TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

___***___

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

ĐỀ TÀI

THIẾT KẾ HỆ THỐNG PHÂN TÍCH THỜI GIAN THỰC MỨC ĐỘ TƯƠNG TÁC CỦA NGƯỜI CHƠI VÀ PHÂN TÍCH DỮ LIỆU CHO CÁC NỀN TẢNG GAME HIỆN ĐẠI

Giảng viên hướng dẫn:

TS. Trần Hồng Việt
ThS. Ngô Minh Hương
.....

Sinh viên thực hiện:

Mai Phan Anh Tùng
Phạm Hải Tiến
Phạm Quốc Hùng

Tóm tắt nội dung

Báo cáo này trình bày quá trình thiết kế và xây dựng một hệ thống phân tích mức độ tương tác người chơi trong game theo thời gian thực (Real-Time Player Engagement Analytics), dựa trên sự kết hợp của các công nghệ streaming hiện đại bao gồm Apache Kafka, Spark Structured Streaming, Redis, mô hình học máy (Machine Learning) và một hệ thống dashboard trực quan hoá được xây dựng bằng Flask.

Hệ thống được thiết kế nhằm mô phỏng môi trường hoạt động của các nền tảng game trực tuyến, nơi dữ liệu hành vi người chơi liên tục phát sinh, bao gồm các sự kiện như đăng ký tài khoản (register), đăng nhập (login), đăng xuất (logout), mua vật phẩm (purchase) và lên cấp (level_up). Các sự kiện này được gửi vào Kafka theo thời gian thực bởi một mô-đun mô phỏng (game event simulator). Spark Structured Streaming chịu trách nhiệm đọc luồng dữ liệu này theo micro-batch, cập nhật trạng thái hành vi người chơi trong Redis, tính toán lại các chỉ số phái sinh và kích hoạt mô hình học máy để dự đoán mức độ tương tác (Engagement Level).

Bằng cách kết hợp xử lý thời gian thực, mô hình học máy và kho dữ liệu in-memory, hệ thống có thể theo dõi trạng thái người chơi, đưa ra dự đoán tức thời và hỗ trợ các quyết định cá nhân hoá như điều chỉnh độ khó, gợi ý phần thưởng hoặc cảnh báo hành vi bất thường. Mặc dù dữ liệu được mô phỏng, pipeline này phản ánh cấu trúc thực tế của các hệ thống phân tích hành vi người chơi được sử dụng trong nhiều nền tảng game hiện đại.

Mục lục

1 Giới Thiệu

Trong những năm gần đây, ngành công nghiệp game trực tuyến phát triển mạnh mẽ với hàng trăm triệu người chơi hoạt động hàng ngày trên các nền tảng như Steam, PlayStation Network, Xbox Live và các hệ thống game di động. Khối lượng dữ liệu sinh ra từ hành vi người chơi, từ các tương tác trong game đến các giao dịch mua vật phẩm, tạo nên một nguồn dữ liệu khổng lồ và liên tục theo thời gian.

Để tối ưu trải nghiệm người dùng và tăng khả năng giữ chân (retention), các nền tảng game hiện đại đều ứng dụng các hệ thống phân tích thời gian thực nhằm xử lý lượng lớn sự kiện phát sinh liên tục. Bên cạnh đó, vị thế cạnh tranh trong thị trường game đòi hỏi các nhà phát triển phải hiểu sâu sắc hành vi người chơi để cá nhân hóa nội dung, phát hiện bất thường, điều chỉnh độ khó động (dynamic difficulty) và vận hành các chiến dịch marketing theo thời gian thực.

Một trong những mục tiêu quan trọng nhất của phân tích real-time trong ngành game là **dự đoán mức độ tương tác của người chơi (player engagement prediction)**. Người chơi có mức tương tác cao thường:

- có thời gian chơi dài hơn,
- hoàn thành nhiều nhiệm vụ hơn,
- mua nhiều vật phẩm hơn,
- và có xu hướng gắn bó lâu dài với sản phẩm.

Do đó, việc theo dõi và dự đoán tương tác theo thời gian thực có thể giúp nhà vận hành game:

- tùy chỉnh độ khó theo từng người chơi,
- gợi ý phần thưởng chính xác theo hành vi,
- phát hiện người chơi có nguy cơ rời bỏ game,
- tối ưu hóa hệ thống gợi ý nhiệm vụ,
- điều phối các chiến dịch marketing cá nhân hóa,
- và cân bằng tải hệ thống dựa trên hành vi người chơi.

Ngoài ra, dữ liệu game sở hữu đầy đủ ba đặc trưng của Big Data: *Volume*, *Velocity*, và *Variety*. Điều này làm cho việc thiết kế một pipeline xử lý dữ liệu real-time trở nên thiết yếu để đảm bảo khả năng mở rộng, độ trễ thấp và độ tin cậy cao.

Mặc dù nhu cầu phân tích hành vi real-time là rất lớn, việc thu thập trực tiếp dữ liệu telemetry của game thực tế thường gặp nhiều hạn chế.

1.1 Mục tiêu và phạm vi đề tài

Mục tiêu tổng quát của đề tài là thiết kế và triển khai một pipeline phân tích mức độ tương tác người chơi theo thời gian thực, dựa trên dữ liệu hành vi mô phỏng, có thể mở rộng và dễ dàng tích hợp với các nền tảng game hiện đại.

Các mục tiêu cụ thể:

- Xây dựng pipeline xử lý streaming end-to-end từ khâu sinh sự kiện, truyền tải, xử lý, dự đoán đến trực quan hoá.
- Xây dựng và huấn luyện mô hình học máy dự đoán *EngagementLevel* từ dữ liệu hành vi người chơi.
- Cài đặt hệ thống trên nền tảng container hóa (Docker), đảm bảo khả năng triển khai lặp lại và mở rộng.

