TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA TOÁN - CƠ - TIN HỌC

NHÓM 23

BÁO CÁO
MÔN: HỌC MÁY

DỰ BÁO VỀ KHẢ NĂNG NGHỈ VIỆC CỦA NHÂN VIÊN CÔNG TY IBM TẠI MỸ BẰNG MÔ HÌNH HỌC MÁY

GV Bộ Môn: Cao Văn Chung

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*	*Lớp*
Nguyễn Trọng Thành (TN)	23001934	K68A4
Nguyễn Văn Thắng	23001938	K68A4
Đinh Bá Thi	23001942	K68A4

Hà Nội, 2026

LỜI MỞ ĐẦU

Trong bối cảnh cạnh tranh toàn cầu hiện nay, nguồn nhân lực được xem là tài sản vô giá và là yếu tố then chốt quyết định sự thành công bền vững của một doanh nghiệp. Vấn đề giữ chân nhân tài và giảm tỷ lệ nghỉ việc (employee attrition) luôn là một trong những ưu tiên hàng đầu của bộ phận quản trị nhân sự (HR). Chi phí cho việc tuyển dụng, đào tạo nhân viên mới và những gián đoạn trong công việc do "chảy máu chất xám"gây ra là vô cùng tốn kém.

Đối với một tập đoàn công nghệ hàng đầu như IBM, việc hiểu rõ lý do tại sao nhân viên rời đi và dự đoán sớm những cá nhân có nguy cơ nghỉ việc mang một ý nghĩa chiến lược quan trọng. Nó cho phép công ty đưa ra các chính sách can thiệp kịp thời, cải thiện môi trường làm việc và tối ưu hóa chiến lược giữ chân nhân tài.

Với sự bùng nổ của khoa học dữ liệu, các kỹ thuật Học máy (Machine Learning) đã mở ra một phương pháp tiếp cận mới, hiệu quả hơn trong việc phân tích và dự báo các xu hướng nhân sự. Thay vì dựa trên cảm tính, chúng ta có thể xây dựng các mô hình dự báo dựa trên dữ liệu thực tế để tìm ra các yếu tố cốt lõi ảnh hưởng đến quyết định của nhân viên.

Chính vì lý do đó, chúng tôi đã chọn đề tài: **"Dự báo về khả năng nghỉ việc của nhân viên công ty IBM tại Mỹ bằng mô hình Học máy"** cho bài tiểu luận/dự án môn Học máy.

Mục tiêu của bài làm là áp dụng các kiến thức đã học để xây dựng, huấn luyện và đánh giá các mô hình phân loại khác nhau (như Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree, Support Vector Machine...). Đồng thời, bài làm cũng tập trung vào việc tiền xử lý dữ liệu, bao gồm các kỹ thuật chuẩn hóa, xử lý mất cân bằng và giảm chiều dữ liệu (Principal Component Analysis/Linear Discriminant Analysis/Feature Selection), để tìm

ra mô hình dự báo tối ưu nhất. Quan trọng hơn, nghiên cứu này mong muốn xác định được những yếu tố then chốt (như mức lương, số năm kinh nghiệm, sự hài lòng công việc...) có ảnh hưởng lớn nhất đến khả năng nghỉ việc tại IBM.

Để hoàn thành bài tiểu luận này, chúng tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất đến giảng viên – **Thầy Cao Văn Chung**. Thầy đã tận tình giảng dạy, định hướng và cung cấp những kiến thức quý báu trong suốt học phần, giúp chúng tôi có đủ nền tảng để thực hiện chủ đề này.

Chúng tôi xin chân thành cảm ơn!

MỤC LỤC

MỤC LỤC HÌNH ẢNH

DANH SÁCH BẢNG

Chương 1

Mở đầu

1.1 Đặt vấn đề

Trong bối cảnh cạnh tranh nhân tài khốc liệt hiện nay, quản lý nhân sự (Human Resource Management - HRM) không chỉ đơn thuần là các công tác hành chính, mà đã trở thành một bộ phận chiến lược, trực tiếp ảnh hưởng đến sự thành công của doanh nghiệp. Một trong những thách thức lớn nhất và tốn kém nhất mà bộ phận HRM phải đối mặt chính là tình trạng nhân viên nghỉ việc, hay còn gọi là "chảy máu chất xám".

Việc một nhân viên rời đi không chỉ tạo ra một khoảng trống về nhân sự mà nó còn kéo theo hàng loạt tác hại và chi phí khổng lồ: chi phí tuyển dụng mới, chi phí đào tạo nhân viên thay thế, thời gian để nhân viên mới đạt được năng suất tối đa, và sự gián đoạn trong các dự án đang vận hành. Đối với các tập đoàn công nghệ lớn như IBM, việc mất đi các kỹ sư và chuyên gia có kinh nghiệm còn có thể ảnh hưởng đến khả năng đổi mới và năng lực cạnh tranh.

Trước đây, các doanh nghiệp thường chỉ can thiệp "bị động"khi nhân viên đã nộp đơn xin nghỉ. Tuy nhiên, với sự phát triển của khoa học dữ liệu, một hướng tiếp cận mới, "chủ động" hơn đã ra đời. Bằng cách áp dụng các mô hình Học Máy (Machine Learning), chúng ta hiện có khả năng phân tích các dữ liệu nhân sự (như mức lương, thâm niên, sự hài lòng, số dự án...) để dự báo sớm những trường hợp có nguy cơ nghỉ việc cao.

Chính vì vậy, đề tài này thực hiện việc "Dự báo về khả năng nghỉ việc của nhân viên công ty IBM tại Mỹ" nhằm mục tiêu xây dựng một mô hình dự báo

hiệu quả. Mô hình này có thể giúp bộ phận HR của IBM xác định đúng các yếu tố then chốt dẫn đến việc nghỉ việc, từ đó đưa ra các chính sách can thiệp và giữ chân nhân tài kịp thời, giảm thiểu chi phí và ổn định tổ chức.

1.2 Mục tiêu nghiên cứu

Đề tài nghiên cứu này tập trung vào các giải quyết các vấn đề như sau:

1. Xây dựng và đánh giá các mô hình học máy để dự đoán khả năng nghỉ việc của nhân viên dựa trên dataset của IBM.
2. Áp dụng các kỹ thuật chuẩn hóa dữ liệu và giảm chiều để cải thiện hiệu suất mô hình và phân tích ảnh hưởng của chúng.

1.3 Cấu trúc của bài tiểu luận

Bài tiểu luận được tổ chức gồm năm chương chính như sau:

Chương 1 – Mở đầu: Trình bày bối cảnh, đặt vấn đề, mục tiêu nghiên cứu và phạm vi của đề tài, đồng thời phác thảo hướng tiếp cận tổng quan.

Chương 2 – Cơ sở lý thuyết: Giới thiệu các khái niệm và kiến thức nền tảng bao gồm bài toán phân loại, các mô hình học máy sử dụng (Naive Bayes, KNN, Logistic Regression, Decision Tree, SVM, MLP), các kỹ thuật tiền xử lý dữ liệu, giảm chiều (PCA, LDA, Feature Selection), xử lý mất cân bằng dữ liệu, các thuật toán phân cụm (K-Means, DBSCAN) và các thước đo đánh giá mô hình

Chương 3 – Phương pháp và Quy trình thực nghiệm: Trình bày bộ dữ liệu, quy trình tiền xử lý, quy trình thực nghiệm và phương pháp đánh giá mô hình.

Chương 4 – Kết quả và thảo luận: Cung cấp kết quả thử nghiệm của các mô hình, so sánh ưu nhược điểm, phân tích và lý giải hiệu suất của từng mô hình.

Chương 5 – Tổng kết: Tổng kết lại toàn bộ nội dung nghiên cứu, nêu các hạn chế của đề tài và đề xuất các hướng phát triển trong tương lai.

Chương 2

Cơ sở lý thuyết

2.1 Tổng quan về bài toán phân loại

2.1.1 Phân loại là gì?

Phân loại là một dạng bài toán học có giám sát (Supervised Learning). Mục tiêu của nó là dự đoán một nhãn lớp rời rạc cho một đối tượng đầu vào dựa trên các đặc trưng (features) của đối tượng đó.

Ví dụ: Dựa vào nội dung email (đặc trưng), mô hình dự đoán email đó thuộc nhóm "Thư rác" (Spam) hay "Không phải thư rác" (Not spam).

2.1.2 Các loại bài toán phân loại

Bài toán phân loại được chia thành ba dạng chính dựa trên số lượng và tính chất của các nhãn lớp:

Phân loại nhị phân (Binary Classification)

Đây là dạng bài toán cơ bản nhất, trong đó tập nhãn lớp chỉ bao gồm hai lớp (ví dụ: $Y = \{0, 1\}$ hoặc $Y = \{-1, 1\}$). Các lớp này thường mang tính đối lập hoặc loại trừ lẫn nhau.

Ví dụ 1: Chẩn đoán y tế (Bệnh / Không bệnh).

Ví dụ 2: Phát hiện giao dịch gian lận (Gian lận / Không gian lận).

Phân loại đa lớp (Multiclass Classification)

Bài toán mà tập nhãn có nhiều hơn hai lớp (ví dụ: $Y = \{A, B, C\}$), và mỗi mẫu dữ liệu đầu vào chỉ thuộc về duy nhất một lớp trong số đó.

Ví dụ 1: Nhận dạng chữ số viết tay (các lớp là {0, 1, 2, ..., 9}).

Phân loại đa nhãn (Multilabel Classification)

Đây là dạng bài toán phức tạp hơn, trong đó mỗi mẫu dữ liệu đầu vào có thể được gán một hoặc nhiều nhãn cùng một lúc. Các nhãn không nhất thiết phải loại trừ lẫn nhau.

Ví dụ 1: Gán thể loại cho một bộ phim (có thể là {Hành động, Hài hước, Lãng mạn} cùng lúc).

Ví dụ 2: Phân loại chủ đề cho một tài liệu khoa học (có thể thuộc cả {Học máy} và {Y sinh}).

2.1.3 Các thuật toán phân loại phổ biến

1. Naive Bayes
2. K-Nearest Neighbors
3. Hồi quy Logistic (Logistic Regression)
4. Máy Vector hỗ trợ (Support Vector Machine - SVM)
5. Cây quyết định (Decision Tree)
6. Rừng ngẫu nhiên (Random Forest)
7. Mạng nơ-ron (Neural Networks)
8. Và một số thuật toán khác ...

2.2 Các mô hình học máy được sử dụng

2.2.1 Naive Bayes

Naive Bayes Classifier

Naive Bayes Classifier (NBC) là thuật toán phân loại dựa trên định lý Bayes và giả định độc lập có điều kiện giữa các đặc trưng. Mục tiêu là tìm nhãn lớp c sao cho xác suất hậu nghiệm $p(c|\mathbf{x})$ là lớn nhất [1].

Giả thuyết "ngây ngô"(Naive) cốt lõi của thuật toán cho rằng: Nếu biết trước nhãn lớp c, các thành phần $x_1, \ldots, x_d$ của dữ liệu đầu vào hoàn toàn độc lập với nhau:

$$p(\mathbf{x}|c) = \prod_{i=1}^{d} p(x_i|c) \tag{2.1}$$

Dựa trên quy tắc Bayes và giả thuyết độc lập, nhãn của điểm dữ liệu $\mathbf{x}$ được xác định bởi:

$$c = \arg \max_{c \in \{1,\ldots,C\}} p(c) \prod_{i=1}^{d} p(x_i|c) \tag{2.2}$$

Để tránh sai số tính toán khi nhân nhiều số xác suất quá nhỏ (underflow), ta chuyển sang không gian Logarit (hàm log đồng biến nên không ảnh hưởng kết quả tối ưu):

$$c = \arg \max_{c \in \{1,\ldots,C\}} \left(\log(p(c)) + \sum_{i=1}^{d} \log(p(x_i|c)) \right) \tag{2.3}$$

Trong đó:

- $p(c)$: Xác suất tiên nghiệm (Prior), tính bằng tần suất xuất hiện của lớp c.
- $p(x_i|c)$: Xác suất có điều kiện (Likelihood), phụ thuộc vào loại dữ liệu.

Gaussian Naive Bayes

Áp dụng cho dữ liệu liên tục. Giả định x_i tuân theo phân phối chuẩn $\mathcal{N}(\mu_{ci}, \sigma_{ci}^2)$:

$$p(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}\right) \tag{2.4}$$

Các tham số trung bình μ_{ci} và phương sai σ_{ci}^2 được ước lượng từ dữ liệu huấn luyện.

Bernoulli Naive Bayes

Áp dụng cho dữ liệu nhị phân (0 hoặc 1), ví dụ như sự xuất hiện của từ trong văn bản (có/không):

$$p(x_i|c) = p(i|c)^{x_i}(1 - p(i|c))^{1-x_i} \tag{2.5}$$

NBC có tốc độ huấn luyện và kiểm thử cực nhanh, hoạt động hiệu quả trên dữ liệu lớn (large-scale) và đặc biệt tốt trong các bài toán phân loại văn bản (spam filtering), mặc dù giả thuyết độc lập hiếm khi đúng trong thực tế.

2.2.2 K-Nearest Neighbors (KNN)

Giới thiệu

K-Nearest Neighbors (KNN) là một trong những thuật toán **Supervised Learning** đơn giản nhất [2].

Thuật toán này thuộc loại "học lười" (lazy learning), tức là nó không học bất cứ điều gì trong giai đoạn training. Thay vào đó, nó lưu trữ toàn bộ dữ liệu training.

Mọi tính toán (chủ yếu là tính khoảng cách) chỉ được thực hiện khi cần dự đoán kết quả cho một điểm dữ liệu mới.

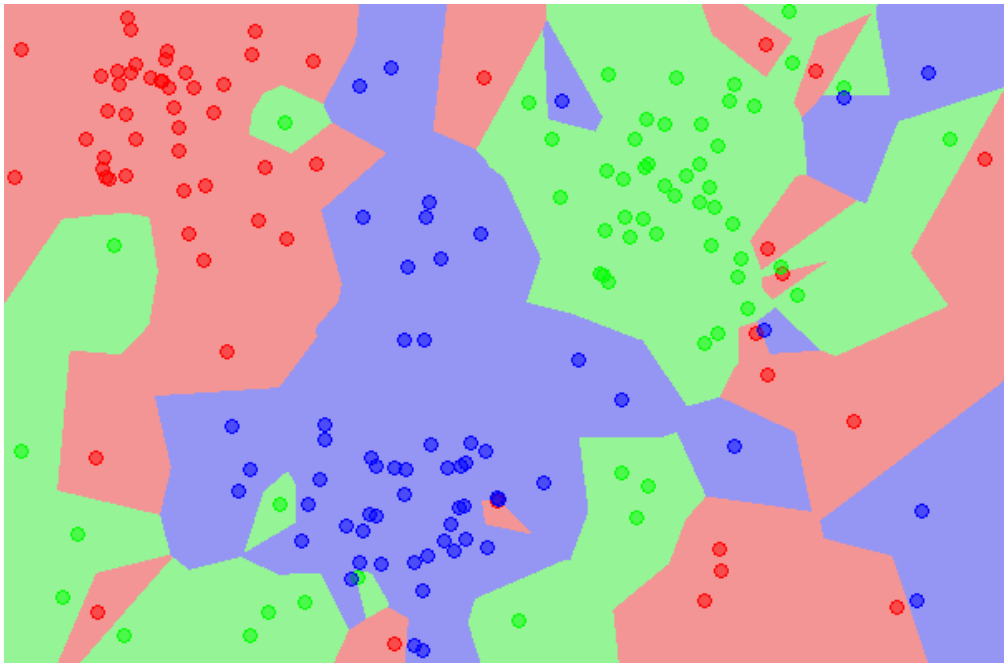
KNN có thể được áp dụng cho cả bài toán classification (phân loại) và regression (hồi quy).

Cách hoạt động

Với classification: Nhãn của một điểm dữ liệu mới được quyết định dựa trên K điểm lân cận gần nó nhất trong tập training. Quyết định này thường dựa trên "bầu chọn theo đa số" (majority voting).

Với regression: Đầu ra của điểm mới thường là giá trị trung bình (hoặc trung bình có trọng số) của đầu ra của K điểm lân cận gần nhất.

KNN chỉ dựa vào thông tin của K lân cận, bất kể việc một vài điểm trong đó có thể là nhiễu.



Hình 2.1: Minh họa KNN với K=1 cho bài toán 3 lớp. Các vùng nhỏ xen kẽ (như vùng Lục) có thể là nhiễu, dễ gây dự đoán sai.

Về khoảng cách: "Khoảng cách" trong KNN có thể được định nghĩa bằng nhiều hàm số khác nhau (ví dụ: khoảng cách Euclidean, Manhattan, v.v.).

2.2.3 Logistic Regression

Giới thiệu

Các mô hình tuyến tính (linear models) có dạng chung $y = f(\mathbf{w}^T\mathbf{x})$.

- **Linear Regression:** $f(s) = s$. Đầu ra là số thực không bị chặn.
- **PLA:** $f(s) = \text{sgn}(s)$. Đầu ra là $\{-1, 1\}$.

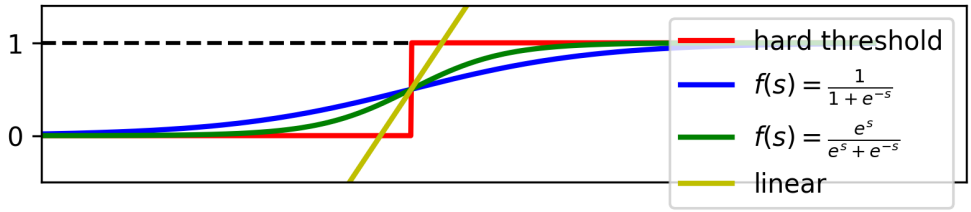
- **Logistic Regression:** Cung cấp đầu ra dạng xác suất, bị chặn trong khoảng $[0, 1]$. Mặc dù có tên *regression*, mô hình này thường được dùng cho bài toán *classification* [3].

Mô hình Logistic Regression

Đầu ra dự đoán có dạng:

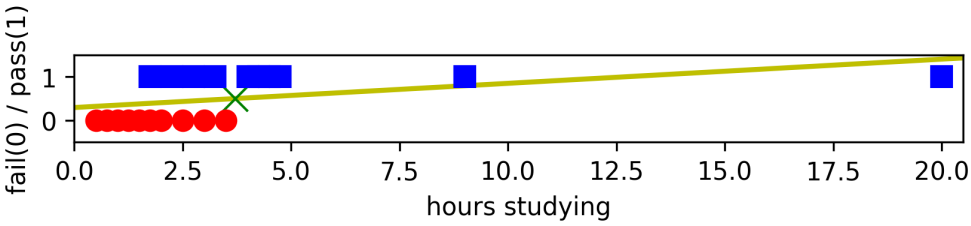
$$f(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$$

Trong đó θ là *logistic function*.



Hình 2.2: Các activation function khác nhau.

Linear regression (đường vàng) không phù hợp cho bài toán phân loại nhị phân vì nó không bị chặn và bị ảnh hưởng bởi các điểm dữ liệu ngoại lai (outliers), làm lệch ngưỡng quyết định.



Hình 2.3: Tại sao Linear Regression không phù hợp?

Chúng ta cần một hàm *mượt* (smooth, có đạo hàm) và bị chặn, ví dụ như các đường màu xanh.

Sigmoid function

Hàm được sử dụng nhiều nhất là hàm *sigmoid*:

$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s)$$

Hàm này bị chặn trong khoảng $(0, 1)$ và có một tính chất đặc biệt về đạo hàm:

$$\begin{aligned}\sigma'(s) &= \frac{e^{-s}}{(1+e^{-s})^2} \\ &= \sigma(s)(1-\sigma(s))\end{aligned}$$

Công thức đạo hàm đơn giản này rất hữu ích cho việc tối ưu.

Xây dựng hàm mất mát

Ta diễn giải đầu ra của mô hình là xác suất:

$$P(y_i = 1|\mathbf{x}_i; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_i) \triangleq z_i \tag{2.6}$$

$$P(y_i = 0|\mathbf{x}_i; \mathbf{w}) = 1 - \sigma(\mathbf{w}^T\mathbf{x}_i) = 1 - z_i \tag{2.7}$$

Hai biểu thức trên có thể viết gộp thành:

$$P(y_i|\mathbf{x}_i; \mathbf{w}) = z_i^{y_i}(1-z_i)^{1-y_i}$$

Chúng ta muốn tìm $\mathbf{w}$ để tối đa hóa xác suất này trên toàn bộ tập dữ liệu (bài toán *Maximum Likelihood Estimation - MLE*). Giả sử các điểm dữ liệu là độc lập:

$$P(\mathbf{y}|\mathbf{X}; \mathbf{w}) = \prod_{i=1}^{N} P(y_i|\mathbf{x}_i; \mathbf{w}) = \prod_{i=1}^{N} z_i^{y_i}(1-z_i)^{1-y_i}$$

Để đơn giản hóa việc tính toán (biến phép nhân thành phép cộng) và tránh lỗi số học, ta lấy *negative log-likelihood* và coi đó là hàm mất mát $J(\mathbf{w})$ cần tối ưu

(minimize). Hàm này còn gọi là *Cross Entropy*:

$$\begin{aligned} J(\mathbf{w}) &= -\log P(\mathbf{y}|\mathbf{X};\mathbf{w}) \\ &= -\sum_{i=1}^{N}(y_i \log z_i + (1-y_i)\log(1-z_i)) \end{aligned}$$

Tối ưu hàm mất mát

Chúng ta sử dụng *Stochastic Gradient Descent* (SGD). Xét hàm mất mát với 1 điểm dữ liệu $(\mathbf{x}_i, y_i)$:

$$J_i(\mathbf{w}) = -(y_i \log z_i + (1-y_i)\log(1-z_i))$$

Ta cần tính đạo hàm $\nabla_{\mathbf{w}} J_i(\mathbf{w})$.

$$\frac{\partial J_i}{\partial \mathbf{w}} = \frac{\partial J_i}{\partial z_i}\frac{\partial z_i}{\partial s_i}\frac{\partial s_i}{\partial \mathbf{w}}$$

với $s_i = \mathbf{w}^T\mathbf{x}_i$ và $z_i = \sigma(s_i)$.

