ liệu gồm 9 tệp ghi lại hành vi người dùng (mua hàng, quảng cáo, tương tác trong game) và 6 tệp dữ liệu trò chuyện. Tất cả được tạo bởi đội ngũ khoa học dữ liệu nhằm mô phỏng hoạt động thực tế trong game.

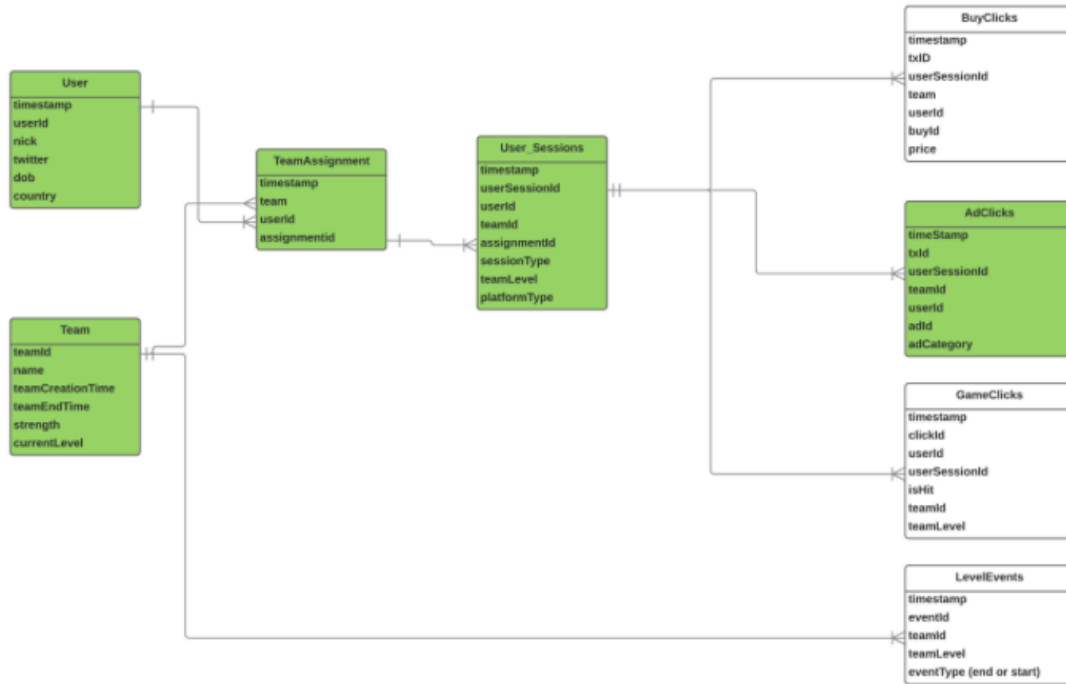
Phân tích sử dụng các kỹ thuật: phân loại (cây quyết định), phân cụm (k-means), phân tích đồ thị (người dùng/team hoạt động tích cực nhất). Công cụ gồm: Spark, Python, Splunk, KNIME, Neo4j.

Dữ liệu hành vi người dùng

Tên tệp	Nội dung chính	Một số trường dữ liệu
ad-clicks.csv	Nhấp vào quảng cáo	timestamp, userid, adId, adCategory
buy-clicks.csv	Mua hàng trong game	timestamp, userid, buyId, price
users.csv	Thông tin người chơi	userid, nickname, ngày sinh, quốc gia
team.csv	Thông tin đội chơi	teamId, thời gian tạo/kết thúc, cấp độ
team-assignments.csv	Thành viên gia nhập đội	timestamp, userid, teamId
level-events.csv	Bắt đầu/kết thúc cấp độ	timestamp, teamId, cấp độ, loại sự kiện
user-session.csv	Thời gian chơi của người dùng	timestamp, sessionId, userId, teamLevel
game-clicks.csv	Nhấp chuột trong game	timestamp, userId, isHit, teamLevel
combined_data.csv	Dữ liệu tổng hợp user-session.csv, buy-clicks.csv, and game-clicks.csv.	userId, sessionId, lượt click, lượt mua, giá trung bình

Để giải quyết bài toán chúng em xử lý và phân tích dữ liệu từ file combined_data.csv gồm:

- (1) userid: User ID.
- (2) userSessionid: User session ID.
- (3) team_level: User's team level.
- (4) platformType: Platform used by user.
- (5) count_gameclicks: Total number of game clicks for user session.
- (6) count_hits: Total number of game hits for user session.
- (7) count_buyid: Total number of purchases for user session.
- (8) avg_price: Average purchase price for user session.



2.3 Gán nhãn và trực quan hoá bằng KNIME

Để gán nhãn cho người chơi là "big spender" hay "low spender", chúng tôi sử dụng công cụ KNIME với hai hướng tiếp cận: Rule Engine và KMeans.

2.2.1 Gán nhãn bằng Rule Engine.

Dựa trên phân tích sơ bộ, chúng tôi định nghĩa ngưỡng chỉ tiêu để phân loại. Ví dụ, có thể lấy 5% số người chi nhiều tiền nhất làm "big spender". Trong trường hợp cụ thể này, ngưỡng chỉ tiêu được xác định là 70 đô la dựa trên phân vị (ví dụ: `user_stats.approxQuantile("total_spent", [0.95], 0.01)[0]`).

Người chơi có `total_spent >= 70` được gán nhãn "big spender", ngược lại là "low spender".

(Hình 1 trong Nội dung B: Tính ngưỡng chỉ tiêu theo Rule Engine)

```

`quantile` = user_stats.approxQuantile("total_spent", [0.95], 0.01)[0]
`quantile` -> `70.0`
  
```

2.2.2 Gán nhãn bằng KMeans.

Một hướng tiếp cận khác là sử dụng thuật toán phân cụm KMeans với số cụm `k = 2` để tự động phân

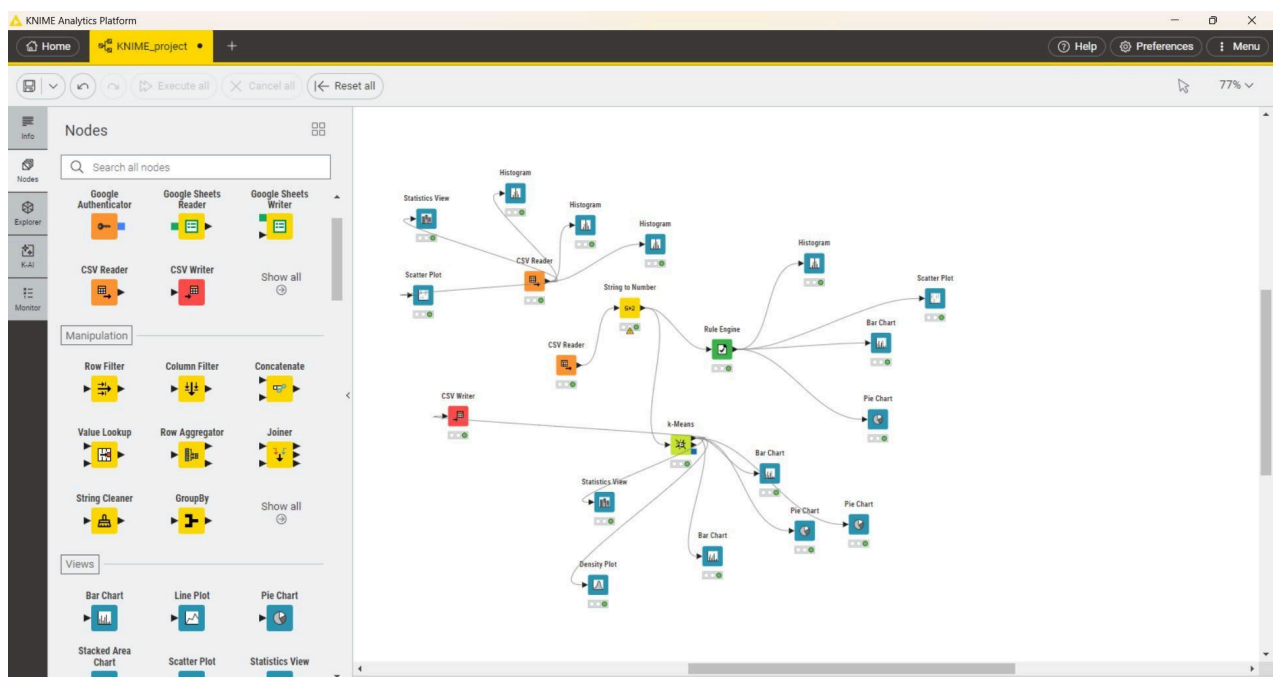
chia người chơi thành hai nhóm dựa trên các đặc trưng chi tiêu của họ. Kết quả cho thấy KMeans có thể nhận diện được một tập "big spender" hơi khác so với Rule Engine, có khả năng bao quát hơn.

```
[ ] quantile = user_stats.approxQuantile("total_spent", [0.95], 0.01)[0]
```

