ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
VIỆN TRÍ TUỆ NHÂN TẠO

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

ĐỀ TÀI

MÔ HÌNH HÓA RỦI RO TÀI CHÍNH VÀ ĐÁNH GIÁ TÍN DỤNG

Giảng viên hướng dẫn: TS. Trần Hồng Việt
ThS. Ngô Minh Hương

Nhóm sinh viên thực hiện: **Nhóm 8 - K68 UET**
1. Nguyễn Công Cường - 23020338
2. Hoàng Tiến Đạt - 23020352

HÀ NỘI - 28/11/2025

Mục lục

Tóm tắt nội dung

Bài báo trình bày quy trình xây dựng hệ thống dự đoán rủi ro tín dụng trên nền tảng dữ liệu lớn, kết hợp xử lý batch và thời gian thực (realtime). Về phía xử lý batch, Apache Spark được sử dụng để làm sạch, tích hợp và trích xuất đặc trưng từ bộ dữ liệu tín dụng phức tạp gồm nhiều bảng quan hệ, phục vụ huấn luyện mô hình XGBoost dự đoán xác suất vỡ nợ của khách hàng (biến mục tiêu *Target* = 1). Về phía xử lý realtime, nhóm thiết kế kiến trúc sử dụng Apache Kafka và Spark Structured Streaming để triển khai pipeline chấm điểm tín dụng trực tuyến, ghi kết quả dự đoán xuống cơ sở dữ liệu quan hệ (PostgreSQL) và hệ quản trị cơ sở dữ liệu phân tán (HBase) phục vụ truy vấn nhanh.

Kết quả cho thấy kiến trúc đề xuất cho phép vừa đảm bảo khả năng xử lý khối lượng dữ liệu lớn trong giai đoạn huấn luyện, vừa đáp ứng được yêu cầu độ trễ thấp trong giai đoạn suy luận mô hình (scoring) thời gian thực. Bài báo đồng thời thảo luận một số vấn đề thực tế trong quá trình triển khai, như hiện tượng trễ xử lý (batch bị chậm so với chu kỳ trigger).

Từ khóa: rủi ro tín dụng, dữ liệu lớn, Apache Spark, Apache Kafka, Spark Structured Streaming, XGBoost, realtime scoring.

1 Giới thiệu

Rủi ro tín dụng là khả năng người đi vay không thực hiện được nghĩa vụ trả nợ theo cam kết. Việc đánh giá chính xác rủi ro này là quan trọng đối với các tổ chức tài chính. Các phương pháp truyền thống thường dựa vào điểm FICO hoặc lịch sử vay nợ ngân hàng, nhưng phương pháp này bỏ sót một lượng lớn khách hàng tiềm năng nhưng chưa có lịch sử tín dụng.

Sự phát triển của các nền tảng dữ liệu lớn (Big Data) và các framework xử lý phân tán như Apache Spark, Apache Kafka, cùng với các thuật toán học máy phi tuyến như XGBoost, cho phép xây dựng các hệ thống chấm điểm tín dụng vừa có độ chính xác cao, vừa đáp ứng được yêu cầu về hiệu năng và khả năng mở rộng.

Dự án được hiện thực trong bài báo này hướng tới các mục tiêu cụ thể sau:

- **Xây dựng pipeline xử lý dữ liệu lớn:** Sử dụng Apache Spark để làm sạch, tích hợp và trích xuất đặc trưng từ bộ dữ liệu tín dụng phức tạp gồm nhiều bảng quan hệ, đảm bảo tính nhất quán và khả năng mở rộng.
- **Phát triển mô hình dự đoán rủi ro tín dụng:** Áp dụng thuật toán XGBoost để xây dựng mô hình dự đoán xác suất vỡ nợ (biến mục tiêu *Target* = 1), tối ưu siêu tham số nhằm nâng cao độ chính xác dự đoán.
- **Thiết kế kiến trúc hệ thống cho cả xử lý batch và realtime:** Đề xuất và triển khai kiến trúc hỗ trợ đồng thời xử lý hàng loạt (batch processing) và xử lý dòng dữ liệu thời gian thực (stream processing) dựa trên Kafka và Spark Structured Streaming, bảo đảm độ trễ thấp, khả năng mở rộng và khả năng tích hợp vào hệ thống nghiệp vụ.

Phần còn lại của bài báo được tổ chức như sau: Mục 2 mô tả bài toán và bộ dữ liệu sử dụng. Mục 3 trình bày kiến trúc hệ thống đề xuất. Mục 4 mô tả chi tiết các bước xử lý dữ liệu và xây dựng đặc trưng trên Apache Spark. Mục 5 trình bày mô hình XGBoost dùng để dự đoán rủi ro tín dụng. Mục 6 trình bày pipeline chấm điểm realtime với Spark Structured Streaming và Kafka. Mục 7 thảo luận kết quả thực nghiệm và đánh giá. Cuối cùng, Mục 8 kết luận và đề xuất hướng phát triển tiếp theo.

2 Mô tả bài toán và dữ liệu

2.1 Bài toán dự đoán rủi ro tín dụng

Bài toán được xem xét là bài toán phân loại nhị phân: dự đoán khả năng vỡ nợ của khách hàng trong tương lai dựa trên thông tin hồ sơ và lịch sử tín dụng, với nhãn:

- *Target* = 1: khách hàng có khả năng vỡ nợ/có vấn đề về nghĩa vụ trả nợ.
- *Target* = 0: khách hàng không vỡ nợ.