- $\frac{\partial J_i}{\partial z_i} = -\left(\frac{y_i}{z_i} - \frac{1-y_i}{1-z_i}\right) = -\frac{y_i - z_i}{z_i(1-z_i)}$
- $\frac{\partial z_i}{\partial s_i} = \sigma'(s_i) = z_i(1-z_i)$ (tính chất đặc biệt của sigmoid)
- $\frac{\partial s_i}{\partial \mathbf{w}} = \mathbf{x}_i$

Kết hợp lại, đạo hàm được rút gọn rất đẹp:

$$\frac{\partial J_i}{\partial \mathbf{w}} = -\frac{y_i - z_i}{z_i(1-z_i)} \cdot (z_i(1-z_i)) \cdot \mathbf{x}_i = -(y_i - z_i)\mathbf{x}_i = (z_i - y_i)\mathbf{x}_i$$

Công thức cập nhật (theo SGD) cho Logistic Regression là:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta(z_i - y_i)\mathbf{x}_i \text{ (hoặc } \mathbf{w} \leftarrow \mathbf{w} + \eta(y_i - z_i)\mathbf{x}_i)$$

Tính chất: Boundary tuyến tính

Logistic Regression được dùng cho classification. Ta dự đoán $y = 1$ nếu $P(y = 1|\mathbf{x}; \mathbf{w}) > 0.5$.

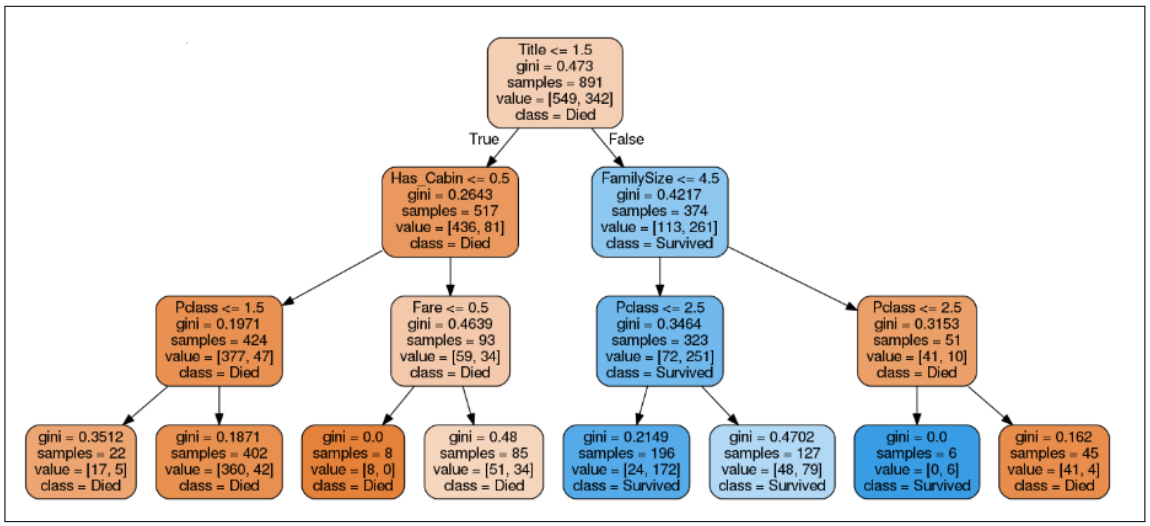
$$\begin{aligned} & P(y = 1|\mathbf{x}; \mathbf{w}) > 0.5 \\ \Leftrightarrow & \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} > 0.5 \\ \Leftrightarrow & 1 > 0.5(1 + e^{-\mathbf{w}^T\mathbf{x}}) \\ \Leftrightarrow & 2 > 1 + e^{-\mathbf{w}^T\mathbf{x}} \\ \Leftrightarrow & 1 > e^{-\mathbf{w}^T\mathbf{x}} \\ \Leftrightarrow & e^0 > e^{-\mathbf{w}^T\mathbf{x}} \\ \Leftrightarrow & 0 > -\mathbf{w}^T\mathbf{x} \\ \Leftrightarrow & \mathbf{w}^T\mathbf{x} > 0 \end{aligned}$$

Đường ranh giới (boundary) phân chia hai lớp là siêu mặt phẳng $\mathbf{w}^T\mathbf{x} = 0$. Đây là một boundary có dạng tuyến tính.

2.2.4 Decision Tree

Giới thiệu

Decision Tree là thuật toán học có giám sát dạng cây, có thể giải quyết cả bài toán regression (hồi quy) và classification (phân loại) [4]. Decision Tree hoạt động bằng cách chia dữ liệu thành các tập con thuần nhất hơn dựa trên các thuộc tính đầu vào. Ví dụ, như với dữ liệu Titanic, thuật toán Decision Tree sẽ học ra mô hình dạng cây như thế này.



Hình 2.4: Sơ đồ cây quyết định với dữ liệu Titanic

- **Nút gốc (Root node)**: chứa toàn bộ tập dữ liệu.
- **Nút nội bộ (Internal nodes)**: đại diện cho các quyết định dựa trên thuộc tính.
- **Nhánh (Branches)**: đại diện cho kết quả của quyết định.
- **Nút lá (Leaf nodes)**: chứa nhãn dự đoán cuối cùng.

Tiêu chí phân tách (Độ không thuần khiết - Impurity)

Quá trình xây dựng cây quyết định liên quan đến việc tối ưu hóa một hàm mất mát, thường là giảm thiểu độ không thuần khiết (impurity) của các nút. Mục tiêu là chọn thuộc tính và ngưỡng phân tách để tạo ra các nút con thuần khiết nhất.

Chỉ số Gini (Gini Impurity)

Chỉ số Gini là một cách để đo độ không thuần khiết của một nút. Một nút thuần khiết (tất cả các mẫu đều thuộc một lớp) sẽ có chỉ số Gini bằng 0.

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2 \tag{2.8}$$

Trong đó:

- p_i là xác suất của việc một mẫu thuộc lớp i tại nút t.
- C là tổng số lớp.

Entropy

Entropy là một thước đo khác về độ không thuần khiết, đo mức độ hỗn loạn hoặc không chắc chắn của một tập hợp các mẫu.

$$Entropy(t) = -\sum_{i=1}^{C} p_i \log_2(p_i) \tag{2.9}$$

Tương tự như chỉ số Gini, Entropy đạt giá trị nhỏ nhất khi nút hoàn toàn thuần khiết.

Giảm độ không thuần khiết (Information Gain)

Khi một đặc trưng được chọn để phân tách, mục tiêu là làm giảm độ không thuần khiết của các nút con so với nút cha. Sự giảm này được gọi là Information Gain (đối với Entropy) hoặc Gini Gain (đối với chỉ số Gini).

$$InformationGain = Impurity(t) - \sum_{k \in \{\text{trái, phải}\}} \frac{N_k}{N} Impurity(t_k) \tag{2.10}$$

Trong đó:

- $Impurity(t)$ là độ không thuần khiết tại nút cha.
- $Impurity(t_k)$ là độ không thuần khiết tại các nút con k sau khi phân tách.
- N là số lượng mẫu tại nút cha, và N_k là số lượng mẫu tại nút con k.

Cây quyết định sẽ chọn thuộc tính mang lại Information Gain lớn nhất.

2.2.5 Multi-Layer Perceptron (MLP)

Giới thiệu

Các mô hình tuyến tính đơn giản (như Perceptron Learning Algorithm - PLA hay Logistic Regression) chỉ giải quyết được các bài toán phân biệt tuyến

tính. Chúng thất bại trước các dữ liệu phân bố phức tạp, ví dụ điển hình là bài toán XOR (không thể kẻ một đường thẳng để chia cắt hai lớp).

Multi-Layer Perceptron (MLP), hay còn gọi là mạng nơ-ron truyền thẳng (Feedforward Neural Network), khắc phục nhược điểm này bằng cách ghép nhiều tầng (layer) các nơ-ron lại với nhau, tạo nên các đường phân chia phi tuyến tính phức tạp [5].

Cấu trúc của MLP

Một mạng MLP cơ bản bao gồm 3 thành phần chính:

1. **Input Layer (Lớp đầu vào):** Nhận dữ liệu đầu vào $\mathbf{x}$.
2. **Hidden Layers (Các lớp ẩn):** Nằm giữa đầu vào và đầu ra. Tại đây các phép biến đổi phi tuyến diễn ra. Một mạng có thể có một hoặc nhiều lớp ẩn.
3. **Output Layer (Lớp đầu ra):** Trả về kết quả dự đoán (class score hoặc giá trị hồi quy).

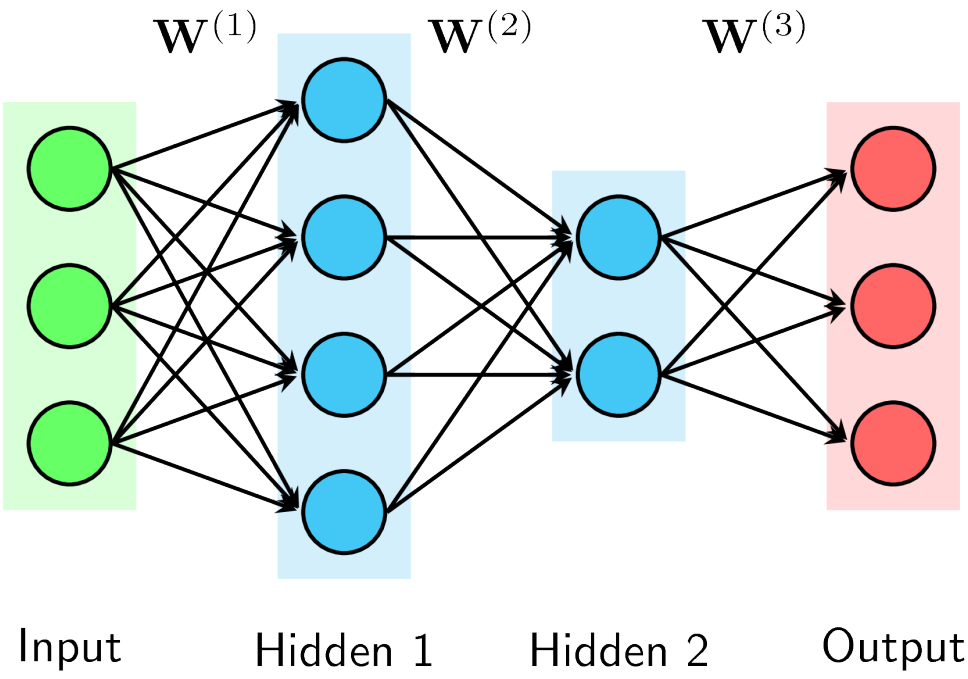
Quá trình tính toán tại mỗi nơ-ron diễn ra theo 2 bước:

- **Tổng hợp tuyến tính:** Tính tổng có trọng số của các đầu vào cộng với bias.
$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{2.11}$$
- **Kích hoạt phi tuyến:** Áp dụng hàm kích hoạt (activation function) lên kết quả vừa tính.
$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}) \tag{2.12}$$

Trong đó $\mathbf{W}^{(l)}, \mathbf{b}^{(l)}$ là ma trận trọng số và bias tại lớp thứ l, $\mathbf{a}^{(l-1)}$ là đầu ra của lớp trước đó.

Hàm kích hoạt (Activation Functions)

Hàm kích hoạt là thành phần quan trọng nhất giúp MLP có khả năng học các quan hệ phi tuyến. Nếu không có hàm này, MLP dù có bao nhiêu lớp cũng chỉ tương đương với một hàm tuyến tính đơn giản. Các hàm phổ biến gồm:



Hình 2.5: MLP với hai hidden layers (các biases đã bị ẩn)

- **Sigmoid:** $f(z) = \frac{1}{1+e^{-z}}$ (đưa giá trị về khoảng $(0, 1)$).
- **Tanh:** $f(z) = \tanh(z)$ (đưa giá trị về khoảng $(-1, 1)$).
- **ReLU (Rectified Linear Unit):** $f(z) = \max(0, z)$. Đây là hàm được sử dụng phổ biến nhất hiện nay do tính toán đơn giản và hạn chế triệt tiêu đạo hàm (vanishing gradient).

Huấn luyện mô hình (Backpropagation)

Quá trình huấn luyện MLP dựa trên việc tối thiểu hóa hàm mất mát (Loss Function), ví dụ như Cross Entropy cho bài toán phân loại. Thuật toán chính được sử dụng là **Lan truyền ngược (Backpropagation)**.

Cơ chế:

- **Feedforward:** Tính toán output dự đoán từ input.
- **Tính Loss:** So sánh dự đoán với nhãn thực tế.
- **Backpropagation:** Sử dụng quy tắc chuỗi (Chain Rule) để tính đạo hàm (gradient) của hàm mất mát theo từng trọng số từ lớp cuối cùng ngược về lớp đầu tiên.

- **Cập nhật trọng số:** Sử dụng Gradient Descent để điều chỉnh $\mathbf{W}$ và $\mathbf{b}$ nhằm giảm thiểu sai số.

2.2.6 Support Vector Machine (SVM)

Giới thiệu

Support Vector Machine (SVM) là một thuật toán học máy giám sát (Supervised Learning) được sử dụng cho bài toán phân lớp (classification) hoặc hồi quy (regression). Ý tưởng cốt lõi của SVM là tìm một không gian phân chia (siêu mặt phẳng - hyperplane) sao cho khoảng cách giữa các điểm dữ liệu gần nhất của các lớp khác nhau tới mặt phẳng này là lớn nhất. Khoảng cách này được gọi là Margin (lề) [6].

Khoảng cách từ một điểm tới một siêu mặt phẳng

Trong không gian d chiều, một siêu mặt phẳng được định nghĩa bởi phương trình:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.13}$$

Trong đó:

- $\mathbf{w} \in \mathbb{R}^d$ là vector pháp tuyến của siêu mặt phẳng.
- $b \in \mathbb{R}$ là bias (độ lệch).

Khoảng cách từ một điểm $\mathbf{x}_0$ bất kỳ tới siêu mặt phẳng này được tính theo công thức:

$$d(\mathbf{x}_0, \text{Hyperplane}) = \frac{|\mathbf{w}^T\mathbf{x}_0 + b|}{||\mathbf{w}||_2} \tag{2.14}$$

Giả định bài toán phân lớp nhị phân

Giả sử chúng ta có tập dữ liệu training: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, với $y_i \in \{-1, 1\}$. Mục tiêu là tìm $\mathbf{w}$ và b sao cho:

- $\mathbf{w}^T\mathbf{x}_i + b \geq 1$ nếu $y_i = 1$
- $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ nếu $y_i = -1$

Điều kiện này có thể viết gọn lại thành:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i = 1, \ldots, N \tag{2.15}$$

Tính toán Margin

Các điểm nằm gần siêu mặt phẳng nhất (gọi là *Support Vectors*) sẽ thỏa mãn $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$. Khoảng cách từ các điểm này tới siêu mặt phẳng phân chia là:

$$\frac{1}{||\mathbf{w}||_2} \tag{2.16}$$

Do đó, độ rộng của lề (tổng khoảng cách từ 2 phía) là:

$$\text{Margin} = \frac{2}{||\mathbf{w}||_2} \tag{2.17}$$

Thiết lập hàm mục tiêu

Để tối đa hóa Margin, ta cần tối đa hóa $\frac{2}{||\mathbf{w}||_2}$, điều này tương đương với việc tối thiểu hóa $||\mathbf{w}||_2$. Để thuận tiện cho việc đạo hàm, ta tối thiểu hóa $\frac{1}{2}||\mathbf{w}||_2^2$.

Bài toán tối ưu (Primal Problem) trở thành:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||_2^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \ \forall i = 1, \ldots, N \end{aligned} \tag{2.18}$$

2.2.7 Soft Margin Support Vector Machine

Giới hạn của Hard Margin SVM

Trong thực tế, dữ liệu thường bị nhiễu (noise) hoặc chứa các điểm ngoại lai (outliers), khiến cho hai lớp dữ liệu không thể phân biệt tuyến tính hoàn toàn (not linearly separable). Khi đó, bài toán Hard Margin SVM sẽ vô nghiệm. Để giải quyết vấn đề này, Soft Margin SVM [7] được đề xuất nhằm nới lỏng các ràng buộc, chấp nhận một số điểm dữ liệu có thể bị phân lớp sai hoặc nằm trong vùng an toàn (margin).

Biến bù (Slack Variables)

Để cho phép các vi phạm xảy ra, ta đưa vào các biến bù không âm $\xi_i \geq 0$ (slack variables) cho từng điểm dữ liệu $\mathbf{x}_i$. Ràng buộc của bài toán thay đổi thành:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1, \ldots, N \tag{2.19}$$

Trong đó:

- Nếu $\xi_i = 0$: Điểm dữ liệu được phân loại đúng và nằm an toàn phía ngoài margin (hoặc trên biên).
- Nếu $0 < \xi_i < 1$: Điểm nằm trong vùng margin nhưng vẫn được phân loại đúng.
- Nếu $\xi_i \geq 1$: Điểm bị phân loại sai.

Bài toán tối ưu (Primal Problem)

Mục tiêu mới là vừa tối đa hóa độ rộng lề (tối thiểu $||\mathbf{w}||$), vừa tối thiểu hóa sai số phân lớp (tổng các ξ_i). Hàm mục tiêu trở thành:

$$\underset{\mathbf{w},b,\xi}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{N}\xi_i \tag{2.20}$$

Thỏa mãn: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ và $\xi_i \geq 0$.

Vai trò của tham số C

C là một hằng số dương do người dùng thiết lập, đóng vai trò cân bằng (trade-off) giữa độ rộng của margin và sai số huấn luyện:

- C lớn: Phạt nặng các sai số. Mô hình cố gắng phân loại đúng tất cả các điểm, dẫn đến margin hẹp (Narrow Margin). Dễ bị Overfitting.
- C nhỏ: Chấp nhận nhiều sai số hơn để có margin rộng hơn (Wide Margin). Dữ liệu tổng quát hóa tốt hơn nhưng có thể bị Underfitting.

Bài toán đối ngẫu (Dual Problem)

Sử dụng phương pháp nhân tử Lagrange, bài toán gốc được chuyển về bài toán đối ngẫu để loại bỏ $\mathbf{w}, b$ và ξ. Mục tiêu là tìm các nhân tử Lagrange λ:

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{N} \lambda_i y_i = 0 \\ & 0 \leq \lambda_i \leq C, \quad \forall i = 1, \ldots, N \end{aligned} \tag{2.21}$$

Nhận xét quan trọng: Sự khác biệt duy nhất giữa bài toán đối ngẫu của Soft Margin và Hard Margin nằm ở ràng buộc của λ_i.

- Hard Margin: $0 \leq \lambda_i$
- Soft Margin: $0 \leq \lambda_i \leq C$ (chặn trên bởi C).

Biểu thức xuất hiện tích vô hướng $\mathbf{x}_i^T \mathbf{x}_j$ chính là cơ sở để áp dụng Kernel Trick cho các bài toán phi tuyến sau này.

Góc nhìn qua hàm Hinge Loss

Bài toán Soft Margin SVM cũng có thể được coi là bài toán tối thiểu hóa hàm mất mát dạng Hinge Loss kèm theo Regularization:

$$\mathcal{L} = \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \tag{2.22}$$

Trong đó $\lambda = \frac{1}{C}$. Cách nhìn này cho thấy mối liên hệ giữa SVM và các thuật toán Machine Learning khác như Neural Networks.

2.2.8 Kernel Support Vector Machine

Giới hạn của SVM tuyến tính

Các phương pháp Hard Margin và Soft Margin SVM chỉ làm việc hiệu quả khi dữ liệu phân biệt tuyến tính (linearly separable) hoặc gần phân biệt tuyến

tính. Đối với các bài toán mà dữ liệu phân bố phức tạp (ví dụ: bài toán XOR), ta không thể tìm được một siêu mặt phẳng phân chia trong không gian ban đầu.

Ý tưởng: Không gian đặc trưng (Feature Space)

Ý tưởng cốt lõi của Kernel SVM [8] là ánh xạ dữ liệu từ không gian ban đầu (Input Space) sang một không gian mới có số chiều cao hơn (Feature Space), thậm chí là vô hạn chiều. Tại không gian này, dữ liệu hy vọng sẽ trở nên phân biệt tuyến tính.

Gọi hàm ánh xạ là $\Phi : \mathbf{x} \to \Phi(\mathbf{x})$. Bài toán tối ưu đối ngẫu (Dual Problem) thay đổi từ việc sử dụng tích vô hướng $\mathbf{x}_i^T \mathbf{x}_j$ sang:

$$\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \tag{2.23}$$

Kernel Trick (Thủ thuật Kernel)

Việc tính toán trực tiếp $\Phi(\mathbf{x})$ và tích vô hướng trong không gian cao chiều tốn kém chi phí tính toán và bộ nhớ. Kernel Trick cho phép ta tính tích vô hướng trong không gian mới ngay tại không gian ban đầu thông qua hàm Kernel $k(\cdot, \cdot)$:

$$k(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x})^T \Phi(\mathbf{z}) \tag{2.24}$$

Nhờ đó, ta không cần xác định cụ thể hàm $\Phi(\mathbf{x})$ mà chỉ cần biết hàm $k(\mathbf{x}, \mathbf{z})$.

