TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO MÔN HỌC
HỌC MÁY

DỰ ĐOÁN KHẢ NĂNG CHẤP THUẬN CẤP THẺ TÍN DỤNG DỰA TRÊN HỒ SƠ ĐĂNG KÍ

Hà Nội, 2025

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO MÔN HỌC
HỌC MÁY

DỰ ĐOÁN KHẢ NĂNG CHẤP THUẬN CẤP THẺ TÍN DỤNG DỰA TRÊN HỒ SƠ ĐĂNG KÍ

Nguyễn Hữu An - 23001493
Nguyễn Thị Quỳnh Anh - 23001496
Đào Thị Ngọc Bích - 23001501

Giảng viên bộ môn:
TS. CAO VĂN CHUNG

Hà Nội, 2025

PHÂN CÔNG NHIỆM VỤ

Công việc	Người phụ trách
Tiền xử lý và giảm chiều dữ liệu, xây dựng mô hình hồi quy	Đào Thị Ngọc Bích
Phân cụm dữ liệu và mô hình MLP	Nguyễn Thị Quỳnh Anh
Tiền xử lý dữ liệu, xây dựng mô hình Gradient Boosting, SVM cho bài toán phân loại	Nguyễn Hữu An
Viết bài báo cáo	Tất cả các thành viên
Chuẩn bị slide	Tất cả các thành viên

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG

MỞ ĐẦU

Trong lĩnh vực tài chính ngân hàng, việc thẩm định và phê duyệt tín dụng là một quy trình nghiệp vụ cốt lõi. Khả năng dự đoán chính xác rủi ro tín dụng—xác định liệu một khách hàng có khả năng không hoàn trả khoản vay hay không—có tầm quan trọng sống còn. Một mô hình dự đoán hiệu quả không chỉ giúp các tổ chức tài chính giảm thiểu tổn thất (nợ xấu) và tối ưu hóa việc phân bổ vốn, mà còn đảm bảo việc tiếp cận tín dụng một cách công bằng và nhanh chóng cho những khách hàng đủ điều kiện.

Với sự phát triển của học máy, các phương pháp phân tích dữ liệu tiên tiến đang dần thay thế các mô hình chấm điểm tín dụng truyền thống, mang lại khả năng dự đoán chính xác hơn bằng cách phân tích các mối quan hệ phức tạp và phi tuyến tính trong dữ liệu khách hàng.

Mục tiêu của dự án này là giải quyết bài toán phân loại nhị phân (binary classification). Dựa trên dữ liệu nhân khẩu học, tài chính và lịch sử tín dụng của khách hàng, chúng ta cần xây dựng và đánh giá các mô hình học máy để dự đoán xem một khách hàng thuộc nhóm:

- **”Rủi ro thấp”**: Khách hàng có khả năng hoàn trả khoản vay.
- **”Rủi ro cao”**: Khách hàng có khả năng không thể thanh toán (vỡ nợ).

Bộ dữ liệu này có tên là Credit Card Approval Prediction, được công bố trên Kaggle. Bộ dữ liệu này gồm hai tệp tin riêng biệt:

- `application_record.csv`: Chứa các thông tin tĩnh của người nộp đơn, ví dụ như giới tính, độ tuổi, thu nhập, tình trạng sở hữu bất động sản, trình độ học vấn, loại hình công việc, v.v.
- `credit_record.csv`: Ghi lại lịch sử trạng thái tín dụng hàng tháng của các khách hàng, cho biết tình trạng thanh toán của họ theo thời gian (ví dụ: đã thanh toán, quá hạn 30 ngày, quá hạn 60 ngày, v.v.).

Với bộ dữ liệu phong phú này, chúng ta có thể huấn luyện các mô hình chính xác, mô hình trở nên khách quan và đáng tin cậy.

Quá trình giải quyết bài toán này, chúng tôi gặp môn thách thức lớn đó là sự mất cân bằng dữ liệu nghiêm trọng. Sau khi xử lý và tạo biến mục tiêu (một khách hàng được coi là ”rủi ro cao” nếu có ít nhất một lần quá hạn trên 60 ngày), phân tích cho thấy bộ dữ liệu mất cân bằng cực đoan: chỉ có 1.69% mẫu thuộc lớp ”Rủi ro cao”, trong khi 98.31% thuộc lớp ”Rủi ro thấp”. Nếu không xử lý, mô hình sẽ bị thiên vị nặng nề, có xu hướng dự đoán mọi trường hợp đều là ”Rủi ro thấp” và đạt độ chính xác cao giả tạo.

Xử lý và tích hợp dữ liệu: Dữ liệu đến từ hai nguồn khác nhau, đòi hỏi phải có sự kết hợp cẩn thận để liên kết lịch sử tín dụng với thông tin cá nhân. Ngoài ra, việc xử lý dữ liệu thiếu

và mã hóa các đặc trưng phân loại thành dạng số là bắt buộc. Song, như được thể hiện qua phân tích PCA sau này, ranh giới quyết định giữa hai lớp là phức tạp và không thể tách biệt một cách tuyến tính. Điều này đòi hỏi các mô hình có khả năng nắm bắt các mối quan hệ phi tuyến.

Để giải quyết bài toán và các thách thức trên, báo cáo này sẽ trình bày một quy trình toàn diện:

- **Chương 1 - Cơ sở lý thuyết**: Nêu ý tưởng, những mô hình dùng để huấn luyện, tối ưu hóa mô hình, hàm mất mát. Làm sạch, kết hợp hai bộ dữ liệu, và thực hiện kỹ thuật đặc trưng để tạo ra biến mục tiêu. Áp dụng kỹ thuật SMOTE (Synthetic Minority Over-sampling Technique) để tạo ra các mẫu tổng hợp cho lớp thiểu số, giúp cân bằng bộ dữ liệu huấn luyện.

- **Chương 2 - Giảm chiều dữ liệu và trực quan hóa**: Phân tích dữ liệu thăm dò để hiểu rõ hơn về sự phân bố của dữ liệu, dữ liệu thiếu, và mối quan hệ giữa các đặc trưng (đặc biệt là sau khi áp dụng PCA).

- **Chương 3 - Xây dựng và Huấn luyện mô hình**: Triển khai một số mô hình học có giám sát (như KNN, Hồi quy Logistic, v.v). Huấn luyện mô hình với dữ liệu có tỷ lệ chia khác nhau và dữ liệu giảm chiều. Sử dụng các số đo như Accuracy, Precision, Recall, F1-Score và Ma trận nhầm lẫn để so sánh hiệu suất, rút ra kết luận về mô hình hoạt động tốt nhất cho bài toán này.

Dựa vào kết quả của bài toán phân loại, chúng tôi đưa ra giải pháp hợp lý nhằm tự động hóa quy trình phê duyệt thẻ tín dụng. Ngân hàng có thể phát hiện chính xác hơn những khách hàng ”Rủi ro cao”, từ chối cấp thẻ cho nhóm này sẽ trực tiếp giảm tỷ lệ nợ xấu và giảm tổn thất tài chính cho tổ chức. Trong thực tế, các sự kiện rủi ro luôn hiếm. Bài toán này chứng minh rằng bằng cách sử dụng các kỹ thuật như SMOTE, chúng ta vẫn có thể xây dựng các mô hình hiệu quả, thay vì các mô hình ”ngây thơ” chỉ đoán ”Rủi ro thấp” cho tất cả mọi người.

Chương 1
CƠ SỞ LÝ THUYẾT

1.1 Ý tưởng thực hiện

Mục tiêu cốt lõi của đồ án này là xây dựng một hệ thống hỗ trợ ra quyết định tín dụng tự động dựa trên hồ sơ đăng ký và lịch sử tín dụng. Về mặt kỹ thuật, đây là bài toán phân loại nhị phân nhằm dự đoán khả năng trả nợ của khách hàng. Mô hình sẽ xử lý hai nguồn dữ liệu chính:

- `application_record.csv`: Chứa các đặc trưng tĩnh về nhân khẩu học và tài chính (tuổi, giới tính, thu nhập, sở hữu tài sản, trình độ học vấn, công việc...).
- `credit_record.csv`: Chuỗi thời gian ghi lại lịch sử thanh toán nợ hàng tháng, dùng để gán nhãn uy tín.

Bài toán được định nghĩa là phân loại khách hàng vào một trong hai nhóm:

- **Lớp 0 (An toàn)**: Khách hàng có khả năng thanh toán tốt, đủ điều kiện cấp tín dụng.
- **Lớp 1 (Rủi ro)**: Khách hàng có nguy cơ nợ xấu cao, từ chối cấp tín dụng.

Do dữ liệu có số lượng thuộc tính lớn và phức tạp, quy trình thực hiện đồ án được triển khai theo các bước phân tích sâu như sau:

1.1.1 Giảm chiều dữ liệu và Trực quan hóa

Để xử lý vấn đề về số chiều lớn (curse of dimensionality) và có cái nhìn trực quan về phân bố dữ liệu, chúng tôi áp dụng hai kỹ thuật:

- **PCA (Principal Component Analysis)**: Kỹ thuật không giám sát giúp giảm số chiều nhưng vẫn bảo toàn tối đa phương sai của dữ liệu gốc, hỗ trợ đánh giá tổng quan cấu trúc không gian dữ liệu.
- **LDA (Linear Discriminant Analysis)**: Kỹ thuật có giám sát nhằm tìm ra các trục chiếu giúp tối đa hóa khoảng cách giữa hai lớp (Rủi ro/An toàn), hỗ trợ việc phân tách dữ liệu tốt hơn.

1.1.2 Phân cụm dữ liệu (Clustering)

Trước khi đi vào phân loại có giám sát, chúng tôi thực hiện phân cụm để khám phá các cấu trúc ngầm định và nhóm khách hàng tự nhiên trong dữ liệu mà không cần nhãn:

- **DBSCAN**: Thuật toán phân cụm dựa trên mật độ không gian. Phương pháp này không yêu cầu xác định trước số lượng cụm và đặc biệt hiệu quả trong việc phát hiện các điểm dữ liệu nhiễu (noise/outliers) – những hồ sơ có tính chất bất thường so với đa số.
- **Gaussian Mixture Model (GMM)**: Mô hình xác suất giả định rằng các điểm dữ liệu được sinh ra từ hỗn hợp của các phân phối chuẩn, cho phép phân cụm mềm dẻo hơn đối với các hình dạng dữ liệu phức tạp.

1.1.3 Mô hình Phân loại (Classification)

Chúng tôi xây dựng, huấn luyện và so sánh hiệu năng của các mô hình học máy tiên tiến đại diện cho các hướng tiếp cận khác nhau:

- **Gradient Boosting**: Phương pháp học kết hợp (Ensemble Learning) mạnh mẽ, xây dựng mô hình dự đoán dưới dạng tập hợp các cây quyết định yếu theo tuần tự.
- **Multi-Layer Perceptron (MLP)**: Mạng nơ-ron nhân tạo với khả năng học các đặc trưng phi tuyến tính phức tạp thông qua các lớp ẩn.
- **SVM (Support Vector Machine) với Kernel RBF**: Sử dụng hàm nhân Radial Basis Function để ánh xạ dữ liệu sang không gian chiều cao hơn, giải quyết tốt bài toán phân lớp phi tuyến.

1.1.4 Mở rộng bài toán Hồi quy (Regression)

Bên cạnh phân loại nhị phân cứng nhắc (0 hoặc 1), chúng tôi mở rộng bài toán sang dạng hồi quy để ước lượng ”điểm tín dụng” hoặc xác suất rủi ro cụ thể:

- Sử dụng giá trị hàm quyết định (decision function) hoặc xác suất dự đoán từ mô hình **Gradient Boosting** làm biến mục tiêu mới.
- Thực hiện bài toán hồi quy để dự báo mức độ rủi ro liên tục, giúp tổ chức tín dụng có cái nhìn chi tiết hơn về ngưỡng chấp nhận rủi ro.

1.2 Phân tích thành phần chính (PCA)

1.2.1 Tổng quan

Phân tích thành phần chính (Principal Component Analysis - PCA) là một kỹ thuật giảm chiều dữ liệu tuyến tính không giám sát phổ biến trong học máy và thống kê. Về mặt hình học, PCA thực hiện một phép biến đổi trực giao để chuyển đổi hệ tọa độ của dữ liệu gốc sang một hệ tọa độ mới. Trong không gian mới này, các trục tọa độ được gọi là các *thành phần chính* (Principal Components), được xác định sao cho phương sai của dữ liệu khi chiếu lên các trục này là lớn nhất.

Mục tiêu của PCA là tìm ra không gian con có số chiều thấp hơn ($k < d$) mà vẫn bảo toàn được lượng thông tin (phương sai) lớn nhất có thể của dữ liệu ban đầu, đồng thời loại bỏ nhiễu và sự đa cộng tuyến (multicollinearity) giữa các đặc trưng.

1.2.2 Cơ sở toán học và Thuật toán

Giả sử tập dữ liệu đầu vào là ma trận X có kích thước $n \times d$, trong đó n là số lượng mẫu và d là số lượng đặc trưng ban đầu. Quy trình thực hiện PCA bao gồm các bước sau:

Bước 1: Chuẩn hóa dữ liệu (Data Standardization)

PCA rất nhạy cảm với sự chênh lệch về độ lớn (scale) của các biến. Do đó, bước đầu tiên là chuẩn hóa dữ liệu để đưa các đặc trưng về cùng một miền giá trị (thường là Z-score normalization):

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

Trong đó:

- $\bar{x}_j$ là giá trị trung bình của đặc trưng thứ j.
- σ_j là độ lệch chuẩn của đặc trưng thứ j.

Sau bước này, ma trận dữ liệu đã chuẩn hóa X' sẽ có kỳ vọng bằng 0 và phương sai bằng 1 cho mỗi cột.

Bước 2: Tính ma trận hiệp phương sai (Covariance Matrix)

Để xác định mối quan hệ tuyến tính giữa các đặc trưng, ta tính ma trận hiệp phương sai Σ (hoặc C) kích thước $d \times d$:

$$\Sigma = \frac{1}{n-1}(X')^T X'$$

Phần tử Σ_{ij} biểu thị mức độ tương quan giữa đặc trưng thứ i và thứ j. Nếu Σ là ma trận chéo, dữ liệu hoàn toàn không có tương quan. Mục đích của PCA là chéo hóa ma trận này trong không gian mới.

Bước 3: Phân rã trị riêng (Eigendecomposition)

Ta tìm các vector riêng (eigenvectors) và trị riêng (eigenvalues) của ma trận Σ bằng cách giải phương trình đặc trưng:

$$\det(\Sigma - \lambda I) = 0$$

Với mỗi nghiệm λ, ta tìm được vector v thỏa mãn: $\Sigma v = \lambda v$.

- **Vector riêng (v)**: Xác định hướng của trục mới (thành phần chính).
- **Trị riêng (λ)**: Đại diện cho độ lớn phương sai của dữ liệu khi chiếu lên trục tương ứng. λ càng lớn, trục đó càng chứa nhiều thông tin.

Bước 4: Sắp xếp và Chọn lọc thành phần chính

Sắp xếp các trị riêng theo thứ tự giảm dần: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d \geq 0$. Tương ứng với đó là các vector riêng $v_1, v_2, ..., v_d$.

Để giảm chiều dữ liệu từ d xuống k, ta chọn k vector riêng đầu tiên ứng với k trị riêng lớn nhất để tạo thành ma trận chiếu W_k kích thước $d \times k$:

$$W_k = [v_1, v_2, ..., v_k]$$

Tỷ lệ phương sai được giữ lại (Explained Variance Ratio) của thành phần thứ i được tính bằng:

$$\text{Explained Variance}_i = \Sigma_j = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

Tổng phương sai tích lũy khi chọn k thành phần giúp đánh giá lượng thông tin được bảo toàn.

Bước 5: Chiếu dữ liệu sang không gian mới

Dữ liệu mới Y (kích thước $n \times k$) được tính bằng phép nhân ma trận giữa dữ liệu gốc đã chuẩn hóa và ma trận chiếu:

$$Y = X'.W_k$$

Lúc này, các cột của Y là các đặc trưng mới độc lập tuyến tính với nhau.

1.2.3 Ưu điểm và Hạn chế

Ưu điểm:

- **Giảm chiều dữ liệu hiệu quả:** Loại bỏ sự dư thừa, giúp giảm chi phí tính toán và không gian lưu trữ, đồng thời tăng tốc độ huấn luyện mô hình.
- **Khử tương quan (Decorrelation):** Các thành phần chính hoàn toàn độc lập tuyến tính, giúp loại bỏ vấn đề đa cộng tuyến trong các mô hình hồi quy.
- **Hỗ trợ trực quan hóa:** Giúp biểu diễn dữ liệu nhiều chiều trên không gian 2D hoặc 3D để phát hiện các cụm (clusters) hoặc xu hướng phân bố.
- **Giảm nhiễu:** Bằng cách loại bỏ các thành phần có trị riêng nhỏ (thường đại diện cho nhiễu), PCA giúp cải thiện tín hiệu của dữ liệu.