Đầu vào của mô hình là vector đặc trưng mô tả khách hàng, có thể bao gồm:

- Thông tin nhân khẩu học: loại hình thu nhập (`NAME_INCOME_TYPE`), loại công việc (`OCCUPATION_TYPE`), tình trạng hôn nhân, số người phụ thuộc, v.v.
- Thông tin về khoản vay: số tiền vay, loại sản phẩm, thời hạn, tỷ lệ giữa số tiền vay và thu nhập, v.v.
- Thông tin lịch sử tín dụng và lịch sử giao dịch, nếu có.

Trong nghiên cứu này, chúng tôi tập trung khai thác ba nhóm đặc trưng chính từ bộ dữ liệu Home Credit Deufault Risk (Kaggle):

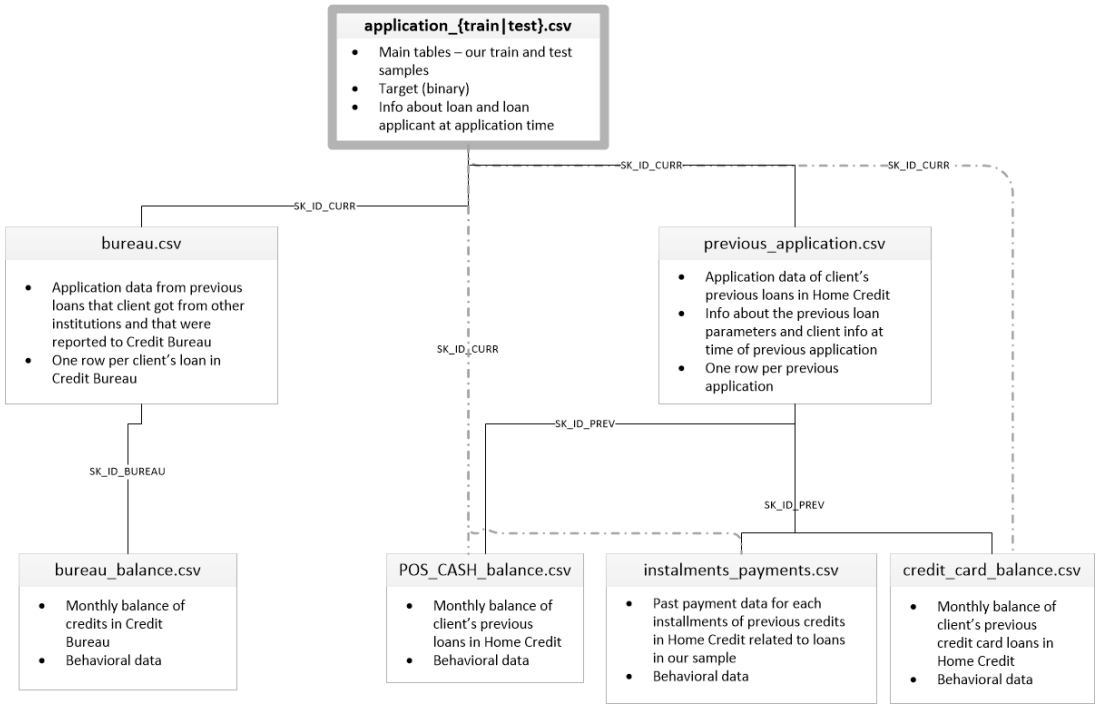
- **Nhóm nhân khẩu học (Demographic Features):** Bao gồm các thông tin cá nhân cơ bản định hình hồ sơ khách hàng như giới tính (`CODE_GENDER`), trình độ học vấn (`NAME_EDUCATION_TYPE`), tình trạng hôn nhân (`NAME_FAMILY_STATUS`), số lượng con cái (`CNT_CHILDREN`), và quyền sở hữu tài sản (xe hơi, nhà đất).
- **Nhóm tài chính và khả năng chi trả (Financial Features):** Các biến số liên quan trực tiếp đến năng lực tài chính như tổng thu nhập (`AMT_INCOME_TOTAL`), số tiền vay (`AMT_CREDIT`), số tiền phải trả hàng năm (`AMT_ANNUITY`), và giá trị tài sản đảm bảo (`AMT_GOODS_PRICE`).
- **Nhóm lịch sử tín dụng và hành vi (Credit History & Behavioral Features):** Đây là nhóm biến quan trọng nhất, bao gồm dữ liệu từ Trung tâm thông tin tín dụng (Credit Bureau) về các khoản vay tại tổ chức khác, lịch sử các khoản vay trước đó tại Home Credit (`previous_application`), hành vi trả nợ thẻ tín dụng (`credit_card_balance`) và lịch sử trả góp (`installments_payments`). Ngoài ra, các biến chuẩn hóa từ nguồn ngoài (`EXT_SOURCE_1`, `EXT_SOURCE_2`, `EXT_SOURCE_3`) đóng vai trò quan trọng trong việc đánh giá điểm tín dụng.