Bài toán tối ưu trở thành:

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{N} \lambda_i y_i = 0, \quad 0 \leq \lambda_i \leq C \end{aligned} \tag{2.25}$$

Điều kiện Mercer

Để một hàm $k(\mathbf{x}, \mathbf{z})$ là một Kernel hợp lệ, nó cần thỏa mãn điều kiện Mercer (đảm bảo ma trận Kernel là nửa xác định dương), giúp bài toán tối ưu lồi và có nghiệm duy nhất.

Các loại Kernel phổ biến

- **Linear Kernel:** Dành cho dữ liệu gần phân biệt tuyến tính.

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z} \tag{2.26}$$

- **Polynomial Kernel:** Phù hợp cho dữ liệu có biên giới hạn cong đa thức.

$$k(\mathbf{x}, \mathbf{z}) = (\gamma \mathbf{x}^T \mathbf{z} + r)^d \tag{2.27}$$

  Trong đó d là bậc của đa thức.

- **Radial Basis Function (RBF) Kernel (Gaussian):** Đây là kernel phổ biến nhất, hoạt động tốt trên hầu hết các loại dữ liệu, tương ứng với không gian vô hạn chiều.

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma ||\mathbf{x} - \mathbf{z}||^2), \quad \gamma > 0 \tag{2.28}$$

- **Sigmoid Kernel:** Tương tự như mạng Neural Network hai lớp.

$$k(\mathbf{x}, \mathbf{z}) = \tanh(\gamma \mathbf{x}^T \mathbf{z} + r) \tag{2.29}$$

2.3 Kỹ thuật phân cụm dữ liệu

2.3.1 K-Means

Giới thiệu chung

K-Means là một thuật toán thuộc nhóm Học không giám sát (Unsupervised Learning) và cụ thể là bài toán **Phân cụm (Clustering)** [**?**]. Mục tiêu của thuật toán là phân chia tập dữ liệu thành K cụm (cluster) khác nhau, sao cho các điểm dữ liệu trong cùng một cụm có tính chất tương đồng nhau nhất, và các điểm khác cụm có sự khác biệt lớn nhất.

Cơ sở toán học và hàm mục tiêu

Giả sử tập dữ liệu là $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ và chúng ta muốn chia thành K cụm. Gọi $\mathbf{m}_k$ là tâm (centroid) của cụm thứ k.

Hàm mục tiêu (Objective Function) của K-Means là tối thiểu hóa tổng bình phương khoảng cách từ mỗi điểm dữ liệu đến tâm cụm gần nhất của nó:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \mathbf{m}_k||_2^2 \tag{2.30}$$

Trong đó:

- $r_{nk} = 1$ nếu điểm $\mathbf{x}_n$ thuộc cụm k, ngược lại $r_{nk} = 0$.
- $|| \cdot ||_2$ là khoảng cách Euclid (Norm 2).

Quy trình thuật toán

Thuật toán K-Means hoạt động theo phương pháp lặp (iterative) gồm các bước sau:

1. **Khởi tạo:** Chọn ngẫu nhiên K điểm làm tâm các cụm ban đầu $\mathbf{m}_1, \ldots, \mathbf{m}_K$.
2. **Phân nhóm (Assignment Step):** Với mỗi điểm dữ liệu $\mathbf{x}_i$, gán nó vào cụm có tâm gần nhất:

$$y_i = \arg\min_k ||\mathbf{x}_i - \mathbf{m}_k||_2^2 \tag{2.31}$$

3. **Cập nhật tâm (Update Step):** Tính lại vị trí tâm cụm mới bằng cách lấy trung bình cộng (mean) tọa độ của tất cả các điểm thuộc cụm đó:

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} r_{ik}} \tag{2.32}$$

4. **Điều kiện dừng:** Lặp lại bước 2 và 3 cho đến khi vị trí các tâm cụm không thay đổi (hội tụ) hoặc đạt số vòng lặp tối đa.

Đánh giá K-Means bằng Adjusted Rand Index (ARI)

Để kiểm tra chất lượng phân cụm của K-Means khi đã có nhãn thực tế (Ground Truth), ta sử dụng chỉ số Adjusted Rand Index (ARI).

- **Ý nghĩa:** ARI đo lường sự tương đồng giữa kết quả phân chia của K-Means và nhãn thật của dữ liệu, bỏ qua yếu tố ngẫu nhiên.
- **Giá trị:** ARI có giá trị từ -1 đến 1.
  - $ARI = 1$: K-Means phân cụm hoàn hảo (trùng khớp với nhãn thật).
  - $ARI \approx 0$: Kết quả tương đương với việc gán nhãn ngẫu nhiên.
- **Lợi ích:** Giúp xác định xem số lượng cụm K và hàm khoảng cách Euclid có phù hợp với cấu trúc dữ liệu thực tế hay không.

2.3.2 DBSCAN

Giới thiệu

DBSCAN là thuật toán phân cụm dựa trên mật độ (density-based) [10]. Khác với K-Means (dựa trên khoảng cách và hình cầu), DBSCAN định nghĩa cụm là các vùng có mật độ điểm dữ liệu cao, được ngăn cách bởi các vùng có mật độ thấp.

Hai tham số chính

Thuật toán yêu cầu 2 tham số đầu vào:

- ε (Epsilon): Bán kính dùng để xác định vùng lân cận của một điểm.
- minPts: Số lượng điểm tối thiểu nằm trong bán kính ε để tạo thành một vùng đặc (dense region).

Phân loại điểm dữ liệu

Dựa trên ε và minPts, một điểm p được phân thành 3 loại:

1. **Core Point (Điểm lõi):** Là điểm có ít nhất minPts điểm lân cận trong bán kính ε (bao gồm cả chính nó).

2. **Border Point (Điểm biên):** Là điểm có ít hơn minPts điểm lân cận, nhưng nằm trong vùng lân cận của một Core Point.

3. **Noise Point (Điểm nhiễu):** Là điểm không phải Core Point cũng không phải Border Point (còn gọi là Outlier).

Đánh giá DBSCAN bằng Adjusted Rand Index (ARI)

Đối với DBSCAN, chỉ số ARI được dùng để đánh giá khả năng phát hiện đúng các cụm có hình dạng bất kỳ và xử lý nhiễu.

- **Tương thích với nhiễu:** ARI đánh giá xem DBSCAN có tách biệt đúng các vùng mật độ cao và gán đúng các điểm nhiễu (outliers) so với nhãn thực tế hay không.

- **Không phụ thuộc số lượng cụm:** Do DBSCAN tự động xác định số lượng cụm (có thể khác với số lượng lớp thực tế), ARI là thước đo phù hợp vì nó chỉ quan tâm đến quan hệ cặp (pairwise) giữa các điểm dữ liệu mà không yêu cầu số lượng cụm tìm được phải khớp chính xác với số lượng lớp ban đầu.

- **Kết quả:** Giá trị ARI càng cao chứng tỏ bộ tham số ε và minPts đã được chọn tối ưu cho việc tách mật độ.

Các khái niệm kết nối

- **Directly Density-Reachable (Tiếp cận mật độ trực tiếp):** Điểm q được gọi là tiếp cận trực tiếp từ p nếu p là Core Point và q nằm trong bán kính ε của p.

- **Density-Connected (Liên thông mật độ):** 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 được tiếp cận mật độ từ o. Đây là cơ sở để gộp các Core Point lại thành một cụm.

Quy trình thuật toán

1. Chọn một điểm p bất kỳ chưa được thăm.

2. Kiểm tra vùng lân cận ε của p.

3. Nếu $|N_\varepsilon(p)| \geq \text{minPts}$, tạo một cụm mới và mở rộng cụm đó bằng cách thêm tất cả các điểm tiếp cận mật độ từ p.

4. Nếu $|N_\varepsilon(p)| < \text{minPts}$, đánh dấu p là Noise (tuy nhiên p có thể được xác định lại là Border Point sau này nếu nó nằm trong vùng lân cận của một Core Point khác).

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

2.4 Kỹ thuật tiền xử lý dữ liệu

2.4.1 Ép kiểu dữ liệu

Ép kiểu dữ liệu là quá trình chuyển đổi dữ liệu từ kiểu có độ chính xác cao hoặc kích thước bộ nhớ lớn sang kiểu nhỏ hơn, nhằm mục đích tối ưu hóa hiệu suất tính toán và dung lượng lưu trữ.

Ép về kiểu `float32`

Ở đây việc ép kiểu từ `float64` về `float32` cho các trường dữ liệu thực, số thập phân đã giảm đi đáng kể trong việc lưu trữ bộ nhớ, các trường dữ liệu sau khi được one hot thường sẽ tự động được đưa về kiểu `float64`, đi kèm với đó là các trường số thực không quá lớn, vậy nên ép kiểu về `float32` vừa tăng được tốc độ tính toán, vừa có thể tiết kiệm bộ nhớ lưu trữ với bộ dữ liệu lớn.

Ép về kiểu `int16`

Đối với dữ liệu kiểu số nguyên như: tuổi (Age), số năm thâm niên (Year at company), và các trường dữ liệu phân loại có thứ tự (1, 2, 3, 4...) ban đầu ở kiểu dữ liệu `int64` tuy nhiên điều này là rất thừa thãi và lãng phí, vậy nên ở đây, chúng tôi đã chuyển đổi kiểu dữ liệu của nó về `int16` là vừa đủ.

2.4.2 Mã hóa tiền xử lý

Label Encoding

Chúng ta sử dụng kỹ thuật mã hóa dữ liệu phân loại này khi đặc trưng phân loại là thứ tự. Trong trường hợp này, việc duy trì thứ tự là rất quan trọng. Do đó, mã hóa phải phản ánh trình tự.

Trong mã hóa nhãn, mỗi nhãn (Category) được chuyển đổi thành một giá trị số nguyên.

One Hot Encoding

Chúng ta sử dụng kỹ thuật mã hóa dữ liệu phân loại này khi các đặc trưng không có thứ tự. Trong mã hóa one-hot, với mỗi cấp độ (level) của một đặc trưng phân loại, chúng ta tạo ra một biến mới. Mỗi danh mục được ánh xạ với một biến nhị phân chứa 0 hoặc 1. Ở đây, 0 đại diện cho sự vắng mặt, và 1 đại diện cho sự hiện diện của danh mục đó.

Các đặc trưng nhị phân mới được tạo ra này được gọi là Biến giả (Dummy variables). Số lượng biến giả phụ thuộc vào các cấp độ có trong biến phân loại. Điều này nghe có vẻ phức tạp. Chúng ta hãy lấy một ví dụ để hiểu rõ hơn. Giả sử chúng ta có một tập dữ liệu với một danh mục là `animal` (động vật), có các loài động vật khác nhau như Chó, Mèo, Cừu, Bò, Sư tử. Bây giờ chúng ta phải mã hóa one-hot dữ liệu này.

2.4.3 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu (còn gọi là Data Scaling) là một bước thiết yếu trong tiền xử lý dữ liệu. Mục đích chính là đưa tất cả các cột dữ liệu (các đặc trưng) về một thang đo chung mà không làm mất đi thông tin về sự khác biệt tương đối giữa chúng.

Vậy tại sao phải chuẩn hóa? Hãy tưởng tượng bạn có một tập dữ liệu về bệnh nhân với hai đặc trưng:

- **Tuổi:** (ví dụ: từ 20 đến 70 tuổi)
- **Huyết áp:** (ví dụ: từ 110 đến 180)

Nếu bạn đưa thẳng vào mô hình, các thuật toán máy học có thể lầm tưởng rằng "Huyết áp"quan trọng hơn "Tuổi"chỉ vì các con số của nó (110, 180) lớn hơn (20, 70).

Chuẩn hóa dữ liệu giải quyết vấn đề này bằng cách đặt tất cả các đặc trưng vào "sân chơi"bình đẳng (ví dụ: đưa cả hai về thang đo từ 0 đến 1). Bằng cách này, mô hình có thể so sánh và đánh giá tầm quan trọng của chúng một cách công bằng.

Các phương pháp chuẩn hóa phổ biến

Có hai kỹ thuật chính thường sẽ gặp:

1. **Normalization (Chuẩn hóa Min-Max):**
   - **Cách làm:** "Co"hoặc "giãn"dữ liệu để tất cả các giá trị nằm gọn trong một phạm vi cụ thể, thường là [0, 1].
   - **Công thức:** $X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$
   - **Kết quả:** Giá trị nhỏ nhất trong cột sẽ trở thành 0, giá trị lớn nhất sẽ trở thành 1.
2. **Standardization (Chuẩn hóa Z-score):**
   - **Cách làm:** Biến đổi dữ liệu sao cho nó có giá trị trung bình (mean) = 0 và độ lệch chuẩn (std) = 1.
   - **Công thức:** $X_{\text{scaled}} = \frac{X - \mu}{\sigma}$ (với μ là trung bình và σ là độ lệch chuẩn)
   - **Kết quả:** Dữ liệu sau khi chuẩn hóa sẽ phân bố xung quanh số 0. Đây là phương pháp rất phổ biến và hoạt động tốt với nhiều thuật toán.

Khi nào thì cần dùng?

Chuẩn hóa là rất quan trọng (thường là bắt buộc) đối với các thuật toán nhạy cảm với "độ lớn"hoặc "khoảng cách"của giá trị:

- **Thuật toán dựa trên khoảng cách:**
  - K-Nearest Neighbors (KNN)

  - K-Means Clustering
  - Support Vector Machines (SVM)

- **Thuật toán tối ưu (Gradient Descent):**
  - Hồi quy tuyến tính (Linear Regression)
  - Hồi quy Logistic (Logistic Regression)
  - Mạng nơ-ron (Neural Networks)

- **Phân tích thành phần chính (PCA):** PCA tìm cách tối đa hóa phương sai, nên các đặc trưng có thang đo lớn sẽ thống trị.

Không nhất thiết phải chuẩn hóa cho các thuật toán dựa trên cây (như Decision Tree, Random Forest) vì chúng chỉ quan tâm đến cách chia dữ liệu (ví dụ: `tuổi > 50?`) chứ không quan tâm giá trị đó lớn đến mức nào.

2.5 Kỹ thuật giảm chiều dữ liệu

2.5.1 Principal Component Analysis (PCA)

Giới thiệu

Dimensionality Reduction (Giảm chiều dữ liệu) là một kỹ thuật nén dữ liệu quan trọng trong Machine Learning, nhằm giải quyết vấn đề chi phí lưu trữ và tốc độ tính toán khi làm việc với các vector đặc trưng có số chiều lớn.

Về mặt toán học, mục tiêu của kỹ thuật này là tìm một hàm ánh xạ biến đổi điểm dữ liệu từ không gian gốc $\mathbf{x} \in \mathbb{R}^D$ sang một không gian mới $\mathbf{z} \in \mathbb{R}^K$ với số chiều thấp hơn ($K < D$).

Principal Component Analysis (PCA) là thuật toán giảm chiều đơn giản và phổ biến nhất dựa trên mô hình tuyến tính. PCA hoạt động dựa trên giả định rằng dữ liệu thực tế thường phân bố tập trung xung quanh các cấu trúc tuyến tính (không gian con - subspace) thay vì rải rác ngẫu nhiên trong toàn bộ không gian.

Các bước thực hiện PCA

1. Tính vector kỳ vọng của toàn bộ dữ liệu:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

2. Trừ mỗi điểm dữ liệu đi vector kỳ vọng của toàn bộ dữ liệu:

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

3. Tính ma trận hiệp phương sai:

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

4. Tính các trị riêng và vector riêng có norm bằng 1 của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.

5. Chọn K vector riêng ứng với K trị riêng lớn nhất để xây dựng ma trận $\mathbf{U}_K$ có các cột tạo thành một hệ trực giao. K vectors này, còn được gọi là các thành phần chính, tạo thành một không gian con **gần** với phân bố của dữ liệu ban đầu đã chuẩn hoá.

6. Chiếu dữ liệu ban đầu đã chuẩn hoá $\hat{\mathbf{X}}$ xuống không gian con tìm được.

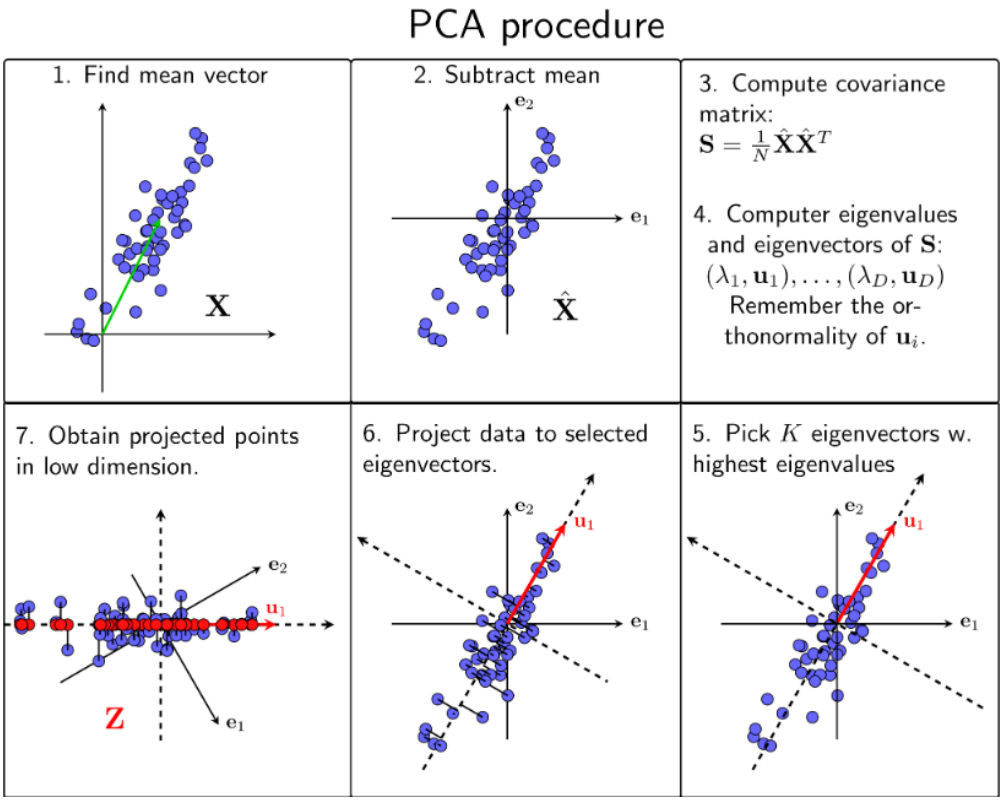
7. Dữ liệu mới chính là toạ độ của các điểm dữ liệu trên không gian mới.

$$\mathbf{Z} = \mathbf{U}_K^T\hat{\mathbf{X}}$$

Dữ liệu ban đầu có thể tính được xấp xỉ theo dữ liệu mới như sau:

$$\mathbf{x} \approx \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{x}}$$

Các bước thực hiện PCA có thể được xem trong hình dưới đây:



Hình 2.6: Các bước thực hiện PCA.

2.5.2 Linear Discriminant Analysis (LDA)

Động lực và So sánh với PCA

Trong khi PCA (Học không giám sát) cố gắng giữ lại phương sai lớn nhất của dữ liệu, điều này không đảm bảo khả năng phân loại tốt vì các lớp có thể bị chồng lấn. LDA (Học có giám sát) khắc phục điều này bằng cách tìm chiều không gian chiếu sao cho khả năng **phân biệt giữa các lớp (discriminant)** là cao nhất.

Ý tưởng cốt lõi (Tiêu chí Fisher)

Để phân tách tốt hai lớp dữ liệu sau khi chiếu, LDA tối ưu hóa đồng thời hai tiêu chí:

1. **Maximize Between-class variance:** Khoảng cách giữa các kỳ vọng (mean) của các lớp phải lớn nhất.
2. **Minimize Within-class variance:** Phương sai nội bộ (độ phân tán) của mỗi lớp phải nhỏ nhất (dữ liệu co cụm lại).

Mục tiêu là tìm vector hệ số $\mathbf{w}$ để tối đa hóa tỉ số:

$$\mathbf{w} = \arg\max_{\mathbf{w}} J(\mathbf{w}) = \frac{\text{Between-class variance}}{\text{Within-class variance}} \tag{2.33}$$

Công thức toán học

Hàm mục tiêu được cụ thể hóa thông qua hai ma trận hiệp phương sai:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{2.34}$$

Trong đó:

- $\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$: Ma trận hiệp phương sai liên lớp (Between-class).
- $\mathbf{S}_W$: Tổng các ma trận hiệp phương sai nội lớp (Within-class).

Nghiệm của bài toán

Việc tối đa hóa hàm $J(\mathbf{w})$ quy về bài toán tìm trị riêng lớn nhất của ma trận $\mathbf{S}_W^{-1}\mathbf{S}_B$. Nghiệm tối ưu (hướng chiếu) $\mathbf{w}$ tỉ lệ thuận với:

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \tag{2.35}$$

Đây là kết quả quan trọng nhất của LDA cho bài toán 2 lớp, cho phép tính toán trực tiếp vector $\mathbf{w}$ mà không cần lặp.

2.5.3 Feature Selection

Feature Selection (Lựa chọn đặc trưng) là một quá trình quan trọng trong học máy và khai phá dữ liệu, nhằm mục đích chọn ra một tập hợp con tối ưu các đặc trưng (features) từ tập dữ liệu ban đầu.

Mục tiêu chính là giữ lại những đặc trưng quan trọng nhất, có liên quan và có khả năng dự đoán cao nhất, đồng thời loại bỏ các đặc trưng thừa thãi, không liên quan, hoặc nhiễu.

Tại sao cần Feature Selection?

Lựa chọn đặc trưng giúp giải quyết ba vấn đề chính khi làm việc với dữ liệu có chiều (dimension) cao:

1. **Giảm Chiều Dữ liệu (Dimensionality Reduction):** Giảm số lượng đặc trưng, giúp tăng tốc độ huấn luyện mô hình và giảm yêu cầu về bộ nhớ.