Phạm vi của đề tài:

- Dữ liệu hành vi được lấy từ tập dữ liệu công khai trên Kaggle và được mô phỏng thành luồng sự kiện; không kết nối với server game thực tế.
- Tập trung vào dự đoán mức độ tương tác (EngagementLevel), chưa triển khai các bài toán khác như churn prediction, fraud detection.
- Hệ thống được triển khai trên một máy chủ đơn (single node) phục vụ mục đích thử nghiệm, chưa đánh giá ở quy mô cluster lớn.

1.2 Đóng góp chính của đề tài

Những đóng góp chính có thể tóm tắt như sau:

- Đề xuất và triển khai một kiến trúc pipeline real-time cho bài toán phân tích hành vi người chơi, kết hợp Kafka, Spark Structured Streaming, Redis, Flask và mô hình học máy.
- Xây dựng mô-đun mô phỏng sự kiện (event simulator) biến dữ liệu batch thành luồng telemetry gần với thực tế.
- Phân tích dữ liệu offline để hiểu các yếu tố ảnh hưởng đến EngagementLevel và thiết kế bộ đặc trưng phù hợp cho mô hình.
- Huấn luyện và tối ưu mô hình Random Forest trên PySpark ML với độ chính xác đạt khoảng 88%.

1.3 Một số nghiên cứu liên quan

Trong những năm gần đây, nhiều công trình đã tập trung vào khai thác telemetry game cho các bài toán phân tích hành vi và dự đoán mức độ tương tác. Chen et al. đề xuất các kiến trúc real-time analytics cho nền tảng game trực tuyến, nhấn mạnh tầm quan trọng của pipeline streaming và lưu trữ in-memory. Kumar et al. sử dụng dữ liệu hành vi chi tiết để dự đoán mức độ gắn bó của người chơi, từ đó hỗ trợ thiết kế cơ chế thưởng và nhiệm vụ phù hợp.

Các nghiên cứu này thường được triển khai trên dữ liệu telemetry nội bộ của các hãng game lớn, vốn không được công khai. Đề tài này tiếp cận theo hướng khác: sử dụng dữ liệu công khai, mô phỏng luồng sự kiện và tập trung vào thiết kế kiến trúc pipeline để có thể dễ dàng tích hợp với nguồn dữ liệu real-time trong tương lai.

1.4 Hạn Chế Thu Thập Dữ Liệu Real-Time và Giải Pháp Mô Phỏng

Một trong những thách thức lớn nhất khi xây dựng hệ thống phân tích hành vi người chơi theo thời gian thực là việc không thể tiếp cận trực tiếp nguồn dữ liệu streaming từ các nền tảng game thương mại. Hầu hết các trò chơi trực tuyến phổ biến đều không cung cấp API công khai cho phép truy cập dữ liệu telemetry real-time do các yếu tố liên quan đến:

- bảo mật và quyền riêng tư người dùng,
- hạn chế truy cập hạ tầng backend,
- rủi ro tải hệ thống nếu mở API real-time,
- chính sách bản quyền và khai thác dữ liệu,
- nguồn dữ liệu huấn luyện bắt buộc phải lấy từ kho dữ liệu công khai thay vì hệ thống sản xuất.

Vì vậy, dữ liệu thời gian thực không thể thu thập trực tiếp từ môi trường game thực tế trong phạm vi nghiên cứu này. Do đó, nhóm sử dụng bộ dữ liệu công khai từ Kaggle:

- **Predict Online Gaming Behavior Dataset** – Kaggle
  https://www.kaggle.com/datasets/rabieelkharoua/predict-online-gaming-behavior-da

Bộ dữ liệu này chứa các thông tin mô tả hành vi và hồ sơ người chơi như tần suất chơi, thời lượng trung bình mỗi phiên, cấp độ nhân vật, số lượng thành tựu và hành vi chi tiêu trong game. Tuy nhiên, đây là dữ liệu dạng *batch*, không phải dữ liệu thời gian thực (streaming).

Để đáp ứng yêu cầu của pipeline real-time, dữ liệu được chuyển đổi thành dạng sự kiện (event stream) bằng một mô-đun mô phỏng hành vi người chơi (Kafka Producer):

- mỗi dòng dữ liệu hồ sơ người chơi được nạp vào hệ thống như trạng thái khởi tạo,
- các sự kiện như login/logout/purchase/level_up được sinh ngẫu nhiên theo phân phối hợp lý,
- dữ liệu mô phỏng được đẩy liên tục vào Kafka tạo thành luồng sự kiện gần giống với hành vi của người chơi thật,
- Spark Structured Streaming xử lý các sự kiện này như thể chúng là telemetry thực tế.

Cách tiếp cận này đảm bảo rằng pipeline có thể được thử nghiệm, đánh giá và tinh chỉnh trong điều kiện gần giống môi trường game sản xuất, mặc dù dữ liệu gốc không đến từ API real-time. Quan trọng hơn, kiến trúc pipeline được thiết kế sao cho nó có thể dễ dàng được tích hợp với dữ liệu thực khi nguồn telemetry real-time khả dụng trong tương lai.

2 Giới Thiệu Các Công Nghệ Sử Dụng

Hệ thống được xây dựng dựa trên nhiều công nghệ xử lý dữ liệu hiện đại, mỗi thành phần đảm nhiệm một vai trò khác nhau trong pipeline real-time. Các công nghệ này không chỉ giúp đảm bảo khả năng xử lý luồng dữ liệu có tốc độ cao, độ trễ thấp, mà còn hỗ trợ việc triển khai, mở rộng và giám sát hệ thống một cách linh hoạt. Phần này trình bày tổng quan về từng công nghệ chính được sử dụng và lý do chúng phù hợp với bài toán phân tích hành vi người chơi.

2.1 Apache Kafka

Apache Kafka là nền tảng xử lý dữ liệu dạng streaming phân tán, được thiết kế để xử lý lượng sự kiện rất lớn theo mô hình publish–subscribe với độ trễ thấp và tính sẵn sàng cao. Kafka lưu trữ dữ liệu theo dạng log phân vùng (partitioned log), giúp hệ thống có thể mở rộng tuyến tính khi số lượng sự kiện tăng lên.