2.2 Bộ dữ liệu và các bảng quan hệ

Bộ dữ liệu Home Credit Default Risk được cung cấp dưới dạng các tập tin CSV (Comma-separated values), bao gồm 7 bảng dữ liệu quan hệ chính được liên kết với nhau qua mã định danh khách hàng (`SK_ID_CURR`) hoặc mã khoản vay (`SK_ID_PREV`):

- **application_train.csv**: Bảng dữ liệu chính chứa thông tin tĩnh của hồ sơ vay hiện tại. Kích thước: 307,511 bản ghi và 122 cột đặc trưng. Đây là bảng chứa nhãn mục tiêu (`TARGET`).
- **bureau.csv**: Thông tin về các khoản vay của khách hàng tại các tổ chức tài chính khác, được báo cáo bởi Credit Bureau. Kích thước: khoảng 1.7 triệu bản ghi.
- **bureau_balance.csv**: Lịch sử trạng thái hàng tháng của các khoản vay trong bảng *bureau*.
- **previous_application.csv**: Lịch sử các đơn vay vốn trước đây của khách hàng tại Home Credit. Kích thước: khoảng 1.6 triệu bản ghi.
- **POS_CASH_balance.csv**: Dữ liệu số dư hàng tháng của các khoản vay mua hàng trả góp (POS) hoặc vay tiền mặt. Kích thước: khoảng 10 triệu bản ghi.

- **installments_payments.csv**: Lịch sử thanh toán thực tế so với kế hoạch của các khoản vay trước đó. Kích thước: hơn 13 triệu bản ghi.
- **credit_card_balance.csv**: Dữ liệu số dư hàng tháng của thẻ tín dụng khách hàng sở hữu.



Hình 1: Sơ đồ quan hệ giữa các bảng dữ liệu trong Home Credit

Bảng 1 trình bày một ví dụ tóm tắt một số biến đặc trưng chính được sử dụng trong mô hình.

Bảng 1: Một số biến đặc trưng chính trong bộ dữ liệu.

Tên biến	Kiểu	Mô tả
`TARGET`	Nhị phân	1 nếu khách hàng gặp khó khăn trả nợ, 0 nếu không.
`EXT_SOURCE_1, 2, 3`	Liên tục	Điểm số chuẩn hóa từ dữ liệu nguồn bên ngoài (Feature quan trọng nhất).
`AMT_ANNUITY`	Liên tục	Số tiền khách hàng phải trả định kỳ cho khoản vay.
`AMT_CREDIT`	Liên tục	Tổng số tiền của khoản vay hiện tại.
`AMT_INCOME_TOTAL`	Liên tục	Tổng thu nhập của khách hàng.
`NAME_EDUCATION_TYPE`	Phân loại	Trình độ học vấn của khách hàng.
`DAYS_BIRTH`	Liên tục	Tuổi khách hàng (tính theo ngày, giá trị âm).
`DAYS_EMPLOYED`	Liên tục	Số ngày làm việc tại công ty hiện tại (giá trị âm).
`CODE_GENDER`	Phân loại	Giới tính của khách hàng (M/F).
`DAYS_LAST_PHONE_CHANGE`	Liên tục	Số ngày kể từ lần cuối khách hàng thay đổi số điện thoại.

2.3 Chỉ tiêu đánh giá

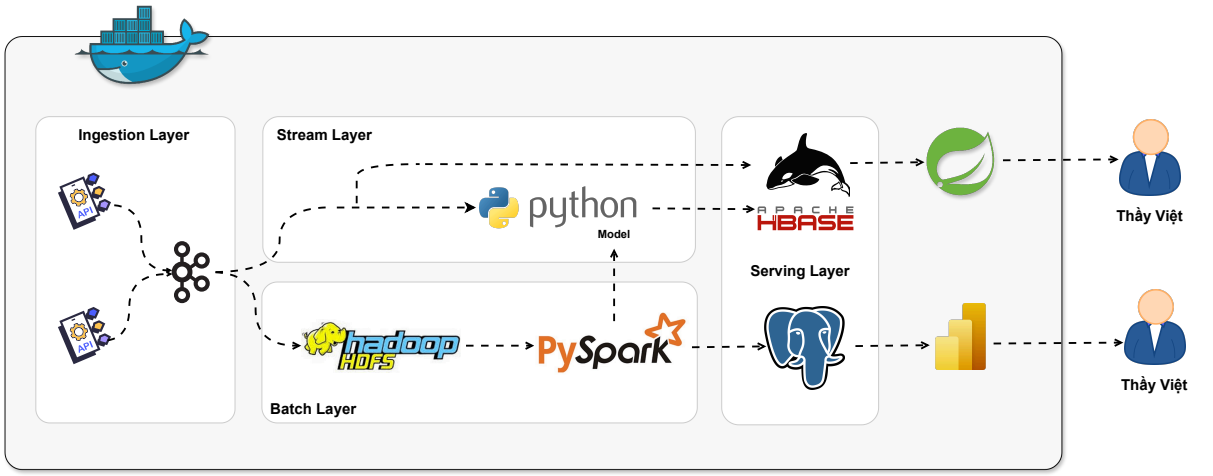
Do bài toán liên quan đến rủi ro tín dụng với lớp dương ($Target = 1$) thường hiếm, các chỉ tiêu đánh giá được quan tâm bao gồm:

- ROC (AUC-ROC).
- Accuracy, độ phủ (Recall) và độ chính xác dương (Precision) cho lớp $Target = 1$.

3 Kiến trúc hệ thống đề xuất

Hệ thống được thiết kế gồm hai luồng xử lý chính:

1. **Luồng batch (offline):** Xử lý dữ liệu lịch sử với Apache Spark để làm sạch, tích hợp, xây dựng đặc trưng, huấn luyện mô hình XGBoost và lưu mô hình đã huấn luyện.
2. **Luồng realtime (online):** Tiếp nhận yêu cầu chấm điểm tín dụng từ ứng dụng qua Kafka, sử dụng Spark Structured Streaming để áp dụng cùng pipeline xử lý đặc trưng và mô hình XGBoost, sau đó ghi kết quả xuống PostgreSQL/HBase.