Hạn chế:

- **Giả định tuyến tính:** PCA chỉ hiệu quả khi các biến có mối quan hệ tuyến tính. Đối với dữ liệu có cấu trúc phi tuyến phức tạp (ví dụ: hình xoắn ốc), PCA không thể ”trải phẳng” dữ liệu hiệu quả (cần dùng t-SNE hoặc Kernel PCA).
- **Mất khả năng diễn giải (Interpretability):** Các thành phần chính là tổ hợp tuyến tính của các đặc trưng gốc, do đó rất khó để giải thích ý nghĩa vật lý của chúng (ví dụ: một trục mới là $0.5 \times$ Tuổi $- 0.3 \times$ Thu nhập rất khó định nghĩa).
- **Nhạy cảm với ngoại lai (Outliers):** Do dựa trên phương sai và độ lệch chuẩn, các giá trị ngoại lai có thể làm sai lệch hướng của các thành phần chính.
- **Không tối ưu cho phân loại:** PCA là thuật toán không giám sát (unsupervised), nó tối ưu hóa phương sai chung chứ không tối ưu hóa khả năng phân tách giữa các lớp (class separability).

1.3 Phân tích phân biệt tuyến tính (Linear Discriminant Analysis - LDA)

1.3.1 Tổng quan và Mục tiêu

Phân tích Phân biệt Tuyến tính (LDA) là một kỹ thuật giảm chiều dữ liệu và phân loại có giám sát (supervised learning). Nếu như PCA là thuật toán không giám sát nhằm tìm kiếm các hướng có phương sai lớn nhất để biểu diễn dữ liệu, thì LDA lại tập trung vào việc tìm kiếm không gian con sao cho khả năng *phân tách giữa các lớp* (class separability) là tốt nhất.

Về mặt hình học, LDA tìm kiếm một phép chiếu tuyến tính (linear projection) dữ liệu từ không gian d chiều xuống không gian k chiều ($k < d$), sao cho trên không gian mới:

1. Khoảng cách giữa các trung tâm (mean) của các lớp là lớn nhất.

2. Độ phân tán (variance) của dữ liệu trong cùng một lớp là nhỏ nhất.

Điều này đảm bảo các mẫu dữ liệu cùng lớp sẽ co cụm lại gần nhau, trong khi các cụm dữ liệu khác lớp sẽ tách rời xa nhau.

1.3.2 Cơ sở toán học

Giả sử bài toán phân loại có C lớp, với N mẫu dữ liệu trong không gian d chiều. Ký hiệu N_i là số lượng mẫu thuộc lớp i, và tập dữ liệu của lớp i là D_i.

Các đại lượng thống kê cơ bản

Trước hết, ta xác định vector trung bình của lớp i (μ_i) và vector trung bình toàn cục (μ):

$$\mu_i = \frac{1}{N_i} \sum_{x \in D_i} x$$

$$\mu = \frac{1}{N} \sum_{i=1}^{C} N_i \mu_i$$

Ma trận phân tán (Scatter Matrices)

LDA xây dựng dựa trên hai ma trận quan trọng:

1. Ma trận phân tán trong lớp (Within-class scatter matrix - S_W): Đại diện cho mức độ phân tán của các điểm dữ liệu xung quanh trung bình của lớp đó. Mục tiêu của LDA là *tối thiểu hóa* đại lượng này.

$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T$$

2. Ma trận phân tán giữa các lớp (Between-class scatter matrix - S_B): Đại diện cho khoảng cách giữa các vector trung bình của các lớp so với trung bình toàn cục. Mục tiêu của LDA là *tối đa hóa* đại lượng này.

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Tiêu chuẩn Fisher (Fisher Criterion)

Để tìm hướng chiếu tối ưu w, LDA tối đa hóa hàm mục tiêu $J(w)$, được định nghĩa là tỷ số giữa phương sai giữa các lớp và phương sai trong lớp sau khi chiếu:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Việc tối đa hóa $J(w)$ dẫn đến bài toán trị riêng tổng quát (Generalized Eigenvalue Problem):

$$S_B w = \lambda S_W w$$

Nếu ma trận S_W khả nghịch, phương trình trở thành:

$$(S_W^{-1} S_B) w = \lambda w$$

Như vậy, hướng chiếu tối ưu w chính là các vector riêng ứng với các trị riêng lớn nhất của ma trận $S_W^{-1} S_B$.

1.3.3 Quy trình thực hiện thuật toán LDA

Quy trình tính toán để giảm chiều dữ liệu bằng LDA bao gồm các bước sau:

1. **Chuẩn hóa dữ liệu:** Mặc dù không bắt buộc nghiêm ngặt như PCA, việc đưa dữ liệu về cùng tỷ lệ (standardization) giúp thuật toán ổn định hơn.
2. **Tính toán vector trung bình:** Tính μ_i cho từng lớp và μ cho toàn bộ tập dữ liệu.
3. **Tính toán các ma trận phân tán:** Xây dựng ma trận S_W và S_B dựa trên công thức đã nêu.
4. **Phân rã trị riêng:** Tìm trị riêng λ và vector riêng v của ma trận $A = S_W^{-1} S_B$.
5. **Sắp xếp và chọn lọc:** Sắp xếp các vector riêng theo thứ tự giảm dần của trị riêng. Chọn k vector riêng đầu tiên tương ứng với k trị riêng lớn nhất để tạo thành ma trận chiếu $W = [v_1, v_2, \ldots, v_k]$.
6. **Chiếu dữ liệu:** Biến đổi dữ liệu gốc X sang không gian mới Y:

$$Y = X \cdot W$$

Lưu ý: Số chiều tối đa của không gian mới trong LDA bị giới hạn bởi $C - 1$ (với C là số lớp), do hạng của ma trận S_B tối đa là $C - 1$.

1.3.4 Đánh giá ưu điểm và hạn chế

Ưu điểm:

- **Hiệu quả phân lớp:** Do tận dụng thông tin nhãn lớp, LDA thường hoạt động tốt hơn PCA trong các bài toán mà sự biến thiên dữ liệu (variance) không đồng nhất với tính phân biệt (separability).
- **Giảm chiều mạnh mẽ:** Có thể giảm số chiều dữ liệu xuống rất thấp (ví dụ: bài toán 2 lớp sẽ giảm về 1 chiều) mà vẫn giữ được thông tin quan trọng nhất để phân loại.

Hạn chế:

- **Giới hạn số chiều:** Không thể giảm chiều về số lượng lớn hơn $C - 1$. Nếu số lớp ít nhưng số đặc trưng cần giữ lại nhiều, LDA không khả thi.
- **Giả định phân phối chuẩn:** LDA hoạt động tối ưu khi dữ liệu tuân theo phân phối chuẩn (Gaussian) và các lớp có ma trận hiệp phương sai giống nhau (Homoscedasticity). Nếu dữ liệu bị lệch hoặc có outlier, hiệu quả sẽ giảm.
- **Vấn đề tuyến tính:** LDA tạo ra ranh giới quyết định tuyến tính. Đối với dữ liệu phi tuyến phức tạp, LDA có thể không tách biệt được các lớp một cách hiệu quả (cần sử dụng Kernel LDA).

1.4 Phân cụm dữ liệu với Mô hình Hỗn hợp Gaussian (GMM)

1.4.1 Tổng quan

Mô hình Hỗn hợp Gaussian (Gaussian Mixture Model - GMM) là một mô hình xác suất được sử dụng phổ biến trong các bài toán phân cụm dữ liệu và ước lượng mật độ. Khác với K-Means là thuật toán phân cụm ”cứng” (hard clustering) – nơi mỗi điểm dữ liệu chỉ thuộc về duy nhất một cụm, GMM là thuật toán phân cụm ”mềm” (soft clustering).

Trong GMM, mỗi điểm dữ liệu được giả định là được sinh ra từ một hỗn hợp của hữu hạn các phân phối chuẩn (Gaussian distributions) với các tham số chưa biết. Mục tiêu của thuật toán là ước lượng các tham số này để mô tả tốt nhất cấu trúc của dữ liệu.

1.4.2 Mô hình Toán học

Giả sử tập dữ liệu $X = \{x_1, x_2, \ldots, x_N\}$ gồm N điểm dữ liệu trong không gian d chiều. GMM giả định rằng phân phối xác suất của X là tổng trọng số của K thành phần Gaussian:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Trong đó:

- K: Số lượng cụm (thành phần).
- π_k: Hệ số trộn (mixing coefficient) của thành phần thứ k. Hệ số này đóng vai trò là xác suất tiên nghiệm (prior probability) để một điểm dữ liệu thuộc về cụm k. Điều kiện ràng buộc là:
$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{và} \quad 0 \leq \pi_k \leq 1$$
- $\mathcal{N}(x|\mu_k, \Sigma_k)$: Hàm mật độ xác suất của phân phối chuẩn đa biến cho thành phần thứ k, được định nghĩa bởi:
$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right)$$
Với μ_k là vector trung bình và Σ_k là ma trận hiệp phương sai của cụm k.

1.4.3 Ước lượng tham số với thuật toán EM

Việc tìm các tham số tối ưu (π, μ, Σ) trực tiếp thông qua phương pháp Ước lượng hợp lý cực đại (Maximum Likelihood Estimation - MLE) là rất khó khăn do sự xuất hiện của biến ẩn (latent variable). Do đó, GMM sử dụng thuật toán **Cực đại hóa Kỳ vọng (Expectation-Maximization - EM)**.

Thuật toán EM là một quy trình lặp bao gồm hai bước chính (E-step và M-step) nhằm tối đa hóa hàm log-likelihood:

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)\right)$$

Bước 1: Khởi tạo (Initialization)

Khởi tạo ngẫu nhiên các tham số μ_k, Σ_k, π_k (thường sử dụng kết quả của K-Means để khởi tạo μ_k).

Bước 2: Bước Kỳ vọng (E-Step)

Tính toán *trách nhiệm* (responsibility) $\gamma(z_{nk})$, là xác suất hậu nghiệm (posterior probability) để điểm dữ liệu x_n thuộc về cụm k, dựa trên các tham số hiện tại:

$$\gamma(z_{nk}) = p(k|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

Bước 3: Bước Cực đại hóa (M-Step)

Cập nhật lại các tham số mô hình dựa trên các giá trị trách nhiệm vừa tính được ở bước E:

1. Tính tổng trách nhiệm của cụm k (số lượng điểm "mềm" trong cụm):

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

2. Cập nhật vector trung bình mới:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

3. Cập nhật ma trận hiệp phương sai mới:

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

4. Cập nhật hệ số trộn mới:

$$\pi_k^{new} = \frac{N_k}{N}$$

Bước 4: Kiểm tra hội tụ

Lặp lại bước E và bước M cho đến khi sự thay đổi của hàm log-likelihood nhỏ hơn một ngưỡng ϵ cho trước, hoặc đạt số vòng lặp tối đa.

1.4.4 Ưu điểm và Hạn chế

Ưu điểm:

- **Tính linh hoạt về hình dạng:** Nhờ sử dụng ma trận hiệp phương sai Σ, GMM có thể mô hình hóa các cụm có hình dạng elip với kích thước và hướng xoay khác nhau, khắc phục hạn chế chỉ tạo ra cụm hình cầu của K-Means.
- **Phân cụm mềm:** Cung cấp xác suất thuộc về từng cụm cho mỗi điểm dữ liệu, giúp xử lý tốt hơn các điểm nằm ở vùng biên giới hoặc các cụm chồng lấn lên nhau (overlapping clusters).

Hạn chế:

- **Nhạy cảm với khởi tạo:** Thuật toán EM có thể hội tụ tại cực trị địa phương (local optima), do đó kết quả phụ thuộc vào giá trị khởi tạo ban đầu.

- **Chi phí tính toán cao:** Việc tính toán ma trận nghịch đảo và định thức trong mỗi vòng lặp tốn kém tài nguyên hơn nhiều so với K-Means, đặc biệt với dữ liệu nhiều chiều.
- **Khó khăn xác định số cụm K:** Tương tự K-Means, GMM yêu cầu xác định trước số cụm. Thường phải dùng các tiêu chuẩn thông tin như AIC (Akaike Information Criterion) hoặc BIC (Bayesian Information Criterion) để chọn K phù hợp.

1.5 Phân cụm dữ liệu dựa trên mật độ (DBSCAN)

1.5.1 Tổng quan

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán phân cụm không giám sát dựa trên mật độ. Khác với K-Means (dựa trên khoảng cách tới tâm) hay GMM (dựa trên phân phối xác suất), DBSCAN định nghĩa một cụm là một vùng không gian có mật độ điểm dữ liệu cao, được ngăn cách với các cụm khác bởi các vùng có mật độ thấp.

Cách tiếp cận này cho phép DBSCAN giải quyết hai vấn đề lớn mà các thuật toán truyền thống thường gặp khó khăn:

1. Phát hiện các cụm có hình dạng bất kỳ (phi tuyến tính, hình lưỡi liềm, hình bao nhau...).
2. Nhận diện và loại bỏ các điểm nhiễu (outliers) một cách tự động.

1.5.2 Các tham số và Định nghĩa cơ bản

DBSCAN hoạt động dựa trên hai tham số chính cần được xác định trước:

- ε (Epsilon): Bán kính của vùng lân cận xung quanh một điểm dữ liệu.
- MinPts: Số lượng điểm tối thiểu nằm trong bán kính ε để vùng đó được coi là ”đậm đặc” (dense).

Dựa trên hai tham số này, DBSCAN phân loại các điểm dữ liệu thành ba loại:

1. **Điểm lõi (Core Point):** Một điểm p được gọi là điểm lõi nếu trong vùng bán kính ε của nó có ít nhất MinPts điểm (bao gồm cả chính nó).

   $$|N_\varepsilon(p)| \geq \text{MinPts}$$

   Trong đó $N_\varepsilon(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\}$ là tập lân cận của p.

2. **Điểm biên (Border Point):** Một điểm q không phải là điểm lõi (có ít hơn MinPts trong lân cận) nhưng nằm trong vùng lân cận của một điểm lõi p.

3. **Điểm nhiễu (Noise Point / Outlier):** Một điểm không phải là điểm lõi cũng không phải là điểm biên. Đây là các điểm nằm ở vùng thưa thớt.

1.5.3 Khái niệm về Tính liên thông mật độ

Để hình thành nên một cụm, DBSCAN sử dụng các khái niệm về khả năng "với tới" (reachability):

- **Mật độ trực tiếp (Directly Density-Reachable):** Điểm q được gọi là đến được trực tiếp từ p nếu p là điểm lõi và $q \in N_\varepsilon(p)$.
- **Mật độ khả đạt (Density-Reachable):** Điểm q được gọi là đến được từ p nếu tồn tại một chuỗi các điểm $p_1, p_2, \ldots, p_n$ với $p_1 = p$ và $p_n = q$ sao cho p_{i+1} đến được trực tiếp từ p_i.
- **Liên thông mật độ (Density-Connected):** Hai điểm p và q được gọi là liên thông mật độ nếu tồn tại một điểm o sao cho cả p và q đều đến được từ o theo nghĩa mật độ khả đạt.

Định nghĩa Cụm (Cluster): Một cụm C là tập hợp con khác rỗng của tập dữ liệu thỏa mãn hai điều kiện:

1. *Tính tối đại (Maximality):* Với mọi p, q, nếu $p \in C$ và q đến được từ p thì $q \in C$.
2. *Tính liên thông (Connectivity):* Với mọi $p, q \in C$, p và q phải liên thông mật độ với nhau.

1.5.4 Quy trình thuật toán

Thuật toán DBSCAN thực hiện theo các bước sau:

1. Khởi tạo tất cả các điểm dữ liệu với trạng thái "chưa thăm" (unvisited).
2. Chọn ngẫu nhiên một điểm p chưa thăm.
3. Nếu p là điểm nhiễu (số láng giềng < MinPts), đánh dấu là "đã thăm" và chuyển sang điểm tiếp theo (Lưu ý: điểm này sau này có thể trở thành điểm biên nếu nó thuộc lân cận của một điểm lõi khác).
4. Nếu p là điểm lõi, tạo một cụm mới và bắt đầu mở rộng cụm:
    - Thêm tất cả các điểm trong $N_\varepsilon(p)$ vào cụm.
    - Với mỗi điểm q trong $N_\varepsilon(p)$, nếu q cũng là điểm lõi, tiếp tục thêm các láng giềng của q vào cụm.

- Quá trình lặp lại cho đến khi không còn điểm nào có thể thêm vào cụm (liên thông mật độ hoàn tất).

5. Lặp lại quy trình cho đến khi tất cả các điểm đều đã được thăm.

1.5.5 Ưu điểm và Hạn chế

Ưu điểm:

- **Không cần giả định số cụm:** Không cần chọn trước số lượng cụm K như K-Means hay GMM.
- **Hình dạng linh hoạt:** Có thể phát hiện các cụm có hình dạng phức tạp (ví dụ: các vòng tròn lồng nhau) mà các thuật toán dựa trên khoảng cách trung bình không làm được.
- **Xử lý nhiễu tốt:** Tự động loại bỏ các điểm ngoại lai, giúp mô hình sạch và chính xác hơn.