Trong hệ thống đề xuất, Kafka được sử dụng như lớp *ingestion* (tiếp nhận dữ liệu) trung tâm:

- tiếp nhận luồng sự kiện hành vi người chơi được mô phỏng từ Kafka Producer,
- lưu trữ tạm thời các sự kiện trong các topic với cơ chế nhân bản (replication) để đảm bảo tính bền vững,
- phân phối đồng thời luồng dữ liệu cho Spark Structured Streaming hoặc các consumer khác nếu cần mở rộng hệ thống trong tương lai.

Việc sử dụng Kafka giúp tách biệt rõ ràng giữa lớp sinh dữ liệu (game/event simulator) và lớp xử lý (Spark), từ đó tăng tính linh hoạt và khả năng mở rộng của pipeline.

Hình 1: Logo Apache Kafka

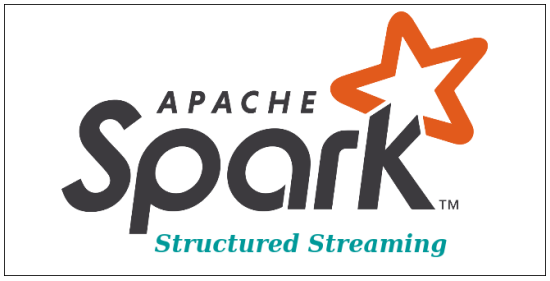
2.2 Apache Spark Structured Streaming

Apache Spark là framework xử lý dữ liệu phân tán in-memory phổ biến, hỗ trợ cả batch và streaming. Structured Streaming là mô hình streaming của Spark dựa trên khái niệm *unbounded table*, cho phép lập trình luồng dữ liệu bằng cùng một API với DataFrame/Dataset trong chế độ batch.

Trong hệ thống:

- Spark Structured Streaming đọc luồng sự kiện từ Kafka theo chu kỳ micro-batch (2 giây),
- chuyển đổi chuỗi JSON nhận được thành các bản ghi có schema rõ ràng,
- cập nhật các chỉ số liên quan đến hành vi người chơi (số phiên chơi, tổng thời gian chơi, tổng chi tiêu, cấp độ hiện tại, …),
- tính toán các đặc trưng phái sinh phục vụ mô hình dự đoán,
- gọi mô hình học máy đã được huấn luyện offline và ghi kết quả dự đoán trở lại hệ thống lưu trữ real-time.

Việc tận dụng Structured Streaming giúp đơn giản hóa việc lập trình xử lý luồng (không cần thao tác trực tiếp ở mức thấp) nhưng vẫn đảm bảo tính mở rộng và khả năng chịu lỗi nhờ kiến trúc phân tán của Spark.



Hình 2: Apache Spark Structured Streaming

2.3 Redis

Redis là hệ quản trị cơ sở dữ liệu in-memory key–value có hiệu năng rất cao, hỗ trợ nhiều cấu trúc dữ liệu như string, hash, list, set, sorted set, … cùng với thời gian truy cập chỉ ở mức mili–giây. Nhờ đặc điểm này, Redis thường được sử dụng như một lớp *fast storage* hoặc cache trong các hệ thống real-time.

Trong pipeline đề xuất, Redis đảm nhiệm vai trò:

- lưu trạng thái mới nhất của từng người chơi dưới dạng `Hash` (ví dụ: thông tin nhân khẩu học, số phiên chơi, tổng thời lượng, tổng chi tiêu, mức độ tương tác dự đoán),
- lưu danh sách sự kiện gần nhất dưới dạng `List` để hiển thị luồng sự kiện (live event feed) trên dashboard,
- lưu các chỉ số tổng quan (live metrics) như số người chơi online, tổng số người dùng, tổng số sự kiện đã xử lý.

Việc dùng Redis giúp dashboard có thể truy vấn dữ liệu gần như ngay lập tức mà không làm ảnh hưởng đến hiệu năng của Spark hoặc Kafka.

Hình 3: Logo Redis

2.4 PySpark MLlib (Machine Learning)

PySpark MLlib là thư viện học máy của Spark, cung cấp các công cụ cho tiền xử lý dữ liệu, trích chọn đặc trưng, xây dựng pipeline và huấn luyện nhiều thuật toán học máy trên cụm phân tán. Khác với các thư viện đơn nút như Scikit–Learn, MLlib được thiết kế để xử lý các tập dữ liệu có kích thước lớn trong môi trường Big Data.

Trong đề tài, MLlib được sử dụng để:

- chuẩn hóa và mã hóa các biến phân loại (Gender, Location, GameGenre, GameDifficulty) bằng `StringIndexer` và `OneHotEncoder`,
- ghép tất cả đặc trưng đầu vào thành một vector duy nhất bằng `VectorAssembler`,

- huấn luyện mô hình *Random Forest Classifier* nhằm dự đoán nhãn `EngagementLevel`,
- tối ưu siêu tham số bằng `CrossValidator` kết hợp với lưới tham số (Grid Search),
- lưu pipeline đã huấn luyện thành mô hình có thể nạp lại để suy luận trong pha streaming.

Sau khi tối ưu, mô hình Random Forest đạt độ chính xác khoảng **88.15%** trên tập kiểm thử, đủ tốt để tích hợp vào pipeline real-time và cung cấp dự đoán mức độ tương tác cho từng người chơi.

Hình 4: Spark MLlib cho bài toán dự đoán EngagementLevel

2.5 Flask

Flask là một micro–framework Python dùng để xây dựng các ứng dụng web và API REST một cách đơn giản nhưng linh hoạt. Ưu điểm của Flask là cấu trúc gọn nhẹ, dễ tích hợp với các thư viện Python khác và phù hợp cho việc phát triển nhanh các hệ thống dashboard hoặc dịch vụ nội bộ.