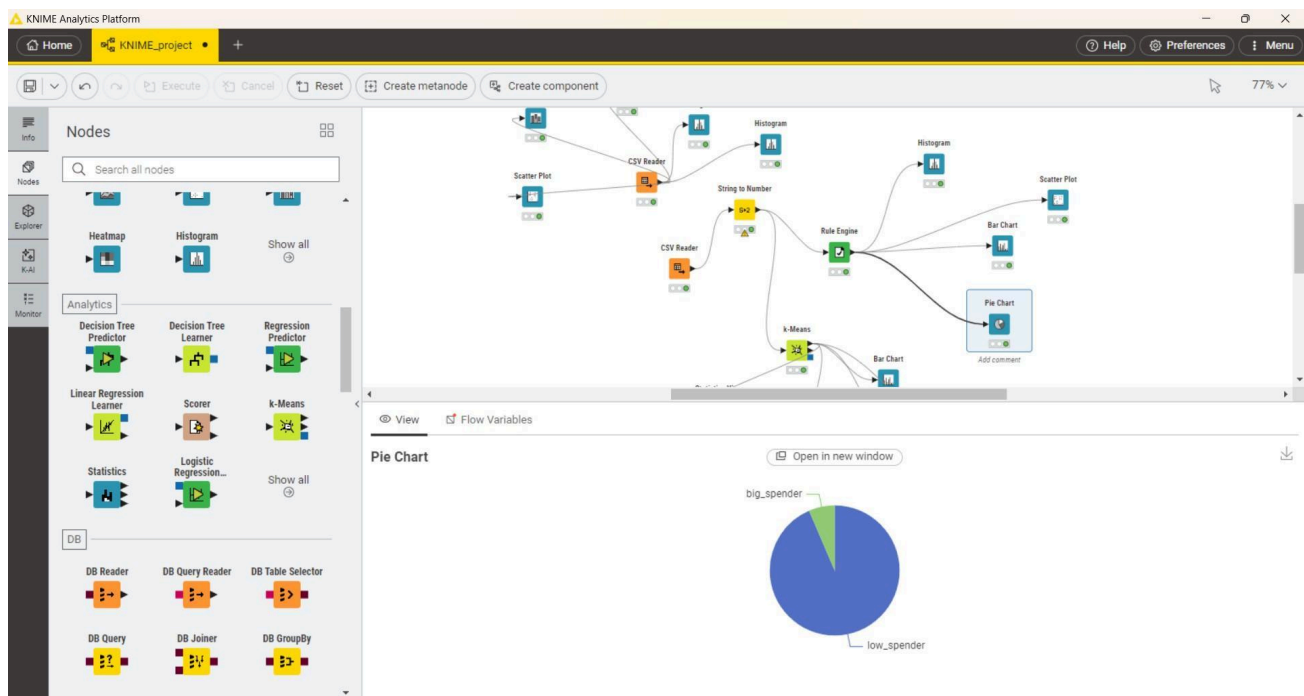
```
[ ] quantile
```

↔ 70.0

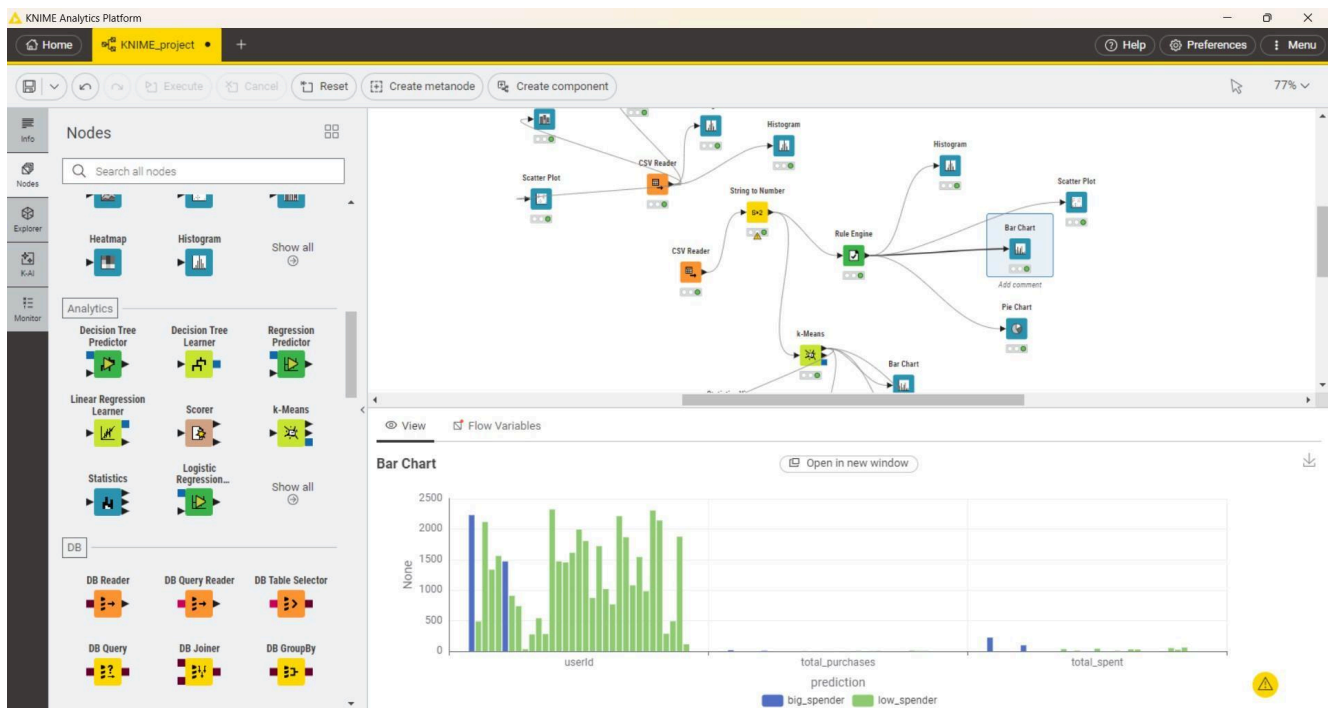
Hình 1: Tính ngưỡng chi tiêu để phân loại phân loại big spender và low spender theo Rule Engine