2. **Cải thiện Hiệu suất Mô hình:** Các đặc trưng nhiễu hoặc không liên quan có thể làm giảm độ chính xác của mô hình. Loại bỏ chúng giúp mô hình tập trung vào thông tin cốt lõi, từ đó cải thiện độ chính xác (Accuracy) và khả năng khái quát hóa (Generalization).

3. **Tăng Khả năng Giải thích (Interpretability):** Mô hình chỉ sử dụng một số lượng đặc trưng nhỏ, quan trọng sẽ dễ hiểu và dễ giải thích hơn.

2.6 Xử lý dữ liệu mất cân bằng và SMOTE

2.6.1 Vấn đề mất cân bằng dữ liệu

Mất cân bằng dữ liệu xảy ra khi sự phân bố giữa các lớp có sự chênh lệch lớn (lớp đa số chiếm ưu thế so với lớp thiểu số), dẫn đến việc mô hình học máy bị thiên vị và dự đoán kém trên lớp thiểu số. Các phương pháp xử lý phổ biến bao gồm: Oversampling (lấy mẫu quá mức), Undersampling (lấy mẫu dưới mức) và tạo dữ liệu nhân tạo.

2.6.2 Thuật toán SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) là kỹ thuật lấy mẫu quá mức nâng cao. Thay vì chỉ nhân bản (duplicate) các điểm dữ liệu cũ gây ra hiện tượng quá khớp (overfitting), SMOTE tạo ra các điểm dữ liệu **tổng hợp mới** (synthetic samples) dựa trên cấu trúc không gian của lớp thiểu số.

Cơ chế hoạt động

Quy trình sinh dữ liệu của SMOTE gồm 3 bước chính:

1. **Chọn điểm gốc:** Chọn một điểm dữ liệu $\mathbf{x}$ thuộc lớp thiểu số.

2. **Tìm lân cận:** Xác định k lân cận gần nhất (k-Nearest Neighbors) của $\mathbf{x}$ trong không gian đặc trưng.

3. **Nội suy tuyến tính:** Chọn ngẫu nhiên một lân cận $\mathbf{x}_{neighbor}$. Điểm dữ liệu mới $\mathbf{x}_{new}$ sẽ được tạo ra nằm ngẫu nhiên trên đoạn thẳng nối giữa $\mathbf{x}$ và $\mathbf{x}_{neighbor}$ theo công thức:

$$\mathbf{x}_{new} = \mathbf{x} + \text{rand}(0, 1) \times (\mathbf{x}_{neighbor} - \mathbf{x}) \tag{2.36}$$

Quá trình này được lặp lại cho đến khi tỷ lệ giữa lớp thiểu số và đa số đạt mức cân bằng mong muốn.

2.7 Các thước đo đánh giá mô hình

Tùy theo từng loại bài toán học máy, các thước đo đánh giá được lựa chọn sao cho phản ánh đúng hiệu suất và đặc điểm của mô hình. Trong nghiên cứu này, các mô hình được đánh giá theo ba nhóm chính: phân loại, hồi quy và phân cụm.

2.7.1 Đánh giá mô hình phân loại

Để đánh giá hiệu suất của mô hình phân loại, ta sử dụng **Confusion Matrix** (Ma trận nhầm lẫn), bao gồm bốn thành phần cơ bản:

- **TP (True Positive):** Dự đoán đúng lớp dương tính.
- **TN (True Negative):** Dự đoán đúng lớp âm tính.
- **FP (False Positive):** Dự đoán sai là dương tính (báo động giả).
- **FN (False Negative):** Dự đoán sai là âm tính (bỏ sót).

Dựa trên các thành phần này, các chỉ số đánh giá chính được xác định như sau:

Accuracy

Là tỷ lệ tổng số dự đoán đúng trên toàn bộ tập dữ liệu:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.37}$$

Chỉ số này có thể gây hiểu lầm trong trường hợp dữ liệu bị mất cân bằng.

Precision

Đo lường độ tin cậy của các dự đoán dương tính, đặc biệt quan trọng khi chi phí cho báo động giả là cao:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.38}$$

Recall

Đo lường khả năng phát hiện đầy đủ các mẫu dương tính thực tế của mô hình, thường được ưu tiên trong các bài toán y tế hoặc phát hiện lỗi:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.39}$$

F1-Score

Là trung bình điều hòa giữa Precision và Recall, được sử dụng khi cần sự cân bằng giữa độ chính xác và độ bao phủ:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.40}$$

2.7.2 Đánh giá mô hình hồi quy

Đối với các bài toán hồi quy, hiệu suất mô hình được đánh giá thông qua mức độ sai lệch giữa giá trị dự đoán và giá trị thực.

R^2 Score

Hệ số xác định R^2 đo lường mức độ mà mô hình giải thích được sự biến thiên của biến mục tiêu:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

trong đó y_i là giá trị thực, $\hat{y}_i$ là giá trị dự đoán, và $\bar{y}$ là giá trị trung bình của biến mục tiêu.

Giá trị R^2 càng gần 1 thì mô hình càng phù hợp với dữ liệu; $R^2 < 0$ cho thấy mô hình dự đoán kém hơn mô hình cơ sở.

MAE – Mean Absolute Error

MAE đo lường sai số tuyệt đối trung bình giữa giá trị dự đoán và giá trị thực:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|.$$

MAE có cùng đơn vị đo với biến mục tiêu và ít nhạy cảm với các giá trị ngoại lai.

RMSE – Root Mean Squared Error

RMSE phản ánh sai số trung bình, trong đó các sai số lớn bị phạt mạnh hơn:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

RMSE nhạy cảm hơn với ngoại lai so với MAE và thường được sử dụng khi cần hạn chế các sai số lớn.

2.7.3 Đánh giá mô hình phân cụm

Đối với bài toán phân cụm, do không có nhãn lớp trong quá trình huấn luyện, việc đánh giá thường dựa trên các thước đo so sánh giữa nhãn phân cụm và nhãn chuẩn (nếu có).

ARI – Adjusted Rand Index

Chỉ số Rand điều chỉnh (Adjusted Rand Index – ARI) là thước đo được sử dụng phổ biến để đánh giá chất lượng của mô hình phân cụm khi tồn tại nhãn chuẩn (ground truth). ARI đo lường mức độ tương đồng giữa kết quả phân cụm của mô hình và phân hoạch thực tế, đồng thời hiệu chỉnh ảnh hưởng của sự trùng khớp ngẫu nhiên.

Giả sử $U = \{U_1, U_2, \ldots, U_r\}$ là tập các cụm do mô hình tạo ra và $V = \{V_1, V_2, \ldots, V_s\}$ là tập các lớp thực. Xét mọi cặp điểm dữ liệu, ARI đánh giá mức độ nhất quán của việc hai điểm bất kỳ được xếp cùng cụm hoặc khác cụm trong hai phân hoạch U và V.

Chỉ số ARI được định nghĩa dựa trên chỉ số Rand (Rand Index – RI) theo công thức:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{\max(\text{RI}) - \mathbb{E}[\text{RI}]},$$

trong đó $\mathbb{E}[\text{RI}]$ là giá trị kỳ vọng của RI trong trường hợp phân cụm ngẫu nhiên.

Một cách biểu diễn tương đương của ARI dựa trên bảng liên hợp giữa các cụm và nhãn thực như sau:

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}},$$

trong đó n_{ij} là số mẫu chung giữa cụm U_i và lớp V_j, $a_i = \sum_j n_{ij}$, $b_j = \sum_i n_{ij}$, và n là tổng số mẫu.

Giá trị của ARI nằm trong khoảng $[-1, 1]$. ARI đạt giá trị 1 khi hai phân hoạch trùng khớp hoàn toàn; ARI xấp xỉ 0 tương ứng với phân cụm ngẫu nhiên; trong khi ARI âm cho thấy kết quả phân cụm kém hơn so với ngẫu nhiên.

Ưu điểm chính của ARI là không bị ảnh hưởng bởi số lượng cụm và cho phép so sánh công bằng giữa các mô hình phân cụm khác nhau. Do đó, ARI thường được sử dụng để đánh giá hiệu suất của các thuật toán phân cụm như K-means, Agglomerative Clustering và Spectral Clustering khi có nhãn chuẩn.

Chương 3

Phương pháp và Quy trình thực nghiệm

3.1 Mô tả bộ dữ liệu

3.1.1 Giới thiệu nguồn gốc (IBM)

Để thực hiện bài nghiên cứu, chúng tôi đã sử dụng bộ dữ liệu "IBM Employee Dataset" [12].

Bộ dữ liệu bao gồm 1470 mẫu quan sát với 35 đặc tính khác nhau liên quan đến đời sống làm việc và đặc điểm cá nhân của nhân viên tại Hoa Kỳ bao gồm:

Bảng 3.1: Mô tả biến

STT	**Thuộc tính**	**Loại dữ liệu**	**Định nghĩa thuộc tính**	**Mô tả thuộc tính**
1	Age	Số nguyên	Tuổi	
2	Attrition	Phân loại	Quyết định nghỉ việc	
3	Business travel	Phân loại	Mức độ đi công tác	Non_Travel Travel_Rarely Travel_Frequently

STT	Thuộc tính	Loại dữ liệu	Định nghĩa thuộc tính	Mô tả thuộc tính
4	Daily rate	Số nguyên	Mức lương theo ngày	
5	Department	Phân loại	Phòng ban làm việc	Human Resources Research & Development Sales
6	Distance from home	Số nguyên	Khoảng cách giữa nơi làm việc và nhà	
7	Education	Phân loại	Trình độ giáo dục	1. Below College 2. College 3. Bachelor 4. Master 5. Doctor
8	Education field	Phân loại	Lĩnh vực giáo dục	Human Resources Life Sciences Medical Marketing Technical Degree Other
9	Employee count	Số nguyên	Số lượng nhân viên	
10	Employee number	Số nguyên	Mã số nhân viên	
11	Environment satisfaction	Phân loại	Mức độ hài lòng về môi trường làm việc	1. Low 2. Medium 3. High 4. Very High

STT	Thuộc tính	Loại dữ liệu	Định nghĩa thuộc tính	Mô tả thuộc tính
12	Gender	Phân loại	Giới tính	Male Female
13	Hourly rate	Số nguyên	Mức lương theo giờ	
14	Job involvement	Phân loại	Mức độ tham gia vào công việc	1. Low 2. Medium 3. High 4. Very High
15	Job level	Phân loại	Cấp bậc công việc	
16	Job role	Phân loại	Lĩnh vực làm việc	Sales Executive Research Scientist Others
17	Job satisfaction	Phân loại	Mức độ hài lòng về công việc	1. Low 2. Medium 3. High 4. Very High
18	Marital status	Phân loại	Tình trạng hôn nhân	Divorced Single Married
19	Monthly income	Số nguyên	Thu thập hàng tháng	
20	Monthly rate	Số nguyên	Mức lương theo tháng	
21	Number of company workers	Số nguyên	Số lượng công ty đã làm trước đây	
22	Over 18	Phân loại	Trên 18 tuổi	
23	Overtime	Phân loại	Tăng ca	No Yes

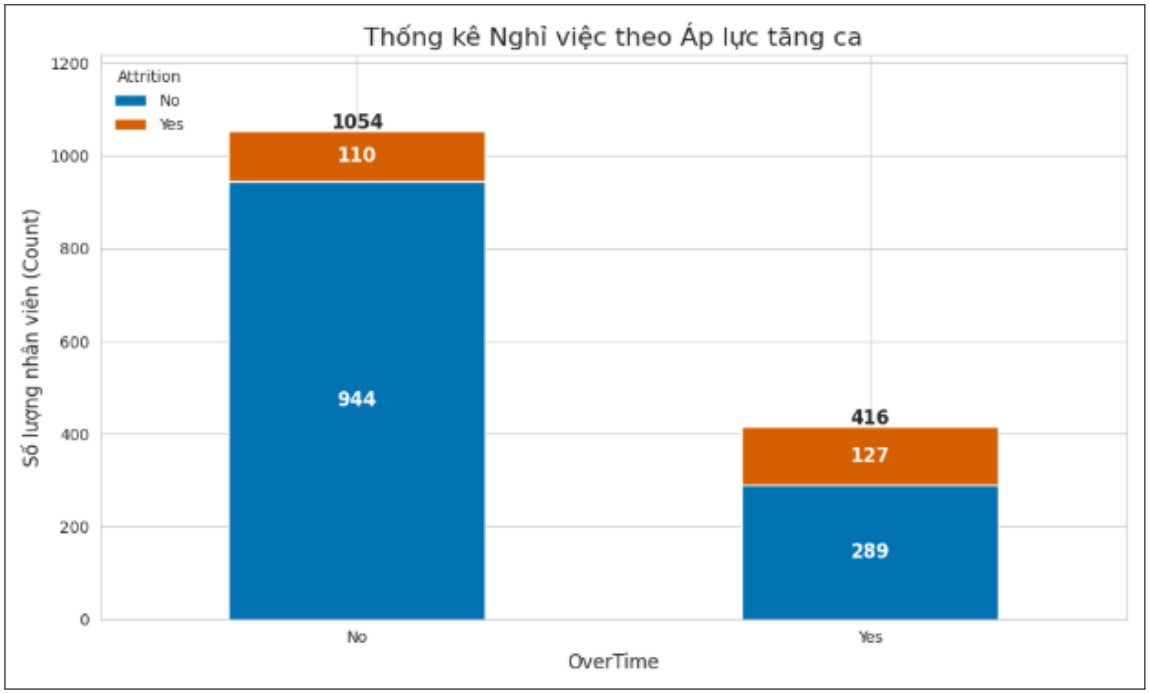
STT	Thuộc tính	Loại dữ liệu	Định nghĩa thuộc tính	Mô tả thuộc tính
24	Percent salary hike	Số nguyên	Tỷ lệ thay đổi trong mức lương	
25	Performance rating	Phân loại	Mức độ hoàn thành công việc	1. Low 2. Medium 3. High 4. Very High
26	Relationship satisfaction	Phân loại	Mức độ hài lòng về mối quan hệ	1. Low 2. Medium 3. High 4. Very High
27	Standard hours	Số nguyên	Giờ làm việc tiêu chuẩn	
28	Stock option level	Phân loại	Mức cổ phần nhân viên nắm giữ	
29	Total working years	Số nguyên	Tổng số năm làm việc	
30	Training times last year	Số nguyên	Số lần đào tạo trong năm	
31	Work-life balance	Phân loại	Mức độ cân bằng giữa làm việc và cuộc sống	1. Bad 2. Good 3. Better 4. Best
32	Years at company	Số nguyên	Số năm làm việc tại công ty	
33	Years in current role	Số nguyên	Số năm làm việc ở vị trí hiện tại	
34	Years since last promotion	Số nguyên	Số năm làm việc kể từ khi thăng chức	

STT	Thuộc tính	Loại dữ liệu	Định nghĩa thuộc tính	Mô tả thuộc tính
35	Years with current manager	Số nguyên	Số năm làm việc với quản lý hiện tại	

Bộ dữ liệu có chứa đặc điểm để xác nhận nhân viên có nghỉ việc ở công ty hay không được xác định bởi biến "Attrition": "No" đại diện cho một nhân viên đã không nghỉ việc và "Yes" đại diện cho một nhân viên đã nghỉ việc tại công ty.

3.1.2 Liệt kê một số thuộc tính quan trọng

- Phân tích Biểu đồ: Thống kê nghỉ việc theo áp lực tăng ca.

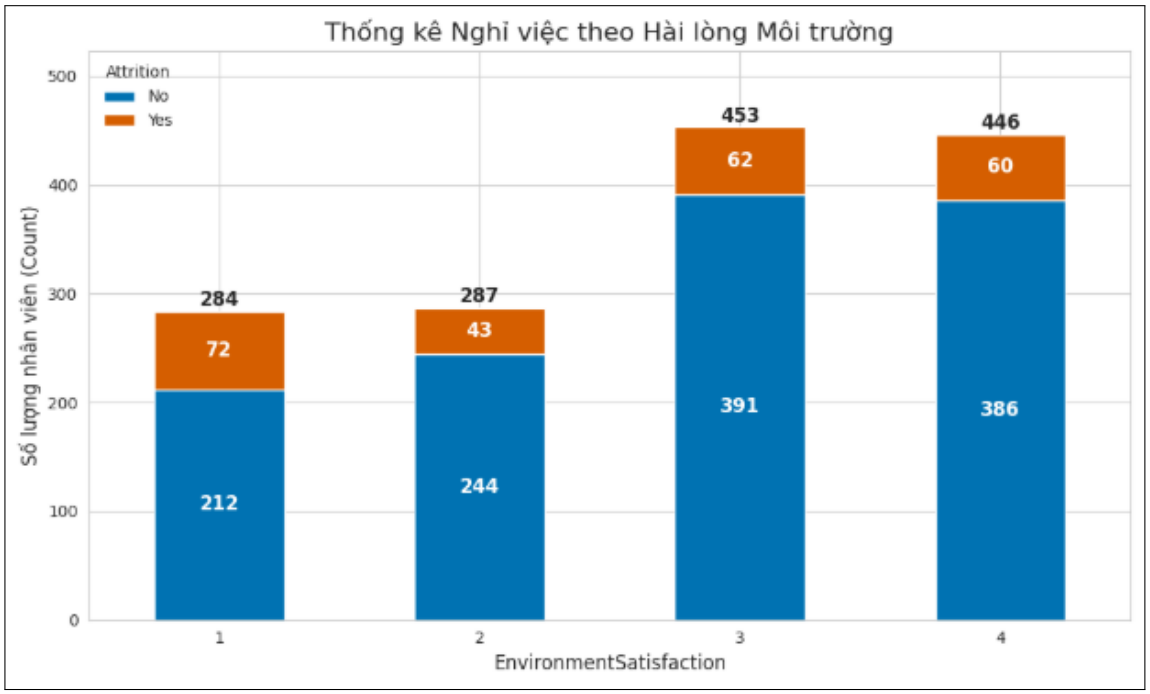


Hình 3.1: Thống kê số lượng nhân viên nghỉ việc theo áp lực tăng ca

- Biểu đồ này cho thấy `OverTime` (Làm thêm giờ) là một yếu tố dự đoán cực kỳ quan trọng.
- Nhóm không làm thêm giờ (No): Có tổng cộng 1054 nhân viên, trong đó chỉ có 110 người nghỉ việc. Tỷ lệ nghỉ việc $\approx 10.4\%$.

- Nhóm có làm thêm giờ (Yes): Có tổng cộng 416 nhân viên, nhưng có tới 127 người nghỉ việc. Tỷ lệ nghỉ việc $\approx 30.5\%$.
- Kết luận: Một nhân viên phải làm thêm giờ có tỷ lệ nghỉ việc cao gần gấp 3 lần (30.5% so với 10.4%) so với người không làm thêm giờ.

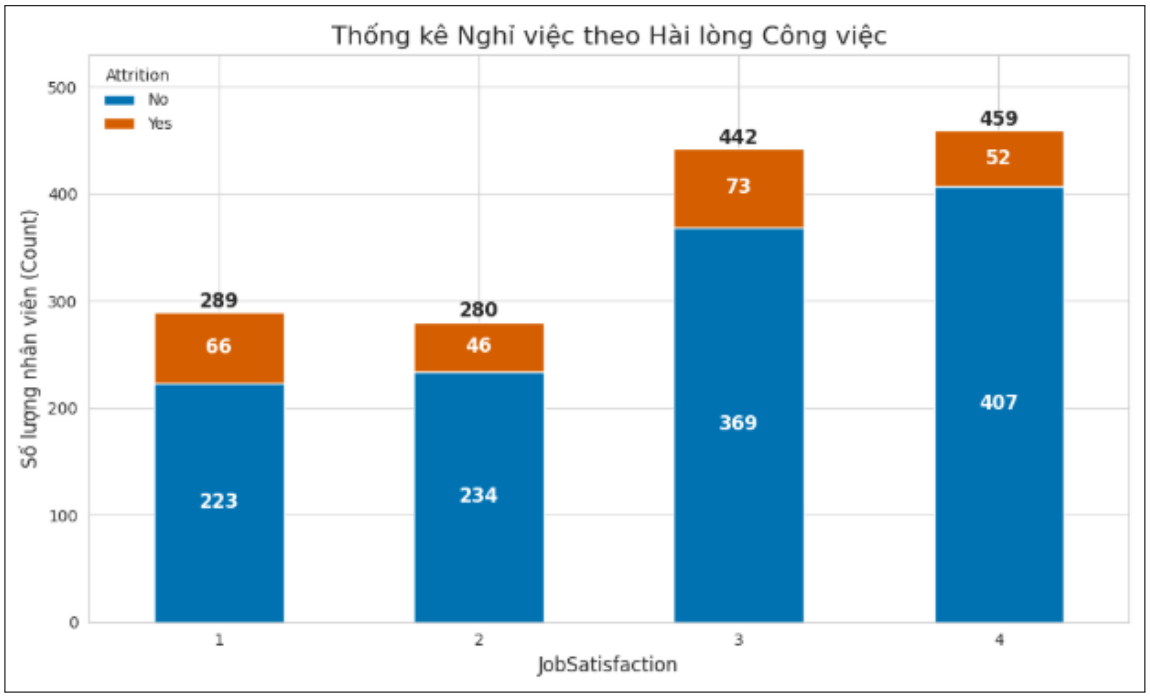
- Phân tích Biểu đồ: Thống kê nghỉ việc theo hài lòng môi trường



Hình 3.2: Thống kê số lượng nhân viên nghỉ việc theo mức độ hài lòng về môi trường làm việc ở công ty

- Biểu đồ này cho thấy mối liên hệ rõ rệt giữa sự hài lòng về môi trường và tỷ lệ nghỉ việc.
- Mức 1 (Rất không hài lòng): Tỷ lệ nghỉ việc cao nhất, với 72 trên 284 nhân viên $\approx 25.4\%$.
- Mức 2 (Không hài lòng): Tỷ lệ nghỉ việc là 43 trên 287 nhân viên $\approx 15.0\%$.
- Mức 3 & 4 (Hài lòng/Rất hài lòng): Tỷ lệ nghỉ việc giảm xuống mức thấp nhất, chỉ khoảng 13.5% - 13.7%.
- Kết luận: Những nhân viên bất mãn nhất với môi trường làm việc (Mức 1) có khả năng rời đi cao gần gấp đôi so với những người hài lòng.