Hình 2: Kiến trúc tổng thể hệ thống chấm điểm tín dụng batch và realtime.

3.1 Các thành phần chính

- **Apache Spark:** Nền tảng xử lý phân tán dùng cho cả batch (Spark SQL, Spark MLlib) và streaming (Spark Structured Streaming).
- **Apache Kafka:** Hệ thống message broker dùng để tiếp nhận và truyền tải các yêu cầu chấm điểm tín dụng (`credit_application`) theo dạng dòng dữ liệu, (`credit_score`) truyền tải dữ liệu sau khi đi qua model ML .
- **PostgreSQL/HBase:** Lưu trữ kết quả chấm điểm, phục vụ báo cáo và truy vấn nhanh. PostgreSQL phù hợp cho nghiệp vụ giao dịch, HBase phù hợp cho lưu trữ phân tán với khối lượng lớn.
- **Docker / Docker Compose:** Dùng để chứa các dịch vụ như Kafka, Zookeeper, Spark, PostgreSQL, HBase, giúp dễ dàng triển khai và tái lập môi trường.

3.2 Môi trường triển khai

Hệ thống được triển khai trong môi trường container, mỗi dịch vụ (Kafka broker, Zookeeper, Spark master/worker, HBase, PostgreSQL, namenode, datanode ...) chạy trong một container riêng, được điều phối bằng `docker-compose`.

Một ví dụ cấu hình liên quan đến HBase:

- Sử dụng image có sẵn của docker: dajobe/hbase
- Sử dụng 2 cổng: 16010 cho UI, 9090 để ứng dụng kết nối với Hbase.

4 Tiền xử lý và xây dựng đặc trưng với Apache Spark

4.1 Nạp và tích hợp dữ liệu

Các bảng dữ liệu nguồn (CSV hoặc từ hệ quản trị cơ sở dữ liệu) được nạp vào Spark thông qua Spark SQL. Các bước điển hình:

1. Đọc dữ liệu từ nhiều bảng: bảng ứng dụng chính, bảng lịch sử khoản vay, bảng thẻ tín dụng, v.v.
2. Chuẩn hóa kiểu dữ liệu (cast về `IntegerType`, `DoubleType`, `StringType`, ...).
3. Join các bảng theo khóa chung (ví dụ: `SK_ID_CURR` hoặc ID khách hàng) để tạo một bảng đặc trưng tổng hợp ở cấp độ khách hàng/khoản vay.

4.2 Làm sạch dữ liệu và xử lý giá trị thiếu

Trong thực tế, dữ liệu chứa nhiều giá trị thiếu, bất thường (outliers) và các giá trị mã hóa đặc biệt (ví dụ: 365243 ngày cho trường `DAYS_EMPLOYED`). Quy trình xử lý có thể bao gồm:

- Thay thế các giá trị đặc biệt bằng `null` hoặc giá trị hợp lý.
- Loại bỏ các bản ghi không hợp lệ (thiếu quá nhiều thông tin quan trọng).
- Áp dụng chiến lược điền giá trị thiếu (imputation) như dùng median/mean cho biến liên tục, mode cho biến phân loại.

Các thao tác này được thực hiện với Spark DataFrame API, đảm bảo khả năng xử lý dữ liệu lớn.

4.3 Mã hóa biến phân loại và chuẩn hóa biến số

Đối với các biến phân loại như `NAME_INCOME_TYPE`, `OCCUPATION_TYPE`, v.v., pipeline xử lý đặc trưng sử dụng các transformer của Spark MLlib:

- `StringIndexer` để ánh xạ giá trị phân loại sang chỉ số số nguyên.
- `OneHotEncoder` để mã hóa one-hot (nếu cần).
- `VectorAssembler` để gộp tất cả các đặc trưng vào một vector đặc trưng duy nhất.

Một điểm cần lưu ý trong triển khai thực tế là một số cột có thể tồn tại trong pipeline mô hình batch nhưng không xuất hiện trong dữ liệu streaming realtime. Ví dụ log cảnh báo từ Spark Structured Streaming:

```
WARN StringIndexerModel: Input column NAME_TYPE_SUITE does not exist
during transformation. Skip StringIndexerModel for this column.
WARN StringIndexerModel: Input column NAME_INCOME_TYPE does not exist
during transformation. Skip StringIndexerModel for this column.
WARN StringIndexerModel: Input column OCCUPATION_TYPE does not exist
during transformation. Skip StringIndexerModel for this column.
```