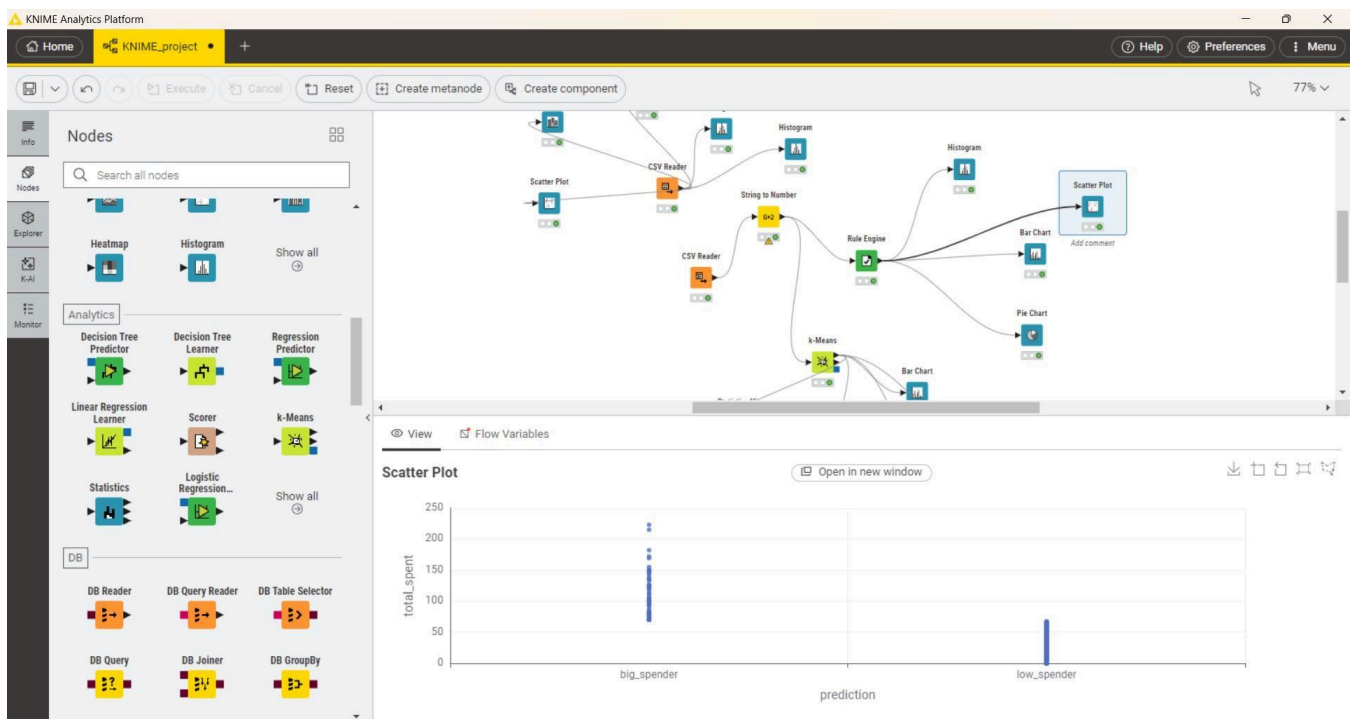
Hình 2: Giao diện làm việc với KNIME



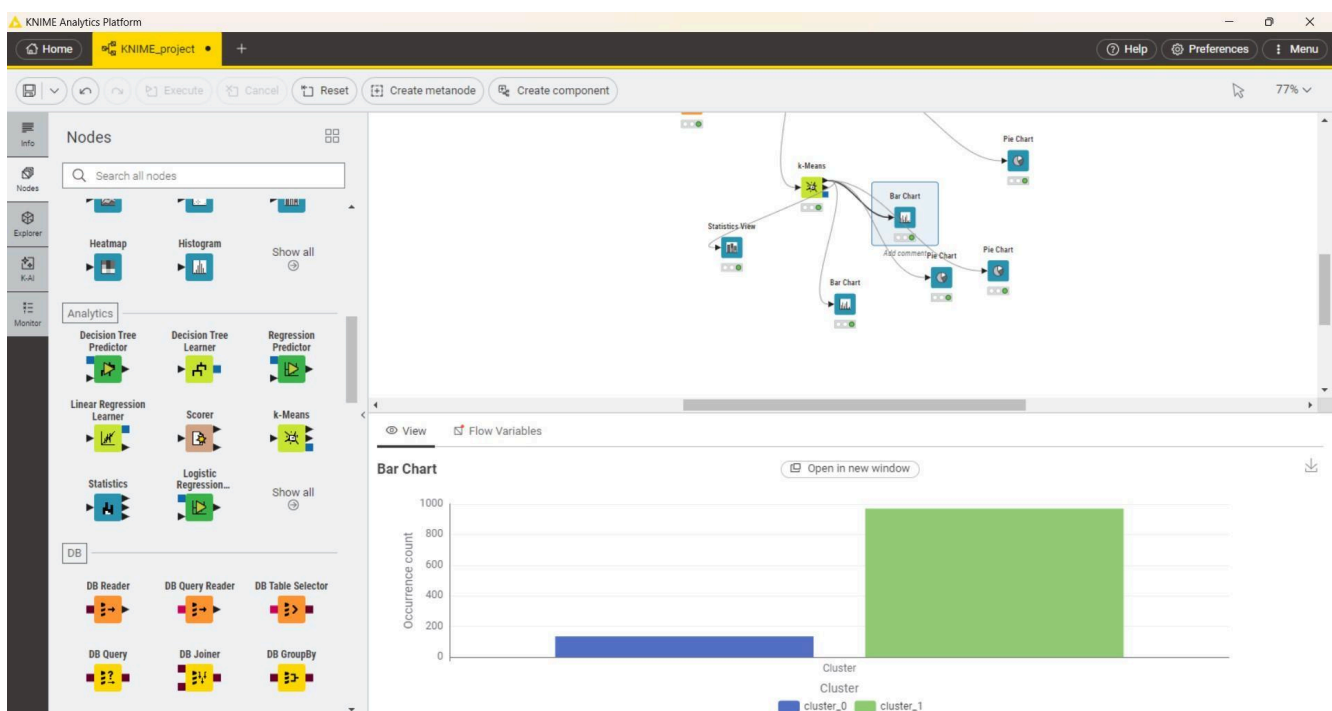
Hình 3: Biểu đồ tròn nếu phân big spender và low spender theo Rule Engine



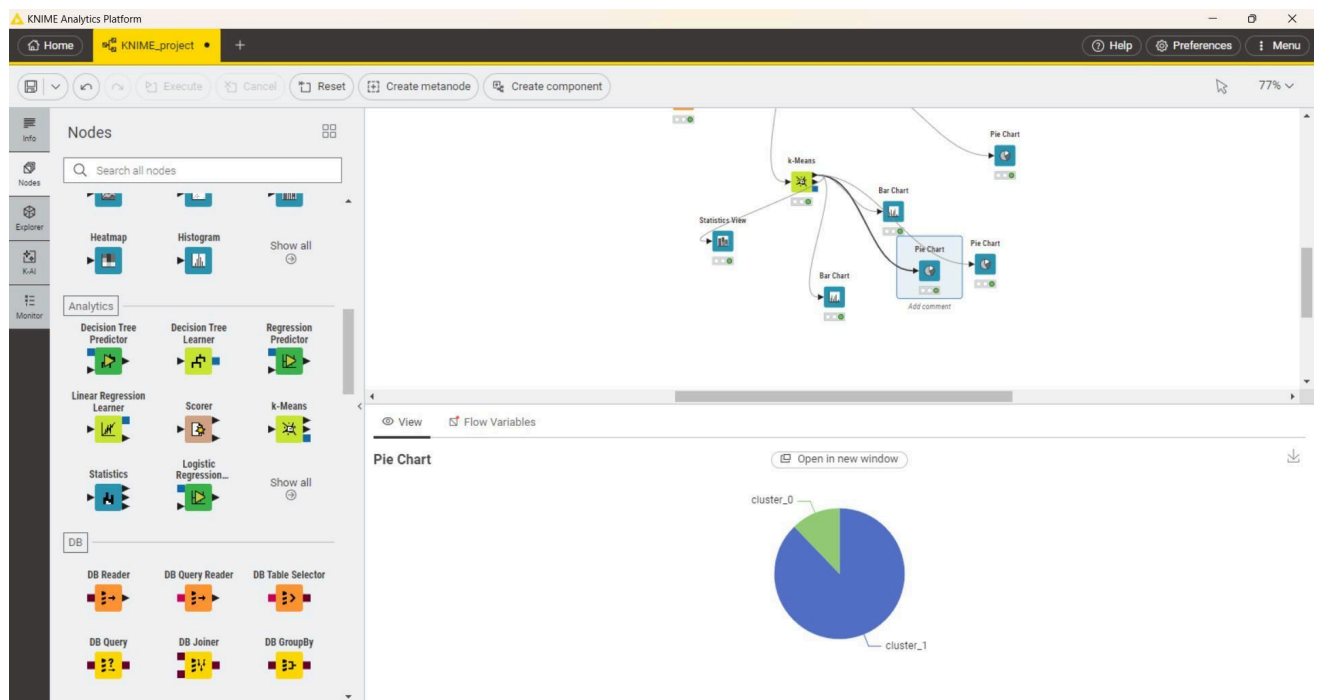
Hình 4: Biểu đồ cột nếu phân big spender và low spender theo Rule Engine