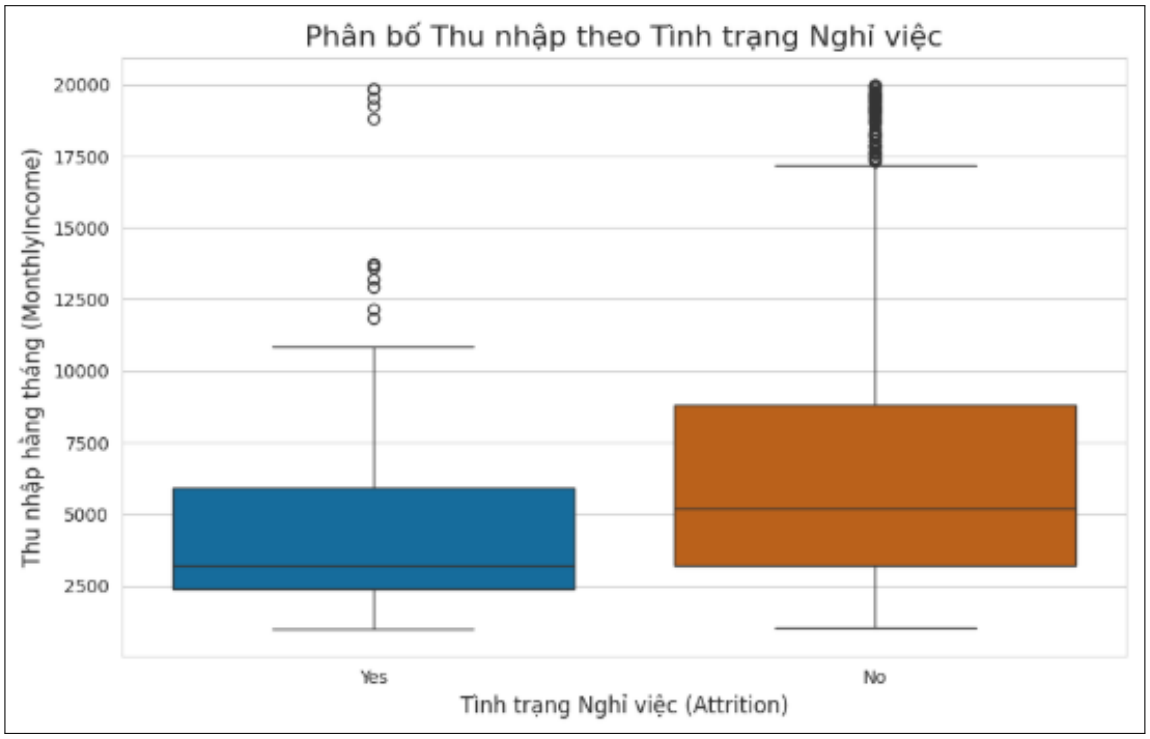
- Phân tích Biểu đồ: Thống kê nghỉ việc theo hài lòng công việc.



Hình 3.3: Thống kê số lượng nhân viên nghỉ việc theo mức độ hài lòng trong công việc ở công ty

- Tương tự như môi trường, mức độ hài lòng với công việc có ảnh hưởng rõ rệt.
- Mức 1 (Rất không hài lòng): Có tỷ lệ nghỉ việc cao nhất, với 66 trên 289 nhân viên $\approx 22.8\%$.
- Mức 4 (Rất hài lòng): Có tỷ lệ nghỉ việc thấp nhất, với chỉ 52 trên 459 nhân viên $\approx 11.3\%$.
- Kết luận: Có một xu hướng rõ ràng: sự hài lòng trong công việc càng tăng (từ 1 đến 4) thì tỷ lệ nghỉ việc càng giảm.

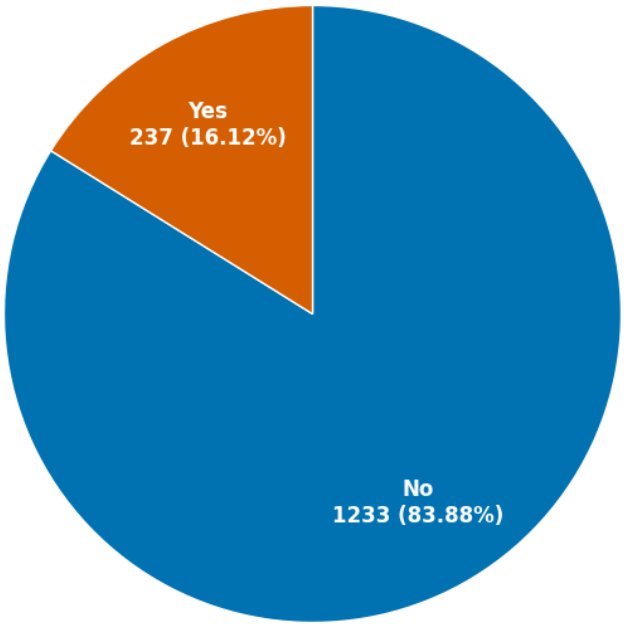
- Phân tích Biểu đồ: Phân bố Thu nhập theo tình trạng nghỉ việc



Hình 3.4: Phân bố thu nhập theo tình trạng nghỉ việc

- Biểu đồ hộp này cho thấy `MonthlyIncome` (Thu nhập) là một yếu tố then chốt.
- Nhóm Nghỉ việc (Yes): Toàn bộ "thân hộp"(IQR) và đường trung vị (median) nằm ở mức thu nhập thấp hơn đáng kể so với nhóm "No".
- Nhóm Ở lại (No): Có mức lương trung vị cao hơn rõ rệt và dải phân bố thu nhập cũng rộng hơn nhiều (bao gồm cả nhân viên mới và quản lý cấp cao).
- Kết luận: Những người nghỉ việc có xu hướng tập trung ở nhóm có thu nhập thấp, cho thấy đây là một trong những yếu tố chính dẫn đến nghỉ việc.

3.1.3 Phân tích biến mục tiêu Attrition



Hình 3.5: Thống kê số lượng nhân viên nghỉ việc tại IBM

Có 237 nhân viên nghỉ việc trong tổng số 1470 nhân viên, chiếm 16,12% tại công ty. Tuy là con số nhỏ nhưng phòng Nhân sự vẫn muốn biết được lý do dẫn đến sự nghỉ việc của nhân viên IBM.

3.2 Quy trình tiền xử lý

3.2.1 Làm sạch dữ liệu

Tách các cột X (các trường dữ liệu), cột y (nhãn của dữ liệu). Xóa bỏ các cột thừa, không cần thiết, có dữ liệu bằng nhau trên các mẫu hoặc không chứa thông tin, bao gồm các trường: Over18, EmployeeNumber, EmployeeCount, StandardHours vì các trường này không mang ý nghĩa trong việc dự đoán, các trường EmployeeCount, StandarHours có giá trị bằng nhau, nếu thêm vào sẽ tạo thêm nhiễu, không mang thông tin (phương sai bằng 0) khiến mô hình dự đoán không tốt.

3.2.2 Ép kiểu dữ liệu

Ép kiểu về các dạng float32 và int16 để tối ưu hiệu suất và bộ nhớ.

3.2.3 Mã hóa nhãn

One Hot Coding

One Hot những cột category không thứ tự như:

```
encode_cols = [
    'BusinessTravel', 'Department', 'EducationField',
    'Gender', 'JobRole', 'MaritalStatus', 'OverTime'
]
```

Các trường dữ liệu có thứ tự được giữ nguyên. Có sử dụng drop='first' nhằm mục đích tránh dư thừa thông tin và đa cộng tuyến.

Dữ liệu X ban đầu code shape là: (1470, 30). Sau khi encode thì X có dạng: (1470, 44).

Label Encoding

Dữ liệu ban đầu chỉ có Yes và No thì sau khi mã hóa thì đã chuyển về tương ứng với 1 và 0 để dễ dàng trong việc tính toán.

3.2.4 Áp dụng Chuẩn hóa

	Age	DailyRate	DistanceFromHome	EmployeeCount	HourlyRate	MonthlyIncome	MonthlyRate	NumCompaniesWorked	PercentS
0	41	1102	1	1	94	5993	19479	8	
1	49	279	8	1	61	5130	24907	1	
2	37	1373	2	1	92	2090	2396	6	
3	33	1392	3	1	56	2909	23159	1	
4	27	591	2	1	40	3468	16632	9	
...	...	...	...	...	...	...	...	...	
1465	36	884	23	1	41	2571	12290	4	
1466	39	613	6	1	42	9991	21457	4	
1467	27	155	4	1	87	6142	5174	1	
1468	49	1023	2	1	63	5390	13243	2	
1469	34	628	8	1	82	4404	10228	2	

1470 rows × 16 columns

Hình 3.6: Dữ liệu liên tục chưa chuẩn hóa

```
array([[ 0.4463504 ,  0.74252653, -1.01090934, ..., -0.0632959 ,
        -0.67914568,  0.24583399],
       [ 1.32236521, -1.2977746 , -0.14714972, ...,  0.76499762,
        -0.36871529,  0.80654148],
       [ 0.008343  ,  1.41436324, -0.88751511, ..., -1.16768726,
        -0.67914568, -1.15593471],
       ...,
       [-1.08667552, -1.60518328, -0.64072665, ..., -0.61549158,
        -0.67914568, -0.31487349],
       [ 1.32236521,  0.54667746, -0.88751511, ...,  0.48889978,
        -0.67914568,  1.08689522],
       [-0.32016256, -0.43256792, -0.14714972, ..., -0.33939374,
        -0.36871529, -0.59522723]], shape=(1470, 16))
```

Hình 3.7: Mảng dữ liệu liên tục đã chuẩn hóa

	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	0.446350	0.742527	-1.010909	0.0	1.383138	-0.108350	0.726020	2.125136	-1.150554	0.0	-0.421642	-2.171982	-0.164613	-0.
1	1.322365	-1.297775	-0.147150	0.0	-0.240677	-0.291719	1.488876	-0.678049	2.129306	0.0	-0.164511	0.155707	0.488508	0.
2	0.008343	1.414363	-0.887515	0.0	1.284725	-0.937654	-1.674841	1.324226	-0.057267	0.0	-0.550208	0.155707	-1.144294	-1.
3	-0.429664	1.461466	-0.764121	0.0	-0.486709	-0.763634	1.243211	-0.678049	-1.150554	0.0	-0.421642	0.155707	0.161947	0.
4	-1.086676	-0.524295	-0.887515	0.0	-1.274014	-0.644858	0.325900	2.525591	-0.877232	0.0	-0.678774	0.155707	-0.817734	-0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1465	-0.101159	0.202082	1.703764	0.0	-1.224807	-0.835451	-0.284329	0.523316	0.489376	0.0	0.735447	0.155707	-0.327893	-0.
1466	0.227347	-0.469754	-0.393938	0.0	-1.175601	0.741140	1.004010	0.523316	-0.057267	0.0	-0.293077	1.707500	-0.001333	0.
1467	-1.086676	-1.605183	-0.640727	0.0	1.038693	-0.076690	-1.284418	-0.678049	1.309341	0.0	-0.678774	-2.171982	-0.164613	-0.
1468	1.322365	0.546677	-0.887515	0.0	-0.142264	-0.236474	-0.150393	-0.277594	-0.330589	0.0	0.735447	0.155707	0.325228	0.
1469	-0.320163	-0.432568	-0.147150	0.0	0.792660	-0.445978	-0.574124	-0.277594	-0.877232	0.0	-0.678774	0.155707	-0.491174	-0.

1470 rows × 16 columns

Hình 3.8: DataFrame dữ liệu đã chuẩn hóa

Mô tả dữ liệu sau khi chuẩn hóa

Trước khi chuẩn hóa, dữ liệu của các trường `Age`, `DailyRate`, ...có độ lớn rất khác nhau, điều này gây mất cân bằng thang đo cho các thuộc tính. Các biến giá trị lớn như `MonthlyIncome` hay `MonthlyRate` có thể sẽ chi phối mô hình.

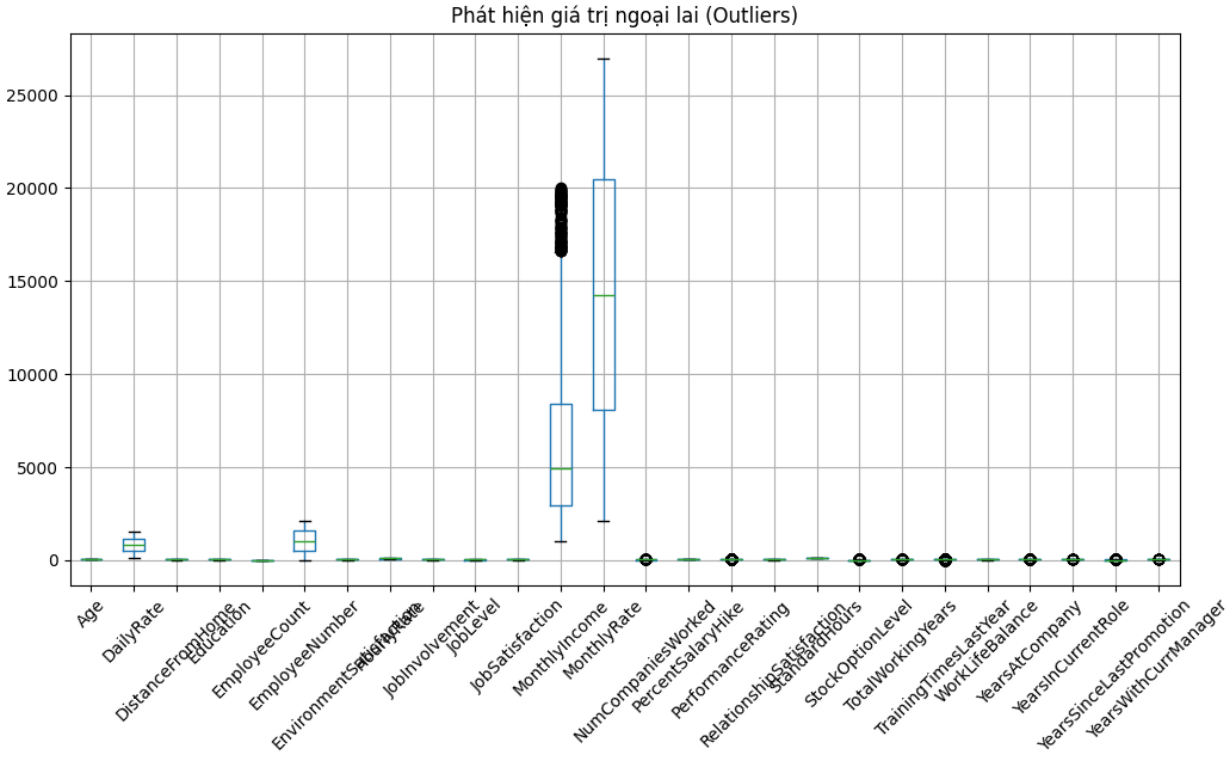
Sau khi chuẩn hóa bằng `StandardScaler`, các giá trị dao động quanh 0, thường nằm trong khoảng từ -3 đến 3. Các cột có trung bình ≈ 0, độ lệch chuẩn ≈ 1.

	0	1	2	3	4	5	6	7	8
count	1.470000e+03	1.470000e+03	1.470000e+03	1470.0	1.470000e+03	1.470000e+03	1.470000e+03	1.470000e+03	1.470000e+03
mean	-3.504377e-17	5.075305e-17	4.350262e-17	0.0	1.691768e-16	-4.471102e-17	3.021015e-17	1.450087e-17	2.271803e-16
std	1.000340e+00	1.000340e+00	1.000340e+00	0.0	1.000340e+00	1.000340e+00	1.000340e+00	1.000340e+00	1.000340e+00
min	-2.072192e+00	-1.736576e+00	-1.010909e+00	0.0	-1.766079e+00	-1.167343e+00	-1.717284e+00	-1.078504e+00	-1.150554e+00
25%	-7.581700e-01	-8.366616e-01	-8.875151e-01	0.0	-8.803615e-01	-7.632087e-01	-8.806440e-01	-6.780494e-01	-8.772324e-01
50%	-1.011589e-01	-1.204135e-03	-2.705440e-01	0.0	5.355811e-03	-3.365516e-01	-1.090645e-02	-2.775943e-01	-3.305891e-01
75%	6.653541e-01	8.788772e-01	5.932157e-01	0.0	8.787715e-01	3.986245e-01	8.641014e-01	5.233157e-01	7.626976e-01
max	2.526886e+00	1.726730e+00	2.444129e+00	0.0	1.678377e+00	2.867626e+00	1.782888e+00	2.525591e+00	2.675949e+00

Hình 3.9: File thống kê chi tiết dữ liệu đã chuẩn hóa

3.3 Trực quan hóa dữ liệu

3.3.1 Biểu đồ phát hiện giá trị ngoại lai của dữ liệu



Hình 3.10: Biểu đồ thống kê giá trị ngoại lai

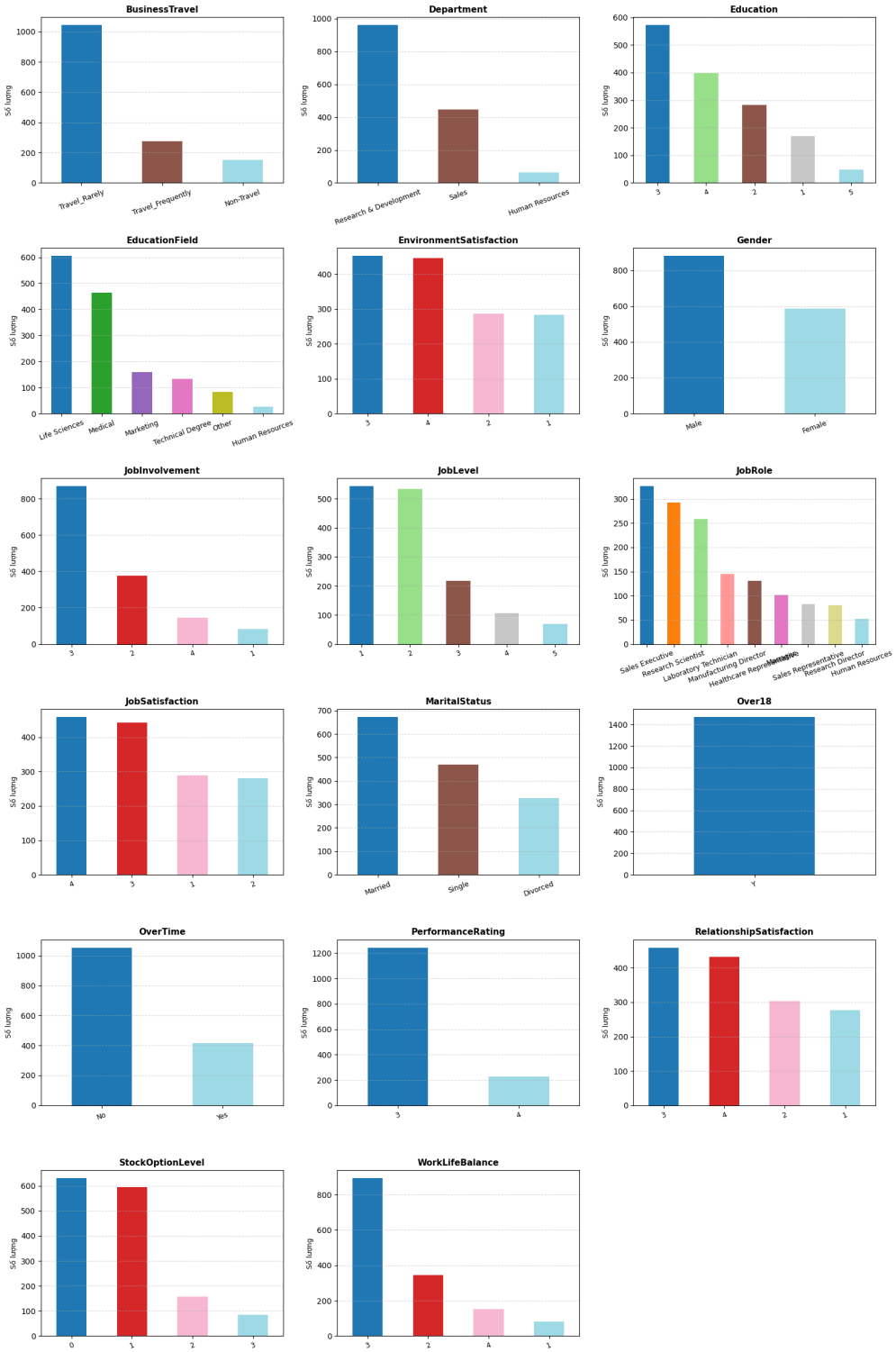
Nhận xét:

Biểu đồ boxplot cho thấy nhiều giá trị ngoại lai, tập trung chủ yếu ở các biến liên quan đến thu nhập và thâm niên làm việc như `MonthlyIncome`, `MonthlyRate`, `TotalWorkingYears` và `YearsAtCompany`.

Các biến đánh giá theo thang điểm rời rạc hầu như không xuất hiện ngoại lai đáng kể do miền giá trị bị giới hạn. Sự tồn tại của các ngoại lai này là hợp lý trong bối cảnh dữ liệu nhân sự và cần được xem xét khi huấn luyện mô hình.