Hạn chế:

- **Độ phức tạp tính toán:** Trong trường hợp xấu nhất không có chỉ mục không gian (spatial indexing), độ phức tạp là $O(N^2)$. Nếu dùng k-d tree hoặc R-tree, có thể giảm xuống $O(N \log N)$.
- **Khó khăn với mật độ biến thiên:** DBSCAN hoạt động kém khi các cụm có mật độ khác nhau quá lớn, vì khó chọn được một cặp $(\varepsilon, \text{MinPts})$ phù hợp cho tất cả các cụm.
- **Dữ liệu cao chiều:** Trong không gian nhiều chiều (high-dimensional space), khái niệm về mật độ và khoảng cách trở nên kém tin cậy (”lời nguyền của số chiều”), khiến việc chọn ε rất khó khăn.

1.6 Mô hình Tăng cường Gradient (Gradient Boosting)

1.6.1 Tổng quan

Gradient Boosting là một kỹ thuật học máy thuộc nhóm *Học kết hợp (Ensemble Learning)*, cụ thể là phương pháp *Boosting*. Ý tưởng cốt lõi của Boosting là kết hợp nhiều mô hình ”học yếu” (weak learners) – thường là các cây quyết định có độ sâu thấp (decision stumps hoặc shallow trees) – để tạo thành một mô hình ”học mạnh” (strong learner) có độ chính xác cao.

Khác với Random Forest xây dựng các cây song song và độc lập (Bagging), Gradient Boosting xây dựng mô hình theo phương thức *tuần tự (sequential)*. Tại mỗi bước lặp, một mô hình mới được thêm vào để sửa chữa những sai số (errors) mà các mô hình trước đó gây ra.

Tên gọi "Gradient Boosting" xuất phát từ việc thuật toán sử dụng phương pháp *Gradient Descent* (xuống đồi) để tối thiểu hóa hàm mất mát. Thay vì cập nhật tham số trọng số như trong mạng nơ-ron, Gradient Boosting cập nhật trực tiếp hàm dự đoán trong không gian hàm (function space).

1.6.2 Cơ sở Toán học

Mô hình Cộng tính (Additive Model)

Giả sử ta cần tìm một hàm dự đoán $F(x)$ để ánh xạ đầu vào x tới đầu ra y sao cho giá trị kỳ vọng của hàm mất mát $L(y, F(x))$ là nhỏ nhất. Gradient Boosting xấp xỉ hàm $F(x)$ dưới dạng tổng có trọng số của M mô hình cơ sở:

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$$

Trong đó:

- $h_m(x)$: Là mô hình cơ sở (weak learner) tại bước thứ m.
- γ_m: Là trọng số đóng góp của mô hình cơ sở m.
- M: Tổng số vòng lặp (số lượng cây).

Mô hình được xây dựng theo quy trình tham lam (greedy) từng bước:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Thuật toán Gradient Boosting tổng quát

Quy trình huấn luyện diễn ra như sau:

Bước 1: Khởi tạo mô hình Tìm một giá trị hằng số tối ưu để khởi tạo mô hình ban đầu (thường là trung bình hoặc log-odds):

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$$

Bước 2: Lặp từ $m = 1$ **đến** M

1. **Tính toán Pseudo-Residuals (Phần dư giả):** Tại mỗi điểm dữ liệu, ta tính toán đạo hàm riêng của hàm mất mát theo giá trị dự đoán của mô hình hiện tại. Giá trị này đại diện cho hướng giảm nhanh nhất của sai số:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

Với bài toán hồi quy dùng MSE, r_{im} chính là phần dư $(y_i - F_{m-1}(x_i))$. Với bài toán phân loại, nó mang ý nghĩa khác (xem phần dưới).

2. **Huấn luyện mô hình cơ sở:** Huấn luyện một cây quyết định $h_m(x)$ để khớp (fit) với các giá trị phần dư r_{im} vừa tính được (tức là dùng tập huấn luyện $\{(x_i, r_{im})\}$).

3. **Tối ưu hóa bước nhảy (Step size):** Tìm hệ số γ_m để tối thiểu hóa hàm mất mát khi thêm cây mới vào:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

4. **Cập nhật mô hình:**

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x)$$

Trong đó ν ($0 < \nu \leq 1$) là tốc độ học (learning rate), giúp mô hình tránh bị quá khớp (overfitting).

1.6.3 Áp dụng cho bài toán Phân loại (Classification)

Đối với bài toán phân loại nhị phân (0 hoặc 1), Gradient Boosting không dự đoán trực tiếp nhãn lớp mà dự đoán *log-odds* (logits).

Hàm mất mát Log-Loss

Hàm mất mát được sử dụng là **Binomial Log-likelihood Loss** (tương tự Logistic Regression):

$$L(y, F(x)) = -[y \log(p) + (1 - y) \log(1 - p)]$$

Trong đó $p = \sigma(F(x)) = \frac{1}{1+e^{-F(x)}}$ là xác suất dự đoán $P(y = 1|x)$.

Pseudo-residuals trong phân loại

Đạo hàm của hàm Log-Loss đối với $F(x)$ cho ta công thức tính phần dư rất đơn giản:

$$r_{im} = y_i - p_{m-1}(x_i)$$

Tức là: *Phần dư = Nhãn thực tế - Xác suất dự đoán hiện tại*. Tại mỗi bước lặp, cây mới sẽ học cách dự đoán sự chênh lệch xác suất này.

1.6.4 Chiến lược Hiệu chỉnh (Regularization)

Gradient Boosting là một mô hình rất mạnh và dễ bị Overfitting. Các kỹ thuật hiệu chỉnh quan trọng bao gồm:

- **Learning Rate (Shrinkage) ν:** Giảm đóng góp của mỗi cây mới. Tốc độ học thấp (ví dụ $\nu < 0.1$) kết hợp với số lượng cây M lớn thường cho kết quả tốt hơn.
- **Subsampling (Stochastic Gradient Boosting):** Thay vì dùng toàn bộ dữ liệu, mỗi cây chỉ được huấn luyện trên một phần ngẫu nhiên của tập dữ liệu (ví dụ 80%). Việc này giúp giảm phương sai của mô hình.
- **Ràng buộc trên cây (Tree Constraints):** Giới hạn độ sâu tối đa của cây (max_depth), số lượng mẫu tối thiểu trong nút lá (min_samples_leaf) để đảm bảo tính ”yếu” (weak) của mô hình cơ sở.

1.6.5 Ưu điểm và Hạn chế

Ưu điểm:

- Độ chính xác cực cao, thường xuyên chiến thắng trong các cuộc thi dữ liệu dạng bảng (tabular data).
- Linh hoạt với nhiều loại hàm mất mát khác nhau.
- Xử lý tốt cả dữ liệu số và dữ liệu phân loại, không yêu cầu khắt khe về tiền xử lý dữ liệu (như chuẩn hóa) so với SVM hay Neural Networks.

Hạn chế:

- **Khó huấn luyện song song:** Do tính chất tuần tự (cây sau phụ thuộc cây trước), Gradient Boosting khó tận dụng phần cứng song song, dẫn đến thời gian huấn luyện lâu hơn Random Forest.
- **Nhiều siêu tham số:** Cần tinh chỉnh kỹ lưỡng (learning rate, số cây, độ sâu cây...) để đạt hiệu quả tối ưu và tránh overfitting.
- **Nhạy cảm với nhiễu:** Nếu hàm mất mát không được chọn phù hợp, mô hình có thể cố gắng khớp cả các điểm dữ liệu nhiễu (outliers).

1.7 Mô hình Mạng nơ-ron Đa tầng (Multi-Layer Perceptron - MLP)

1.7.1 Tổng quan

Multi-Layer Perceptron (MLP) là một loại mạng nơ-ron nhân tạo truyền thẳng (Feedforward Artificial Neural Network). Đây là mô hình nền tảng của Deep Learning, có khả năng học các mối quan hệ phi tuyến tính phức tạp giữa dữ liệu đầu vào và đầu ra.

Về mặt cấu trúc, một mạng MLP bao gồm ít nhất ba lớp nút (node) hay còn gọi là nơ-ron:

1. **Lớp đầu vào (Input Layer):** Nhận tín hiệu từ dữ liệu gốc. Số lượng nơ-ron tương ứng với số chiều (đặc trưng) của dữ liệu.
2. **Các lớp ẩn (Hidden Layers):** Nằm giữa lớp đầu vào và đầu ra. Đây là nơi thực hiện các phép biến đổi phi tuyến để trích xuất đặc trưng. Một MLP có thể có một hoặc nhiều lớp ẩn.
3. **Lớp đầu ra (Output Layer):** Đưa ra kết quả dự đoán của mô hình. Số lượng nơ-ron phụ thuộc vào bài toán (ví dụ: số lượng lớp trong bài toán phân loại).

Mỗi nơ-ron trong một lớp (trừ lớp đầu vào) được kết nối với tất cả các nơ-ron của lớp trước đó thông qua các trọng số (weights).

1.7.2 Mô hình Toán học

Cơ chế Lan truyền xuôi (Forward Propagation)

Xét một nơ-ron j tại lớp thứ l. Giá trị đầu ra của nơ-ron này được tính toán qua hai bước:

1. **Tổng hợp tuyến tính:** Tính tổng có trọng số của các đầu vào từ lớp trước $(l-1)$ cộng với hệ số chệch (bias).
$$z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}$$
Trong đó:
   - $w_{ji}^{(l)}$: Trọng số kết nối từ nơ-ron i lớp $(l-1)$ đến nơ-ron j lớp l.
   - $a_i^{(l-1)}$: Đầu ra (hoạt hóa) của nơ-ron i lớp trước.
   - $b_j^{(l)}$: Hệ số bias của nơ-ron j lớp l.
2. **Hàm kích hoạt (Activation Function):** Áp dụng một hàm phi tuyến $f(\cdot)$ lên tổng z để tạo ra đầu ra của nơ-ron.
$$a_j^{(l)} = f(z_j^{(l)})$$

Dưới dạng ma trận, công thức cho lớp l được viết gọn là:

$$Z^{(l)} = W^{(l)} A^{(l-1)} + b^{(l)}$$

$$A^{(l)} = f(Z^{(l)})$$

Các hàm kích hoạt phổ biến

Hàm kích hoạt là thành phần quan trọng giúp mạng nơ-ron có khả năng học các hàm phi tuyến. Nếu không có hàm kích hoạt phi tuyến, dù mạng có bao nhiêu lớp thì nó vẫn chỉ tương đương với một mô hình hồi quy tuyến tính.

- **ReLU (Rectified Linear Unit):** Thường dùng cho các lớp ẩn vì tính toán nhanh và giảm thiểu vấn đề biến mất đạo hàm (vanishing gradient).

$$f(z) = \max(0, z)$$

- **Sigmoid:** Chuyển giá trị về khoảng $(0, 1)$, thường dùng trong bài toán phân loại nhị phân.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- **Softmax:** Dùng cho lớp đầu ra trong bài toán phân loại đa lớp để chuyển đổi các giá trị thô (logits) thành phân phối xác suất.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

1.7.3 Quá trình Huấn luyện và Lan truyền ngược (Backpropagation)

Mục tiêu của huấn luyện MLP là tìm bộ tham số $\theta = \{W, b\}$ sao cho sai số giữa dự đoán của mô hình và nhãn thực tế là nhỏ nhất.

Hàm mất mát (Loss Function)

Đối với bài toán phân loại đa lớp, hàm mất mát phổ biến nhất là **Cross-Entropy Loss**:

$$L(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{C} y_{nk} \log(\hat{y}_{nk})$$

Trong đó:

- N: Số lượng mẫu dữ liệu.

- C: Số lượng lớp phân loại.
- y_{nk}: Bằng 1 nếu mẫu n thuộc lớp k, ngược lại bằng 0 (One-hot encoding).
- $\hat{y}_{nk}$: Xác suất dự đoán mẫu n thuộc lớp k (đầu ra của Softmax).

Thuật toán Lan truyền ngược (Backpropagation)

Để tối thiểu hóa hàm mất mát, ta sử dụng thuật toán Gradient Descent (hoặc các biến thể như Adam, RMSprop). Thuật toán cần tính đạo hàm riêng của hàm mất mát theo từng trọng số $\frac{\partial L}{\partial w}$. Quá trình này áp dụng quy tắc chuỗi (Chain Rule) đi ngược từ lớp đầu ra về lớp đầu vào:

$$\frac{\partial L}{\partial w^{(l)}} = \frac{\partial L}{\partial a^{(l)}} \cdot \frac{\partial a^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial w^{(l)}}$$

Sau khi có gradient, trọng số được cập nhật theo công thức:

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}}$$

với η là tốc độ học (learning rate).

1.7.4 Ưu điểm và Hạn chế

Ưu điểm:

- **Khả năng xấp xỉ vạn năng (Universal Approximation):** Về lý thuyết, một MLP với ít nhất một lớp ẩn đủ lớn có thể xấp xỉ bất kỳ hàm liên tục nào.
- **Xử lý dữ liệu phức tạp:** Hiệu quả cao trong việc mô hình hóa các mối quan hệ phi tuyến tính phức tạp mà các mô hình truyền thống (như Logistic Regression) không làm được.

Hạn chế:

- **Hộp đen (Black-box):** Khó giải thích được tại sao mô hình đưa ra quyết định cụ thể (tính minh bạch thấp).
- **Dễ bị Overfitting:** Với số lượng tham số lớn, MLP dễ bị quá khớp với dữ liệu huấn luyện nếu không có các kỹ thuật điều chỉnh (Regularization) như Dropout hoặc L2.
- **Khó tối ưu hóa:** Hàm mất mát của MLP là không lồi (non-convex), do đó thuật toán tối ưu có thể bị kẹt ở các điểm cực trị địa phương (local minima).

1.8 Chuyển từ phân loại sang hồi quy

1.8.1 Tổng quan

Trong bài toán gốc, mục tiêu là phân loại rủi ro tín dụng thành hai lớp: *High Risk* (1) và *Low Risk* (0). Mô hình phân loại sử dụng là hồi quy logistic nhị phân với đầu ra là xác suất được tính qua hàm sigmoid:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

Để chuyển sang bài toán hồi quy, phương pháp này lựa chọn **xác suất thuộc lớp High Risk** (hoặc giá trị hàm quyết định) làm mục tiêu hồi quy. Cách tiếp cận này cho phép mô hình hóa **"mức độ rủi ro"** của khách hàng theo thang liên tục, thay vì phân loại nhị phân rời rạc. Giá trị hồi quy càng cao thể hiện rủi ro càng lớn, cung cấp thông tin chi tiết hơn cho việc ra quyết định tín dụng.

1.8.2 XGBoost Regressor

XGBoost (Extreme Gradient Boosting) là một thuật toán boosting tiên tiến dựa trên gradient, kết hợp nhiều mô hình cây quyết định yếu thành một mô hình tổng thể mạnh mẽ và chính xác cao. Phiên bản **XGBoost Regressor** được thiết kế đặc biệt cho các bài toán hồi quy, tức dự đoán các biến đầu ra liên tục, thay vì phân loại các nhãn rời rạc.

Mặc dù cùng chia sẻ kiến trúc cây quyết định và cơ chế boosting với phiên bản phân loại, điểm khác biệt cốt lõi của XGBoost Regressor nằm ở **hàm mất mát** được tối ưu hóa – thường là các hàm đo sai số hồi quy như Sai số Bình phương Trung bình (MSE), thay vì hàm mất mát chéo (cross-entropy) như trong bài toán phân loại.

Nguyên lý hoạt động

XGBoost Regressor xây dựng mô hình dự đoán dưới dạng tổng có trọng số của một chuỗi các cây quyết định:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F}$$

Trong đó: - f_t là cây quyết định thứ t, ánh xạ các đặc trưng đầu vào x_i đến một giá trị thực (giá trị dự đoán) tại nút lá. - $\mathcal{F}$ là không gian của tất cả các cây quyết định có thể có. - $\hat{y}_i$ là đầu ra dự đoán cuối cùng cho mẫu i sau T vòng boosting.

Hàm mục tiêu tại vòng boosting thứ t được định nghĩa như sau:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \sum_{k=1}^{t} \Omega(f_k)$$

Trong đó: - l là hàm mất mát hồi quy, ví dụ phổ biến là **hàm bình phương lỗi**: $l(y, \hat{y}) = (y - \hat{y})^2$. - $\Omega(f)$ là **hàm điều chuẩn (regularization)** để phạt độ phức tạp của mô hình, giúp kiểm soát overfitting:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

với T là số lượng nút lá trong cây và w_j là giá trị dự đoán (trọng số) tại nút lá thứ j. Tham số γ kiểm soát độ phức tạp tối thiểu để tách nhánh, λ điều chuẩn cho trọng số tại lá.