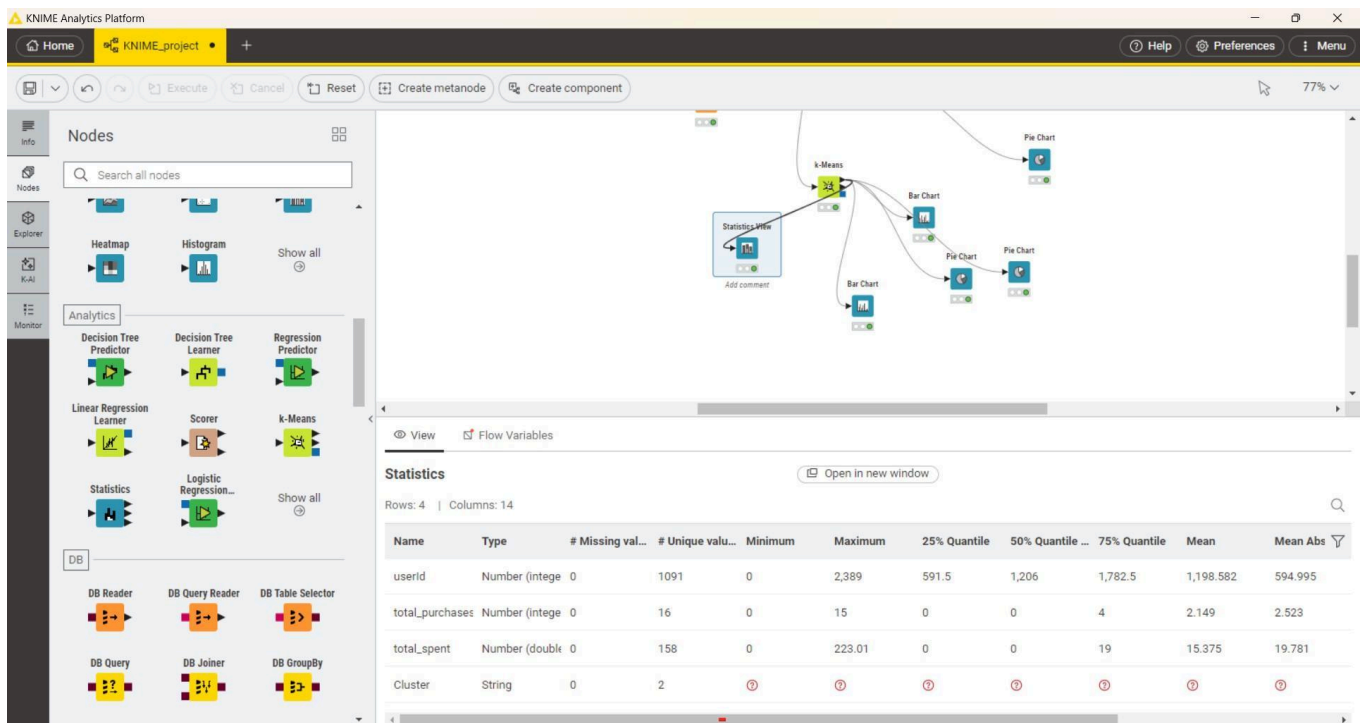
Hình 5: Biểu đồ phân tán nếu phân big spender và low spender theo Rule Engine



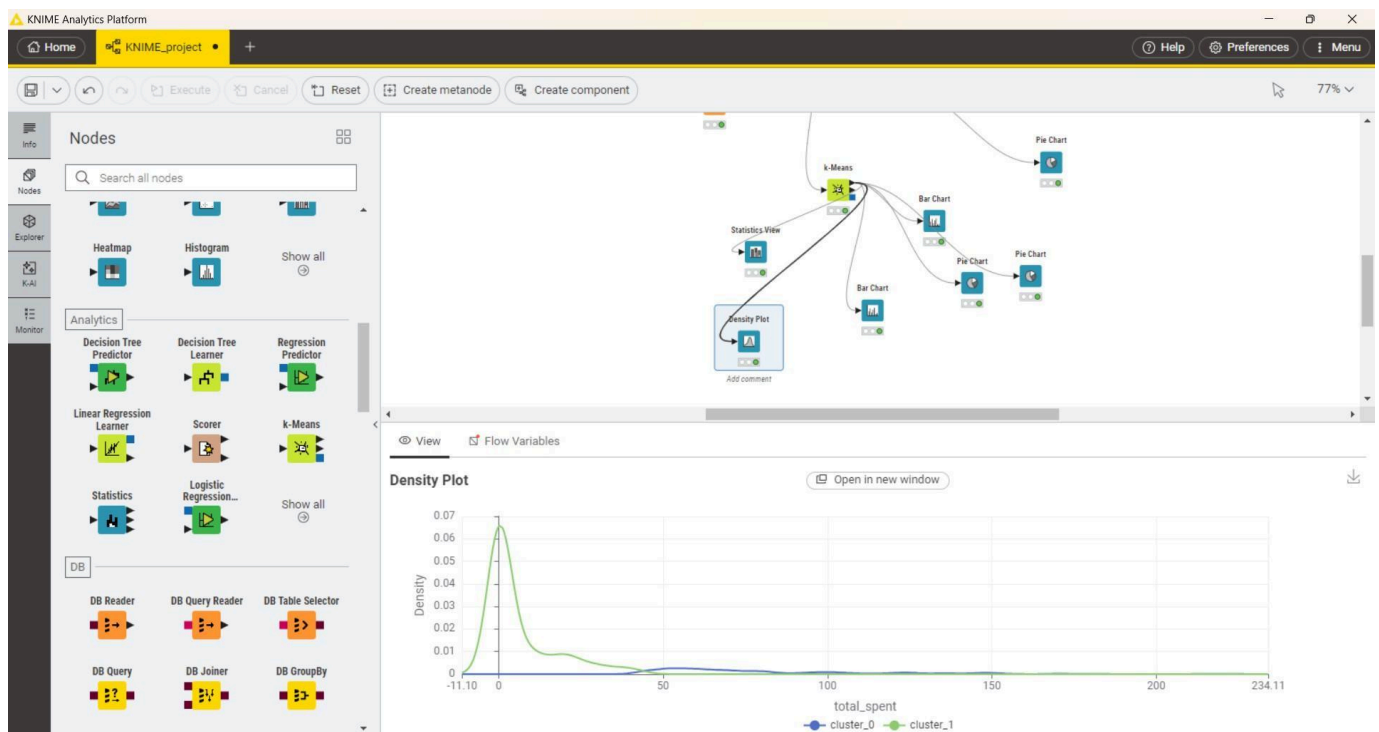
Hình 6: Biểu đồ cột nếu phân big spender (cluster 0) và low spender (cluster 1) theo KMeans



Hình 7: Biểu đồ tròn nếu phân big spender (cluster 0) và low spender (cluster 1) theo KMeans



Hình 8: Các chỉ số thống kê nếu phân big spender (cluster 0) và low spender (cluster 1) theo KMeans



Hình 9: Biểu đồ đường theo nhãn big spender (cluster 0) và low spender (cluster 1) theo KMeans

The figure shows the KNIME Analytics Platform interface. On the left, the 'Nodes' panel is visible, categorized into 'Manipulation' and 'Views'. The main workspace displays a workflow starting with a 'CSV Reader' node, followed by a 'Statistics View' node, then a 'k-Means' node. The 'k-Means' node is connected to several visualization nodes: 'Bar Chart', 'Pie Chart', and 'Density Plot'. The bottom panel shows a table with 12 rows of data, including columns for RowID, userid, platformType, total_purchases, total_spent, and Cluster. The table is sorted by 'total_spent' in descending order.

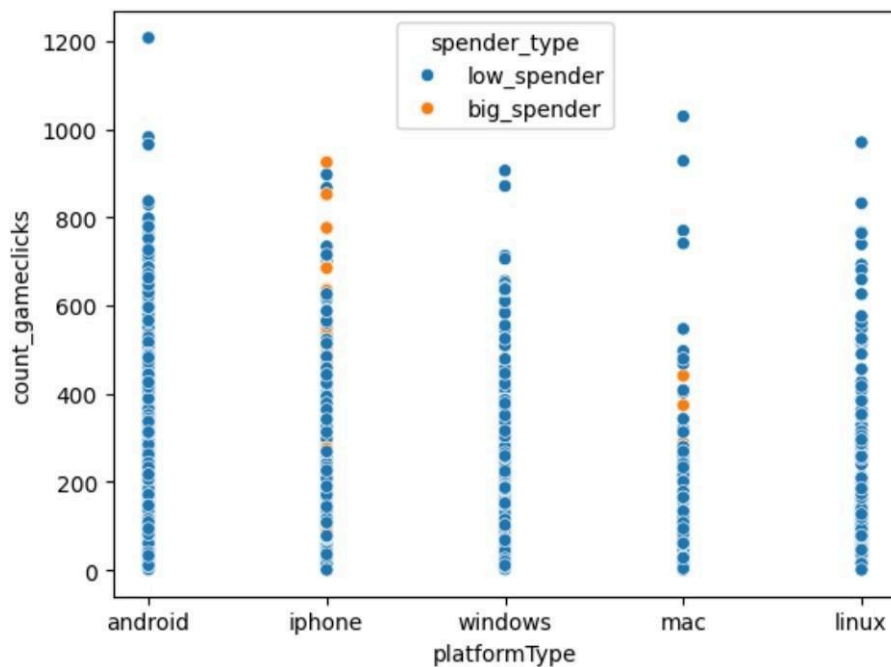
#	RowID	userid	platformType	total_purchases	total_spent	Cluster
1	Row0	2229	iphone	15	223.01	cluster_0
2	Row1	487	iphone	0	0	cluster_1
3	Row2	2116	android	0	0	cluster_1
4	Row3	1335	windows	0	0	cluster_1
5	Row4	1560	android	0	0	cluster_1
6	Row5	1471	iphone	8	96	cluster_0
7	Row6	908	mac	0	0	cluster_1
8	Row7	738	iphone	0	0	cluster_1
9	Row8	32	android	0	0	cluster_1
10	Row9	275	android	0	0	cluster_1
11	Row10	540	android	0	0	cluster_1
12	Row11	281	android	5	34	cluster_1