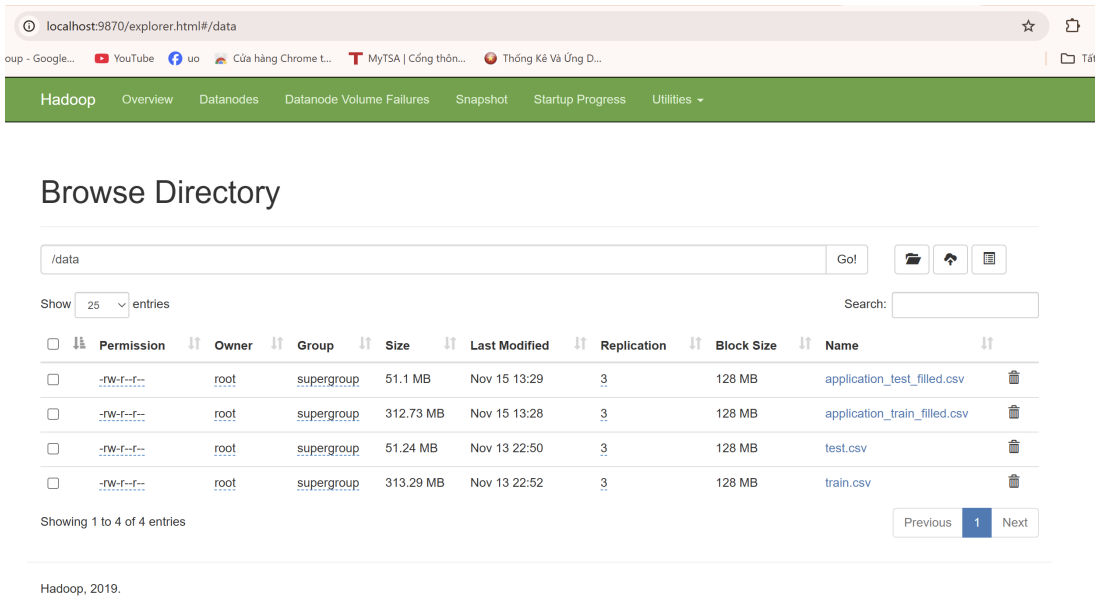
Để xử lý vấn đề này, có thể áp dụng một trong các hướng:

- Đảm bảo schema của dữ liệu realtime đầy đủ các cột như schema batch, với giá trị mặc định nếu không có thông tin.
- Hoặc hiệu chỉnh lại pipeline đặc trưng, loại bỏ các cột không khả dụng trong môi trường realtime.

4.4 Chia tập dữ liệu và lưu pipeline đặc trưng

Sau khi xây dựng xong pipeline xử lý đặc trưng, dữ liệu được chia thành:

- Tập huấn luyện (train set): 80
- Tập kiểm thử (test set) để đánh giá cuối cùng: 20
- Bộ dữ liệu được lưu trữ trên giao diện quản trị HDFS(localhost:9870)



Hình 3: Nơi lưu trữ dataset

Pipeline (gồm các bước `StringIndexer`, `OneHotEncoder`, `VectorAssembler`) được `fit` trên tập huấn luyện và sau đó được lưu lại (save) để tái sử dụng trong giai đoạn huấn luyện mô hình XGBoost và trong pipeline realtime.

5 Xây dựng mô hình XGBoost dự đoán rủi ro tín dụng

5.1 Huấn luyện mô hình

XGBoost là một thuật toán boosting trên cây quyết định, nổi tiếng với khả năng xử lý tốt dữ liệu dạng bảng, hỗ trợ tự động xử lý missing value và có nhiều siêu tham số linh hoạt.

Trong dự án, mô hình XGBoost được cấu hình cho bài toán phân loại nhị phân với các tham số cơ bản như:

- `objective = "binary:logistic"`.
- `eval_metric` (ví dụ: `auc`, `logloss`).
- Các siêu tham số chính: `max_depth`, `learning_rate`, `n_estimators`, `subsample`, `colsample_bytree`, ...

Sau khi huấn luyện, mô hình tốt nhất được lưu lại dưới dạng file (`.model`) lưu trữ trên HDFS namenode để sử dụng trong pipeline realtime.

5.2 Lưu trữ và triển khai mô hình

Mô hình và pipeline đặc trưng được lưu trữ thống nhất để đảm bảo:

- Pipeline đặc trưng trong batch và realtime là đồng nhất.
- Việc thay đổi sang phiên bản mô hình mới có thể thực hiện có kiểm soát (model versioning).

6 Triển khai pipeline chấm điểm realtime với Spark Structured Streaming

6.1 Luồng dữ liệu realtime qua Kafka

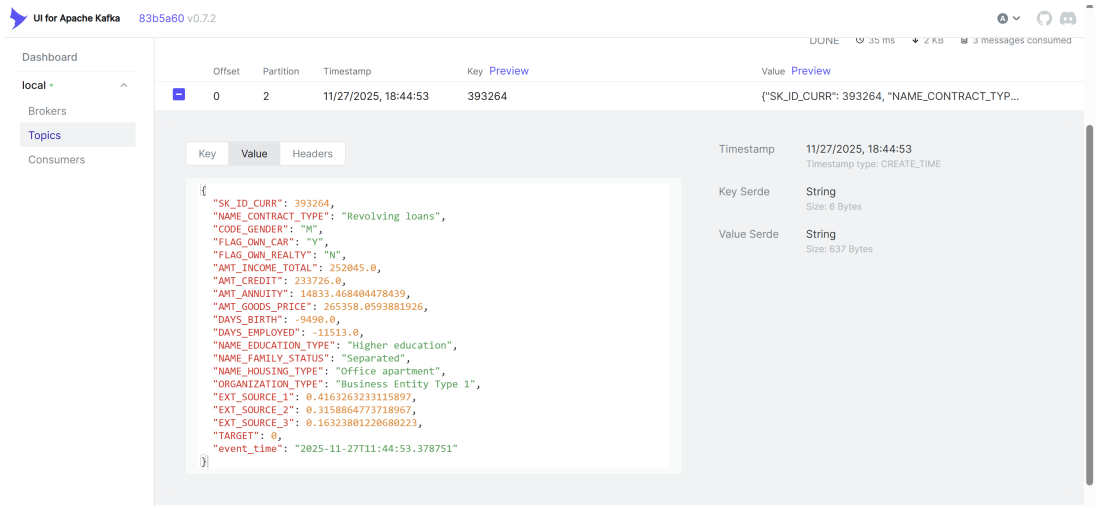
Trong môi trường vận hành, các yêu cầu chấm điểm tín dụng được hệ thống nghiệp vụ (phần mềm front–end hoặc hệ thống lõi) gửi tới một dịch vụ trung gian (REST API). Dịch vụ này đóng vai trò *producer* và đẩy thông điệp vào Kafka topic `credit_application`. Thông điệp (payload) được chuẩn hóa dưới dạng JSON, bao gồm:

- Thông tin định danh yêu cầu (ví dụ: `request_id`, `application_id`);
- Tập các trường đặc trưng đầu vào phục vụ mô hình chấm điểm;
- Thời điểm tạo yêu cầu và một số siêu dữ liệu (metadata) khác nếu cần.

6.2 Job Spark Structured Streaming

Một job Spark Structured Streaming được cấu hình với trigger theo chu kỳ (ví dụ: 5 giây) để:

1. Đọc dữ liệu từ Kafka topic `credit_application`.
2. Parse JSON thành DataFrame với schema đã định nghĩa trước.
3. Áp dụng pipeline đặc trưng (đã lưu từ bước batch).
4. Gọi mô hình XGBoost để dự đoán xác suất vỡ nợ (`probability`) và nhãn dự đoán (`prediction`).
5. Ghi kết quả xuống PostgreSQL hoặc HBase (bao gồm request ID, xác suất, nhãn dự đoán, timestamp xử lý).



Hình 4: Data realtime được gửi qua kafka

6.3 Tối ưu độ trễ và xử lý trùng lặp

Một số điểm tối ưu cần xem xét:

- Tối ưu pipeline xử lý đặc trưng để giảm chi phí tính toán mỗi batch.
- Tối ưu kết nối đến cơ sở dữ liệu đích (PostgreSQL/HBase), tránh ghi từng bản ghi một.
- Điều chỉnh kích thước batch, mức độ song song (số partition), cấu hình resource cho Spark.

Ngoài ra, trong thực tế có thể xảy ra hiện tượng một message gửi lên Kafka nhưng có nhiều lần ghi xuống database (do reprocessing, retry của Spark hoặc do checkpoint). Để giảm thiểu, có thể:

- Thiết kế khóa chính/unique constraint trên bảng kết quả (ví dụ: theo request ID) và thực hiện upsert thay vì insert thuần.
- Kiểm soát logic commit offset và checkpoint của Structured Streaming.

7 Kết quả và đánh giá

7.1 Thiết lập thực nghiệm

Thực nghiệm được triển khai trên cụm Spark chạy trong môi trường Docker, với cấu hình tài nguyên và số lượng node cụ thể.

- **Kafka**
  - 01 container Zookeeper (`confluentinc/cp-zookeeper:7.5.11`), lắng nghe trên cổng 2181.
  - 01 container Kafka broker (`confluentinc/cp-kafka:7.5.11`), expose cổng 9092 cho các dịch vụ trong mạng Docker và cổng 29092 cho client trên máy host.
  - 01 container giao diện quản lý `kafka-ui` (`provectuslabs/kafka-ui:latest`), truy cập qua HTTP tại `localhost:8083`.
- **Tầng ingestion:**

- 01 dịch vụ REST `ingestion-api` (container `ingestion_api`) đóng vai trò producer, nhận yêu cầu chấm điểm từ phía ứng dụng và đẩy vào Kafka topic `credit_applications` qua địa chỉ `kafka:9092`.
- Tham số `DEFAULT_FRAUD_RATE=0.002` được sử dụng để sinh dữ liệu mô phỏng theo tỷ lệ vỡ nợ mong muốn trong giai đoạn thử nghiệm.

- **Hệ thống lưu trữ phân tán HDFS:**
  - 01 NameNode (`bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8`), expose cổng 9000 (RPC) và 9870 (Web UI NameNode).
  - 01 DataNode (`bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8`), expose cổng 9864 (Web UI DataNode).
  - HDFS được cấu hình với địa chỉ `hdfs://namenode:9000`.