Để tối ưu hóa hàm mục tiêu này một cách hiệu quả, XGBoost sử dụng **phép khai triển Taylor bậc hai** để xấp xỉ giá trị hàm mất mát tại vòng lặp hiện tại:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t)$$

Với: - **Gradient bậc nhất:** $g_i = \dfrac{\partial l(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ - **Hessian (Đạo hàm bậc hai):** $h_i = \dfrac{\partial^2 l(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$

Việc sử dụng thông tin bậc hai (Hessian) cho phép thuật toán ước lượng chính xác hơn hướng và khoảng cách cần thiết để giảm thiểu lỗi, dẫn đến tốc độ hội tụ nhanh và ổn định hơn so với các phương pháp gradient boosting chỉ dùng đạo hàm bậc nhất.

Sự khác biệt chính so với XGBoost Phân loại

Mặc dù có chung một khung kiến trúc boosting, XGBoost Regressor và XGBoost Phân loại khác nhau ở những điểm cơ bản sau:

Ưu điểm của XGBoost Regressor:

- **Độ chính xác cao:** Nhờ cơ chế boosting kết hợp nhiều cây yếu và khả năng bắt các mối quan hệ phi tuyến phức tạp.
- **Kiểm soát overfitting tốt:** Được tích hợp sẵn thông qua các thành phần điều chuẩn (L1/L2) trên cấu trúc cây (γ) và trọng số lá (λ).
- **Tốc độ và hiệu suất:** Tối ưu hóa bằng khai triển Taylor và các kỹ thuật xử lý song song làm cho quá trình huấn luyện rất nhanh.
- **Linh hoạt:** Hỗ trợ nhiều hàm mất mát tùy chỉnh và xử lý được các loại dữ liệu đa dạng.

Nhờ những đặc điểm này, XGBoost Regressor đã trở thành một công cụ mạnh mẽ và phổ biến trong nhiều bài toán hồi quy thực tế, đặc biệt hiệu quả với dữ liệu có nhiễu, có tính phi tuyến cao hoặc khi cần một mô hình dự đoán có độ chính xác vượt trội.

Đặc điểm	XGBoost Regressor	XGBoost Phân loại
Mục tiêu	Dự đoán một giá trị liên tục.	Dự đoán xác suất hoặc nhãn lớp rời rạc.
Hàm mất mát	Tối ưu các hàm sai số hồi quy như **MSE (Mean Squared Error)**, **MAE (Mean Absolute Error)**.	Tối ưu hàm mất mát chéo như **Log Loss (Binary/Multinomial Cross-Entropy)**.
Đầu ra tại lá cây	Mỗi lá cây chứa một **giá trị thực** (ví dụ: 15.7, -2.4), đại diện trực tiếp cho phần đóng góp vào kết quả dự đoán cuối cùng.	Mỗi lá cây đóng góp vào **log-odds** (hoặc logit). Các giá trị này sau đó được chuyển đổi qua hàm sigmoid (cho bài toán nhị phân) hoặc softmax (cho bài toán đa lớp) để ra xác suất.
Chiến lược dự đoán	Giá trị dự đoán là tổng trực tiếp các giá trị từ tất cả các cây.	Tổng các giá trị từ cây được chuyển đổi thành xác suất thuộc về mỗi lớp.

1.8.3 Random Forest Regressor

Random Forest Regressor là một thuật toán học tổ hợp (ensemble learning) dựa trên nhiều cây quyết định hồi quy (CART Regression Trees). Thuật toán này hoạt động theo nguyên lý ”wisdom of the crowd”, trong đó dự đoán cuối cùng là trung bình của các dự đoán từ nhiều cây hồi quy độc lập.

Nguyên lý hoạt động

Random Forest Regressor xây dựng một tập hợp B cây hồi quy và tổng hợp kết quả:

$$\hat{y}_i = \frac{1}{B} \sum_{b=1}^{B} T_b(x_i)$$

Trong đó: - $T_b(x_i)$ là dự đoán từ cây hồi quy thứ b cho mẫu x_i - B là số lượng cây trong rừng (thường từ 100 đến 1000) - $\hat{y}_i$ là giá trị dự đoán cuối cùng

Quá trình xây dựng mỗi cây trong rừng sử dụng hai kỹ thuật ngẫu nhiên:

1. **Bootstrapping:** Mỗi cây được huấn luyện trên một tập dữ liệu con được tạo bằng phương pháp lấy mẫu có hoàn lại (bootstrap sampling) từ tập huấn luyện gốc.

2. **Feature Randomness:** Tại mỗi lần tách nút, chỉ một tập con ngẫu nhiên m đặc trưng được xem xét để tìm kiếm phân tách tốt nhất. Thông thường $m \approx \sqrt{p}$ hoặc $m \approx p/3$ với p là tổng số đặc trưng.

Hàm mục tiêu tại mỗi nút tách là tối thiểu hóa phương sai trong các nút con:

$$\text{MSE}_{\text{split}} = \frac{n_{\text{left}}}{n}\text{MSE}_{\text{left}} + \frac{n_{\text{right}}}{n}\text{MSE}_{\text{right}}$$

với $\text{MSE}_{\text{node}} = \frac{1}{n_{\text{node}}} \sum_{i \in \text{node}} (y_i - \bar{y}_{\text{node}})^2$, trong đó $\bar{y}_{\text{node}}$ là giá trị trung bình của các mẫu trong nút.

Ưu điểm của Random Forest Regressor

- **Kiểm soát overfitting tốt:** Nhờ kỹ thuật bagging và feature randomness, rừng ngẫu nhiên ít bị overfitting hơn so với cây quyết định đơn lẻ.
- **Độ ổn định cao:** Dự đoán ít nhạy cảm với nhiễu trong dữ liệu nhờ cơ chế tổng hợp từ nhiều cây.
- **Ước lượng độ quan trọng đặc trưng:** Có thể xác định đặc trưng nào quan trọng nhất cho bài toán hồi quy dựa trên mức độ giảm MSE khi sử dụng đặc trưng đó.
- **Xử lý dữ liệu thiếu:** Có khả năng xử lý dữ liệu bị thiếu mà không cần imputation phức tạp.
- **Không cần chuẩn hóa dữ liệu:** Không yêu cầu chuẩn hóa đặc trưng vì dựa trên phân tách ngưỡng.

Các tham số quan trọng

- `n_estimators`: Số lượng cây trong rừng (tăng độ chính xác nhưng tăng thời gian)
- `max_depth`: Độ sâu tối đa của mỗi cây
- `max_features`: Số lượng đặc trưng xem xét tại mỗi lần tách
- `min_samples_split`: Số mẫu tối thiểu để tách một nút
- `min_samples_leaf`: Số mẫu tối thiểu tại nút lá
- `bootstrap`: Có sử dụng bootstrapping hay không

Random Forest Regressor đặc biệt hiệu quả khi:

- Dữ liệu có nhiều nhiễu hoặc outlier
- Cần mô hình robust và ổn định
- Cần ước lượng độ quan trọng của đặc trưng
- Yêu cầu thời gian huấn luyện nhanh

Nhờ tính đơn giản, hiệu quả và khả năng chống nhiễu tốt, Random Forest Regressor là một trong những thuật toán hồi quy được sử dụng phổ biến nhất trong thực tế.

Chương 2
TIỀN XỬ LÝ DỮ LIỆU VÀ GIẢM CHIỀU DỮ LIỆU

2.1 Trực quan hóa và thống kê dữ liệu

2.1.1 Mô tả tập dữ liệu

Bộ dữ liệu sau khi được thu thập và hợp nhất bao gồm 36.457 bản ghi với 20 thuộc tính, chứa đựng các thông tin đa chiều về nhân khẩu học, tình hình tài chính, hành vi sử dụng dịch vụ và nhãn rủi ro tín dụng (`Is_high_risk`). Các thuộc tính bao gồm nhiều kiểu dữ liệu khác nhau như số nguyên, số thực và các biến phân loại, cụ thể như sau:

- **Các biến định lượng (Numeric variables):**
  - `Children_Count`: Số lượng con cái. Đây là biến rời rạc với giá trị thường nhỏ.
  - `Income`: Thu nhập hàng năm của cá nhân (đơn vị tiền tệ gốc). Phân phối của biến này thường bị lệch phải (right-skewed).
  - `Age`: Tuổi của khách hàng (được chuyển đổi từ số ngày sinh).
  - `Employment_Length`: Thâm niên làm việc (số năm).
  - `Family_Member_Count`: Tổng số thành viên trong gia đình.
  - `Account_Age`: Tuổi của tài khoản tín dụng (tính theo tháng), thể hiện lịch sử gắn bó của khách hàng.
- **Các biến nhị phân và định tính (Categorical variables):**
  - `Gender`, `Has_a_Car`, `Has_a_Property`: Các biến định danh cơ bản.
  - `Has_a_Mobile_Phone`, `Has_a_Work_Phone`, `Has_a_Phone`, `Has_an_Email`: Các biến nhị phân (0/1) thể hiện khả năng liên lạc.
  - `Employment_Status`: Trạng thái nghề nghiệp (Working, Commercial associate, Pensioner, v.v.).
  - `Education_Level`: Trình độ học vấn (Secondary, Higher education, v.v.), mang tính thứ bậc.
  - `Marital_Status`: Tình trạng hôn nhân.
  - `Dwelling`: Loại hình cư trú.
  - `Job_Title`: Chức danh công việc. Thuộc tính này có tỷ lệ dữ liệu khuyết (missing value) cao, với khoảng 31% số quan sát không có thông tin.

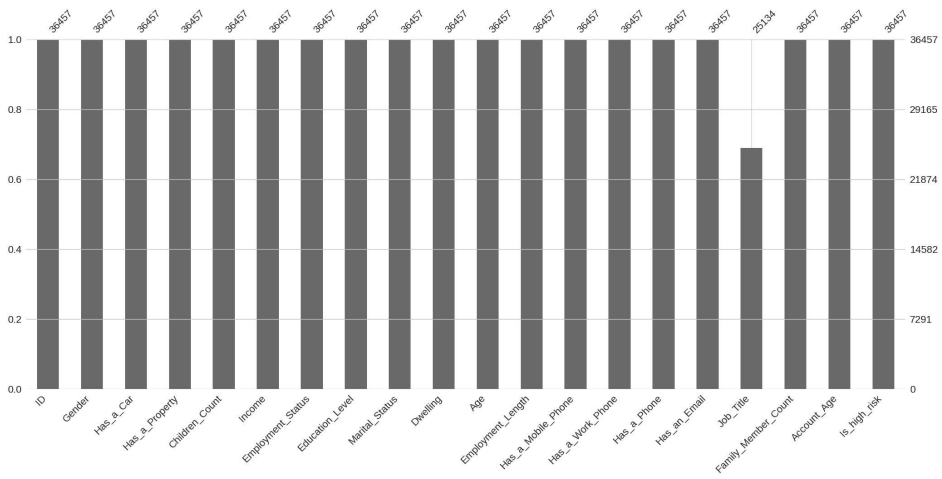
– `Is_high_risk`: Biến mục tiêu (Target variable), nhận giá trị 1 nếu khách hàng có rủi ro nợ xấu và 0 nếu an toàn.

```
RangeIndex: 36457 entries, 0 to 36456
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   36457 non-null  int64
 1   Gender               36457 non-null  object
 2   Has_a_Car            36457 non-null  object
 3   Has_a_Property       36457 non-null  object
 4   Children_Count       36457 non-null  int64
 5   Income               36457 non-null  float64
 6   Employment_Status    36457 non-null  object
 7   Education_Level      36457 non-null  object
 8   Marital_Status       36457 non-null  object
 9   Dwelling             36457 non-null  object
 10  Age                  36457 non-null  int64
 11  Employment_Length    36457 non-null  int64
 12  Has_a_Mobile_Phone   36457 non-null  int64
 13  Has_a_Work_Phone     36457 non-null  int64
 14  Has_a_Phone          36457 non-null  int64
 15  Has_an_Email         36457 non-null  int64
 16  Job_Title            25134 non-null  object
 17  Family_Member_Count  36457 non-null  float64
 18  Account_Age          36457 non-null  float64
 19  Is_high_risk         36457 non-null  object
dtypes: float64(3), int64(8), object(9)
```

Hình 2.1: Tổng quan các trường dữ liệu và kiểu dữ liệu

2.1.2 Phân tích tương quan và phân phối

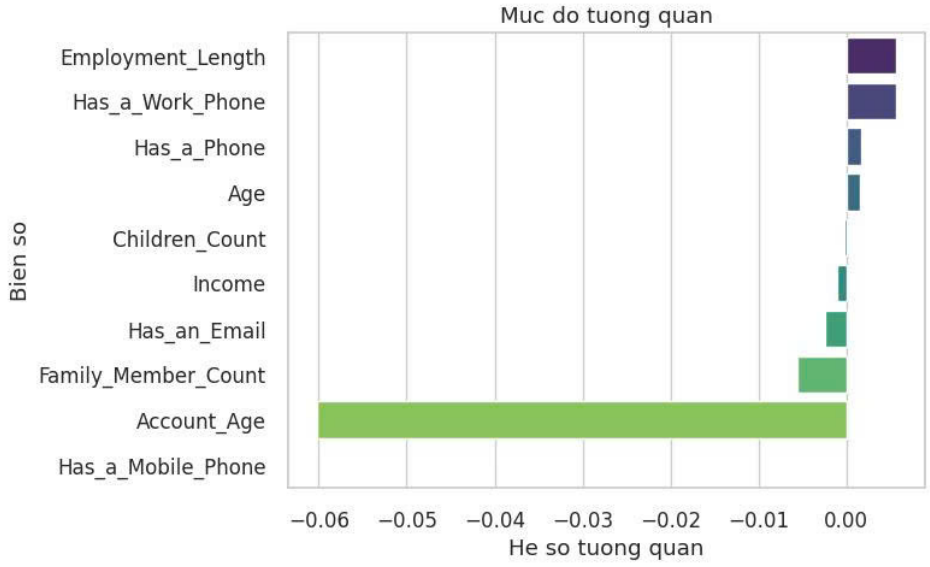
Việc phân tích dữ liệu khuyết cho thấy sự thiếu hụt nghiêm trọng ở trường `Job_Title`, trong khi các trường khác đảm bảo độ đầy đủ tốt.



Hình 2.2: Biểu đồ trực quan hóa dữ liệu khuyết (Missingno matrix)

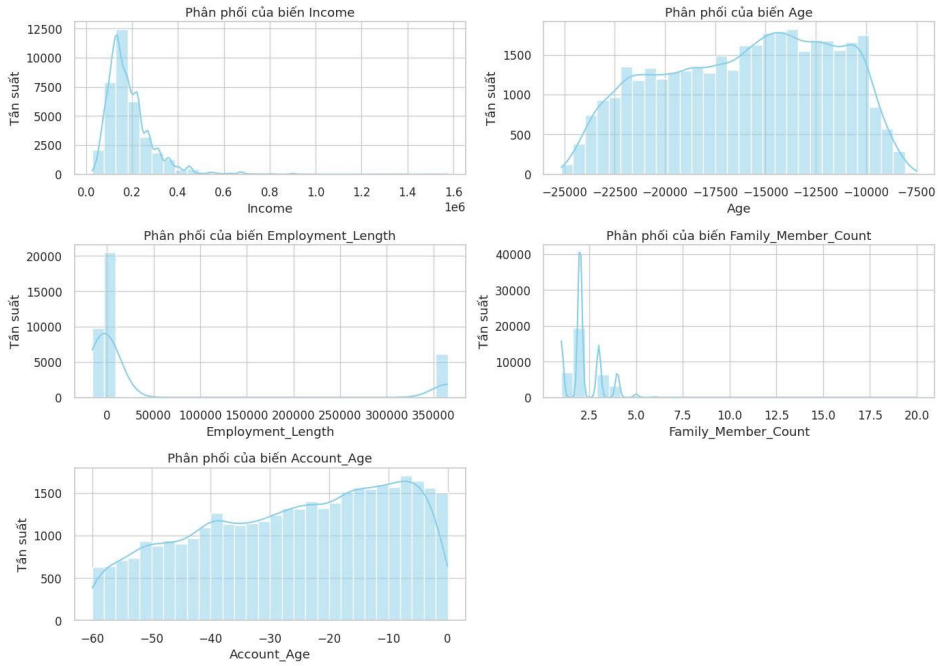
Phân tích tương quan (Correlation Analysis) giữa các biến số và biến mục tiêu cho thấy:

- Biến `Account_Age` có mối tương quan rõ rệt nhất với biến mục tiêu. Điều này dễ hiểu vì lịch sử tài khoản càng dài thì thông tin về hành vi trả nợ càng rõ ràng.
- Các biến như `Age` hay `Has_a_Mobile_Phone` có hệ số tương quan rất thấp, gần như bằng 0, cho thấy chúng ít có khả năng phân loại rủi ro nếu đứng độc lập.



Hình 2.3: Ma trận tương quan giữa các biến số

Biểu đồ phân phối của các biến liên tục cho thấy sự chênh lệch lớn về miền giá trị, đòi hỏi phải thực hiện chuẩn hóa dữ liệu trước khi đưa vào mô hình.