Trong hệ thống real-time player analytics:

- Flask cung cấp các API truy vấn Redis để lấy dữ liệu trạng thái người chơi, danh sách sự kiện mới nhất và các chỉ số tổng quan,
- render giao diện dashboard cho phép người vận hành theo dõi số người chơi online, tổng số sự kiện, luồng sự kiện, danh sách người chơi đang hoạt động và chi tiết từng người chơi,
- xử lý yêu cầu từ người dùng (ví dụ: chọn một Player ID để xem chi tiết) và trả về dữ liệu tương ứng dưới dạng JSON hoặc HTML.

Flask đóng vai trò là "lớp trình bày" (presentation layer) của hệ thống, kết nối người dùng cuối với pipeline xử lý dữ liệu ở phía sau.

Hình 5: Logo Flask

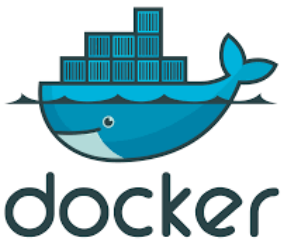
2.6 Docker

Docker là nền tảng container hóa cho phép đóng gói ứng dụng cùng toàn bộ môi trường phụ thuộc (thư viện, cấu hình, hệ điều hành tối thiểu) thành các container nhẹ, có thể triển khai đồng nhất trên nhiều máy khác nhau. So với cách triển khai truyền thống, Docker giúp giảm thiểu lỗi do khác biệt môi trường và đơn giản hóa quá trình cài đặt.

Trong đề tài, Docker được sử dụng để:

- đóng gói từng thành phần của hệ thống (Kafka, Zookeeper, Spark Streaming job, Redis, Flask dashboard) thành các container độc lập,
- điều phối các container bằng `docker-compose`, giúp khởi động hoặc dừng toàn bộ pipeline chỉ với một vài lệnh,
- tạo môi trường triển khai thống nhất giữa máy phát triển và máy trình diễn/báo cáo, đảm bảo tính tái lập thí nghiệm.

Nhờ Docker, kiến trúc pipeline trở nên linh hoạt và dễ mở rộng: có thể tăng số instance của các thành phần như Spark hoặc Kafka khi hệ thống cần xử lý luồng dữ liệu lớn hơn.



Hình 6: Logo Docker

3 Phân Tích Dữ Liệu Offline (Offline Data Analytics)

3.1 Mô tả dữ liệu và quy trình tiền xử lý

Để hỗ trợ quá trình xây dựng mô hình học máy và hiểu rõ hành vi người chơi, một quy trình phân tích dữ liệu (Exploratory Data Analysis – EDA) được tiến hành trên tập dữ liệu `online_gaming_behavior_dat`
Việc phân tích giúp xác định phân phối, mức độ biến động, tương quan giữa các biến và nhận diện những đặc trưng quan trọng ảnh hưởng đến mức độ tương tác (EngagementLevel). Dữ liệu được xử lý bằng Python cùng các thư viện pandas, numpy và matplotlib.

Tập dữ liệu bao gồm các nhóm đặc trưng chính:

- **Nhân khẩu học**: Age, Gender, Location.
- **Hành vi chơi**: SessionsPerWeek, AvgSessionDurationMinutes, PlayTimeHours.
- **Tiến trình**: PlayerLevel, AchievementsUnlocked.
- **Chi tiêu**: InGamePurchases.
- **Độ khó trò chơi**: GameDifficulty.
- **Nhãn mục tiêu**: EngagementLevel.

Ngoài ra, một đặc trưng mới được tạo trong quá trình tiền xử lý:

- **isAddicted**: giá trị bằng 1 khi thời lượng chơi trung bình tuần vượt quá 1280 phút, ngược lại bằng 0.

Quy trình tiền xử lý dữ liệu được thực hiện như sau:

- Kiểm tra và xử lý giá trị thiếu: loại bỏ các dòng bị thiếu nghiêm trọng hoặc điền giá trị trung vị/giá trị phổ biến cho một số cột.
- Phát hiện các giá trị ngoại lai (outlier) dựa trên phân phối PlayTimeHours, SessionsPerWeek và giới hạn ở một ngưỡng hợp lý.
- Chuẩn hóa/cân bằng một số biến số để tránh các đặc trưng có độ lớn quá khác nhau chi phối mô hình.
- Mã hóa các biến phân loại (Gender, Location, GameGenre, GameDifficulty) bằng kỹ thuật phù hợp trong PySpark ML (StringIndexer + OneHotEncoder).

3.2 So Sánh Hành Vi Giữa Các Nhóm EngagementLevel

Dữ liệu được nhóm theo nhãn EngagementLevel và tính trung bình các biến hành vi:

- Nhóm **High Engagement** có trung bình PlayTimeHours, AchievementsUnlocked và PlayerLevel cao hơn hai nhóm còn lại.
- Nhóm **Low Engagement** có tần suất chơi và tổng thời gian chơi thấp nhất.
- Điều này hỗ trợ mô hình phân loại với ngưỡng phân chia rõ ràng giữa các nhóm.