3.3.2 Biểu đồ thống kê dữ liệu

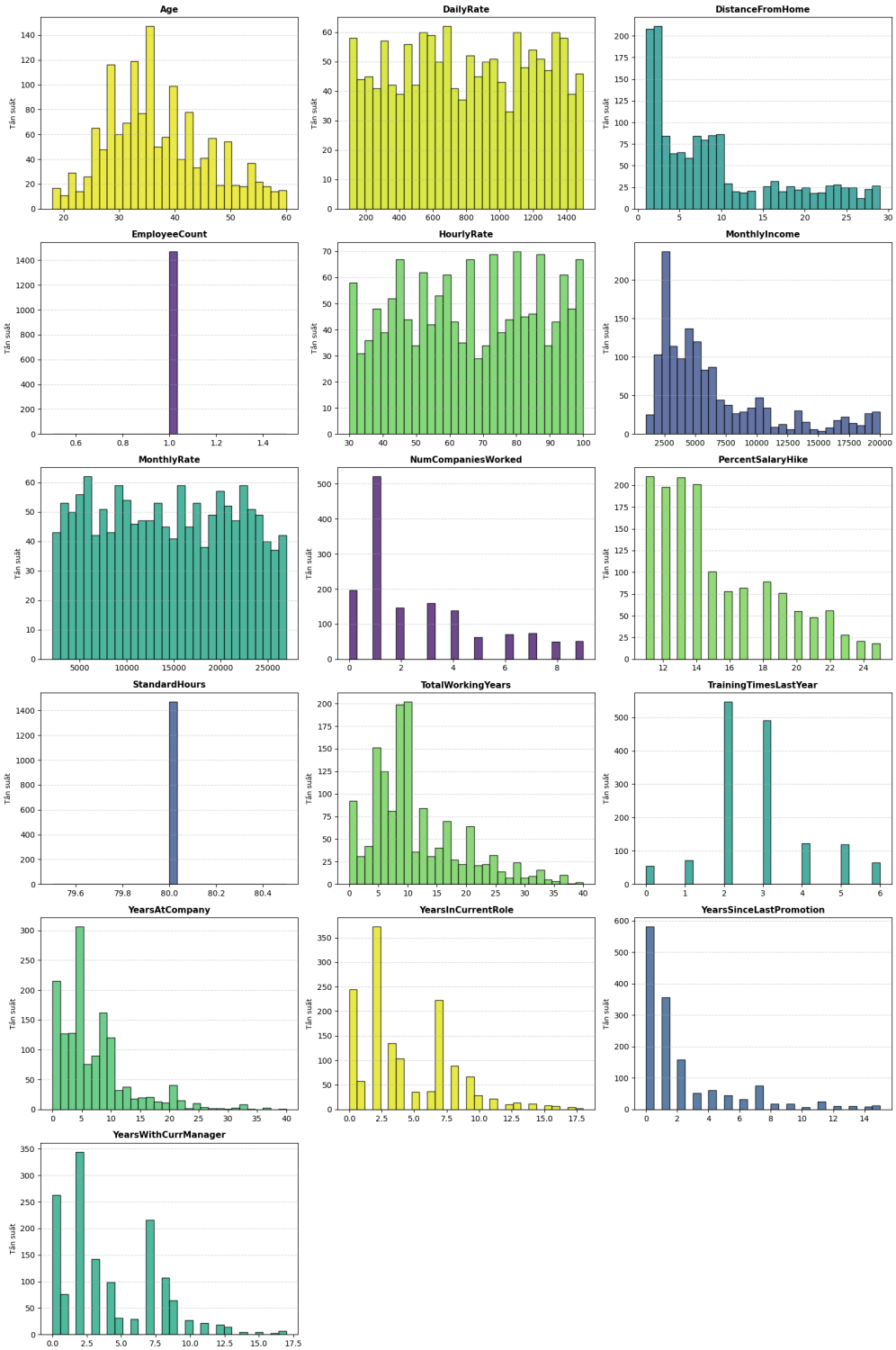
Dữ liệu rời rạc



Hình 3.11: Biểu đồ thống kê giá trị của dữ liệu rời rạc.

Có thể thấy được là trường dữ liệu `Over18` có dữ liệu không đổi, đây cũng là lý do trường này sẽ được loại bỏ khi sử dụng dữ liệu để huấn luyện mô hình.

Dữ liệu liên tục



Hình 3.12: Biểu đồ thống kê giá trị của dữ liệu liên tục.

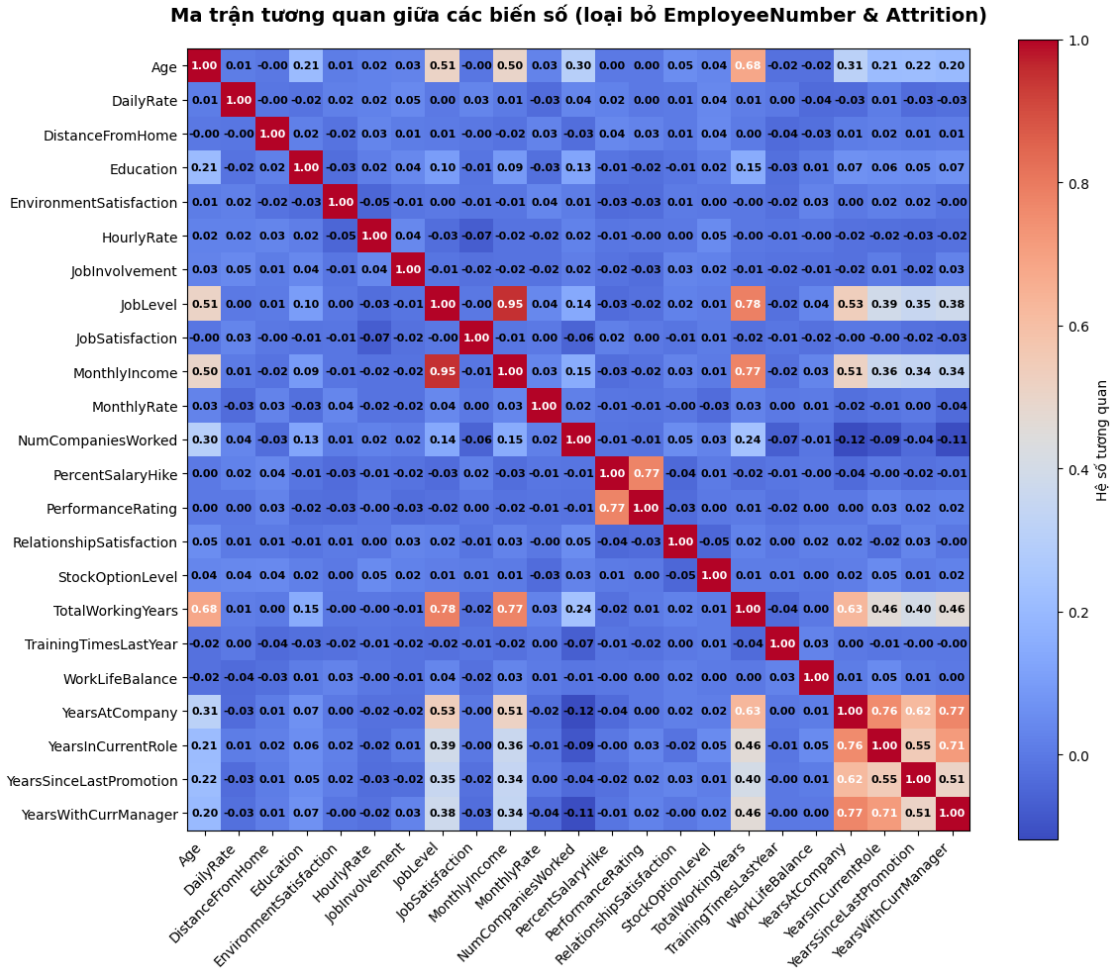
Trường dữ liệu `EmployeeCount`, `StandardHours` tương tự sẽ bị loại bỏ vì có dữ liệu không đổi.

Nhận xét tổng thể:

Các biểu đồ phân bố cho thấy dữ liệu gồm cả biến rời rạc và liên tục với đặc điểm phân bố không đồng đều. Đối với các biến phân loại, dữ liệu có xu hướng mất cân bằng giữa các nhóm, chẳng hạn như `BusinessTravel`, `Department`, `JobRole` và `Gender`, trong đó một số nhóm chiếm tỷ lệ vượt trội so với phần còn lại.

Đối với các biến số liên tục, nhiều biến như `MonthlyIncome`, `TotalWorkingYears`, `YearsAtCompany` và `YearsInCurrentRole` có phân bố lệch phải, phản ánh sự khác biệt lớn về thu nhập và thâm niên làm việc giữa các nhân viên. Một số biến khác như `Age` và `DistanceFromHome` có phân bố tương đối tập trung, trong khi các biến đánh giá theo thang điểm rời rạc có phân bố bị giới hạn trong miền giá trị nhỏ.

3.3.3 Ma trận hệ số tương quan



Hình 3.13: Ma trận hệ số tương quan.

Nhận xét:

Ma trận tương quan cho thấy cặp biến `MonthlyIncome` và `JobLevel` với hệ số $r = 0.95$, phản ánh mối quan hệ tuyến tính gần như tuyệt đối giữa thu nhập và cấp bậc. Bên cạnh đó, thâm niên làm việc (`TotalWorkingYears`) đóng vai trò trung tâm khi có tương quan mạnh với cả cấp bậc (0.78) và thu nhập (0.77), đồng thời biến `PerformanceRating` cũng được xác định là yếu tố ảnh hưởng trực tiếp đến mức tăng lương (`PercentSalaryHike`) với $r = 0.77$.

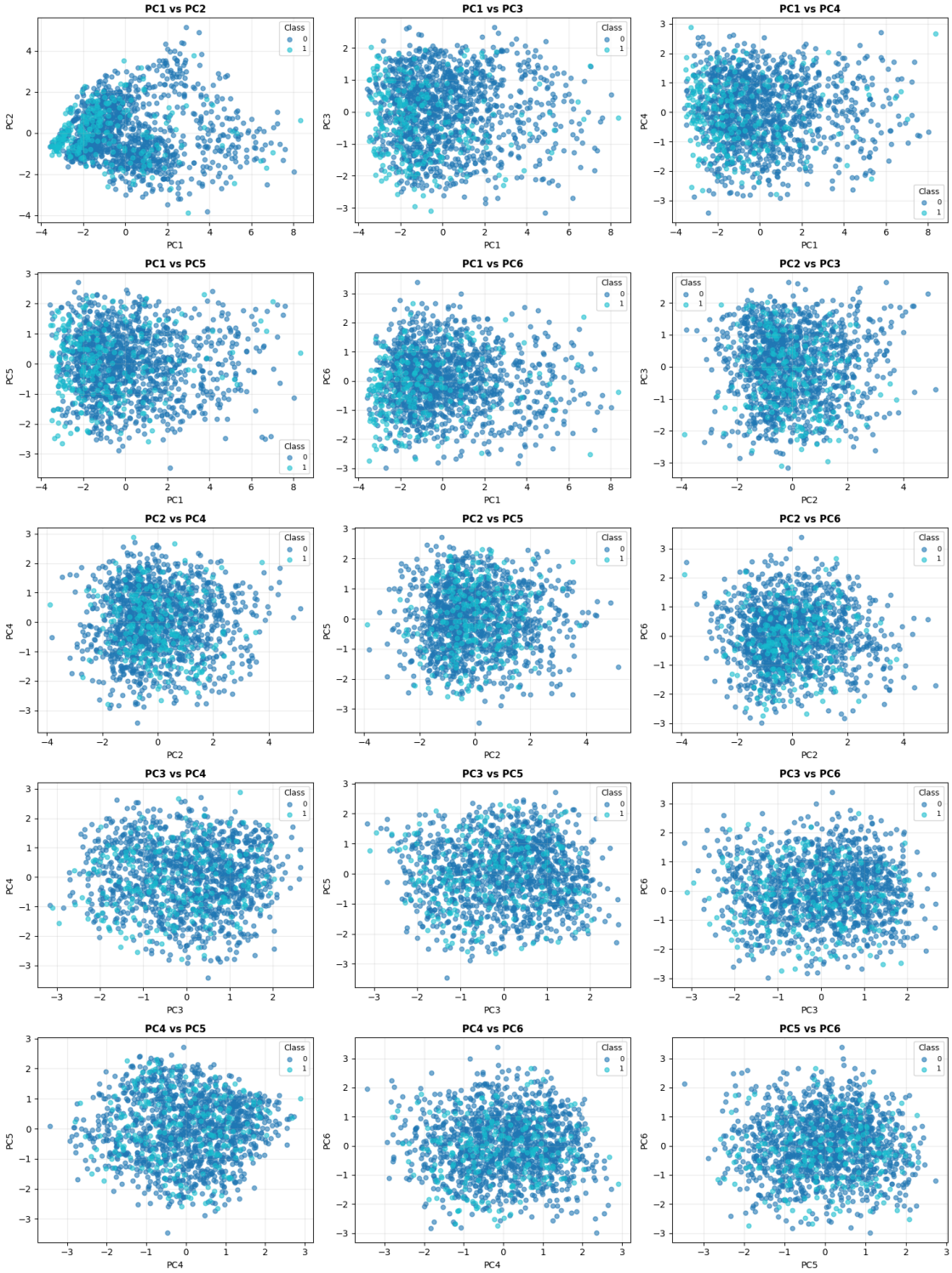
Các trường `YearsAtCompany`, `YearsInCurrentRole` và `YearsWithCurrManager` có sự phụ thuộc lẫn nhau chặt chẽ (hệ số từ 0.71 đến 0.77), cho thấy sự ổn định về vị trí thường đi kèm với việc gắn bó lâu dài với một người quản lý. Ngược lại, các chỉ số về mức độ hài lòng (*Satisfaction*) lại hoàn toàn độc lập và gần như không có tương quan với các yếu tố tài chính hay nhân khẩu học (hệ số

xấp xỉ 0), ngụ ý rằng thu nhập cao hay thâm niên lớn không đồng nghĩa với mức độ thỏa mãn công việc cao hơn.

3.3.4 Giảm chiều dữ liệu

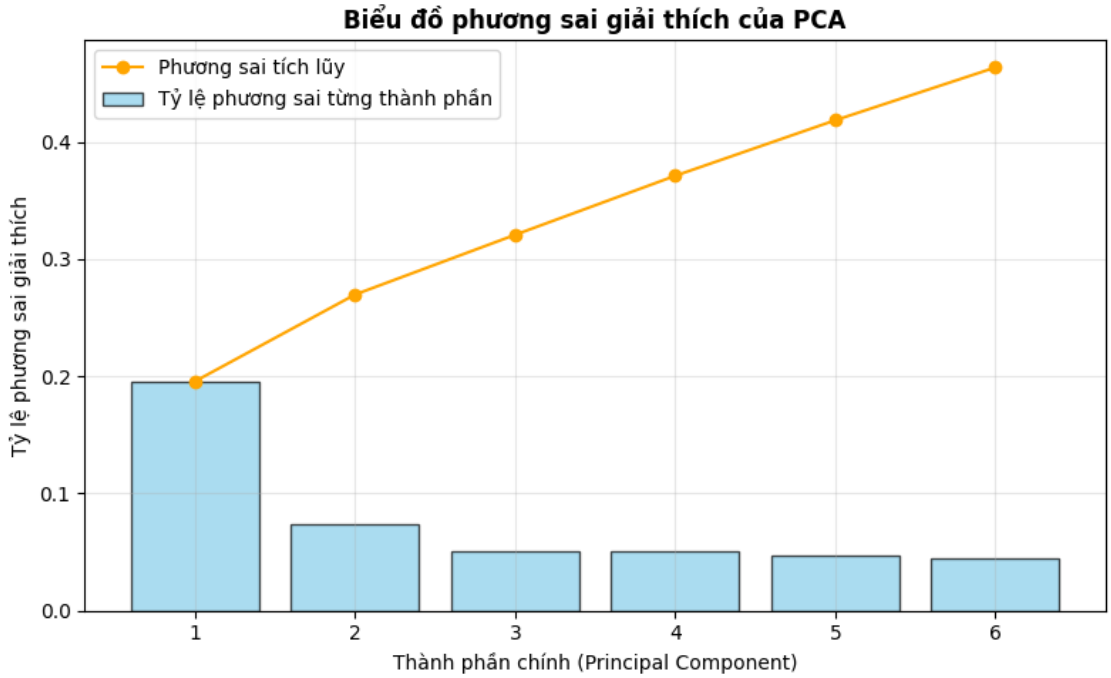
Giảm chiều dữ liệu với PCA

Ở đây chúng tôi sử dụng phương pháp giảm chiều PCA để giảm chiều dữ liệu về còn 6 chiều, rồi trực quan hóa từng cặp dữ liệu với nhau.



Hình 3.14: Trực quan hóa từng cặp dữ liệu.

Biểu đồ phương sai tích lũy của dữ liệu sau khi giảm chiều.



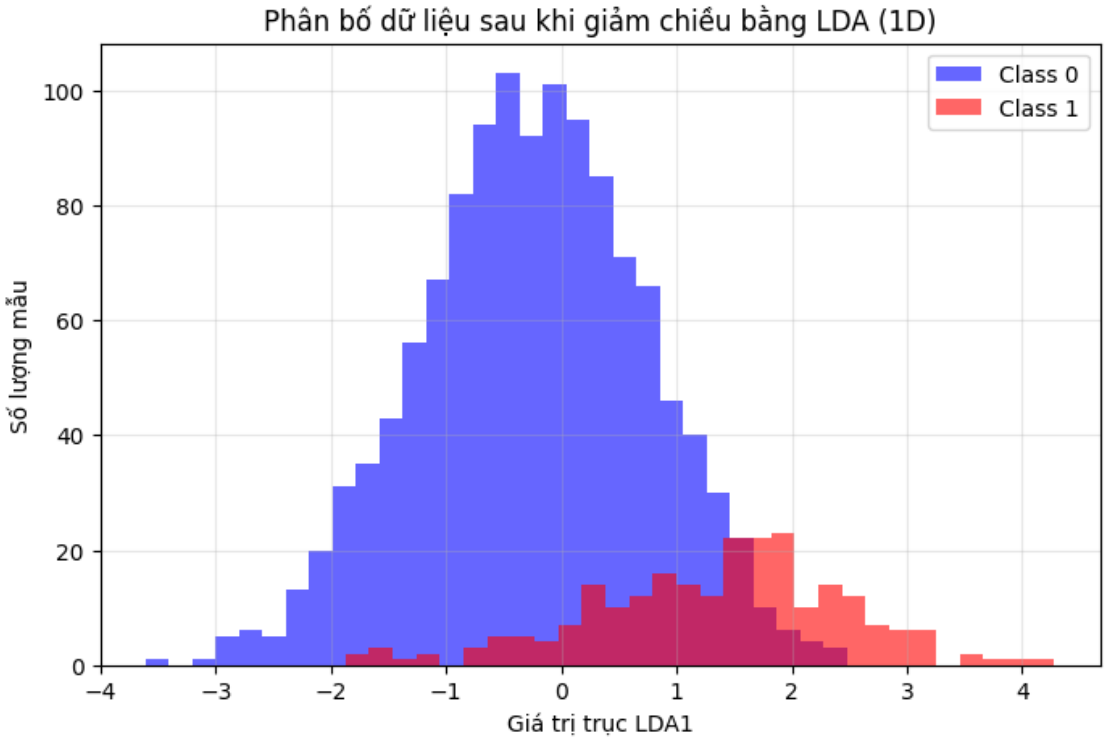
Hình 3.15: Biểu đồ phương sai tích lũy sau khi giảm chiều bằng PCA.

Nhận xét:

Phương sai tích lũy từ PC1 đến PC6 chỉ đạt 46.36% tức nó chỉ giữ lại khoảng hơn 40% dữ liệu quan trọng. Dữ liệu bị phân tán một phần sau khi one hot encode, dữ liệu có nhiều chiều, dẫn đến phương sai bị pha loãng, không thể tập trung. Phần khác là do có nhiều nhiễu của việc dữ liệu nhiều categories nên làm cho PCA kém hiệu quả.

Giảm chiều dữ liệu với LDA

Do dữ liệu phân loại chỉ có 2 lớp (yes/no) nên ở đây chúng tôi chỉ có thể giảm về còn 1 chiều dữ liệu.



Hình 3.16: Biểu đồ thể hiện giảm chiều với LDA.

Nhận xét:

Biểu đồ cho thấy LDA đã bước đầu tách biệt được tâm của hai lớp (Class 0 (No) lệch trái, Class 1 (Yes) lệch phải), tuy nhiên phương sai lớn khiến vùng chồng lấn (overlap) vẫn còn rất rộng, gây khó khăn cho việc phân loại chính xác tuyệt đối. Ngoài ra, sự chênh lệch lớn về chiều cao cột giữa hai lớp phản ánh vấn đề mất cân bằng dữ liệu nghiêm trọng.

3.4 Quy trình thực nghiệm phân cụm

3.4.1 Mô hình

Trong khuôn khổ thực nghiệm phân cụm lần này, chúng tôi sử dụng 2 phương pháp phân cụm:

1. DBSCAN (Density-Based Spatial Clustering of Applications with Noise.)
2. K-Means

3.4.2 Kịch bản

Kịch bản 1

Thực hiện phân cụm với dữ liệu ban đầu với nhiều tham số khác nhau.

Kịch bản 2

Thực hiện phân cụm trên dữ liệu đã được giảm chiều với các tham số khác nhau.

3.5 Quy trình thực nghiệm phân loại

3.5.1 Mô hình

Với bài toán phân loại để kiểm tra xem nhân viên có nghỉ việc hay không (yes/no), chúng tôi sử dụng các mô hình:

1. Naive Bayes.
2. K-Nearest Neighbors.
3. Logistic Regression.
4. Decision Tree.
5. Support Vector Machine.

3.5.2 Kịch bản

Kịch bản 1

Thực hiện huấn luyện mô hình với dữ liệu đã được mã hóa bằng phương pháp One-Hot Encoding, với các trường hợp chia tập huấn luyện và kiểm tra (train/test) theo yêu cầu.