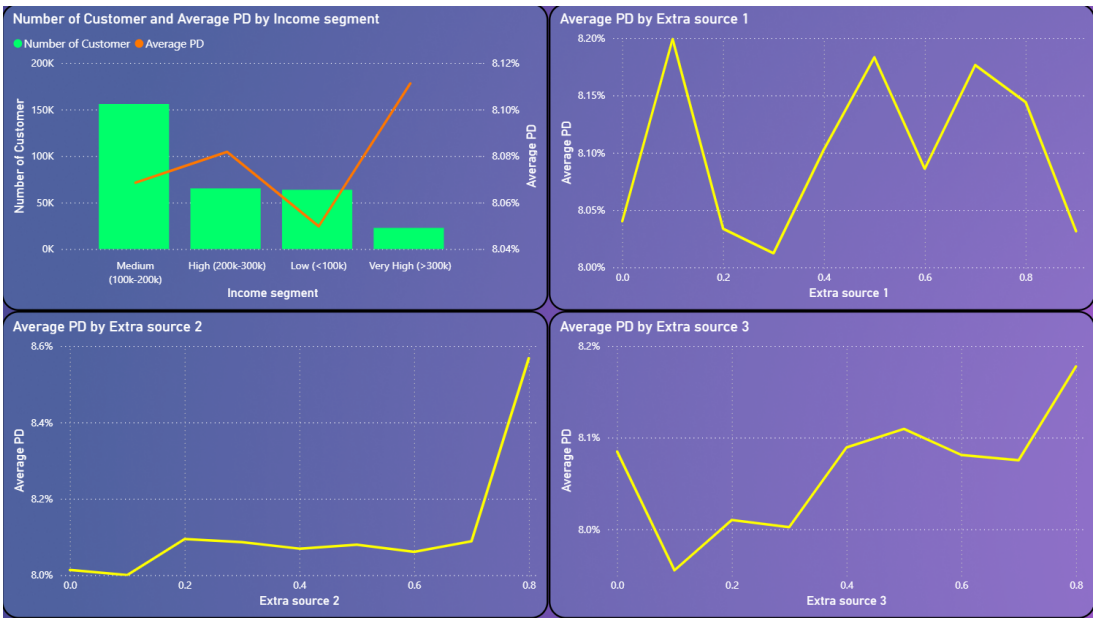
- **Cụm xử lý Spark:**
  - 01 container Spark Master (`spark-master`) chạy tiến trình `org.apache.spark.deploy.master.Mast` ở cổng 7077, web UI tại `http://localhost:8080`.
  - 01 container Spark Worker (`spark-worker`) chạy tiến trình `org.apache.spark.deploy.worker.Work` kết nối tới `spark://spark-master:7077`, web UI tại `http://localhost:8081`.
  - Worker được cấu hình với `SPARK_WORKER_CORES=4` và `SPARK_WORKER_MEMORY=6G`, tương ứng 4 lõi logic và 6 GB RAM dành cho các executor.
  - Bộ nhớ driver được cấu hình `SPARK_DRIVER_MEMORY=4G`.
  - Mã nguồn Spark application được mount vào thư mục `/opt/app`, mô hình đã huấn luyện được mount vào `/opt/model`.
  - Phiên bản Spark sử dụng trong Dockerfile: apache/spark:3.4.1

- **Cơ sở dữ liệu quan hệ (batch view):**
  - 01 container PostgreSQL (`postgres:15`), cơ sở dữ liệu `finrisk`, user `finuser`.
  - 01 container Adminer (`adminer`) làm giao diện web để thao tác với PostgreSQL, truy cập tại `http://localhost:8082`.

- **Cơ sở dữ liệu NoSQL HBase (realtime view):**
  - 01 container HBase (`dajobe/hbase:latest`), web UI HBase Master tại `http://localhost:16010`, dịch vụ Thrift/REST tại cổng 9090.
  - Biến môi trường `HBASE_MANAGES_ZK=true` cho phép HBase tự khởi động một ZooKeeper nội bộ trong container.

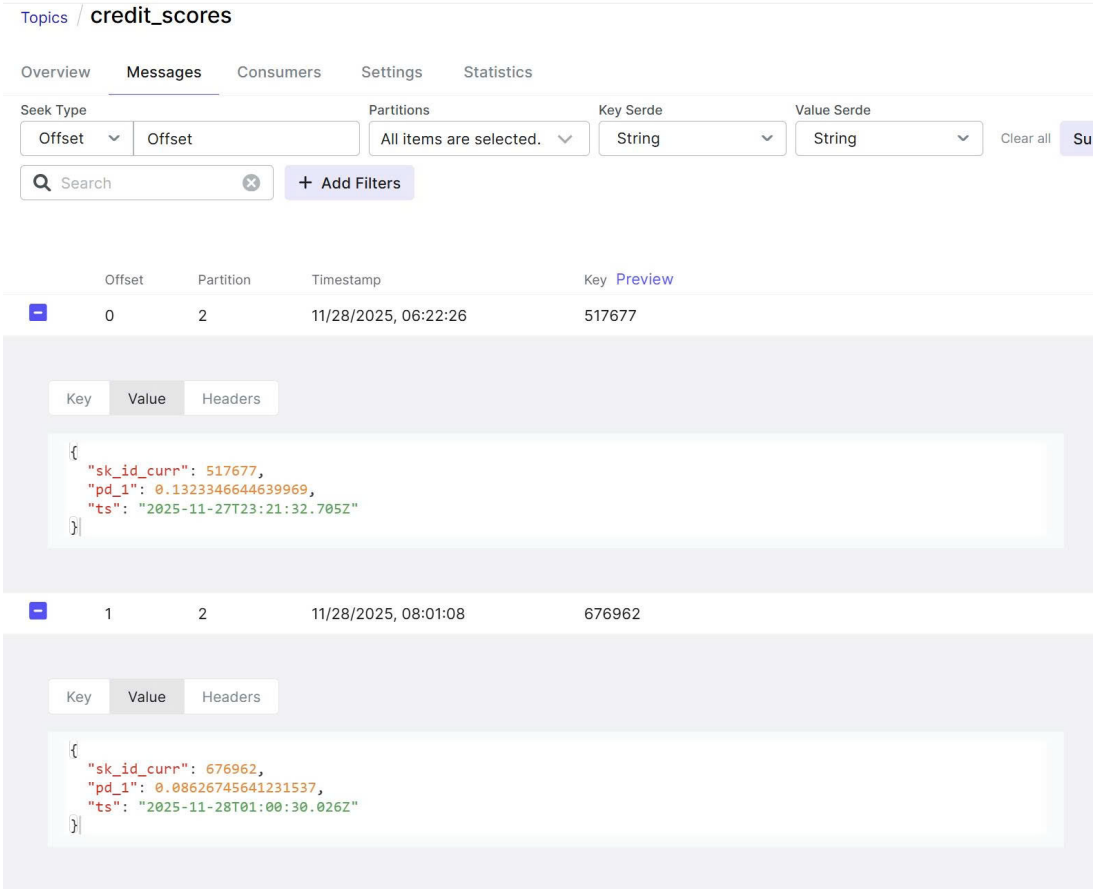
7.2 Kết quả mô hình trên dữ liệu batch

Hình 5: Bảng điều khiển (dashboard) tổng quan hiển thị các chỉ số rủi ro tín dụng.