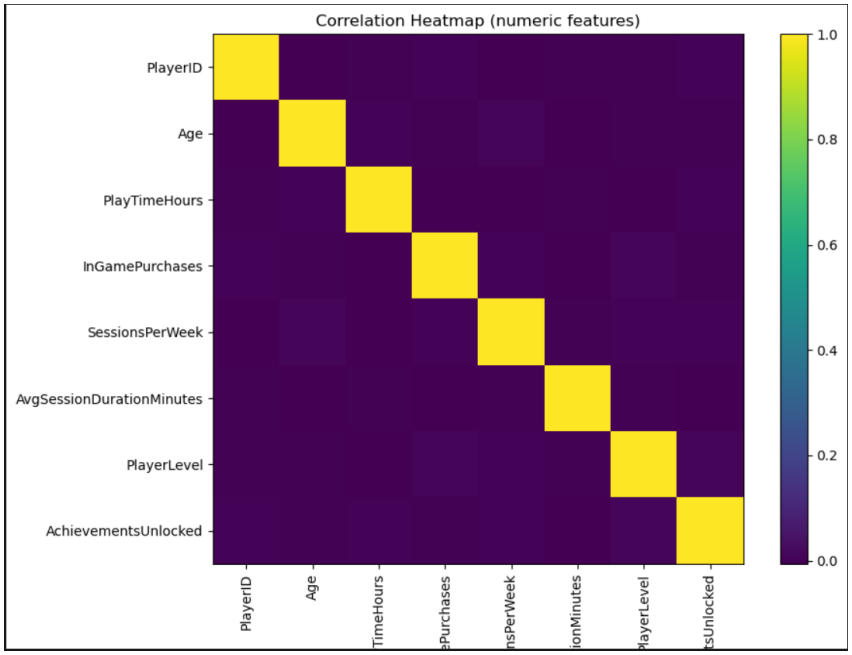
3.3 Ma Trận Tương Quan

Kết quả ma trận tương quan Pearson cho thấy phần lớn các biến số trong dữ liệu không có mối tương quan tuyến tính mạnh với nhau. Hầu hết các giá trị tương quan ngoài đường chéo đều rất nhỏ (gần bằng 0), phản ánh rằng các đặc trưng hành vi như PlayTimeHours, SessionsPerWeek, AvgSessionDurationMinutes, InGamePurchases, AchievementsUnlocked và PlayerLevel hoạt động tương đối độc lập.

- Các biến hành vi và tiến trình (PlayTimeHours, PlayerLevel, AchievementsUnlocked) không có tương quan tuyến tính đáng kể như thường thấy trong các bộ dữ liệu thực tế.
- SessionsPerWeek và AvgSessionDurationMinutes chỉ có mức tương quan nhẹ, cho thấy tần suất chơi và thời lượng trung bình mỗi phiên không ràng buộc chặt chẽ.
- InGamePurchases gần như không tương quan với các đặc trưng khác, phản ánh hành vi chi tiêu độc lập trong dataset mô phỏng.

Điều này cho thấy bộ dữ liệu có cấu trúc đa chiều, không bị chi phối bởi một mối quan hệ tuyến tính mạnh nào. Do đó, mô hình học máy cần kết hợp nhiều đặc trưng khác nhau thay vì dựa vào một quan hệ nổi bật duy nhất để dự đoán EngagementLevel.

Các mối tương quan này xác nhận rằng mô hình dự đoán EngagementLevel nên dựa chủ yếu vào đặc trưng hành vi và tiến trình người chơi.



Hình 7: Ma trận tương quan Pearson giữa các đặc trưng số

3.4 Phân Cụm Người Chơi (Clustering)

K-Means được áp dụng trên bộ đặc trưng gồm PlayTimeHours, SessionsPerWeek, AvgSessionDurationMinutes InGamePurchases, AchievementsUnlocked và PlayerLevel. Dữ liệu được chuẩn hoá trước khi huấn luyện và số cụm được chọn là $k = 4$ dựa trên Silhouette Score.

Kết quả phân cụm cho thấy sự tách biệt tự nhiên giữa bốn nhóm người chơi:

1. **Nhóm casual**: thời lượng chơi thấp, tiến trình chậm và ít mở khoá thành tựu.
2. **Nhóm mid-core**: tần suất chơi ổn định, thời lượng trung bình, tiến trình tăng đều.
3. **Nhóm hardcore**: thời lượng chơi cao, số phiên chơi lớn và đạt level cao.
4. **Nhóm có chi tiêu**: có InGamePurchases cao hơn các nhóm còn lại.

EngagementLevel	High	Low	Medium
Cluster			
0	0.255261	0.263960	0.480779
1	0.264311	0.228977	0.506711
2	0.265390	0.253327	0.481283
3	0.249515	0.284036	0.466449

Hình 8: Phân bố tỉ lệ EngagementLevel (Low–Medium–High) trong từng cụm

Kết quả trên cho thấy mỗi cụm có tỉ lệ EngagementLevel khác nhau, phản ánh sự khác biệt tự nhiên trong hành vi của người chơi. Cụm có tỉ lệ **High** cao hơn thường đi kèm với chỉ số PlayTimeHours, AchievementsUnlocked và PlayerLevel lớn hơn, trong khi các cụm có tỉ lệ

Low cao chủ yếu rơi vào nhóm casual. Điều này xác nhận rằng các cụm K-Means được hình thành phù hợp với cấu trúc nhãn tương tác trong dữ liệu.

Để phân tích sâu hơn nhóm người chơi "giá trị cao", nhóm nghiên cứu tiếp tục tính **giá trị trung bình của các đặc trưng hành vi chỉ trên những người chơi có EngagementLevel = 2**. Cách tiếp cận này giúp giảm nhiễu từ các nhóm tương tác thấp và làm nổi bật đặc trưng cốt lõi của những người chơi hoạt động mạnh. Kết quả cho thấy tâm cụm của nhóm EngagementLevel cao có sự khác biệt rõ ràng về PlayTimeHours, AchievementsUnlocked và PlayerLevel, từ đó hình thành các hồ sơ hành vi (player profiles) đại diện cho từng loại người chơi.

Ứng dụng của kết quả phân cụm đối với hệ thống real-time

Mặc dù phân cụm được trình bày trong phần phân tích offline, kết quả này có giá trị mạnh mẽ trong việc cải thiện hệ thống real-time:

- **Cá nhân hoá theo hồ sơ cụm**: dữ liệu streaming từ Spark có thể được so khớp với tâm cụm gần nhất để phân loại người chơi theo thời gian thực. Điều này cho phép hệ thống gợi ý nhiệm vụ, sự kiện hoặc độ khó phù hợp hơn.
- **Phát hiện bất thường hành vi**: nếu người chơi thuộc cụm hardcore nhưng PlayTimeHours giảm mạnh so với tâm cụm, Spark Streaming có thể cảnh báo sớm nguy cơ giảm tương tác hoặc rời bỏ game.
- **Tối ưu hoá chiến dịch monetization**: cụm có InGamePurchases cao phản ánh nhóm khách hàng có khả năng chi tiêu. Thông tin này hỗ trợ triển khai các gói ưu đãi, bundle hoặc sự kiện khuyến khích nạp tiền ngay trong giao diện real-time.
- **Tăng độ chính xác của mô hình dự đoán**: tâm cụm đóng vai trò như "chuẩn hành vi" cho từng loại người chơi. Mô hình phân loại EngagementLevel có thể so sánh dữ liệu người chơi mới với các tâm cụm, giúp Spark Streaming đưa ra dự đoán ổn định hơn.