Hình 2.4: Phân phối xác suất của các biến liên tục

2.2 Tiền xử lý dữ liệu (Data Preprocessing)

Quá trình tiền xử lý đóng vai trò then chốt để nâng cao hiệu suất mô hình. Các bước thực hiện bao gồm loại bỏ thuộc tính nhiễu, xử lý dữ liệu khuyết và mã hóa dữ liệu.

2.2.1 Lược bỏ các thuộc tính không phù hợp

Dựa trên phân tích thống kê và ý nghĩa nghiệp vụ, các thuộc tính sau đã bị loại bỏ khỏi tập dữ liệu huấn luyện:

- **`Account_Age` (Tuổi tài khoản):**
  - *Lý do:* Rò rỉ dữ liệu (Data Leakage).
  - *Giải thích:* Biến mục tiêu `Is_high_risk` được xây dựng dựa trên lịch sử thanh toán (`MONTHS_BALANCE`). Nếu giữ lại `Account_Age`, mô hình sẽ học được quy tắc trực tiếp từ cách gán nhãn thay vì học hành vi khách hàng, dẫn đến hiện tượng quá khớp (overfitting) và sai lệch khi dự đoán thực tế.
- **`Job_Title` (Chức danh):**
  - *Lý do:* Tỷ lệ dữ liệu khuyết quá lớn ($\approx 31\%$).
  - *Giải thích:* Việc khôi phục (impute) một lượng lớn dữ liệu như vậy sẽ đưa vào nhiều nhiễu, làm giảm độ chính xác của mô hình.

- **`Has_a_Mobile_Phone`:**
  - *Lý do:* Phương sai bằng 0 (Zero Variance).
  - *Giải thích:* 100% khách hàng trong tập dữ liệu đều sở hữu điện thoại di động. Một đặc trưng không có sự biến thiên sẽ không mang lại giá trị phân loại.
- **`Children_Count`:**
  - *Lý do:* Sức mạnh dự đoán thấp.
  - *Giải thích:* Phân tích đơn biến và đa biến cho thấy không có sự khác biệt đáng kể về tỷ lệ rủi ro giữa các nhóm khách hàng có số lượng con cái khác nhau.

2.2.2 Chuyển đổi và Mã hóa dữ liệu

Dữ liệu được chuẩn hóa thông qua quy trình `ColumnTransformer` của thư viện Scikit-learn:

- **Chuẩn hóa biến số (Scaling):** Sử dụng `MinMaxScaler` để đưa các biến liên tục (`Income`, `Age`, `Employment_Length`,...) về khoảng [0, 1]. Điều này giúp các thuật toán dựa trên khoảng cách (như PCA, KNN) hoạt động hiệu quả hơn.
- **Mã hóa biến phân loại (Encoding):**
  - *Binary Encoding:* Các biến Yes/No được chuyển về 0/1.
  - *One-Hot Encoding:* Áp dụng cho các biến định danh không có thứ tự (`Gender`, `Marital_Status`, `Dwelling`,...).
  - *Ordinal Encoding:* Áp dụng cho biến có tính thứ bậc là `Education_Level`.

2.2.3 Thống kê dữ liệu sau xử lý

Sau quá trình làm sạch và mã hóa, tập dữ liệu bao gồm 36.457 mẫu với 32 đặc trưng (do quá trình One-Hot Encoding sinh thêm cột). Một số thống kê đáng chú ý:

- **Phân bố nhãn:** Tập dữ liệu bị mất cân bằng nghiêm trọng với 98,31% là *Low Risk* và chỉ 1,69% là *High Risk*. Điều này đặt ra yêu cầu phải sử dụng các kỹ thuật cân bằng dữ liệu (như SMOTE) trong giai đoạn huấn luyện.
- **Giới tính:** Nữ chiếm đa số (67,01%).
- **Hôn nhân:** Đa phần khách hàng đã kết hôn (68,71%).
- **Tài sản:** 67,22% khách hàng sở hữu bất động sản và 37,97% sở hữu ô tô.

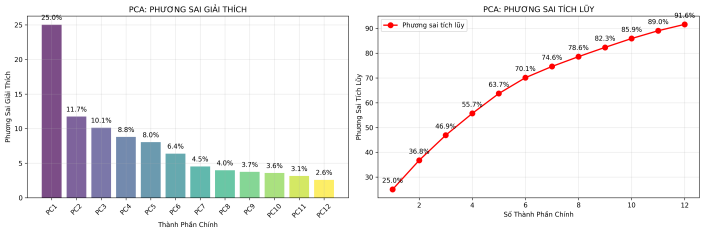
2.3 Giảm chiều dữ liệu (Dimensionality Reduction)

Để giải quyết vấn đề ”lời nguyền số chiều” và tăng tốc độ tính toán, hai phương pháp giảm chiều phổ biến là PCA và LDA đã được thực nghiệm.

2.3.1 Phân tích thành phần chính (PCA)

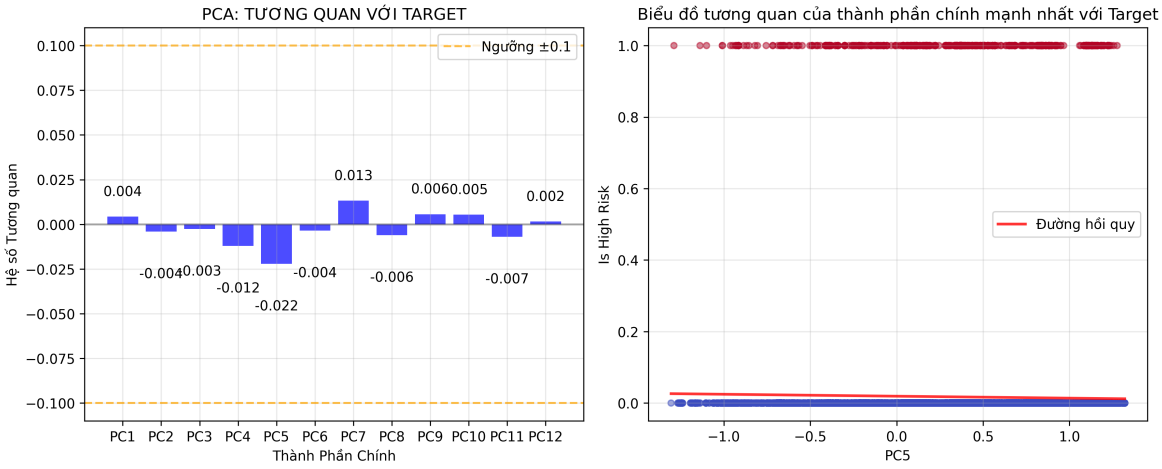
Trước khi thực hiện PCA, dữ liệu đã được chuẩn hóa Min-Max để đảm bảo các biến đóng góp công bằng vào phương sai tổng thể. Kết quả phân tích cho thấy:

- Cần khoảng 12 thành phần chính (Principal Components - PCs) để giải thích được hơn 90% phương sai của dữ liệu gốc.
- **Tỷ lệ phương sai giải thích (Explained Variance Ratio):**
  - PC1: 25,02%
  - PC2: 11,74%
  - PC3: 11,25%
  - ...
  - PC12: 2,6%



Hình 2.5: Biểu đồ phương sai tích lũy theo số lượng thành phần chính

Khi xem xét mối tương quan giữa các thành phần chính và biến mục tiêu `Is_high_risk`, ta thấy các hệ số tương quan đều khá thấp. Thành phần PC5 có tương quan mạnh nhất nhưng vẫn chưa đủ để phân tách rõ ràng hai lớp dữ liệu trên không gian chiếu.

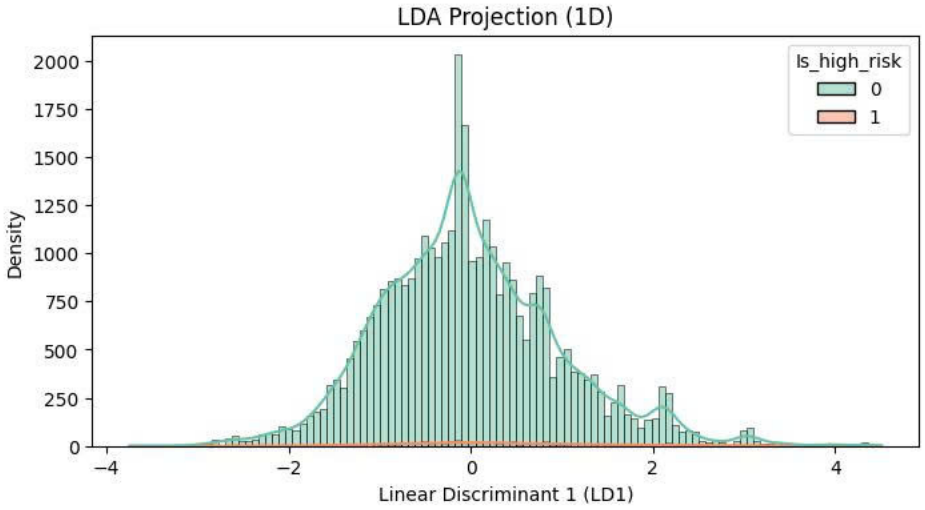


Hình 2.6: Tương quan giữa các thành phần chính và biến mục tiêu

2.3.2 Phân tích biệt thức tuyến tính (LDA)

Khác với PCA (học không giám sát), LDA là thuật toán học có giám sát, mục tiêu là tìm hướng chiếu giúp tối đa hóa khoảng cách giữa các lớp và tối thiểu hóa phương sai trong nội bộ lớp.

Do bài toán phân loại nhị phân (2 lớp: High Risk/Low Risk), LDA giảm chiều dữ liệu xuống còn $C - 1 = 1$ chiều (thành phần LD1).



Hình 2.7: Phân phối mật độ của hai lớp trên trục LDA (LD1)

Đánh giá kết quả LDA:

- **Ưu điểm:** LD1 đã tìm ra được hướng chiếu có khả năng phân tách nhất định. Biểu đồ mật độ cho thấy đỉnh của hai phân phối có sự lệch nhau.

- **Hạn chế:** Vùng chồng lấn (overlap) giữa hai lớp vẫn rất lớn (đặc biệt trong khoảng giá trị trung tâm). Điều này cho thấy các đặc trưng tuyến tính chưa đủ mạnh để phân biệt hoàn toàn rủi ro tín dụng.

2.3.3 So sánh và Kết luận về Giảm chiều

- **PCA:** Tập trung bảo toàn thông tin (phương sai) nhưng không tối ưu cho phân loại. Kết quả trực quan hóa cho thấy các điểm dữ liệu của hai lớp rủi ro trộn lẫn vào nhau trong không gian PCA. PCA thích hợp hơn cho việc loại bỏ nhiễu đa cộng tuyến.
- **LDA:** Tập trung vào khả năng phân lớp. Mặc dù vẫn còn sự chồng lấn, LDA cung cấp tín hiệu phân loại tốt hơn PCA trong bài toán này.
- **Kết luận:** Với tính chất phức tạp và mất cân bằng của dữ liệu tín dụng, việc chỉ sử dụng giảm chiều tuyến tính (PCA/LDA) là chưa đủ để xây dựng mô hình phân loại chính xác cao. Tuy nhiên, các kỹ thuật này giúp hiểu rõ cấu trúc dữ liệu và là bước đệm quan trọng cho các mô hình phi tuyến (như Random Forest hay Neural Networks) ở giai đoạn sau.

Chương 3
PHÂN CỤM DỮ LIỆU

3.1 Phương pháp mô hình hỗn hợp Gaussian (GMM)

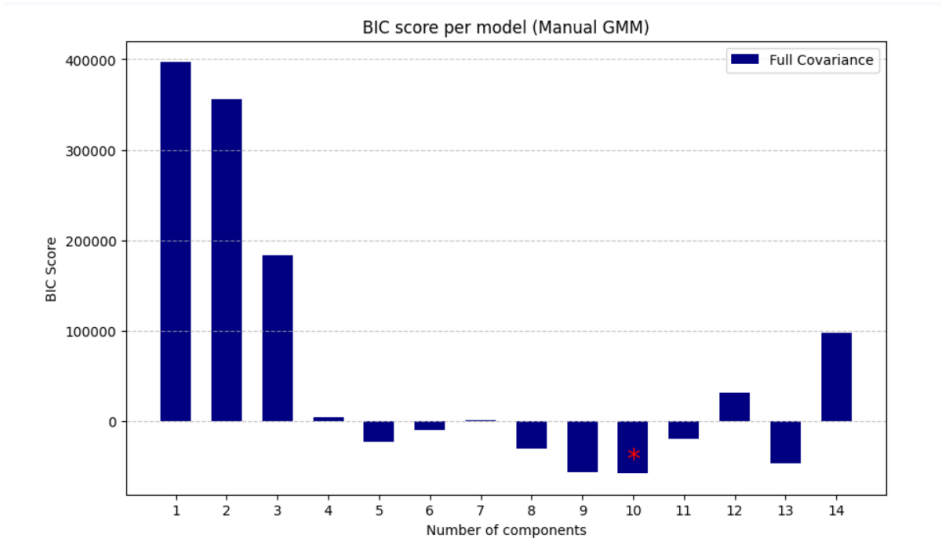
3.1.1 Thực nghiệm phân cụm và đánh giá kết quả

Dữ liệu đầu vào của bài toán phân cụm đã được bỏ qua trường đầu ra và là dữ liệu đã giảm chiều bằng PCA. Dùng dữ liệu giảm chiều giúp mô hình hoạt động ổn định, giảm nhiễu, hội tụ nhanh hơn gấp nhiều lần so với dữ liệu gốc.

Để lựa chọn số cụm tối ưu, nhóm dựa trên mức độ phù hợp được đánh giá thông qua chỉ số BIC. Đây là chỉ số đo lường mức độ hợp lý của mô hình đối với một bộ tham số, được tính dựa trên giá trị tối đa của hàm hợp lý như sau:

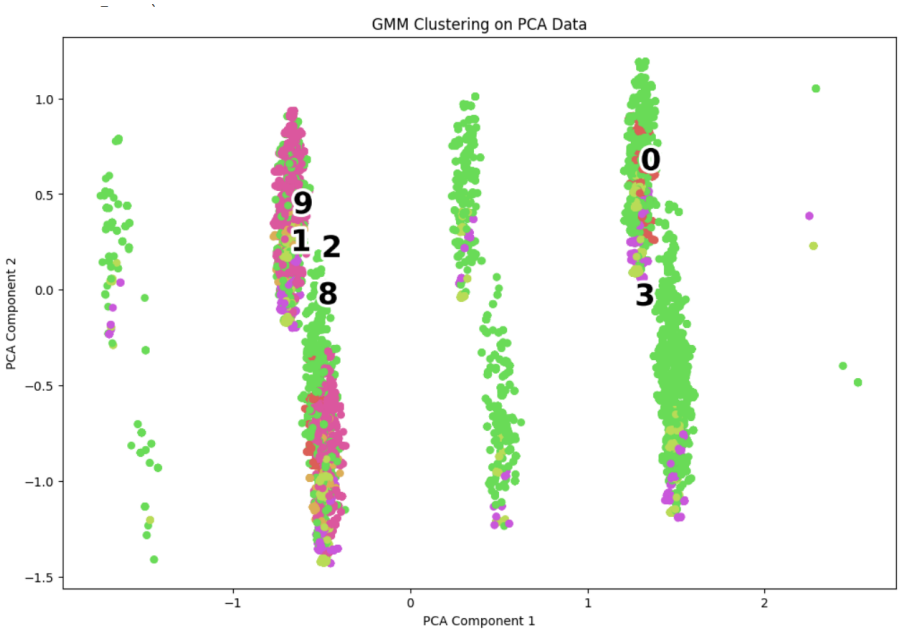
$$\text{BIC} = k \ln(n) - 2 \ln(\hat{L})$$

Với k là số lượng tham số được ước lượng từ mô hình, n là số lượng quan sát của bộ dữ liệu và $\hat{L}$ là giá trị ước lượng tối đa của hàm hợp lý. Mô hình có BIC càng nhỏ thì mức độ hợp lý của mô hình đối với bộ dữ liệu càng cao. Huấn luyện mô hình GMM với nhiều tham số n_components và tìm ra giá trị có BIC là nhỏ nhất. Đó chính là số lượng thành phần phù hợp nhất của bộ dữ liệu được tính theo mô hình GMM. Kết quả thực nghiệm cho thấy mô hình tốt nhất có 10 cụm. Sau quá trình huấn luyện bằng thuật toán EM, số cụm dữ liệu thực tế là 6.



Hình 3.1: Chỉ số BIC cho từng mô hình

Để đánh giá kết quả, ta dự báo cụm và vẽ biểu đồ trên không gian hai chiều, các điểm dữ liệu được đánh dấu bằng màu của cụm tương ứng.