Hình 10: Bảng dữ liệu cuối cùng phân nhãn theo KMeans

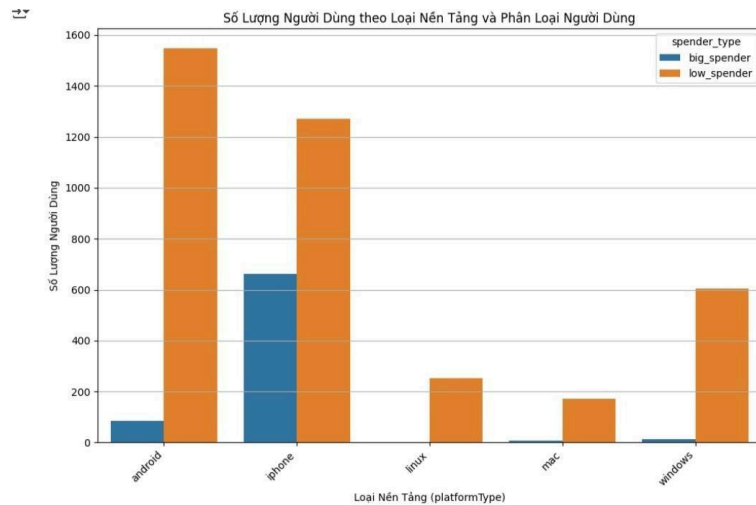
2.4 Phân Tích Dữ Liệu

Sau khi hoàn tất các bước tiền xử lý dữ liệu, bao gồm việc xử lý các giá trị NULL trong cột `count_buyId` và `avg_price` (bằng cách thay thế bằng 0 và chuyển đổi kiểu dữ liệu phù hợp), tạo cột `made_purchase` để xác định các phiên có giao dịch, và tạo cột mục tiêu `spender_type` (big spender/low spender) dựa trên `total_spent` của mỗi người dùng, em đã tiến hành phân tích sâu hơn để hiểu rõ hơn về các đặc điểm của dữ liệu và mối quan hệ giữa các biến.

Nhận xét: Biểu đồ phân tán Hình 11: Biểu đồ phân tán số lượt nhấp chuột trong game theo loại nền tảng và phân loại người chi tiêu cho thấy sự phân bố của `count_gameclicks` đối với hai nhóm big spender và low spender trên các nền tảng khác nhau. Nhìn chung, không có sự khác biệt hoàn toàn về số lượt nhấp chuột giữa hai nhóm trên đối với các nền tảng. Người dùng Android và iPhone, thuộc cả hai nhóm chi tiêu, đều có những phiên với số lượt nhấp chuột rất cao. Kết Luận: `count_gameclicks` có thể không phải là yếu tố quyết định duy nhất để phân loại người dùng.



Hình 11: Biểu đồ phân tán: Số Lượt Nhấp Chuột Trong Game theo Loại Nền Tảng và Phân Loại Người Chi Tiêu.

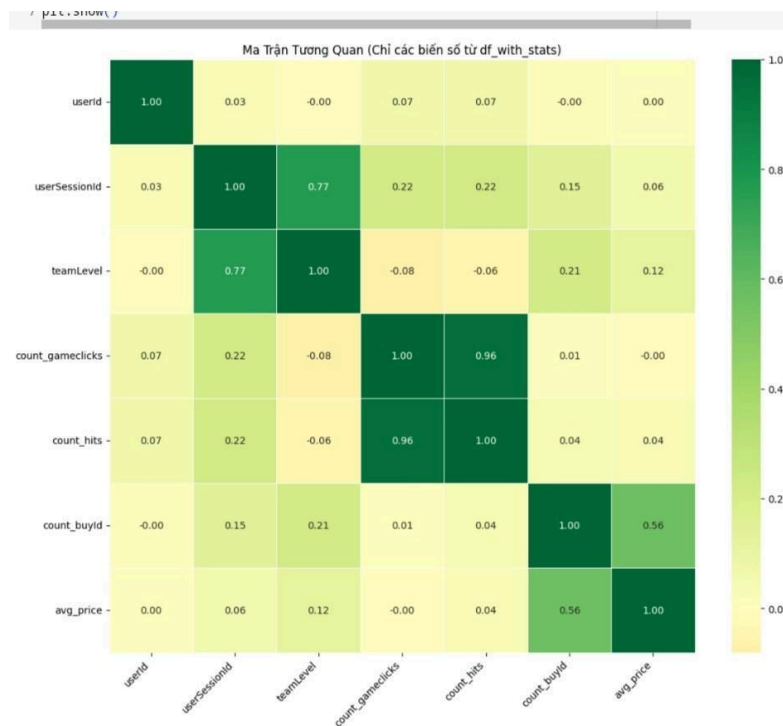


Hình 12: Biểu đồ cột: Số Lượng Người Dùng theo Loại Nền Tảng và Phân Loại Người Chi Tiêu.

Nhận xét: Biểu đồ cột (Hình 12) Biểu đồ cột Số Lượng Người Dùng theo Loại Nền Tảng và Phân Loại Người Chi Tiêu trực quan hóa số lượng phiên người dùng được phân loại là low spenders và big spender trên từng nền tảng.

- Đa số phiên được xác định của big spender diễn ra trên nền tảng iPhone và Android.
- low spenders đa dạng nền tảng: Nhóm low spenders xuất hiện trên tất cả các nền tảng, với số lượng lớn nhất trên iPhone và Android.

Kết Luận: Nền tảng (platformType) là một đặc trưng cực kỳ quan trọng. Người dùng sử dụng iPhone và Android có tỉ lệ trở thành big spender cao hơn các nền tảng (platform) khác.

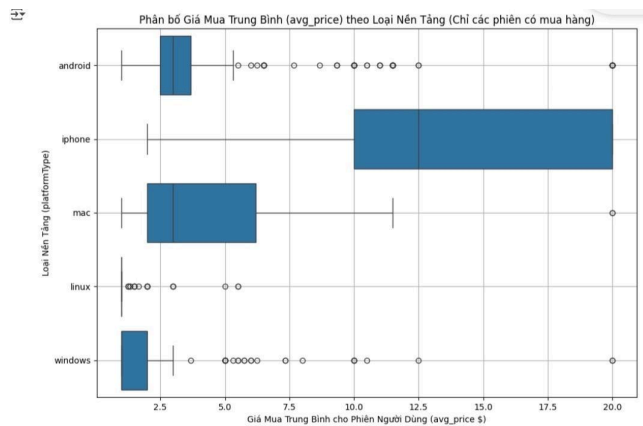


Hình 13: Heatmap: Ma Trận Tương Quan giữa các Chỉ Số.

Nhận xét: Ma trận tương quan (Hình 13) giữa các biến số ban đầu cho thấy:

- `count_gameclicks` có mối tương quan dương rất mạnh (0.96) với `count_hits`.
- `userId` có tương quan dương khá cao với `teamLevel` (0.77).
- `avg_price` có tương quan dương yếu với `count_buyId` (0.56).
- Quan trọng nhất đối với bài toán, `avg_price` gần như không có tương quan tuyến tính đáng kể với các chỉ số hoạt động khác trong game.