Kịch bản 2

Thực hiện chạy mô hình với dữ liệu đã được chuẩn hóa và giảm chiều bằng phương pháp Feature Selection (tùy theo từng mô hình), với các trường hợp chia train/test khác nhau theo yêu cầu.

Kịch bản 3

Thực hiện chạy mô hình với dữ liệu đã được chuẩn hóa và giảm chiều bằng phương pháp PCA (tùy theo từng mô hình), với các trường hợp chia train/test khác nhau theo yêu cầu.

Kịch bản 4

Thực hiện chạy mô hình với dữ liệu đã được chuẩn hóa và giảm chiều bằng phương pháp LDA (tùy theo từng mô hình), với các trường hợp chia train/test khác nhau theo yêu cầu.

Kịch bản 5

Thực hiện trực quan hóa kết quả mô hình đối với một hoặc hai kịch bản tiêu biểu ở trên.

Kịch bản 6

Thực hiện thực nghiệm lại các mô hình trong các kịch bản đã xây dựng, với dữ liệu được cân bằng bằng phương pháp SMOTE.

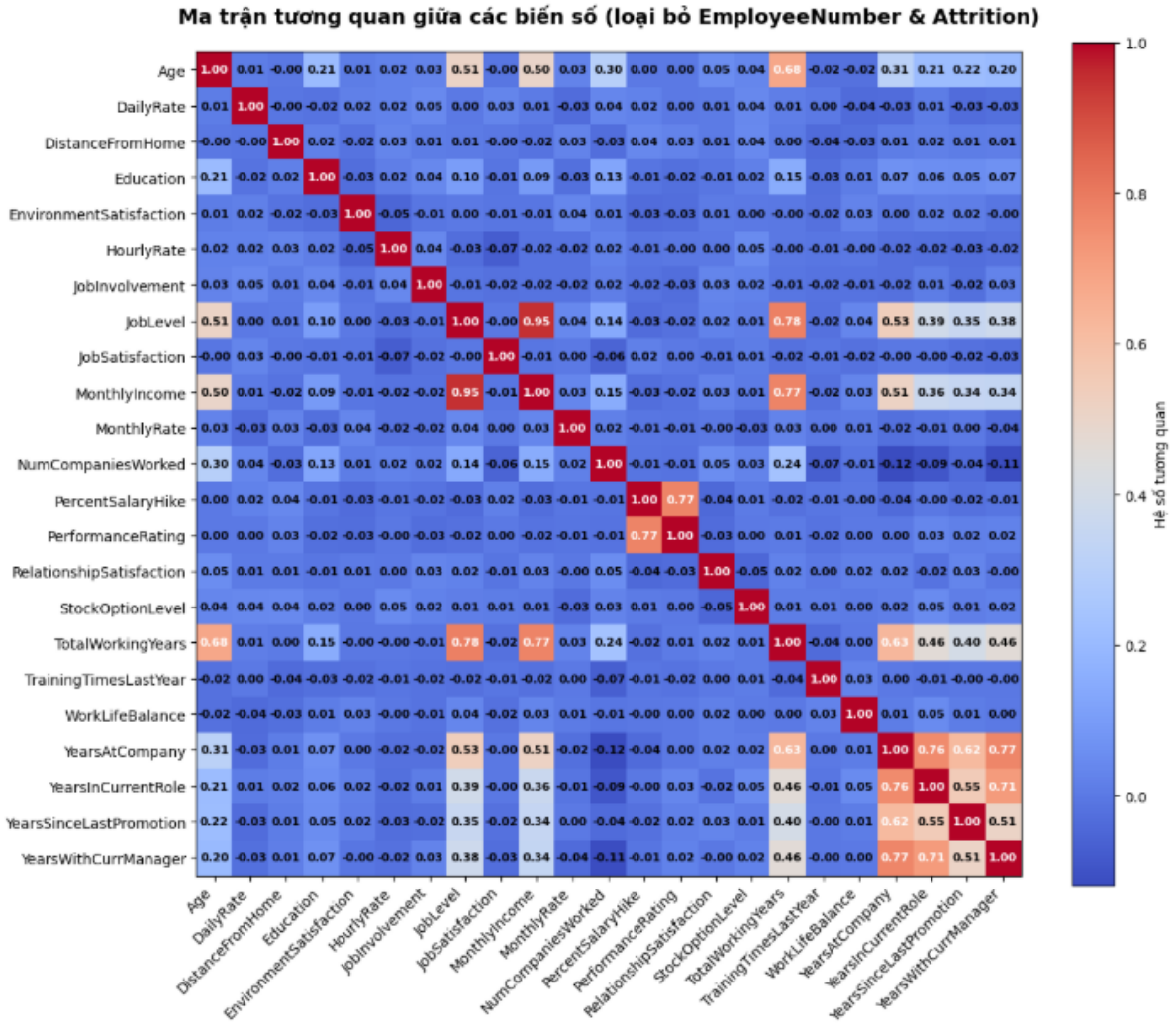
3.6 Quy trình thực nghiệm hồi quy

Với tập dữ liệu IBM HR Analytics Employee Attrition & Performance chúng tôi lựa chọn xây dựng bài toán hồi quy với mục tiêu dự đoán thu nhập hàng tháng của nhân viên dựa trên các đặc trưng của tập dữ liệu.

Trường `MonthlyIncome` là một biến liên tục, do đó các mô hình hồi quy được lựa chọn thay vì các mô hình phân loại, nó đại diện cho thu nhập hàng

tháng của nhân viên dựa vào các dữ liệu được thu thập trước đó, đây là biến liên tục và phù hợp với bài toán hồi quy.

Ngoài ra biến `MonthlyIncome` có tương quan với nhiều biến khác trong dữ liệu, cụ thể:



Hình 3.17: Ma trận hệ số tương quan các trường dữ liệu.

Với các biến có mức độ tương quan cao với biến mục tiêu, ta có:

- `Age` (≈ 0.5)
- `JobLevel` (≈ 0.95)
- `TotalWorkingYears` (≈ 0.77)
- `YearAtCompany` (≈ 0.51)
- `YearInCurrentRole` (≈ 0.36)

Ngoài ra, mô hình còn xem xét nhiều biến khác với mức độ ảnh hưởng khác nhau.

Các biến này phản ánh trực tiếp cơ cấu lương của công ty như:

- Thâm niên làm việc càng cao thì thu nhập của nhân viên càng lớn.
- Trình độ và cấp bậc công việc càng cao thì mức lương tương ứng càng cao.
- Thời gian gắn bó với công ty càng lâu thì khả năng đạt mức lương cao càng lớn.
- Thời gian đảm nhiệm vai trò hiện tại càng dài thì mức lương càng ổn định và có xu hướng tăng.

3.6.1 Mô hình

Trong bài toán hồi quy thu nhập hàng tháng của nhân viên tại IBM, chúng tôi sử dụng các mô hình:

1. Multilayer Perceptron.
2. SVM.

3.6.2 Kịch bản

Kịch bản 1

Thực hiện huấn luyện mô hình trên dữ liệu gốc, không được chuẩn hóa.

Kịch bản 2

Thực hiện huấn luyện mô hình trên dữ liệu gốc đã được chuẩn hóa.

Kịch bản 3

Thực hiện huấn luyện mô hình trên dữ liệu gốc đã được chuẩn hóa, giảm chiều với PCA và LDA.

Chương 4

Kết quả và thảo luận

4.1 Kết quả thực nghiệm phân cụm

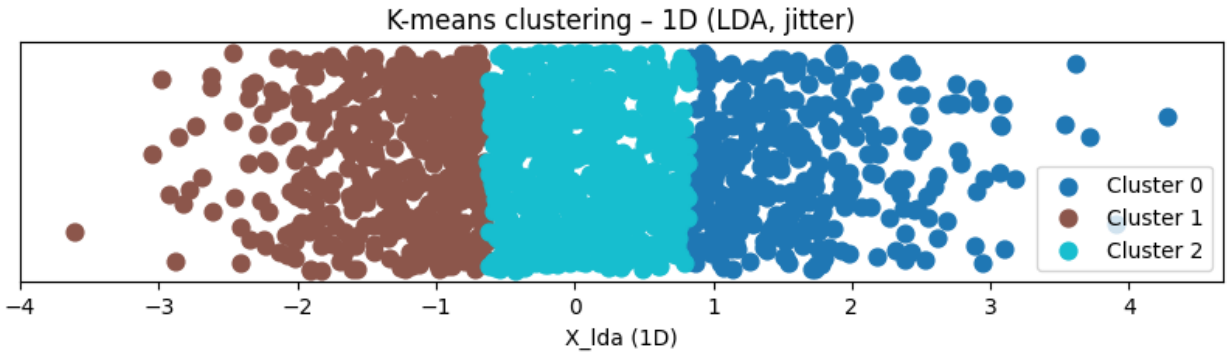
4.1.1 Mô hình DBSCAN

Thực nghiệm

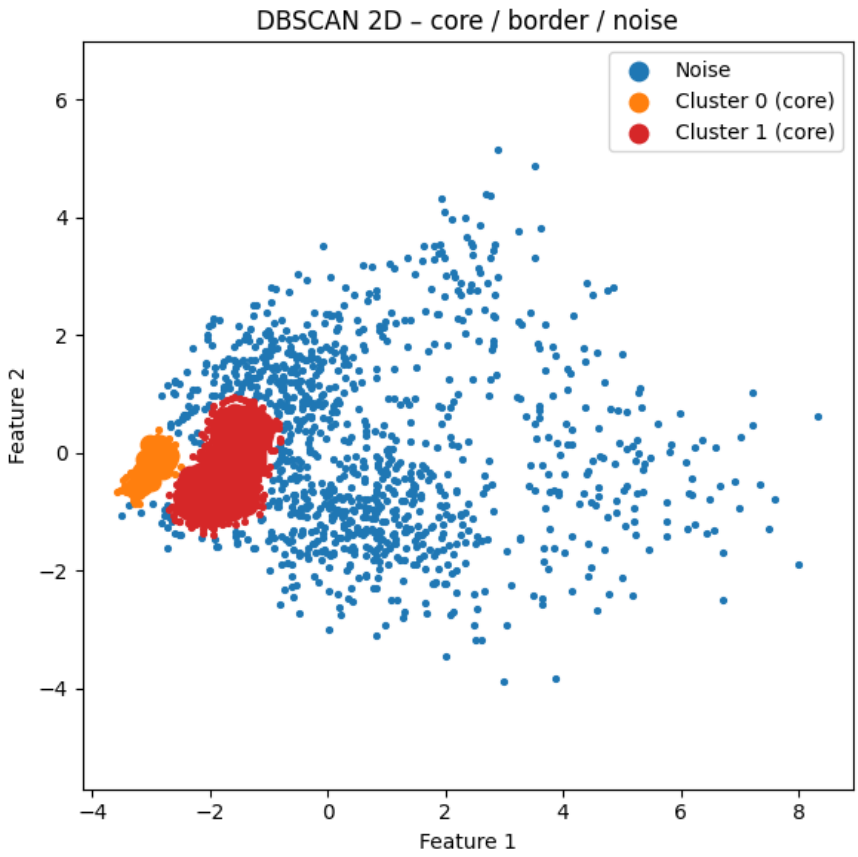
Ở đây dữ liệu thực nghiệm được lấy theo kết quả tốt nhất của chỉ số ARI.

Dữ liệu	ε	**min_samples**	**Metric**	**ARI**
Gốc	0.07	3	cosine	0.0431
PCA (2D)	0.30	30	euclidean	0.1771
PCA (3D)	0.50	20	euclidean	0.0922
PCA (5D)	0.07	44	cosine	0.0443
PCA (90%)	0.20	5	cosine	0.1740
LDA	0.07	5	euclidean	0.2325

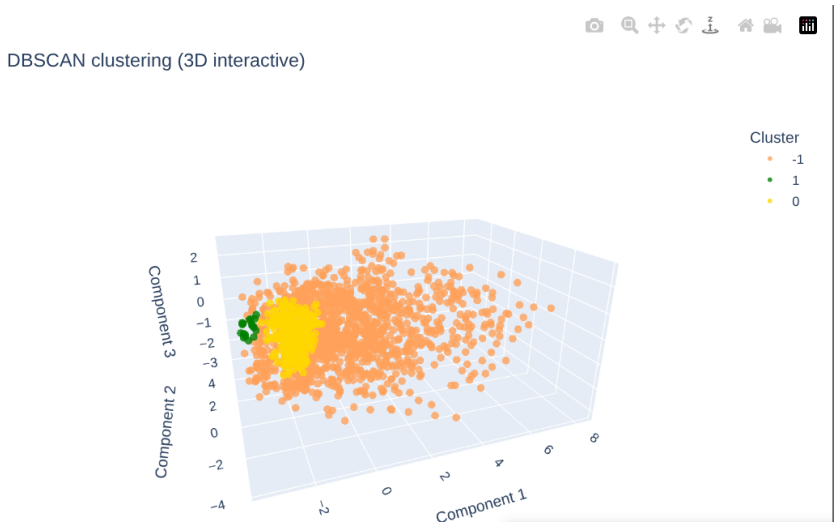
Trực quan hóa dữ liệu sau phân cụm



Hình 4.1: Trực quan dữ liệu phân cụm 1D sử dụng DBSCAN.



Hình 4.2: Trực quan dữ liệu phân cụm 2D sử dụng DBSCAN.



Hình 4.3: Trực quan dữ liệu phân cụm 3D sử dụng DBSCAN.

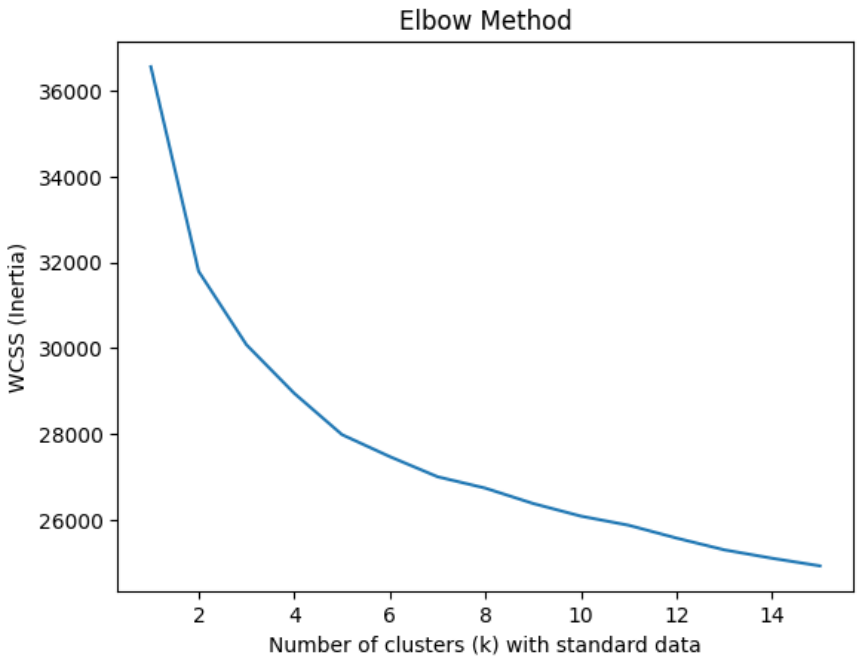
Nhận xét

Kết quả thực nghiệm cho thấy thuật toán DBSCAN hoạt động kém trên dữ liệu ban đầu, khi chỉ số ARI đạt giá trị rất thấp hoặc xấp xỉ 0 đối với hầu hết các cấu hình tham số. Việc giảm chiều dữ liệu bằng PCA giúp cải thiện chất lượng phân cụm trong một số trường hợp, đặc biệt với dữ liệu giảm về 2 chiều hoặc giữ lại 90% phương sai, tuy nhiên kết quả vẫn chưa ổn định.

Ngược lại, phương pháp giảm chiều bằng LDA cho kết quả vượt trội rõ rệt, với ARI đạt giá trị cao nhất trong tất cả các kịch bản. Điều này cho thấy việc khai thác thông tin nhãn trong quá trình giảm chiều giúp dữ liệu trở nên phân tách hơn, từ đó nâng cao hiệu quả của DBSCAN khi kết hợp với khoảng cách Euclidean.

4.1.2 Mô hình K-Means

Thực hiện phân tích số cụm.



Hình 4.4: Phân tích số cụm sử dụng K-Means.

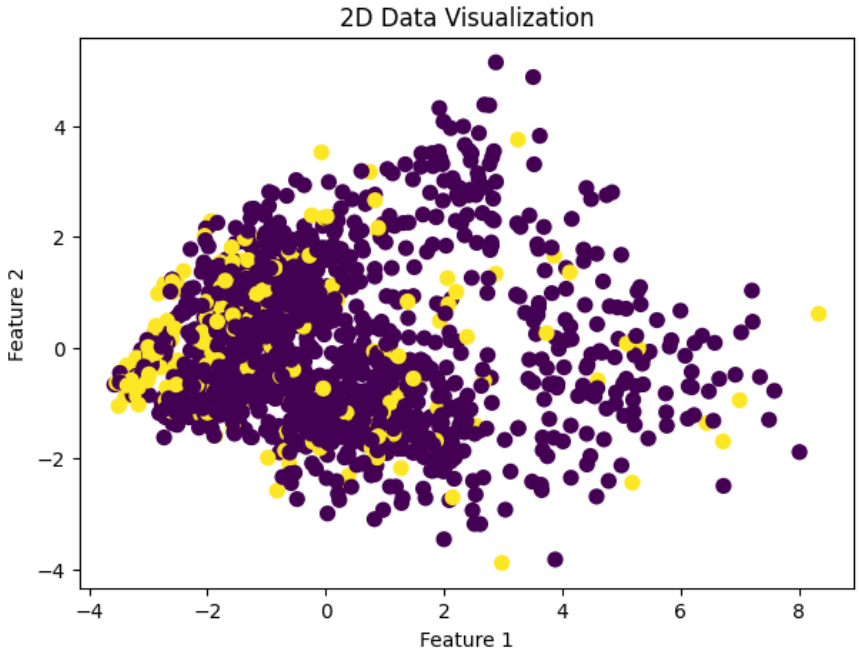
Nhận xét:

Qua biểu đồ, ta có thể thấy được WCSS giảm rất mạnh từ 1 đến 3, 4 cho thấy việc tăng số cụm trong khoảng này giúp mô hình cải thiện đáng kể.

Trong khoảng 4 - 6 thì WCSS vẫn còn giảm tuy nhiên chậm hơn, biểu thị rằng cấu trúc các cụm trở nên ổn định dần.Từ khoảng 6 trở đi thì WCSS giảm tuy nhiên ít hơn ban đầu, điều này cho thấy tăng số cụm không mang lại nhiều ý nghĩa mà chỉ làm chia nhỏ cụm hiện có.

=> Nên ở đây với dữ liệu đã được chuẩn hóa số cụm có thể ước trừng cho mô hình K-Means là khoảng 4 cụm.

Trực quan với dữ liệu đã giảm chiều



Hình 4.5: Trực quan hóa với dữ liệu 2 chiều sau khi giảm.

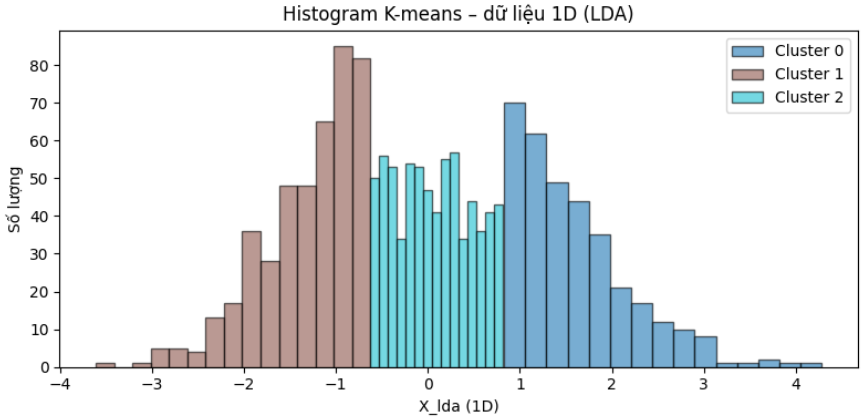
Thực nghiệm

Ở đây chúng tôi sử dụng độ đo ARI để đánh giá chất lượng phân cụm dữ liệu.