Hình 3.2: Kết quả phân cụm GMM

3.1.2 Mối quan hệ giữa các mẫu dữ liệu đầu vào trong các cụm

Kết quả đánh giá cho thấy các độ đo Silhouette (0.00895) và Dunn (0.00868) đều có giá trị rất thấp, trong khi Davies–Bouldin đạt 3.58, cho thấy mức độ phân tán trong các cụm còn lớn và khoảng cách giữa các cụm không đủ rõ ràng. Nói cách khác, xét trong không gian đặc trưng của dữ liệu đầu vào (sau khi giảm chiều bằng PCA), các cụm thu được còn có sự chồng lấn đáng kể và chưa thể hiện được cấu trúc phân cụm rõ ràng.

Ngược lại, độ đo Modularity đạt giá trị 0.736, cho thấy cấu trúc cộng đồng của dữ liệu là tương đối rõ rệt khi xét dưới dạng đồ thị quan hệ giữa các mẫu dữ liệu. Giá trị Modularity cao chứng tỏ rằng các mẫu được gán vào cùng một cụm có mối liên kết trong cụm mạnh hơn đáng kể so với liên kết giữa các cụm, xét trên quan hệ lân cận hoặc độ tương tự giữa các mẫu.

Sự khác biệt này cho thấy rằng các mối quan hệ giữa dữ liệu đầu vào được biểu diễn tốt hơn dưới dạng cấu trúc quan hệ (graph-based) so với khoảng cách hình học trong không gian đặc trưng.

3.1.3 Quan hệ giữa các đầu ra tương ứng trong các cụm

Kết quả thống kê mô tả biến Is_high_risk theo từng cụm cho thấy sự khác biệt không lớn về tỷ lệ rủi ro giữa các cụm. Giá trị trung bình của Is_high_risk dao động trong khoảng 0.011–0.020, cho thấy tỷ lệ các trường hợp rủi ro cao trong mỗi cụm đều ở mức thấp. Cụm 3 có giá

Bảng 3.1: Tóm tắt thống kê biến mục tiêu theo từng cụm từ mô hình GMM

Cụm	Mean	Std	Số lượng mẫu
0	0.011057	0.104599	1628
1	0.017717	0.131984	1016
2	0.015526	0.122560	5308
3	0.019707	0.139000	9844
4	0.014272	0.118638	2032
5	0.015793	0.124681	9751

trị trung bình cao nhất (0.0197), trong khi cụm 0 có giá trị thấp nhất (0.0111); tuy nhiên, mức chênh lệch này là tương đối nhỏ và chưa thể hiện sự phân tách rõ ràng về rủi ro giữa các cụm.

Độ lệch chuẩn của Is_high_risk trong các cụm dao động từ 0.104 đến 0.139, phản ánh sự tồn tại của cả các mẫu rủi ro và không rủi ro trong cùng một cụm. Điều này cho thấy rằng các cụm được hình thành chưa có tính thuần nhất cao về mặt đầu ra, và việc phân cụm chủ yếu dựa trên đặc trưng đầu vào hơn là trực tiếp tách biệt các mức độ rủi ro.

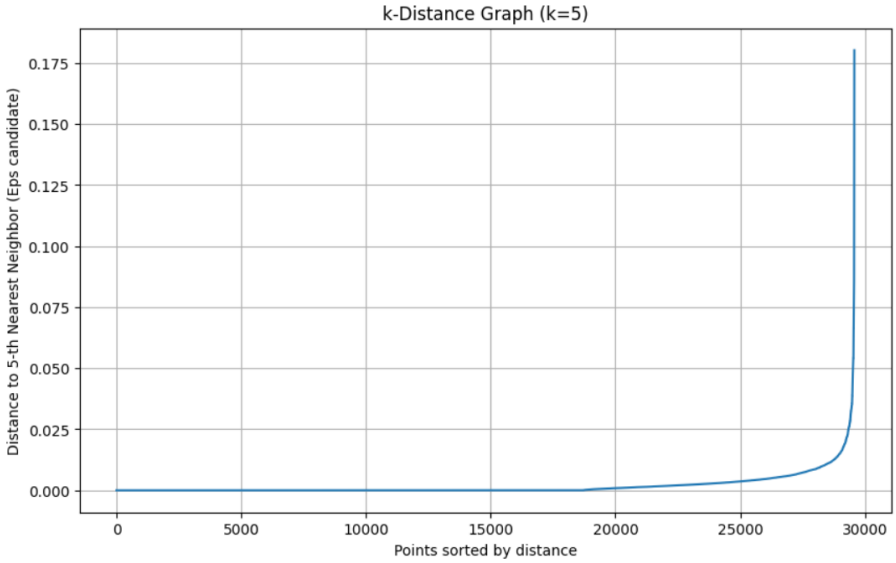
Ngoài ra, số lượng mẫu trong các cụm có sự chênh lệch đáng kể, với các cụm 3 và 9 chiếm tỷ trọng lớn nhất trong dữ liệu. Điều này cho thấy mô hình phân cụm có xu hướng gom các mẫu có đặc trưng tương tự vào các cụm lớn, nhưng chưa tạo ra các cụm chuyên biệt đại diện cho các mức rủi ro khác nhau.

Mặc dù mô hình phân cụm đã phát hiện được cấu trúc trong dữ liệu đầu vào, song các cụm hình thành chưa thể hiện sự phân hóa rõ rệt đối với biến đầu ra, cho thấy phân cụm trong trường hợp này mang tính mô tả hơn là phân loại rủi ro.

3.2 Phương pháp DBSCAN

3.2.1 Thực nghiệm phân cụm và đánh giá kết quả

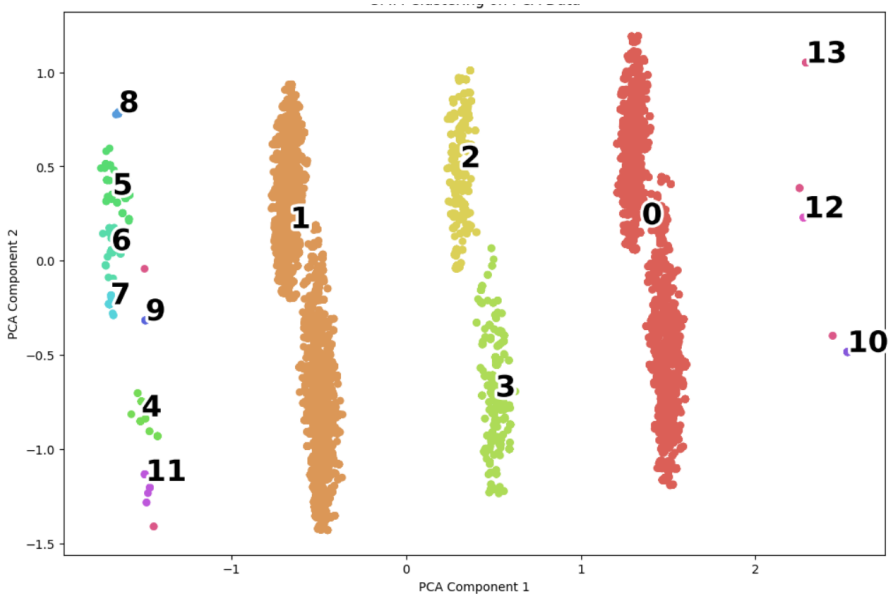
Dữ liệu đầu vào của bài toán phân cụm đã được bỏ qua trường đầu ra và là dữ liệu đã giảm còn 2 chiều bằng PCA. Tìm giới hạn khoảng ε dựa trên biểu đồ khoảng cách k gần nhất, Cuối cùng chúng ta lựa chọn số lượng điểm dữ liệu tối thiểu nằm trong vùng lân cận là k=5.



Hình 3.3: Biểu đồ k-distance

Lựa chọn ε thông qua phương pháp quét lưới (grid search), trong đó chất lượng phân cụm được đánh giá bằng chỉ số Silhouette sau khi loại bỏ các điểm nhiễu.

Để đánh giá kết quả, ta dự báo cụm và vẽ biểu đồ trên không gian hai chiều, các điểm dữ liệu được đánh dấu bằng màu của cụm tương ứng.



Hình 3.4: Kết quả phân cụm DBSCAN

3.2.2 Mối quan hệ giữa các mẫu dữ liệu đầu vào trong các cụm

Kết quả đánh giá cho thấy Silhouette Score đạt 0.2895, phản ánh mức độ gắn kết trong các cụm ở mức trung bình, đồng thời các cụm đã có sự tách biệt nhất định trong không gian đặc

trưng của dữ liệu đầu vào. Giá trị Dunn = 0.2206 cho thấy khoảng cách giữa các cụm lớn hơn đáng kể so với độ phân tán nội cụm, cho thấy cấu trúc phân cụm đã được cải thiện so với các cấu hình có Dunn thấp, mặc dù chưa đạt mức phân tách rõ ràng hoàn toàn.

Chỉ số Davies–Bouldin = 1.85 cho thấy các cụm vẫn tồn tại mức độ chồng lấn nhất định, đặc biệt là đối với các cụm có độ phân tán trong cụm tương đối lớn.

Bên cạnh đó, Modularity đạt 0.466, phản ánh rằng khi biểu diễn dữ liệu dưới dạng đồ thị quan hệ, các mẫu trong cùng một cụm có xu hướng liên kết với nhau mạnh hơn so với các mẫu thuộc cụm khác. Điều này cho thấy cấu trúc cộng đồng của dữ liệu tồn tại ở mức khá, dù chưa thực sự rõ rệt.

Các độ đo đánh giá trong cho thấy cấu trúc phân cụm đã hình thành tương đối rõ ràng trong không gian đặc trưng của dữ liệu đầu vào, mặc dù vẫn tồn tại hiện tượng chồng lấn giữa các cụm.

3.2.3 Quan hệ giữa các đầu ra tương ứng trong các cụm.

Bảng 3.2: Tóm tắt thống kê biến mục tiêu theo từng cụm từ mô hình DBScan

Cluster	Mean	Std	Count
-1	0.000000	0.000000	8
0	0.017593	0.131475	8526
1	0.015748	0.124501	19495
2	0.023356	0.151774	764
3	0.020004	0.140253	549
4	0.000000	0.000000	32
5	0.044444	0.207224	90
6	0.025641	0.160128	39
7	0.050000	0.223607	20
8	0.076923	0.277735	13
9	0.000000	0.000000	5
10	0.000000	0.000000	12
11	0.083333	0.288688	12
12	0.000000	0.000000	8
13	0.000000	0.000000	6

Kết quả cho thấy sự phân bố rõ rệt giữa các cụm: hai cụm lớn nhất là Cluster 0 (8.526 mẫu) và Cluster 1 (19.495 mẫu) đại diện cho nhóm khách hàng có rủi ro rất thấp (Mean lần lượt là 0,0176 và 0,0157), chiếm tỷ lệ áp đảo trong tập dữ liệu và phản ánh đặc trưng Low Risk của phần lớn quan sát.

Các cụm có kích thước trung bình như Cluster 2 (764 mẫu, Mean = 0,0234) và Cluster 3 (549 mẫu, Mean = 0,0200) thể hiện mức rủi ro cao hơn một cách nhẹ, trong khi các cụm nhỏ hơn (Cluster 5 đến Cluster 12, với số lượng mẫu từ 90 trở xuống) ghi nhận giá trị Mean tăng

dần đáng kể (lên đến 0,0833 ở Cluster 12), cho thấy đây là các nhóm khách hàng có nguy cơ rủi ro cao hơn, phù hợp với đặc trưng thiểu số High Risk trong dữ liệu mất cân bằng.

Đặc biệt, một số cụm nhỏ và nhiễu (Cluster -1, 4, 9, 10, 13) có Mean bằng 0, đại diện cho các trường hợp hoàn toàn không có rủi ro. Quan hệ tổng thể giữa các cụm cho thấy xu hướng nghịch rõ nét: kích thước cụm giảm thì mức rủi ro trung bình tăng lên, chứng tỏ mô hình DBSCAN đã hiệu quả trong việc phát hiện và tách biệt các nhóm rủi ro thiểu số, đồng thời hỗ trợ tốt cho việc ưu tiên theo dõi và quản lý rủi ro tín dụng.

Chương 4
XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

4.1 Huấn luyện và Đánh giá mô hình Multilayer Perceptron (MLP)

Sau khi đánh giá hiệu suất của các mô hình tuyến tính (Logistic Regression, LDA) và nhận thấy hạn chế trong việc nắm bắt các mối quan hệ phức tạp của dữ liệu tín dụng, nhóm nghiên cứu đã tiến hành thử nghiệm mô hình mạng nơ-ron nhân tạo đa lớp (Multilayer Perceptron - MLP). Đây là mô hình có khả năng học các đường ranh giới quyết định phi tuyến tính, phù hợp với đặc thù dữ liệu tài chính đa chiều.

4.1.1 Cấu hình mô hình và Quá trình huấn luyện

Mô hình được xây dựng từ đầu với các tham số cấu hình tối ưu như sau:

Cấu trúc mạng (Network Architecture)

Dựa trên số lượng đặc trưng sau khi tiền xử lý ($N \approx 27$ đặc trưng sau One-Hot Encoding), chúng tôi thiết kế kiến trúc mạng như sau để cân bằng giữa khả năng học và chi phí tính toán:

- **Lớp đầu vào (Input Layer):** Số nơ-ron bằng số chiều của dữ liệu ($D_{in} = 27$ hoặc D_{pca} đối với dữ liệu giảm chiều).
- **Lớp ẩn (Hidden Layers):** Chúng tôi sử dụng 02 lớp ẩn.
  - Lớp ẩn 1: 64 nơ-ron, hàm kích hoạt **ReLU**.
  - Lớp ẩn 2: 32 nơ-ron, hàm kích hoạt **ReLU**.

  Việc sử dụng ReLU ($f(x) = \max(0, x)$) giúp giải quyết vấn đề triệt tiêu đạo hàm (vanishing gradient) thường gặp ở hàm Sigmoid/Tanh trong các mạng sâu, đồng thời tăng tốc độ hội tụ.
- **Lớp đầu ra (Output Layer):** 01 nơ-ron với hàm kích hoạt **Sigmoid** để đưa ra xác suất dự đoán $P(y = 1|x)$ (xác suất khách hàng có rủi ro cao).

Công thức tổng quát cho đầu ra của một nơ-ron tại lớp l:

$$a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)}) \tag{4.1}$$

Trong đó W là ma trận trọng số, b là bias, và σ là hàm kích hoạt.

Cơ chế huấn luyện và Hàm mất mát

Để tối ưu hóa trọng số, chúng tôi sử dụng thuật toán **Lan truyền ngược (Backpropagation)** kết hợp với bộ tối ưu hóa **Adam** (Adaptive Moment Estimation). Hàm mất mát được sử dụng là **Binary Cross-Entropy**, phù hợp cho bài toán phân loại nhị phân:

$$J(W, b) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})] \tag{4.2}$$

Để tránh hiện tượng *Overfitting* (khi mô hình học vẹt dữ liệu huấn luyện), chúng tôi áp dụng kỹ thuật **Dropout** (với tỷ lệ $p = 0.2$) sau mỗi lớp ẩn. Kỹ thuật này ngẫu nhiên ”tắt” một số nơ-ron trong quá trình huấn luyện, buộc mạng phải học các đặc trưng mạnh mẽ hơn.

4.1.2 Thiết lập thực nghiệm

Quá trình huấn luyện được thực hiện trên 03 kịch bản phân chia dữ liệu (Train:Validation) khác nhau để đánh giá độ ổn định của mô hình: 80:20, 70:30, và 60:40. Đồng thời, mô hình được kiểm chứng trên cả hai tập dữ liệu: Dữ liệu gốc (đầy đủ chiều) và Dữ liệu sau giảm chiều PCA.

Các siêu tham số (Hyperparameters) được thiết lập:

- **Epochs:** 300 (có sử dụng Early Stopping nếu loss không giảm sau 10 epochs).
- **Batch size:** 32.
- **Learning rate:** 0.005.