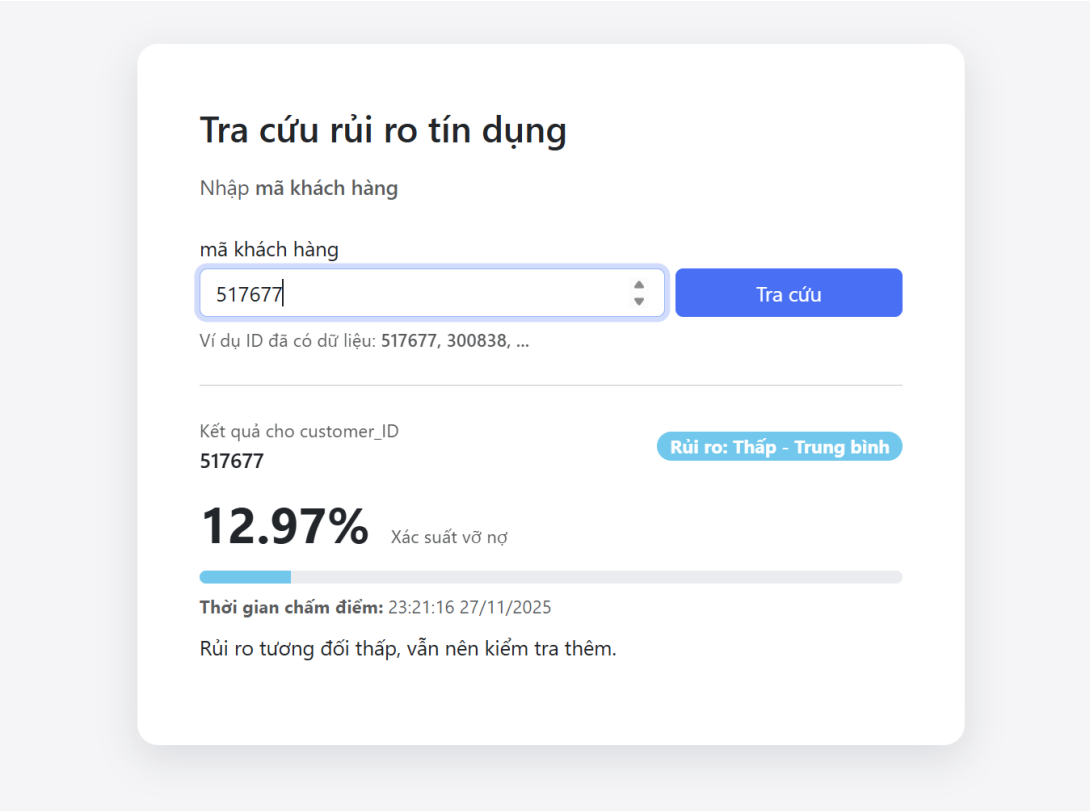


Hình 6: Các biểu đồ chi tiết về kết quả dự báo của mô hình XGBoost

7.3 Kết quả thực nghiệm trên pipeline realtime



Hình 7: Output của mô hình XGBoost sau khi data realtime chảy qua



Hình 8: Giao diện người dùng

8 Kết luận và hướng phát triển

Bài báo đã trình bày một cách có hệ thống quy trình xây dựng hệ thống dự đoán rủi ro tín dụng trên nền tảng dữ liệu lớn, kết hợp xử lý batch và realtime. Các đóng góp chính bao gồm:

- Xây dựng pipeline xử lý dữ liệu lớn với Apache Spark, làm sạch, tích hợp và trích xuất đặc trưng từ nhiều bảng quan hệ.
- Phát triển mô hình XGBoost dự đoán xác suất vỡ nợ, sử dụng các đặc trưng được chuẩn hóa từ pipeline Spark.
- Thiết kế và triển khai kiến trúc xử lý thời gian thực dựa trên Kafka và Spark Structured Streaming, cho phép chấm điểm tín dụng realtime và lưu kết quả xuống PostgreSQL/HBase.
- Phát triển hệ thống theo từng version, cải thiện độ trễ và giao diện thân thiện với người dùng
- Triển khai hệ thống trên nền tảng điều phối container như Kubernetes để nâng cao khả năng mở rộng và tự động hóa.
- Bổ sung hệ thống giám sát chất lượng mô hình (model monitoring), phát hiện drift dữ liệu và cơ chế tự động huấn luyện lại mô hình.

Tài liệu

[1] Apache Spark — A unified analytics engine for large-scale data processing. https://spark.apache.org/.

[2] Apache Kafka — A distributed streaming platform. https://kafka.apache.org/.

[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[4] Anna Montoya, inversion, KirillOdintsov, and Martin Kotek. Home Credit Default Risk. 2018. Kaggle. https://kaggle.com/competitions/home-credit-default-risk