Phân tích phân bố giá mua trung bình (`avg_price`) theo nền tảng (`platformType`)



Hình 14: Biểu đồ hộp: Phân bố Giá Mua Trung Bình theo Loại Nền Tảng (Chỉ các phiên có mua hàng).

Nhận xét: Khi chỉ xem xét các phiên có giao dịch mua hàng (`avg_price > 0`), biểu đồ hộp (Hình 14) của `avg_price` theo từng nền tảng cho thấy:

- Người dùng iPhone có xu hướng có khoảng tứ phân vị và các giá trị ngoại lệ ở mức giá mua trung bình cao hơn rõ rệt so với các nền tảng khác.

- Các nền tảng khác: Android và Windows có phân bố giá rộng hơn ở các mức thấp và trung bình. Mac và Linux có ít dữ liệu hơn nhưng cũng tập trung ở các mức giá thấp hơn iPhone.

Kết luận sơ bộ từ Phân tích dữ liệu:

Qua các phân tích trên, có thể thấy platformType (đặc biệt là iPhone) và avg_price (hoặc các biến liên quan đến tổng chi tiêu) là những đặc trưng rất hứa hẹn cho việc phân loại big spender và low spenders. Các hành vi trong game như count_gameclicks hay teamLevel có thể đóng vai trò hỗ trợ nhưng không phải là yếu tố quyết định duy nhất.

2.5 Tiền xử lý dữ liệu và chọn đặc trưng

- ❖ Chọn đặc trưng
 - Nhóm lần lượt các trường dữ liệu theo spender type để biết được mối liên hệ

```
[40] df_with_stats.groupby("spender_type", "teamLevel").count().orderBy("spender_type", "teamLevel").show()
```

spender_type	teamLevel	count
big_spender	1	82
big_spender	2	94
big_spender	3	102
big_spender	4	117
big_spender	5	128
big_spender	6	129
big_spender	7	117
big_spender	8	1
low_spender	1	375
low_spender	2	432
low_spender	3	506
low_spender	4	570
low_spender	5	639
low_spender	6	690
low_spender	7	634
low_spender	8	3

Hình 15: Bảng phân bố số lượng người chơi dựa trên spender_type và teamLevel

- Nhóm spender type theo team level: ta có thể thấy người chơi theo từng loại spender phân bố khá đều theo từng team level, và có một xu hướng tăng khi team level tăng lên, đây có thể là một đặc trưng có ích nên giữ lại.

```
[42] df_with_stats.groupby("spender_type", "platformType").count().orderBy("spender_type", "platformType").show()
```

spender_type	platformType	count
big_spender	android	86
big_spender	iphone	663
big_spender	mac	7
big_spender	windows	14
low_spender	android	1549
low_spender	iphone	1271
low_spender	linux	252
low_spender	mac	172
low_spender	windows	605

Hình 16: Bảng phân bố số lượng người chơi dựa trên spender_type và platformType

- Nhóm spender type theo platform type: ta có thể thấy sự phân biệt rõ rệt về lượng người big spender và low spender ở các nền tảng (big spender tập trung hoàn toàn ở iphone), đặc trưng này cũng nên giữ lại. Tương tự với các đặc trưng khác, phân tích đánh giá xem có nên giữ lại hay thay thế.

❖ Tiền xử lý dữ liệu:

➤ Vì mục tiêu của bài toán là nhanh chóng dự đoán được người chơi có tiềm năng là big spender không nên ta sẽ chỉ lấy 3 session đầu của mỗi người chơi, và không lấy toàn bộ dữ liệu để huấn luyện. Tránh trường hợp rò rỉ dữ liệu, do ta phân nhãn hoàn toàn dựa trên tổng lượng giá trị người dùng đã dùng.

```
[45] from pyspark.sql.window import Window
      from pyspark.sql.functions import row_number

      window = Window.partitionBy("userId").orderBy("userSessionId")
      df_early = df_with_stats.withColumn("session_rank", row_number().over(window)).filter("session_rank <= 3")
```

df_early.show()

userId	platformType	userSessionId	teamLevel	count_gameclicks	count_hits	count_buyId	avg_price	made_purchase	total_purchases	total_spent	spender_type	session_rank
0	iphone	23473	1	237	28	0	0.0	0	0	0.0	low_spender	1
0	iphone	24943	2	353	35	0	0.0	0	0	0.0	low_spender	2
0	iphone	28377	3	381	32	0	0.0	0	0	0.0	low_spender	3
1	android	5835	1	73	7	0	0.0	0	8	19.0	low_spender	1
1	android	7847	2	95	14	0	0.0	0	8	19.0	low_spender	2
1	android	10041	3	93	9	2	3.0	1	8	19.0	low_spender	3
2	iphone	12107	3	7	0	0	0.0	0	0	0.0	low_spender	1
2	iphone	12411	4	39	3	0	0.0	0	0	0.0	low_spender	2
2	iphone	15899	5	35	6	0	0.0	0	0	0.0	low_spender	3
6	iphone	28628	7	90	5	0	0.0	0	0	0.0	low_spender	1
8	iphone	27410	4	25	1	0	0.0	0	3	33.0	low_spender	1

Hình 17: Bảng các đặc trưng của dữ liệu trong 3 session đầu tiên

➤ Vì người chơi trải qua nhiều session, nên khi lấy dữ liệu để làm đặc trưng thì phải tìm cách gom nhóm lại, ở đây nhóm chúng em chọn cách tính trung bình cho các đặc trưng mang giá trị số có thể tính như lượt click, lượt hit trong game,...

```
[51] agg_df = df.groupBy("userId", "platformType").agg(
      F.avg("count_gameclicks").alias("avg_clicks"),
      F.avg("count_hits").alias("avg_hits"),
      F.sum("count_buyId").alias("total_early_buys"),
      F.avg("avg_price").alias("early_avg_price"),
      F.max("made_purchase").alias("has_early_purchase"),
      F.count("*").alias("session_count")
    )
```

agg_df.show()

userId	platformType	avg_clicks	avg_hits	total_early_buys	early_avg_price	has_early_purchase	session_count
0	iphone	323.6666666666667	31.66666666666668	0	0.0	0	3
1	android	87.0	10.0	2	1.0	1	3
2	iphone	27.0	3.0	0	0.0	0	3
6	iphone	90.0	5.0	0	0.0	0	1
8	iphone	103.5	12.0	3	5.5	1	2
9	iphone	54.0	6.333333333333333	2	6.666666666666667	1	3
10	linux	719.0	81.66666666666667	4	0.6666666666666666	1	3
12	iphone	78.66666666666667	7.333333333333333	2	10.0	1	3

Hình 18: Bảng các đặc trưng của dữ liệu sau khi đã gom nhóm

➤ Và riêng team level, nhóm em nhận thấy số lượng người chơi có vẻ tăng lên theo team level, và do người chơi có thể tham gia ở những team level khác nhau ở những session khác nhau nên cũng cần gom nhóm lại. Ở đây nhóm em thay thế danh sách các team level cho mỗi người dùng thành max team level (giá trị team level lớn nhất).

```
df_with_stats.groupBy('spender_type', 'teamLevel').count().orderBy('teamLevel', 'spender_type').show()
```

```
+-----+-----+-----+
|spender_type|teamLevel|count|
+-----+-----+-----+
|big_spender|1|82|
|low_spender|1|375|
|big_spender|2|94|
|low_spender|2|432|
|big_spender|3|102|
|low_spender|3|506|
|big_spender|4|117|
|low_spender|4|570|
|big_spender|5|128|
|low_spender|5|639|
|big_spender|6|129|
|low_spender|6|690|
|big_spender|7|117|
|low_spender|7|634|
|big_spender|8|1|
|low_spender|8|3|
+-----+-----+-----+
```

Hình 19: Bảng phân bố số lượng người chơi dựa trên spender_type và teamLevel

```
[57] team_feature = df_temp.groupBy("userId").agg(
      max("teamLevel").cast("int").alias("max_teamLevel")
    )
```

```
[58] final_features = df_temp.join(team_feature, on="userId", how="left")
```

```
[59] final_features = final_features.drop('teamLevel').distinct()
```

```
[60] final_features.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|userId|platformType|avg_clicks|avg_hits|total_early_buys|early_avg_price|session_count|spender_type|max_teamLevel|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|iphone|323.6666666666667|31.666666666666668|0|0.0|3|low_spender|3|
|1|android|87.0|10.0|2|1.0|3|low_spender|3|
|2|iphone|27.0|3.0|0|0.0|3|low_spender|5|
|6|iphone|90.0|5.0|0|0.0|1|low_spender|7|
|8|iphone|103.5|12.0|3|5.5|2|low_spender|5|
|9|iphone|54.0|6.333333333333333|2|6.666666666666667|3|big_spender|4|
|10|linux|719.0|81.66666666666667|4|0.6666666666666666|3|low_spender|3|
|12|iphone|78.66666666666667|7.333333333333333|2|10.0|3|big_spender|3|
|13|android|172.0|16.5|4|4.0|2|low_spender|7|
|14|linux|93.0|6.0|0|0.0|1|low_spender|7|
```

Hình 20: Bảng các đặc trưng sau khi chọn và gom nhóm, thay thế.