Kết luận về phân cụm

Phân tích phân cụm không chỉ cho thấy sự tồn tại tự nhiên của các nhóm hành vi casual–midcore–hardcore, mà còn giúp hệ thống xây dựng các hồ sơ người chơi để nâng cao khả năng cá nhân hóa và phát hiện bất thường trong môi trường streaming real-time. Đây là bước quan trọng giúp gắn kết phân tích offline với hệ thống vận hành trực tiếp.

4 Xây dựng mô hình bằng PySpark ML

Quy trình huấn luyện mô hình được triển khai hoàn toàn bằng `pyspark.ml`. Toàn bộ tiến trình bao gồm tiền xử lý dữ liệu, tạo pipeline, tối ưu siêu tham số và đánh giá mô hình.

4.1 Tiền xử lý và Feature Engineering

Các cột được sử dụng gồm:

- Age, Gender, Location, GameGenre,
- InGamePurchases, SessionsPerWeek, AvgSessionDurationMinutes,

- PlayerLevel, AchievementsUnlocked, GameDifficulty, EngagementLevel.

Các bước tiền xử lý:

- Chuẩn hóa các cột chuỗi bằng **StringIndexer**.
- Chuyển thành One-Hot Vector bằng **OneHotEncoder**.
- Ghép toàn bộ đặc trưng vào một vector duy nhất bằng **VectorAssembler**.
- Tạo thêm cột **isAddicted** theo quy tắc:

$$\text{isAddicted} = \begin{cases} 1, & \text{AvgSessionDurationMinutes} > 1280 \\ 0, & \text{otherwise} \end{cases}$$

4.2 Xây dựng Pipeline huấn luyện

Pipeline được xây dựng như sau:

```
pipeline = Pipeline(stages = indexers +
                    [encoder, label_indexer, assembler, rf])
```

Pipeline bao gồm:

- Chuỗi các bước xử lý dữ liệu (indexers),
- Bộ mã hóa One-Hot,
- Mã hóa nhãn mục tiêu,
- VectorAssembler,
- Mô hình RandomForestClassifier.

4.3 Tối ưu siêu tham số bằng Cross-Validation

Mạng lưới siêu tham số (Grid Search):

- Số cây trong rừng: numTrees = [100, 200]
- Độ sâu tối đa của cây: maxDepth = [8, 10]
- Chiến lược chọn đặc trưng: featureSubsetStrategy = ”sqrt”

Cross-Validation:

- 3-fold cross-validation
- Đánh giá bằng thước đo Accuracy
- Số luồng chạy song song: 4

4.4 Kết quả huấn luyện

- Thời gian huấn luyện: **khoảng 5 phút**.
- Độ chính xác tối ưu: **0.8815**.
- Mô hình cuối cùng được lưu tại: `cv_pipeline_model`.

4.5 Tổng quan về mô hình Random Forest

Random Forest là mô hình học máy dạng tập hợp (ensemble) bao gồm nhiều cây quyết định (decision tree) huấn luyện trên các mẫu dữ liệu và tập đặc trưng con khác nhau. Mỗi cây đưa ra một dự đoán và mô hình cuối cùng sử dụng cơ chế bỏ phiếu đa số (majority voting) để quyết định nhãn đầu ra.

Ưu điểm của Random Forest trong bài toán này:

- Xử lý tốt dữ liệu hỗn hợp giữa biến số và biến phân loại.
- Giảm hiện tượng overfitting so với một cây quyết định đơn lẻ nhờ cơ chế bagging.
- Dễ cài đặt và tích hợp trong PySpark ML, hỗ trợ huấn luyện phân tán trên tập dữ liệu lớn.

4.6 Đánh giá chi tiết mô hình

Ngoài độ chính xác tổng thể, mô hình còn được đánh giá bằng các thước đo chi tiết hơn:

- **Confusion matrix**: cho thấy mô hình phân loại nhầm giữa các lớp Low, Medium, High như thế nào.
- **Precision, Recall, F1-score** cho từng lớp, giúp đánh giá khả năng nhận diện đúng người chơi thuộc từng mức độ tương tác.

Kết quả cho thấy các lớp **Medium** và **High Engagement** được mô hình phân loại với độ chính xác cao hơn, trong khi lớp **Low Engagement** có tỉ lệ nhầm lẫn cao hơn do số lượng mẫu ít hơn và ranh giới với lớp Medium chưa thực sự rõ ràng. Điều này gợi ý hướng cải thiện trong tương lai như:

- Cân bằng dữ liệu giữa các lớp (class balancing).
- Thử nghiệm thêm các mô hình khác như Gradient Boosting, XGBoost.

5 Pipeline Xử Lý Dữ Liệu Real-Time

Hệ thống được thiết kế theo kiến trúc pipeline streaming nhằm xử lý dữ liệu người chơi theo thời gian thực từ lúc sự kiện được sinh ra cho đến lúc hiển thị trên dashboard. Pipeline bao gồm bốn lớp chức năng chính: **(1) mô phỏng dữ liệu**, **(2) truyền tải sự kiện**, **(3) xử lý streaming**, **(4) lưu trữ và dự đoán**, và **(5) trực quan hoá**. Toàn bộ thành phần vận hành liên tục và đồng bộ như trong một hệ thống game thực tế.