4.1.3 Kết quả và Đánh giá

So sánh hiệu năng trên các tỷ lệ chia dữ liệu

Bảng dưới đây trình bày kết quả độ chính xác (Accuracy) và F1-Score trên tập kiểm thử (Test set):

Tỉ lệ	Accuracy	Class	Precision	Recall	F1-score
8/2	0.82	0	0.76	0.94	0.84
		1	0.93	0.69	0.79
7/3	0.78	0	0.70	0.97	0.81
		1	0.95	0.59	0.72
6/4	0.74	0	0.66	0.97	0.79
		1	0.94	0.50	0.66

Bảng 4.1: Tổng hợp kết quả trên dữ liệu gốc

Giảm chiều	Accuracy	Class	Precision	Recall	F1-score
PCA	0.82	0	0.78	0.90	0.83
		1	0.88	0.75	0.81
LDA	0.56	0	0.61	0.32	0.42
		1	0.54	0.79	0.64

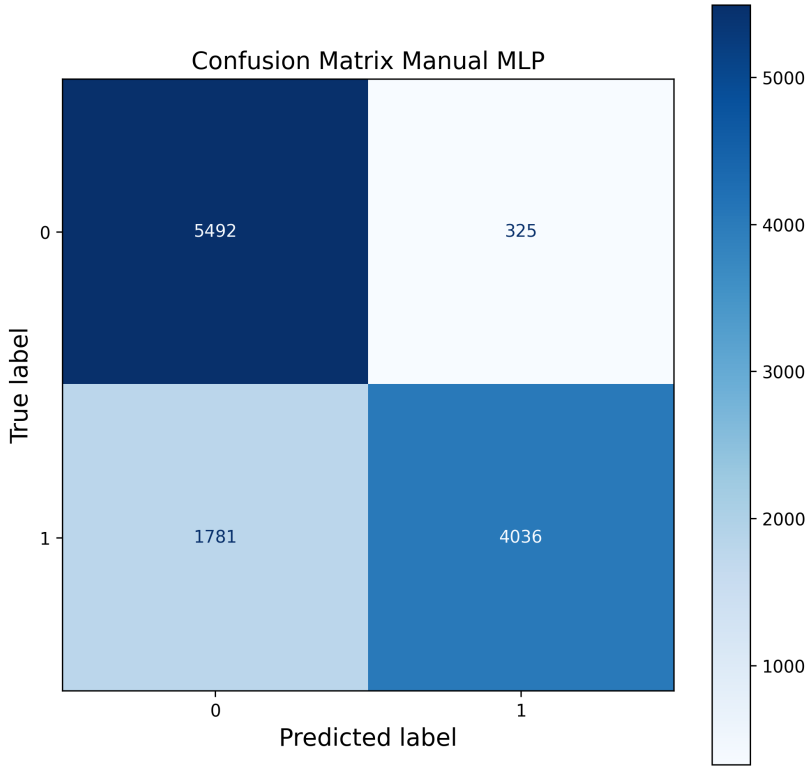
Bảng 4.2: Tổng hợp kết quả trên dữ liệu giảm chiều

Nhận xét:

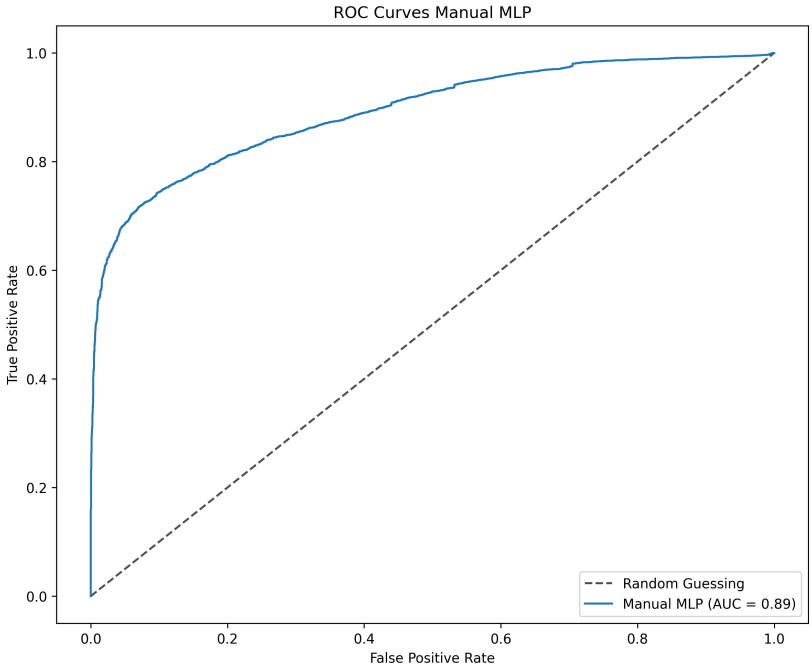
- Với tỷ lệ 8:2, mô hình đạt hiệu suất tốt nhất do có đủ dữ liệu để học các mẫu tổng quát. Khi giảm lượng dữ liệu train (6:4), độ chính xác có xu hướng giảm nhẹ, chứng tỏ mô hình nhạy cảm với kích thước mẫu huấn luyện.
- So sánh giữa dữ liệu gốc và PCA: Dữ liệu PCA tuy đã giảm chiều (mất một lượng nhỏ thông tin phương sai) nhưng vẫn duy trì độ chính xác tương đương so với dữ liệu gốc. Điều này chứng minh tính hiệu quả của bước giảm chiều dữ liệu trong việc loại bỏ nhiễu và giảm độ phức tạp tính toán.
- Đối với dữ liệu LDA, mô hình lại thể hiện sự kém hiệu quả trong quá trình huấn luyện. Mô hình gần như đưa ra kết quả sai hoàn toàn so với dữ liệu gốc và dữ liệu PCA. Qua đó ta thấy được việc cố gắng giảm chiều đối với dữ liệu phức tạp không phải là một chiến lược tốt khi nó làm mất đi phần lớn thông tin của dữ liệu.

Ma trận nhầm lẫn (Confusion Matrix) và ROC-AUC

Đối với bài toán rủi ro tín dụng, việc phát hiện đúng nhóm rủi ro cao (True Positive) quan trọng hơn độ chính xác tổng thể.



Hình 4.1: *Ma trận nhầm lẫn mô hình MLP*



Hình 4.2: *Đường cong ROC mô hình MLP*

- **ROC-AUC Score:** Đạt mức [0.89]. Diện tích dưới đường cong ROC càng lớn cho thấy khả năng phân biệt giữa hai lớp (High Risk vs Low Risk) của MLP càng tốt.

- Dựa vào Ma trận nhầm lẫn, tỷ lệ Recall cho lớp ”High Risk” đạt [0.69]. Đây là chỉ số quan trọng để ngân hàng tránh cấp tín dụng cho khách hàng nợ xấu.

4.1.4 Kết luận cho mô hình MLP

Mô hình MLP cho thấy khả năng học phi tuyến vượt trội hơn so với các mô hình tuyến tính cơ bản. Mặc dù chi phí huấn luyện cao hơn, nhưng với kiến trúc 2 lớp ẩn kết hợp Dropout, mô hình đã xử lý tốt sự phức tạp của dữ liệu tín dụng và hạn chế được Overfitting.

4.2 Mô hình Gradient Boosting

Bên cạnh mạng nơ-ron (MLP), nhóm nghiên cứu cũng triển khai mô hình Gradient Boosting, một trong những thuật toán học máy mạnh mẽ nhất hiện nay dựa trên nguyên lý kết hợp (ensemble) nhiều mô hình yếu (thường là cây quyết định) để tạo thành một mô hình dự báo chính xác.

Nguyên lý Boosting

Ý tưởng cốt lõi của Gradient Boosting là kết hợp nhiều ”mô hình yếu” (weak learners) - thường là các Cây quyết định (Decision Trees) có độ sâu thấp (decision stumps) - để tạo thành một mô hình dự đoán mạnh mẽ. Quá trình huấn luyện diễn ra tuần tự:

1. Mô hình đầu tiên học dữ liệu gốc.
2. Các mô hình tiếp theo không học lại dữ liệu gốc, mà tập trung học các **phần dư (residuals)** - tức là sai số mà các mô hình trước đó chưa giải quyết được.
3. Kết quả cuối cùng là tổng có trọng số của tất cả các cây thành phần.

Mô hình tại bước m được cập nhật theo công thức:

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x) \tag{4.3}$$

Trong đó:

- $F_{m-1}(x)$: Mô hình tổng hợp tại bước trước.
- $h_m(x)$: Cây quyết định mới được huấn luyện để xấp xỉ đạo hàm âm của hàm mất mát (Negative Gradient of Loss Function) đối với $F_{m-1}(x)$.
- ν: Tốc độ học (Learning Rate), tham số kiểm soát mức độ đóng góp của từng cây để tránh Overfitting.

Đối với bài toán phân loại rủi ro tín dụng (nhị phân), hàm mất mát được sử dụng là **Log-Loss (Deviance)**:

$$L(y, F(x)) = -[y \log(\sigma(F(x))) + (1 - y) \log(1 - \sigma(F(x)))] \tag{4.4}$$

4.2.1 Cấu hình mô hình

Mô hình được xây dựng từ đầu với các tham số cấu hình như sau:

- **n_estimators (Số lượng cây):** 100. Đây là số vòng lặp boosting.
- **learning_rate (Tốc độ học):** 0.1. Giá trị nhỏ giúp mô hình hội tụ tốt hơn nhưng cần nhiều cây hơn.
- **max_depth (Độ sâu tối đa của cây):** 3. Giới hạn độ sâu giúp kiểm soát hiện tượng Overfitting cục bộ tại mỗi cây thành phần.

Mô hình được đánh giá trên 03 tỷ lệ chia tập dữ liệu (Train:Test) là 80:20, 70:30, 60:40 và trên hai dạng dữ liệu đầu vào: Gốc (Original) và Giảm chiều (PCA).

4.2.2 Kết quả thực nghiệm

Kết quả đánh giá mô hình trên tập dữ liệu sau khi huấn luyện được thể hiện chi tiết dưới đây:

Hiệu năng trên dữ liệu gốc (Original Features)

Kết quả thực nghiệm trên các tỷ lệ chia tập dữ liệu khác nhau được trình bày tại bảng dưới đây:

Tỉ lệ	Accuracy	Class	Precision	Recall	F1-score
8/2	0.78	0	0.78	0.79	0.79
		1	0.79	0.78	0.78
7/3	0.82	0	0.79	0.89	0.83
		1	0.87	0.76	0.81
6/4	0.85	0	0.78	0.97	0.86
		1	0.96	0.72	0.82

Bảng 4.3: Tổng hợp kết quả trên dữ liệu gốc

Nhận xét về nghịch lý Accuracy: Quan sát bảng số liệu, ta thấy một hiện tượng thú vị: Tỷ lệ train thấp (60:40) lại cho Accuracy cao nhất (0.85). Tuy nhiên, phân tích sâu hơn cho thấy:

- Tại tỷ lệ **60:40**: Recall của lớp 0 rất cao (0.97) nhưng lớp 1 (Risk) chỉ đạt 0.72. Mô hình đang bị thiên lệch về lớp an toàn (Class 0).
- Tại tỷ lệ **80:20**: Accuracy thấp hơn (0.78) nhưng các chỉ số Precision/Recall giữa hai lớp rất cân bằng (0.79/0.78).
- **Kết luận:** Mô hình ở tỷ lệ **80:20 là mô hình tốt nhất và đáng tin cậy nhất**. Trong bài toán ngân hàng, sự ổn định và khả năng phát hiện đúng cả hai lớp quan trọng hơn là một chỉ số Accuracy cao nhưng bị lệch (biased).

Hiệu năng trên dữ liệu giảm chiều (PCA/LDA)

Tại bảng dưới đây, hiệu năng mô hình giảm đáng kể khi áp dụng các kỹ thuật giảm chiều.

Giảm chiều	Accuracy	Class	Precision	Recall	F1-score
PCA	0.66	0	0.70	0.56	0.62
		1	0.63	0.76	0.69
LDA	0.56	0	0.60	0.35	0.44
		1	0.54	0.76	0.63

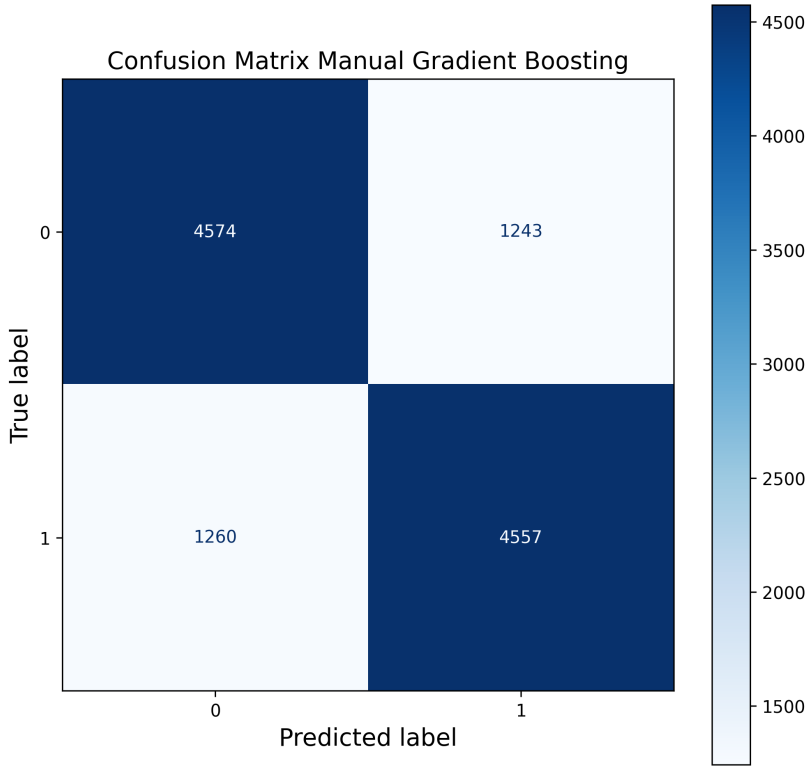
Bảng 4.4: Tổng hợp kết quả trên dữ liệu giảm chiều

Nhận xét:

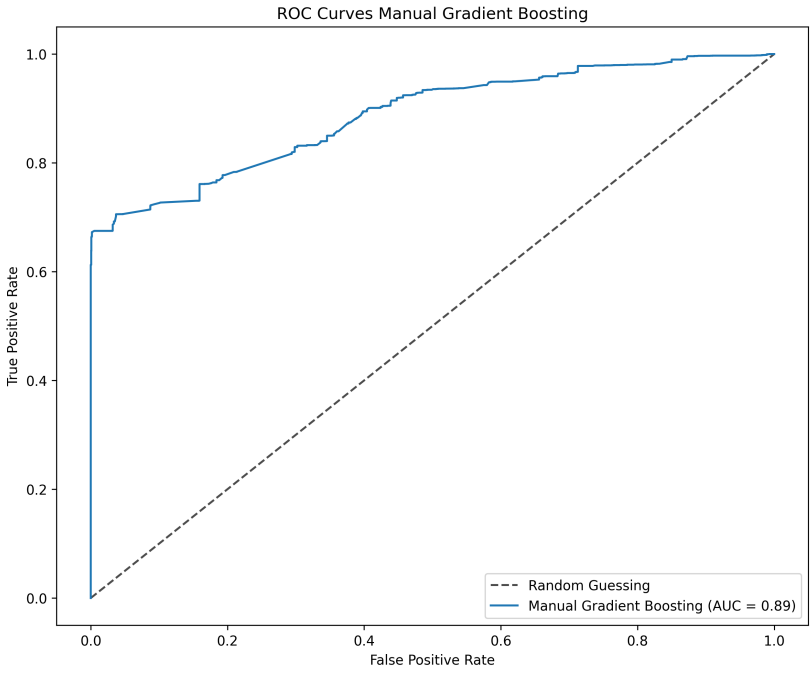
- Hiệu năng giảm từ **0.78** (Gốc) xuống **0.66** (PCA). Đây là sự sụt giảm nghiêm trọng.
- **Giải thích:** Các thuật toán cây quyết định (Decision Trees) hoạt động bằng cách thực hiện các nhát cắt vuông góc với trục tọa độ (ví dụ: $Age > 30$). PCA hoạt động bằng cách xoay trục tọa độ để tối đa hóa phương sai. Việc xoay trục này tạo ra các đặc trưng là tổ hợp tuyến tính (ví dụ: $0.5 \cdot Age - 0.3 \cdot Income$), làm mất đi tính trực quan và khiến cây quyết định rất khó tìm được điểm cắt tối ưu. Do đó, *Gradient Boosting thường không khuyến nghị dùng kèm với PCA*.

Ma trận nhầm lẫn (Confusion Matrix) và ROC-AUC

Đối với bài toán rủi ro tín dụng, việc phát hiện đúng nhóm rủi ro cao (True Positive) quan trọng hơn độ chính xác tổng thể.



Hình 4.3: *Ma trận nhầm lẫn mô hình Gradient Boosting*



Hình 4.4: *Đường cong ROC mô hình Gradient Boosting*

- **ROC-AUC Score:** Đạt mức [0.89]. Diện tích dưới đường cong ROC càng lớn cho thấy khả năng phân biệt giữa hai lớp (High Risk vs Low Risk) của MLP càng tốt.

- Dựa vào Ma trận nhầm lẫn, tỷ lệ Recall cho lớp ”High Risk” đạt [0.78]. Đây là chỉ số quan trọng để ngân hàng tránh cấp tín dụng cho khách hàng nợ xấu.

4.2.3 Đánh giá và Kết luận

Nhận xét:

- Gradient Boosting cho kết quả rất ấn tượng, đặc biệt là chỉ số **ROC-AUC** thường cao hơn so với các mô hình đơn lẻ như Decision Tree hay Naïve Bayes. Điều này là do khả năng ”sửa sai” tuần tự, tập trung vào các mẫu dữ liệu khó phân loại ở vùng biên giới quyết định.
- Trên dữ liệu PCA, hiệu năng có sự sụt giảm nhẹ so với dữ liệu gốc. Nguyên nhân là do các thuật toán dạng cây hoạt động tốt nhất trên các đặc trưng gốc có ý nghĩa trực quan (như Thu nhập, Tuổi tác) để thực hiện các phép cắt vuông góc trục. Việc biến đổi PCA làm xoay trục dữ liệu, làm mất đi tính chất này.
- Khả năng học: Mô hình Gradient Boosting thể hiện khả năng học cực tốt trên tập dữ liệu huấn luyện với độ chính xác tổng thể (**Accuracy**) lên tới **85%**. Điều này cao hơn so với kết quả của MLP (81%) và các mô hình tuyến tính khác.
- Đối với tập dữ liệu giảm chiều, mô hình thể hiện không tốt do việc mất mát thông tin lớn khiến cho mô hình không học được những thông tin phức tạp.