❖ Chuẩn hoá và chuyển đổi kiểu đặc trưng

➤ Các trường chứa dữ liệu kiểu string như spender type và platform type được chuyển đổi về dạng index. Riêng Platform type sẽ được chuyển sang dạng OneHotEncoder sau khi chuyển về dạng index vì để giữa các nền tảng không tính khoảng cách thứ tự.

```
[62] spender_indexer = StringIndexer(inputCol="spender_type", outputCol="spender_type_index")
platform_indexer = StringIndexer(inputCol="platformType", outputCol="platform_index")
final_features = spender_indexer.fit(final_features).transform(final_features)
final_features = platform_indexer.fit(final_features).transform(final_features)
```

```
[63] final_features = final_features.drop('platformType', 'spender_type')
```

```
final_features.show()
```

userId	avg_clicks	avg_hits	total_early_buys	early_avg_price	session_count	max_teamLevel	spender_type_index	platform_index
0	323.6666666666667	31.666666666666668	0	0.0	3	3	0.0	0.0
1	87.0	10.0	2	1.0	3	3	0.0	1.0
2	27.0	3.0	0	0.0	3	5	0.0	0.0
6	90.0	5.0	0	0.0	1	7	0.0	0.0
8	103.5	12.0	3	5.5	2	5	0.0	0.0
9	54.0	6.333333333333333	2	6.666666666666667	3	4	1.0	0.0
10	719.0	81.66666666666667	4	0.6666666666666666	3	3	0.0	3.0
12	78.66666666666667	7.333333333333333	2	10.0	3	3	1.0	0.0
13	172.0	16.5	4	4.0	2	7	0.0	1.0
14	93.0	6.0	0	0.0	1	7	0.0	3.0

```
[65] encoder = OneHotEncoder(
    inputCols=[ "platform_index"],
    outputCols=[ "platform_ohc"]
)
final_features = encoder.fit(final_features).transform(final_features)
```

```
[66] final_features = final_features.drop('platform_index')
```

Hình 21: Các đặc trưng sau khi chuyển đổi

➤ Chuẩn hoá các kiểu dữ liệu số như lượt click trung bình, lượt hit trung bình,... sau khi đã được gom nhóm và trích tạo như ở phần trên.

```
numeric_cols = [
    "avg_clicks",
    "avg_hits",
    "total_early_buys",
    "early_avg_price",
    "session_count",
    "max_teamLevel"
]

pre_assembled = VectorAssembler(inputCols=numeric_cols, outputCol="numeric_vector").transform(final_features)

scaler = StandardScaler(inputCol="numeric_vector", outputCol="scaled_numeric", withMean=True, withStd=True)
scaler_model = scaler.fit(pre_assembled)
scaled_df = scaler_model.transform(pre_assembled)
final_assembler = VectorAssembler(
    inputCols=[
        "scaled_numeric",
        "max_teamLevel", "platform_ohc"],
    outputCol="features"
)
final_df = final_assembler.transform(scaled_df)
```

Do dữ liệu bị mất cân bằng, thường thì mọi người chơi đều hướng tới miễn phí hoặc chỉ trả số lượng tiền nhỏ nên ta cần tạo thêm một cột trọng số để cân bằng lại

```
[25] df_with_stats.groupBy('spender_type').count().show()
```

```
↔ +-----+-----+
  |spender_type|count|
  +-----+-----+
  | big_spender|  770|
  | low_spender| 3849|
  +-----+-----+
```

Hình 22: Tỷ lệ big spender - low spender

```
[90] from pyspark.sql.functions import when, lit
      total = 3849 + 770
      weight1 = total / (2 * 3849)
      weight2 = total / (2 * 770)
      train_df = train_df.withColumn(
          "classWeightCol",
          when(train_df["spender_type_index"] == 0, lit(weight1))
            .otherwise(lit(weight2))
      )
```

2.6 Huấn luyện mô hình:

Vì đây là bài toán liên quan đến dữ liệu mất cân bằng, ta cần những mô hình có thể thêm trọng số và đủ mạnh để không bị thiên về bên low spender hay big spender.

```
▶ rf = RandomForestClassifier(
    featuresCol="features",
    labelCol="spender_type_index",
    weightCol="classWeightCol",
    numTrees=90,
    maxDepth=15,
    maxBins=64
)
rf_model = rf.fit(train_df)
train_preds = rf_model.transform(train_df)

acc_eval = MulticlassClassificationEvaluator(
    labelCol="spender_type_index",
    predictionCol="prediction",
    metricName="accuracy"
)
train_acc = acc_eval.evaluate(train_preds)

auc_eval = BinaryClassificationEvaluator(
    labelCol="spender_type_index",
    rawPredictionCol="rawPrediction",
    metricName="areaUnderROC"
)
```



```

train_auc = auc_eval.evaluate(train_preds)

print(f"[Train] Random Forest - Accuracy = {train_acc:.4f}, AUC = {train_auc:.4f}")


test_preds = rf_model.transform(test_df)

acc_eval = MulticlassClassificationEvaluator(
    labelCol="spender_type_index",
    predictionCol="prediction",
    metricName="accuracy"
)
test_acc = acc_eval.evaluate(test_preds)

auc_eval = BinaryClassificationEvaluator(
    labelCol="spender_type_index",
    rawPredictionCol="rawPrediction",
    metricName="areaUnderROC"
)
test_auc = auc_eval.evaluate(test_preds)

print(f"[Test] Random Forest - Accuracy = {test_acc:.4f}, AUC = {test_auc:.4f}")

```


 [Train] Random Forest - Accuracy = 0.9878, AUC = 0.9993
 [Test] Random Forest - Accuracy = 0.9402, AUC = 0.9699

Hình 23: Kết quả và tham số tốt nhất huấn luyện bằng Random Forest

```

from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(
    featuresCol="features",
    labelCol="spender_type_index",
    weightCol="classWeightCol",
    maxIter=100
)

lr_model = lr.fit(train_df)
predictions = lr_model.transform(train_df)
evaluator = MulticlassClassificationEvaluator(
    labelCol="spender_type_index",
    predictionCol="prediction",
    metricName="accuracy"
)

accuracy = evaluator.evaluate(predictions)
print(f"Train Accuracy (LogReg) = {accuracy:.4f}")

auc_eval = BinaryClassificationEvaluator(
    labelCol="spender_type_index",
    rawPredictionCol="rawPrediction",
    metricName="areaUnderROC"
)

```




```

auc = auc_eval.evaluate(predictions)
print(f"AUC (LogReg Weighted) = {auc:.4f}")
predictions = lr_model.transform(test_df)
evaluator = MulticlassClassificationEvaluator(
    labelCol="spender_type_index",
    predictionCol="prediction",
    metricName="accuracy"
)

accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy (LogReg) = {accuracy:.4f}")
auc = auc_eval.evaluate(predictions)
print(f"AUC (LogReg Weighted) = {auc:.4f}")

```


 Train Accuracy (LogReg) = 0.9290
 AUC (LogReg Weighted) = 0.9647
 Test Accuracy (LogReg) = 0.9022
 AUC (LogReg Weighted) = 0.9587

Hình 24: Kết quả và tham số tốt nhất huấn luyện bằng Logistic Regression

```

gbt = GBTClassifier(
    featuresCol="features",
    labelCol="spender_type_index",
    weightCol="classWeightCol",
    maxIter=40,
    maxDepth=5,
    stepSize=0.7
)
model = gbt.fit(train_df)
predictions = model.transform(train_df)
accuracy = accuracy_eval.evaluate(predictions)
auc = auc_eval.evaluate(predictions)
print(f"GBT Train - Accuracy = {accuracy:.4f}, AUC = {auc:.4f}")
predictions = model.transform(test_df)
accuracy = accuracy_eval.evaluate(predictions)
auc = auc_eval.evaluate(predictions)
print(f"GBT Test - Accuracy = {accuracy:.4f}, AUC = {auc:.4f}")

```

 GBT Train - Accuracy = 0.9956, AUC = 0.9999
 GBT Test - Accuracy = 0.9348, AUC = 0.9526

Hình 25: Kết quả và tham số tốt nhất huấn luyện bằng Gradient Boosted Trees

2.7 Đánh giá mô hình

Tập train và test được chia theo người chơi, có nghĩa là giữa 2 tập sẽ không có bất kì người chơi nào được trùng lặp. Và tập test cũng sẽ được gom nhóm, chuẩn hoá, trích xuất và chuyển đổi đặc trưng tương tự như tập train.