5.1 Tổng quan kiến trúc và môi trường triển khai

Về tổng thể, kiến trúc hệ thống được tổ chức thành chuỗi:

$$\text{Producer} \rightarrow \text{Kafka} \rightarrow \text{Spark Structured Streaming} \rightarrow \text{Redis} \rightarrow \text{Flask Dashboard}.$$

Các thành phần được triển khai dưới dạng container Docker riêng biệt và được điều phối bằng `docker-compose`. Cấu hình minh hoạ:

- 1 container cho Zookeeper, 1 container cho Kafka Broker.
- 1 container cho Spark Streaming job.
- 1 container cho Redis.
- 1 container cho Flask (backend + dashboard).

Hệ thống được chạy thử nghiệm trên máy tính cá nhân với Docker Desktop. Để hệ thống hoạt động ổn định, cấu hình khuyến nghị tối thiểu gồm 4 vCPU và 8GB RAM, đảm bảo Spark và Kafka có đủ tài nguyên để xử lý luồng sự kiện liên tục.

5.2 Mô phỏng sự kiện người chơi (Kafka Producer)

Vì không thể thu thập dữ liệu real-time từ game do hạn chế API public, dữ liệu Kaggle được sử dụng làm nguồn offline và được chuyển hóa thành luồng sự kiện real-time. Kafka Producer:

- đọc dữ liệu từ file CSV,
- tạo sự kiện *register*, *login*, *logout*, *purchase*, *level_up*,
- gửi sự kiện vào Kafka topic `game_events` theo thời gian thực.

Điều này mô phỏng hành vi người chơi tương tự như telemetry của game thật.

5.3 Truyền tải sự kiện (Kafka Broker)

Kafka chịu trách nhiệm phân phối dữ liệu giữa Producer và Spark:

- đảm bảo lưu trữ tạm thời có độ tin cậy cao,
- hỗ trợ scale-out khi có nhiều consumers,
- duy trì độ trễ thấp trong truyền tải.

5.4 Xử lý streaming (Spark Structured Streaming)

Spark là thành phần cốt lõi trong pipeline real-time:

1. đọc sự kiện từ Kafka theo micro-batch 2 giây,
2. parse JSON theo schema định nghĩa sẵn,
3. cập nhật các chỉ số thô vào Redis (TotalSessions, TotalSpent, PlayerLevel),
4. tính toán chỉ số phái sinh như AvgSessionDuration,

5. lấy dữ liệu mới nhất và chạy mô hình học máy dự đoán EngagementLevel,
6. ghi lại kết quả vào Redis.

Spark giữ vai trò "bộ não" của hệ thống, đảm bảo mọi dữ liệu đều được cập nhật liên tục.

5.5 Lưu trữ real-time và dự đoán (Redis + ML Model)

Redis đóng vai trò như kho dữ liệu trạng thái của người chơi:

- lưu trạng thái player dưới dạng Hash,
- lưu danh sách sự kiện gần nhất dưới dạng List,
- lưu các chỉ số tổng quan (live metrics),
- phản hồi dữ liệu gần như tức thời cho dashboard.

Mô hình Random Forest (được huấn luyện offline) dự đoán mức tương tác mỗi khi trạng thái người chơi thay đổi. Kết quả được ghi đè trực tiếp vào Redis.

5.6 Trực quan hoá và API (Flask Dashboard)

Dashboard được xây dựng bằng Flask + HTML/JS:

- API truy vấn Redis theo thời gian thực,
- hiển thị số người chơi online, tổng sự kiện, biểu đồ và chi tiết từng người chơi,
- cập nhật dữ liệu theo chu kỳ ngắn mà không ảnh hưởng hiệu năng hệ thống.

Pipeline đảm bảo tất cả thành phần hoạt động đồng bộ, từ mô phỏng dữ liệu đến hiển thị trên giao diện.

5.7 Đánh giá hiệu năng pipeline real-time

Trong quá trình thử nghiệm, nhóm tiến hành quan sát một số chỉ số hiệu năng cơ bản:

- **Độ trễ end-to-end**: thời gian từ lúc một sự kiện được sinh ra tại Producer cho đến khi kết quả dự đoán tương ứng xuất hiện trên dashboard thường ở mức vài giây (thấp hơn hoặc xấp xỉ chu kỳ micro-batch 2 giây).
- **Thông lượng (throughput)**: hệ thống xử lý ổn định hàng trăm sự kiện mỗi phút trong môi trường thử nghiệm, không xuất hiện hiện tượng backlog nghiêm trọng trong Kafka.
- **Mức sử dụng tài nguyên**: CPU và RAM của các container luôn nằm trong ngưỡng chấp nhận được, cho thấy kiến trúc đề xuất có thể mở rộng thêm khi tăng cấu hình phần cứng.

Các kết quả này chứng minh pipeline có khả năng đáp ứng yêu cầu xử lý real-time với độ trễ thấp trong bối cảnh thử nghiệm, đồng thời còn dư địa để mở rộng cho các kịch bản tải cao hơn.

6 Kết Luận và Hướng Phát Triển

6.1 Tóm tắt kết quả đạt được

Hệ thống được xây dựng đã mô phỏng đầy đủ một nền tảng phân tích dữ liệu người chơi real-time, bao gồm ingestion, streaming processing, fast storage, machine learning và trực quan hoá. Pipeline kết hợp giữa Kafka, Spark Structured Streaming, Redis và Flask cho phép:

- sinh và xử lý luồng sự kiện hành vi người chơi gần với thực tế,
- liên tục cập nhật trạng thái của từng người chơi,
- dự đoán mức độ tương tác dựa trên mô hình Random Forest được huấn luyện offline,
- hiển thị kết quả trên dashboard phục vụ giám sát và ra quyết định.