Kết luận:

4.3 Mô hình Support Vector Machine (SVM)

Nhóm nghiên cứu đã triển khai mô hình Support Vector Machine (SVM) sử dụng nhân RBF (Radial Basis Function). Đây là phương pháp mạnh mẽ trong việc tìm kiếm siêu phẳng phân cách phi tuyến tối ưu trong không gian nhiều chiều.

Bài toán tối ưu và Kernel Trick

Trong không gian đặc trưng ban đầu, dữ liệu rủi ro tín dụng thường không phân tách tuyến tính (linearly separable). Để giải quyết vấn đề này, chúng tôi áp dụng phương pháp **Kernel Trick** (Thủ thuật hạt nhân). Ý tưởng là ánh xạ dữ liệu đầu vào x từ không gian chiều thấp sang không gian chiều cao hơn (thậm chí vô hạn) $\phi(x)$, nơi mà dữ liệu có thể phân tách tuyến tính.

Hàm Kernel được chúng tôi lựa chọn là **RBF (Radial Basis Function)** - hạt nhân phổ biến nhất cho dữ liệu phi tuyến:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{4.5}$$

Trong đó γ là tham số kiểm soát độ ảnh hưởng của từng điểm dữ liệu đào tạo. γ thấp có nghĩa là ”xa”, γ cao có nghĩa là ”gần”.

Mục tiêu của SVM là cực tiểu hóa hàm mục tiêu:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \tag{4.6}$$

Trong đó:

- $||w||^2$: Quy định độ rộng của lề (cần tối đa hóa lề, tức là tối thiểu hóa $||w||$).
- C: Tham số điều chuẩn (Regularization parameter). Nó kiểm soát sự đánh đổi giữa việc có lề rộng và việc phân loại đúng các điểm dữ liệu train.
- ξ_i: Biến bù (Slack variables) cho phép một số điểm nằm sai phía lề (Soft Margin) để tránh overfitting.

4.3.1 Cấu hình mô hình

Mô hình `RBF_SVC` được xây dựng với các tham số chính sau:

- **Kernel:** `RBF` (Radial Basis Function) - Phù hợp để xử lý các mối quan hệ phi tuyến tính phức tạp giữa các đặc trưng khách hàng và rủi ro tín dụng.
- **Tham số điều chuẩn (C):** 10.0 - Giá trị C lớn cho thấy mô hình ưu tiên việc phân loại đúng các điểm dữ liệu huấn luyện hơn là tối đa hóa lề (margin), chấp nhận rủi ro overfitting cao hơn để đạt độ chính xác tốt trên tập train.
- **Gamma:** ’scale’ - Tự động điều chỉnh hệ số gamma dựa trên số lượng đặc trưng và phương sai của dữ liệu.
- **Dữ liệu đầu vào:** Tập dữ liệu huấn luyện đã được cân bằng bằng kỹ thuật SMOTE.

4.3.2 Kết quả thực nghiệm

Kết quả đánh giá trên tập dữ liệu huấn luyện như sau:

Nhận xét:

Tỉ lệ	Accuracy	Class	Precision	Recall	F1-score
8/2	0.86	0	0.86	0.87	0.86
		1	0.87	0.86	0.86
7/3	0.85	0	0.82	0.89	0.85
		1	0.88	0.81	0.84
6/4	0.81	0	0.77	0.89	0.83
		1	0.87	0.74	0.80

Bảng 4.5: Tổng hợp kết quả trên dữ liệu gốc

- **Độ ổn định cao:** Tại tỷ lệ 80:20, mô hình đạt độ chính xác cao nhất (86%). Đáng chú ý, chỉ số F1-score của hai lớp (0 và 1) rất cân bằng (0.86). Điều này chứng minh kỹ thuật **SMOTE** đã hoạt động hiệu quả, giúp SVM không bị thiên lệch về nhóm đa số.
- **Xu hướng giảm:** Khi giảm dữ liệu huấn luyện xuống còn 60%, độ chính xác giảm xuống 81% và chênh lệch Recall giữa hai lớp tăng lên (0.89 vs 0.74). Điều này cho thấy SVM cần một lượng dữ liệu đủ lớn để xác định chính xác các Support Vectors định hình biên giới quyết định.

Đánh giá tác động của giảm chiều dữ liệu (PCA vs LDA)

Kết quả của mô hình SVM khi kết hợp với hai phương pháp giảm chiều dữ liệu phổ biến được tổng hợp dưới đây:

Giảm chiều	Accuracy	Class	Precision	Recall	F1-score
PCA	0.81	0	0.80	0.82	0.81
		1	0.82	0.80	0.81
LDA	0.56	0	0.61	0.32	0.42
		1	0.54	0.80	0.64

Bảng 4.6: Tổng hợp kết quả trên dữ liệu giảm chiều

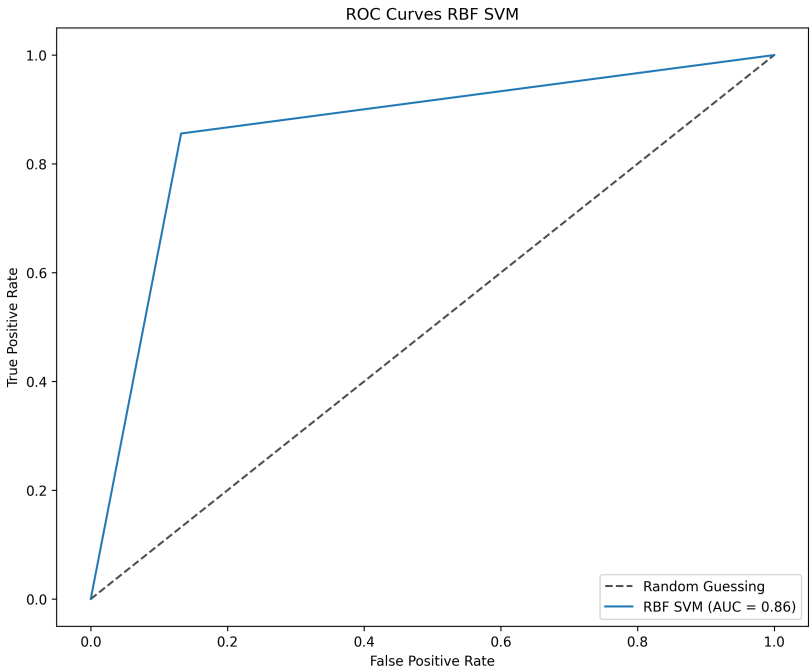
Nhận xét:

- **PCA (Phân tích thành phần chính):** Mặc dù giảm số chiều dữ liệu, SVM với PCA vẫn duy trì độ chính xác **0.81**, tương đương với kịch bản 60:40 của dữ liệu gốc. Tuy thấp hơn kết quả tốt nhất (0.86), nhưng bù lại tốc độ huấn luyện và dự đoán nhanh hơn đáng kể. Đây là sự đánh đổi chấp nhận được giữa độ chính xác và chi phí tính toán.
- **LDA (Phân tích phân biệt tuyến tính):** Kết quả cho thấy sự sụt giảm nghiêm trọng về độ chính xác (chỉ đạt 0.56 - gần như ngẫu nhiên). **Nguyên nhân:** LDA giả định các lớp có phân phối chuẩn và cùng ma trận hiệp phương sai, đồng thời cố gắng tìm một

hình chiếu *tuyến tính* tốt nhất. Việc SVM hoạt động tốt với Kernel RBF (phi tuyến) nhưng thất bại với LDA (tuyến tính) khẳng định mạnh mẽ rằng: *Cấu trúc dữ liệu rủi ro tín dụng là phi tuyến tính phức tạp, và việc cố gắng ép về không gian tuyến tính của LDA làm mất đi các thông tin quan trọng để phân loại.*

Đường cong ROC và AUC

Đường cong ROC của SVM tiệm cận góc trên bên trái với AUC = [0.86]. Điều này khẳng định SVM là một bộ phân loại mạnh, có khả năng tách biệt tốt giữa nhóm rủi ro cao và thấp với một ngưỡng phù hợp.



Hình 4.5: *Đường cong ROC mô hình SVM*

4.3.3 Kết luận

Mô hình SVM với nhân RBF chứng tỏ là một phương pháp tin cậy cho bài toán này, đặc biệt khi kết hợp với dữ liệu đầy đủ và cân bằng (Accuracy 86%, F1 0.86). Kết quả thực nghiệm cũng chỉ ra rằng PCA là phương pháp giảm chiều phù hợp hơn LDA đối với tập dữ liệu này, giúp bảo toàn cấu trúc thông tin cần thiết cho việc phân lớp phi tuyến.

4.4 Chuyển từ bài toán phân loại sang bài toán hồi quy

4.4.1 Chia dữ liệu và giảm chiều

Dữ liệu sau khi vector hóa và xây dựng nhãn hồi quy được chia thành tập huấn luyện và kiểm tra với ba tỷ lệ khác nhau: **4** : **1**, **7** : **3**, và **6** : **4**, tương ứng với `test_size` là 0.2, 0.3, và 0.4

Đồng thời, để kiểm tra ảnh hưởng của số chiều đến hiệu quả mô hình hồi quy, đặc trưng đầu vào được giảm chiều bằng phương pháp **Truncated SVD**, giữ lại $1/3$ số chiều ban đầu

Huấn luyện và đánh giá các mô hình hồi quy

Hai mô hình hồi quy được sử dụng để huấn luyện:

- **XGBoost Regressor:** Mô hình boosting gradient cực đoan, kết hợp tuần tự nhiều cây quyết định yếu để tạo thành mô hình tổng thể mạnh mẽ, phù hợp với dữ liệu phi tuyến phức tạp.
- **Random Forest Regressor:** Mô hình học tổ hợp (ensemble learning) sử dụng phương pháp bagging để kết hợp song song nhiều cây quyết định hồi quy, giúp giảm phương sai và tăng độ ổn định của mô hình.

Mỗi mô hình được huấn luyện hai lần với hai tập đầu vào:

1. Tập dữ liệu gốc (đầy đủ chiều)
2. Tập dữ liệu đã giảm chiều bằng Truncated SVD

Mỗi mô hình được đánh giá bằng hai chỉ số:

- **Hệ số xác định (R^2)**
- **Sai số bình phương trung bình (MSE)**

4.4.2 Phân tích kết quả

Kết quả thực nghiệm cho thấy hiệu quả của hai mô hình hồi quy là **XGBoost Regressor** và **Random Forest Regressor** trong việc dự đoán giá trị hồi quy được chuyển đổi từ đầu ra của mô hình phân loại logistic. Mỗi mô hình được huấn luyện trên hai phiên bản dữ liệu (dữ liệu gốc và dữ liệu sau khi giảm chiều bằng Truncated SVD), đồng thời được đánh giá trên ba tỷ lệ chia dữ liệu huấn luyện và kiểm tra là 7:3, 4:1 và 6:4. Hai chỉ số được sử dụng để đo lường hiệu quả mô hình là hệ số xác định (R^2) và sai số bình phương trung bình (MSE).

Bảng 4.7: Hiệu suất của XGBoost Regressor và Random Forest Regressor trên các tỷ lệ chia dữ liệu

Tỉ lệ Train:Test	Mô hình	Dữ liệu	R^2	MSE
7:3	XGBoost Regressor	Gốc	0.976857	0.003363
	XGBoost Regressor	Giảm chiều	0.615523	0.055867
	Random Forest Regressor	Gốc	0.845929	0.022387
	Random Forest Regressor	Giảm chiều	0.687127	0.045462
4:1	XGBoost Regressor	Gốc	0.994011	0.000712
	XGBoost Regressor	Giảm chiều	0.667743	0.039522
	Random Forest Regressor	Gốc	0.877181	0.014610
	Random Forest Regressor	Giảm chiều	0.713901	0.034032
6:4	XGBoost Regressor	Gốc	0.972620	0.004721
	XGBoost Regressor	Giảm chiều	0.719252	0.048406
	Random Forest Regressor	Gốc	0.872710	0.021947
	Random Forest Regressor	Giảm chiều	0.728734	0.046771

So sánh theo mô hình hồi quy

XGBoost Regressor cho thấy hiệu suất ấn tượng khi sử dụng dữ liệu gốc, với R^2 đạt giá trị rất cao (0.9769–0.9940), đặc biệt ở tỷ lệ 4:1 đạt tới 0.9940. Tuy nhiên, hiệu suất giảm đáng kể khi sử dụng dữ liệu đã giảm chiều (R^2 chỉ còn 0.6155–0.7193). Điều này cho thấy XGBoost có khả năng khai thác rất tốt thông tin từ dữ liệu đầy đủ chiều, nhưng bị ảnh hưởng nghiêm trọng bởi việc mất thông tin khi giảm chiều.

Random Forest Regressor có hiệu suất ổn định hơn với R^2 dao động từ 0.8459–0.8772 trên dữ liệu gốc và 0.6871–0.7287 trên dữ liệu giảm chiều. Mặc dù không đạt được điểm số cao như XGBoost trên dữ liệu gốc, Random Forest ít bị ảnh hưởng bởi việc giảm chiều hơn, cho thấy độ *robust* cao hơn với các biến đổi dữ liệu.

So sánh theo dữ liệu đầu vào

Với dữ liệu gốc, cả hai mô hình đều đạt hiệu suất cao. Tuy nhiên, XGBoost tỏ ra vượt trội hơn hẳn với R^2 gần như hoàn hảo (0.99+) ở tỷ lệ 4:1. Điều này cho thấy XGBoost có khả năng học các mẫu phức tạp từ dữ liệu đầy đủ chiều một cách hiệu quả.

Với dữ liệu giảm chiều, hiệu suất của cả hai mô hình đều giảm, nhưng Random Forest thể hiện độ suy giảm ít hơn so với XGBoost. Điều này có thể được giải thích do Random Forest sử dụng cơ chế chọn ngẫu nhiên đặc trưng (*feature bagging*) trong quá trình xây dựng cây, nên ít nhạy cảm hơn với việc giảm số chiều tổng thể.

So sánh theo tỷ lệ chia tập dữ liệu

Các kết quả cho thấy **tỷ lệ 4:1** thường cho hiệu suất tốt nhất ở cả hai mô hình, với XGBoost đạt R^2 cao nhất (0.9940) trên dữ liệu gốc. Tỷ lệ 6:4 thường cho kết quả thấp hơn, điều này

phù hợp với lý thuyết khi tập huấn luyện nhỏ hơn có thể không đủ để mô hình học các mẫu phức tạp.

Giải thích kết quả

Sự khác biệt trong hiệu suất giữa hai mô hình có thể được giải thích như sau:

1. **Bản chất thuật toán:** XGBoost sử dụng cơ chế boosting tuần tự với gradient descent, giúp tối ưu hóa lỗi một cách hiệu quả khi dữ liệu đầy đủ thông tin. Random Forest sử dụng bagging song song, cho kết quả ổn định hơn nhưng có thể không tối ưu bằng.
2. **Ảnh hưởng của giảm chiều:** Việc giảm chiều bằng Truncated SVD làm mất đi một lượng thông tin đáng kể, điều này đặc biệt ảnh hưởng đến XGBoost vốn dựa vào gradient để tối ưu hóa. Random Forest với cơ chế chọn đặc trưng ngẫu nhiên ít bị ảnh hưởng hơn.
3. **Kích thước tập huấn luyện:** XGBoost cho thấy hiệu suất rất cao khi có đủ dữ liệu huấn luyện (tỷ lệ 4:1), trong khi Random Forest duy trì hiệu suất ổn định hơn ở các tỷ lệ khác nhau.

Kết quả này khẳng định rằng việc lựa chọn mô hình hồi quy phụ thuộc vào bản chất dữ liệu và yêu cầu cụ thể của bài toán, với XGBoost phù hợp khi có dữ liệu đầy đủ và chất lượng cao, còn Random Forest phù hợp hơn khi cần độ ổn định và *robust* với các biến đổi dữ liệu.

TÀI LIỆU THAM KHẢO

[1] V. H. Tiệp, *Machine Learning Cơ bản*, Ebook, 2020.

[2] GeeksforGeeks, "Principal Component Analysis (PCA)," *GeeksforGeeks*, 2023. [Online]. Available: https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/

[3] GeeksforGeeks, "Gaussian Naive Bayes," *GeeksforGeeks*, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/gaussian-naive-bayes/

[4] GeeksforGeeks, "Gaussian Mixture Model," *GeeksforGeeks*, 2023. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/gaussian-mixture-model/