Mô hình	Accuracy trên tập test	AUC trên tập test
Logistic Regression	0.9022	0.9587
Random Forest	0.9402	0.9699
GBT	0.9348	0.9526

Hình 26: Bảng tổng hợp kết quả trên tập test

Kết luận: Trong ba mô hình được so sánh, Random Forest cho thấy hiệu quả tổng thể vượt trội khi đạt Accuracy cao nhất trên tập kiểm tra (0.9402) đồng thời duy trì AUC ở mức cao (0.9699). Điều này cho thấy mô hình không chỉ dự đoán chính xác với đa số mẫu mà còn phân biệt tốt giữa hai lớp trong bài toán phân loại mất cân bằng. So với Logistic Regression và GBT, Random Forest có tốt nhất về cả độ chính xác và khả năng phân biệt, do đó được xem là mô hình phù hợp nhất cho bài toán dự đoán loại người chơi tiềm năng có thể là big spender

CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

3.1 Kết luận

Trong kỷ nguyên số hóa hiện nay, Big Data đã trở thành yếu tố cốt lõi trong nhiều lĩnh vực, đặc biệt là trong việc phân tích hành vi người dùng, cá nhân hóa trải nghiệm và ra quyết định dựa trên dữ liệu. Việc xử lý và khai thác hiệu quả khối lượng dữ liệu khổng lồ, phong phú và liên tục gia tăng đòi hỏi sự kết hợp của các công nghệ tiên tiến như Spark và các công cụ trực quan như KNIME.

Thông qua quá trình thực hiện bài tập lớn, nhóm chúng em đã có cơ hội tiếp cận và triển khai bài toán **phân loại người chi tiêu lớn (Big Spender)** trong trò chơi *Catch the Pink Flamingo*, với hai công cụ chính là **Apache Spark** và **KNIME**.

Cụ thể, nhóm đã thực hiện:

- Tiền xử lý và chuẩn hóa dữ liệu hành vi người chơi thông qua KNIME.
- Áp dụng các thuật toán phân lớp như Decision Tree, Random Forest và Logistic Regression trên nền Spark để xử lý dữ liệu lớn.
- Đánh giá hiệu suất phân loại người chơi có xu hướng chi tiêu cao, giúp xác định nhóm khách hàng tiềm năng hoặc có giá trị cao.

Các kết quả chính mà nhóm đạt được:

- Hiểu rõ quy trình xây dựng pipeline phân tích dữ liệu lớn bằng KNIME kết hợp Spark.
- Biết cách kết nối và khai thác khả năng xử lý song song, phân tán của Spark để huấn luyện mô hình học máy trên tập dữ liệu lớn.
- Nâng cao kỹ năng trực quan hóa dữ liệu và phân tích kết quả qua các công cụ đồ họa trong KNIME.
- Phân tích được hành vi người dùng trong game và đưa ra tiêu chí phân loại người chi tiêu lớn (Big Spender).
- So sánh hiệu năng của nhiều mô hình học máy khi áp dụng trên dữ liệu lớn.
- Rèn luyện kỹ năng làm việc nhóm, lập kế hoạch và xử lý sự cố kỹ thuật khi làm việc với công cụ mới.

Tuy nhiên, nhóm cũng gặp một số hạn chế:

- Do giới hạn tài nguyên phần cứng và thời gian, mô hình mới chỉ chạy trên tập dữ liệu được trích xuất từ một phần nhỏ của trò chơi.
- Việc lựa chọn và tinh chỉnh siêu tham số cho các mô hình học máy còn đơn giản.
- Chưa khai thác hết các chức năng nâng cao của Spark MLlib và KNIME như AutoML, hyperparameter tuning, hoặc xử lý thời gian thực.

3.2 Hướng phát triển

Đề tài có thể được phát triển theo các hướng sau:

- **Mở rộng tập dữ liệu và thử nghiệm trên quy mô lớn hơn:** Sử dụng toàn bộ lịch sử người chơi từ nhiều server để mô hình phân loại có thể học từ đầy đủ hành vi.
- **Tối ưu mô hình phân loại:** Kết hợp thêm các thuật toán khác như Gradient Boosting, XGBoost hoặc mạng neural shallow để cải thiện độ chính xác.
- **Phân tích theo thời gian thực (real-time analytics):** Kết hợp Spark Streaming để phân tích hành vi người dùng ngay khi họ đang chơi, từ đó đề xuất các chiến lược cá nhân hóa.
- **Tích hợp pipeline tự động hóa:** Tự động hóa toàn bộ quy trình từ thu thập dữ liệu → tiền xử lý → huấn luyện mô hình → đánh giá kết quả thông qua KNIME Workflow.
- **Ứng dụng trong các bài toán khác:** Áp dụng quy trình tương tự cho các bài toán như phát hiện người chơi gian lận, tối ưu hóa chiến dịch marketing trong game, hoặc đề xuất gói vật phẩm phù hợp.
- **Triển khai trên môi trường đám mây:** Sử dụng AWS EMR hoặc Databricks để xử lý linh hoạt và mở rộng quy mô khi cần thiết.

Việc triển khai đề tài không chỉ giúp nhóm làm quen với các công nghệ và công cụ quan trọng trong lĩnh vực Big Data, mà còn giúp định hình tư duy giải quyết vấn đề một cách hệ thống và thực tế hơn. Đây là một trải nghiệm quý báu, giúp nhóm tự tin hơn khi tiếp cận với các bài toán dữ liệu trong ngành công nghiệp hiện nay.

Nhóm rất mong nhận được sự góp ý của thầy cô và các bạn để tiếp tục hoàn thiện kiến thức và kỹ năng của mình trong những học kỳ tiếp theo.

NHIỆM VỤ CỦA CÁC THÀNH VIÊN

Họ và Tên	Công việc được phân công
Nguyễn Thị Ngọc Lan	<ul style="list-style-type: none">- Chịu trách nhiệm lựa chọn và xử lý đặc trưng từ tập dữ liệu lớn, đảm bảo dữ liệu đầu vào sạch và đầy đủ.- Vận dụng kiến thức về tiền xử lý dữ liệu trong môi trường phân tán, sử dụng Spark MLlib để huấn luyện và đánh giá mô hình.- Thực hiện phân chia tập huấn luyện và kiểm thử hợp lý theo chuẩn Big Data.- Đánh giá mô hình bằng các tiêu chí phổ biến (độ chính xác, độ bao phủ, F1-score), qua đó cho thấy sự hiểu biết về quy trình phân tích dữ liệu lớn.
Tạ Giang Thùy Loan	<ul style="list-style-type: none">- Tiến hành gán nhãn dữ liệu đầu vào và tổ chức lại dữ liệu để phục vụ cho quá trình huấn luyện mô hình phân cụm.- Ứng dụng thuật toán KMeans trên Spark để xử lý dữ liệu ở quy mô lớn.- Trực quan hóa dữ liệu đầu ra bằng PySpark kết hợp công cụ hỗ trợ như KNIME, thể hiện khả năng khai thác kết quả từ hệ sinh thái Big Data.- Phân tích và đánh giá mức độ chính xác của các cụm sau khi phân nhóm.
Hoàng Ngọc Điệp	<ul style="list-style-type: none">- Phụ trách phân tích kết quả mô hình và trực quan hóa dữ liệu phân cụm sau khi huấn luyện.- Ứng dụng Spark SQL và Spark DataFrame API để xử lý song song tập dữ liệu lớn.- Sử dụng công cụ trực quan hóa như Matplotlib, Seaborn để biểu diễn dữ liệu đầu ra rõ ràng và khoa học.- Nhận xét về hiệu năng xử lý, khả năng mở rộng mô hình và đề xuất cải tiến cho quy trình xử lý dữ liệu lớn.