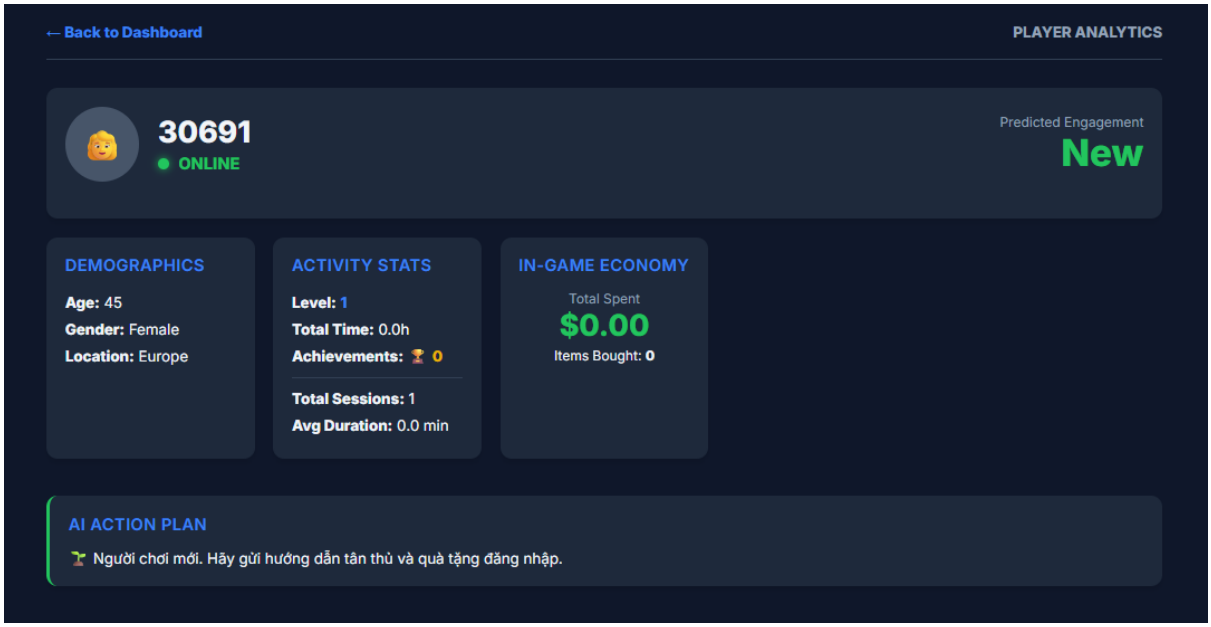
Một số kết quả hiển thị của hệ thống được minh hoạ trong Hình 9 và Hình 10. Dashboard tổng quan cho phép theo dõi số người chơi đang online, tổng số người chơi, tổng số sự kiện đã xử lý theo thời gian thực, cùng luồng sự kiện mới nhất. Màn hình chi tiết cho từng người chơi hiển thị đầy đủ thông tin nhân khẩu học, thống kê hoạt động, chỉ số chi tiêu trong game, mức độ tương tác dự đoán và gợi ý hành động vận hành (AI action plan) tương ứng với hồ sơ hành vi hiện tại.

Hình 9: Dashboard giám sát game real-time: số người chơi online, tổng người dùng, tổng số sự kiện và luồng sự kiện mới nhất.

6.2 Ý nghĩa thực tiễn

Về mặt thực tiễn, kiến trúc pipeline đề xuất có thể được áp dụng cho:

- Các studio game muốn nhanh chóng thử nghiệm các ý tưởng phân tích hành vi người chơi mà không cần thay đổi sâu vào backend hiện tại.



Hình 10: Màn hình phân tích chi tiết một người chơi: thông tin nhân khẩu học, hành vi chơi, chi tiêu, mức độ tương tác dự đoán và gợi ý hành động.

- Hệ thống vận hành (live-ops) cần theo dõi mức độ tương tác để:
  - điều chỉnh độ khó động theo từng người chơi,
  - kích hoạt nhiệm vụ, sự kiện hoặc phần thưởng đúng thời điểm,
  - phát hiện sớm người chơi có nguy cơ rời bỏ game.

6.3 Hạn chế và hướng phát triển

Đề tài vẫn còn một số hạn chế:

- Dữ liệu sử dụng là dữ liệu batch công khai được mô phỏng lại thành streaming, chưa sử dụng dữ liệu telemetry real-time từ hệ thống game sản xuất.
- Mô hình học máy mới dừng ở Random Forest, chưa so sánh với nhiều thuật toán khác hoặc mô hình deep learning chuyên sâu cho hành vi người chơi.
- Hệ thống được triển khai và đánh giá trên môi trường single-node, chưa có thử nghiệm trên cluster phân tán quy mô lớn.

Trong tương lai, có thể mở rộng:

- thêm anomaly detection (phát hiện gian lận, hành vi bất thường),
- thêm phân tích hành vi nâng cao (player journey, funnel analysis),
- thêm A/B testing real-time cho các tính năng hoặc sự kiện trong game,
- tích hợp gợi ý nhiệm vụ hoặc nội dung theo thời gian thực dựa trên Reinforcement Learning,
- thay thế dữ liệu mô phỏng bằng telemetry thực tế khi có thể tiếp cận nguồn dữ liệu này.

Tài Liệu Tham Khảo

1. Apache Kafka Documentation. Available: https://kafka.apache.org/documentation/
2. Apache Spark Structured Streaming Guide. Available: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
3. Redis In-Memory Database Documentation. Available: https://redis.io/documentation
4. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, 2011.
5. Akiba et al., "Optuna: A Next-generation Hyperparameter Optimization Framework," KDD 2019.
6. Géron, A., "Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow," O'Reilly Media, 2019.
7. Chen et al., "Building Real-Time Analytics Systems for Online Gaming Platforms," IEEE Transactions on Games, 2020.
8. Kumar et al., "User Engagement Prediction in Games using Behavioral Telemetry," ACM CHI Play, 2018.

Nhiệm Vụ Của Các Thành Viên

- **Mai Phan Anh Tùng**: Xử lý và phân tích dữ liệu, viết báo cáo, chuẩn bị slide.
- **Phạm Hải Tiến**: Xây dựng mô hình dự đoán engagement.
- **Phạm Quốc Hùng**: Xây dựng pipeline mô phỏng thời gian thực và phát triển giao diện web đơn giản.