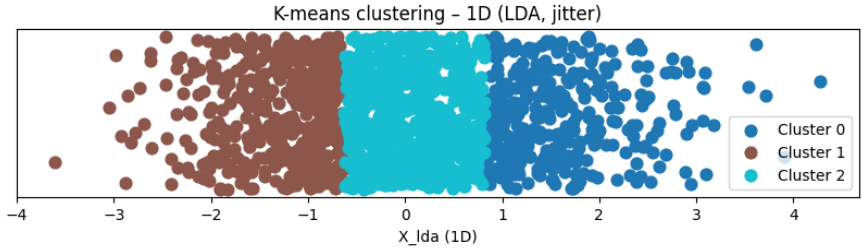
k	**Gốc**	**PCA (2D)**	**PCA (3D)**	**PCA (5D)**	**PCA (90%)**	**LDA**
2	-0.035594	-0.034565	-0.035594	-0.035597	-0.036636	0.119530
3	-0.042245	-0.041807	-0.041807	-0.042057	-0.041624	0.120366
4	-0.005583	-0.025229	-0.026613	-0.027248	-0.005259	0.093000
5	-0.009324	-0.009569	-0.007023	-0.011444	-0.009054	0.078202
6	0.001449	-0.013797	0.002151	-0.001115	0.001021	0.062683
7	0.005852	-0.006845	-0.003518	-0.006104	0.004883	0.049959
8	0.003432	-0.010765	-0.000261	-0.002720	-0.002164	0.042574

Trực quan dữ liệu phân cụm

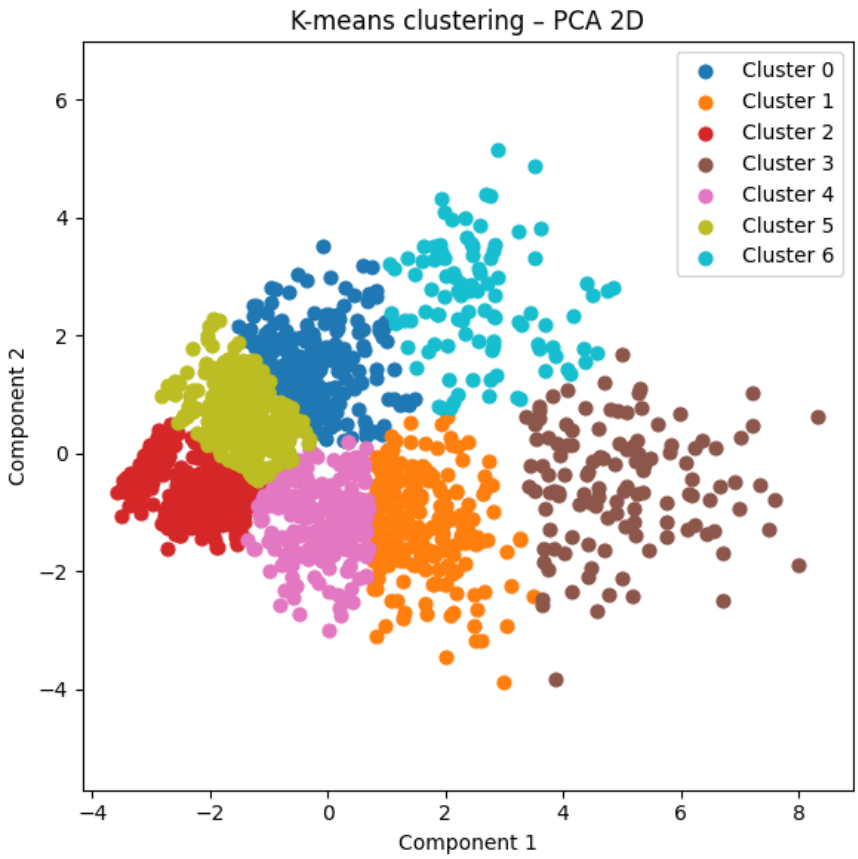
Phần trực quan ở đây được lấy theo trường hợp tốt nhất theo chỉ số ARI.



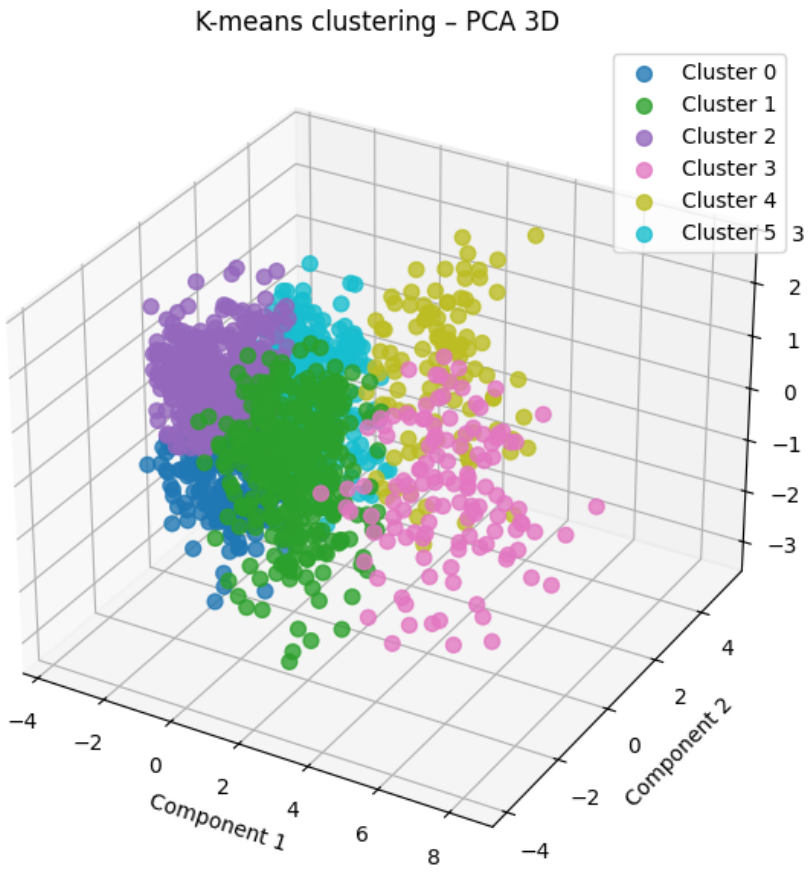
Hình 4.6: Trực quan hóa phân cụm với dữ liệu 1D giảm chiều bằng LDA.



Hình 4.7: Trực quan hóa phân cụm với dữ liệu 1D giảm chiều bằng LDA.



Hình 4.8: Trực quan hóa phân cụm với dữ liệu 2D giảm chiều bằng PCA.



Hình 4.9: Trực quan hóa phân cụm với dữ liệu 3D giảm chiều bằng PCA.

Nhận xét

Kết quả trong bảng cho thấy phương pháp giảm chiều bằng PCA (với các số chiều khác nhau hoặc giữ lại 90% phương sai) không cải thiện đáng kể chất lượng phân cụm, khi chỉ số ARI chủ yếu mang giá trị âm hoặc xấp xỉ 0. Ngược lại, phương pháp LDA cho kết quả vượt trội rõ rệt ở tất cả các giá trị k, với ARI dương và cao hơn nhiều so với các phương pháp còn lại. Điều này cho thấy việc khai thác thông tin nhãn trong LDA giúp tăng khả năng phân tách dữ liệu hiệu quả hơn so với các phương pháp giảm chiều không giám sát.

4.2 Kết quả thực nghiệm phân loại

Kết quả với dữ liệu gốc

Bảng 4.1: Kết quả với tỉ lệ Train / Test = 7 / 3

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Hồi quy Logistic	0.8775	0.7179	0.3943	0.5090
Naive Bayes (GNB)	0.6576	0.2400	0.6900	0.3600
Naive Bayes (GNB + BNB)	0.8526	0.4600	0.4300	0.4400
KNN	0.8639	0.5600	0.0800	0.1400
Decision Tree	0.7823	0.3655	0.4788	0.4146
SVM	0.8934	0.7200	0.3800	0.4900

Bảng 4.2: Kết quả với tỉ lệ lệ Train / Test = 6 / 4

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Hồi quy Logistic	0.8775	0.6949	0.4315	0.5324
Naive Bayes (GNB)	0.6684	0.2500	0.6900	0.3600
Naive Bayes (GNB + BNB)	0.8605	0.4900	0.3600	0.4100
KNN	0.8605	0.4800	0.1600	0.2400
Decision Tree	0.7823	0.3655	0.4788	0.4146
SVM	0.8878	0.6800	0.3500	0.4600

Kết quả với dữ liệu có sử dụng SMOTE

Bảng 4.3: Kết quả với tỉ lệ Train / Test tỉ lệ tốt nhất khi sử dụng SMOTE

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Hồi quy Logistic	0.7806	0.3988	0.7052	0.5095
Naive Bayes (GNB)	0.6973	0.2600	0.6300	0.3600
KNN	0.8520	0.4700	0.4900	0.4800
Decision Tree	0.8112	0.4148	0.4105	0.4126
SVM	0.8277	0.4100	0.5900	0.4900

Kết quả với dữ liệu giảm chiều

Bảng 4.4: Giảm chiều với PCA (n=6) - Kết quả trên tập TEST

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Hồi quy Logistic	0.8401	0.5714	0.0421	0.0784
Naive Bayes (GNB)	0.8673	0.0000	0.0000	0.0000
KNN	0.8350	0.3200	0.1700	0.2200
SVM	0.7976	0.2400	0.2100	0.2200

Bảng 4.5: Giảm chiều với LDA - Kết quả trên tập TEST

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Hồi quy Logistic	0.8707	0.6610	0.4105	0.5064
Naive Bayes (GNB)	0.8776	0.5900	0.3800	0.4600
KNN	0.8690	0.5400	0.3600	0.4300
SVM	0.8844	0.6300	0.3100	0.4600

Bảng 4.6: Giảm chiều với Feature Selection - Kết quả trên tập TEST

Mô hình	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	0.7993	0.4000	0.4842	0.4380

4.3 Kết quả thực nghiệm hồi quy

Bảng 4.7: Kết quả thực nghiệm tốt nhất của Multi Layer Perceptron

Dataset	**Solver**	**Layers**	**Test**	**LR**	**Alpha**	**MAE**	**RMSE**	R^2
Grad.Desc	sgd	(100,75,50)	0.3	0.3	.001	6627	8009	-2.17
Original	adam	(100,75)	0.2	.001	.01	1885	2520	0.71
Normalized	adam	(50,25)	0.2	.001	.01	1150	1514	0.90
Reduced Dim	adam	(100,75,50)	0.3	.001	.01	1368	1803	0.84

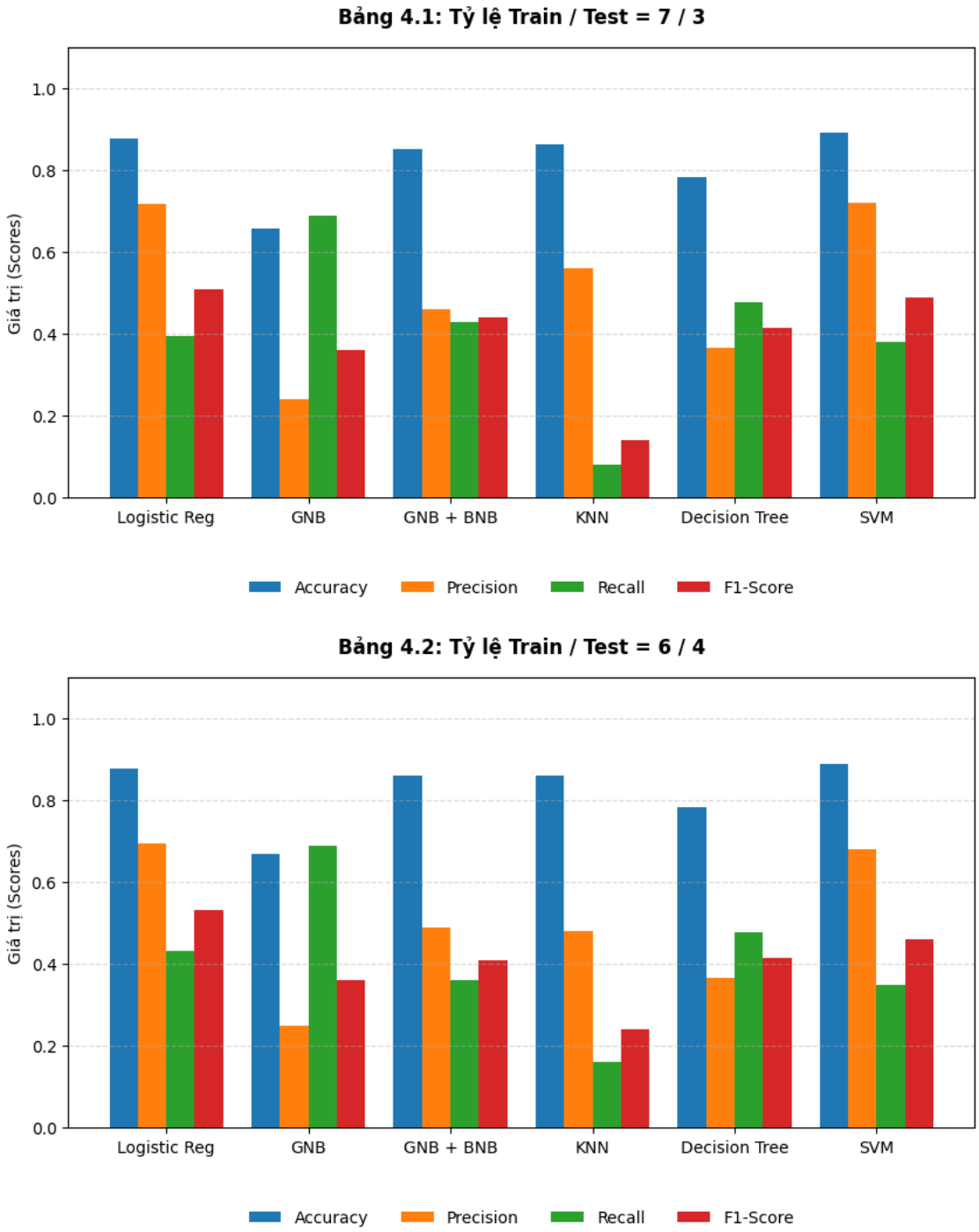
Bảng 4.8: Kết quả thực nghiệm tốt nhất của mô hình SVM

Dataset	**Kernel**	**C**	**Epsilon**	**Test Size**	**MAE**	**RMSE**	R^2
Onehot & Norm	Linear	1000	10.0	0.2	884.66	1181.51	0.9361
PCA	Linear	1000	10.0	0.3	2047.92	2737.43	0.6295

4.4 So sánh và thảo luận thực nghiệm phân loại

Bảng so sánh

Từ kết quả của các mô hình trên tập dữ liệu ban đầu, ta có bảng sau.



Hình 4.10: Bảng thống kê kết quả thực nghiệm phân loại.

Kết quả trên dữ liệu gốc (Tỉ lệ 7/3)

Dựa trên Bảng 4.1, hiệu năng các mô hình trên 44 chiều dữ liệu được tóm tắt như sau:

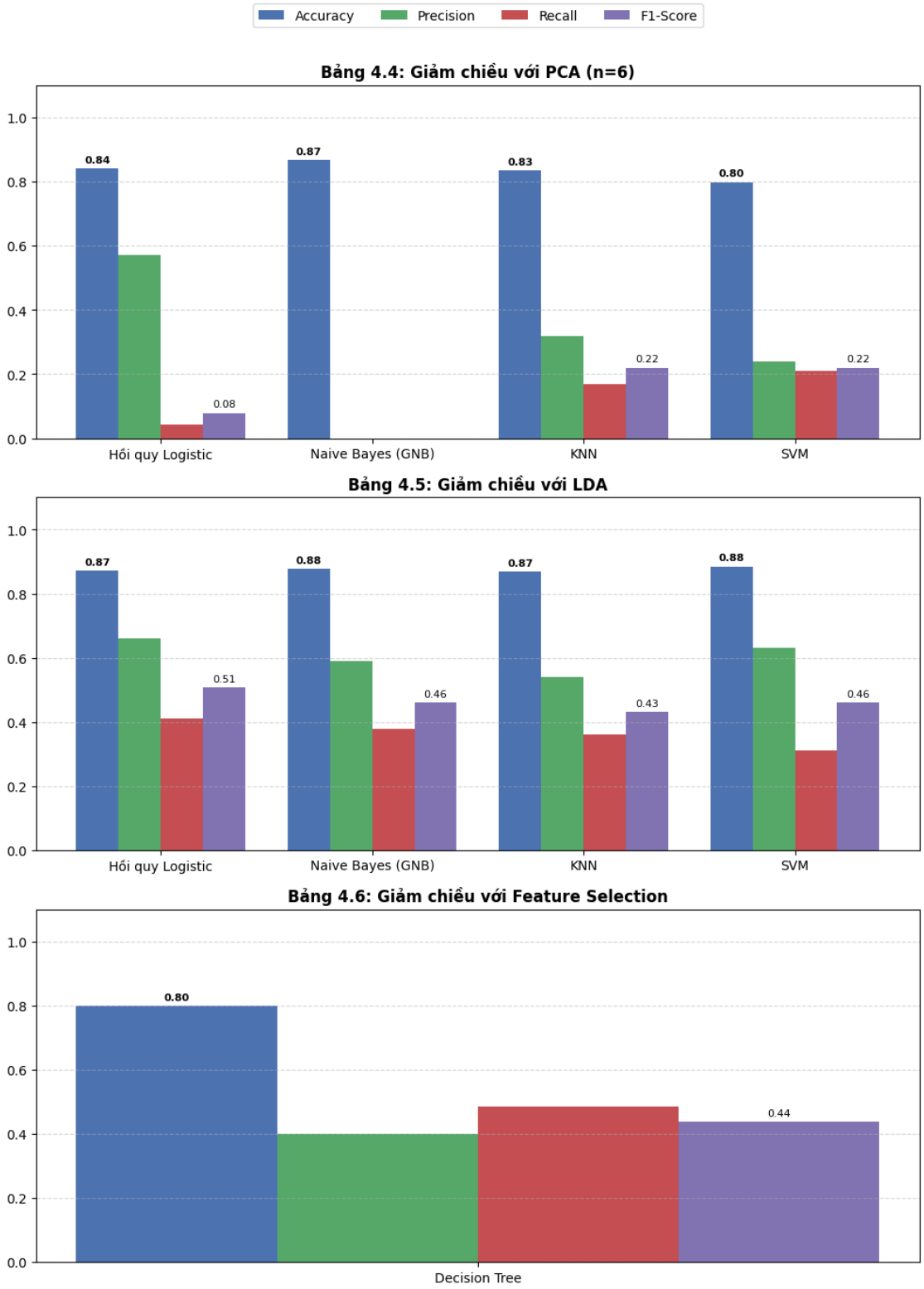
- **Hiệu suất tổng thể tốt nhất:** SVM và Hồi quy Logistic dẫn đầu với Accuracy cao (0.89 và 0.88) cùng Precision tốt (> 0.71). Tuy nhiên, cả

hai đều có hạn chế lớn ở chỉ số Recall ($0.38 - 0.39$), tức là bỏ sót nhiều nhân viên thực sự nghỉ việc.

- **Độ nhạy cao nhất:** Naive Bayes (GNB) đạt Recall cao nhất (0.6900) nhưng Precision lại thấp nhất (0.2400), cho thấy mô hình chấp nhận tỉ lệ báo động giả cao để phát hiện tối đa các ca nghỉ việc.
- **Hiệu suất kém nhất:** KNN gần như thất bại trong việc nhận diện lớp thiểu số (Recall chỉ đạt 0.0800), trong khi Decision Tree cho kết quả trung bình và không vượt trội ở chỉ số nào.

Tác động của giảm chiều dữ liệu (PCA vs. LDA)

Biểu đồ so sánh.



Hình 4.11: Bảng thống kê kết quả thực nghiệm phân loại với dữ liệu đã giảm chiều.

So sánh kết quả giữa Bảng 4.4 (PCA, n=6) và Bảng 4.5 (LDA):

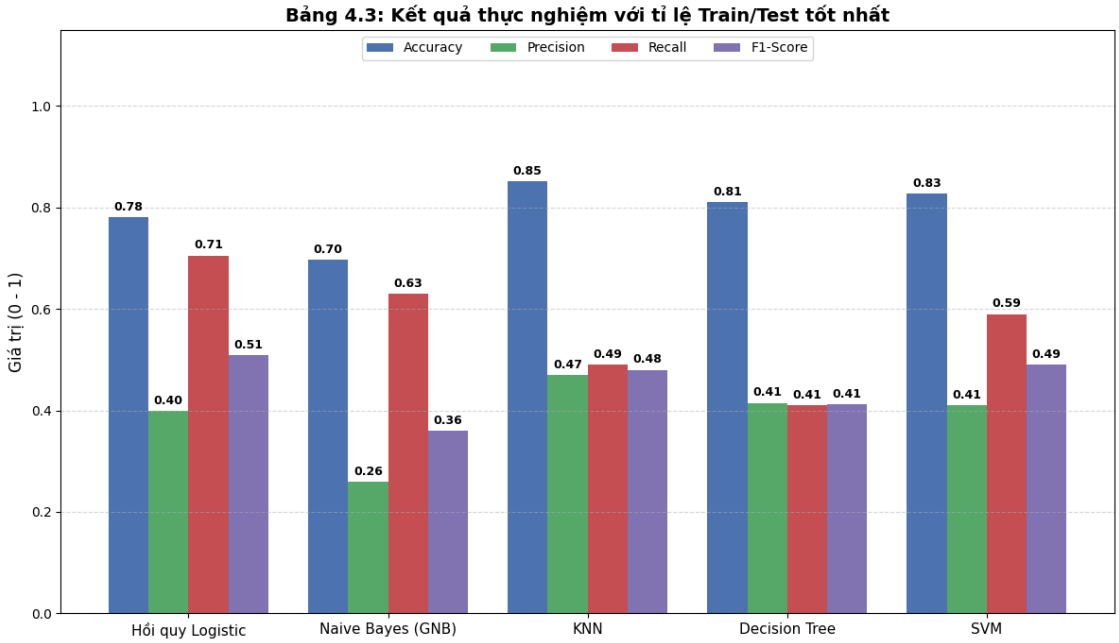
- **PCA (Không hiệu quả):** Việc nén xuống 6 thành phần chính làm

suy giảm nghiêm trọng hiệu suất của mọi mô hình. Đặc biệt, GNB và Logistic Regression gần như mất hoàn toàn khả năng dự đoán lớp "Nghỉ việc"(Recall ≈ 0). Điều này cho thấy PCA đã làm mất thông tin phân lớp quan trọng.

- **LDA (Hiệu quả vượt trội):** Ngược lại, LDA giúp duy trì Accuracy ổn định trên 86% cho tất cả mô hình. Hồi quy Logistic hoạt động tốt nhất trên không gian LDA với F1-Score cao nhất (0.5064). LDA cũng giúp khắc phục điểm yếu chí mạng của KNN và GNB, biến chúng thành các mô hình cân bằng hơn.

Tác động của cân bằng dữ liệu (SMOTE)

Dựa trên Bảng 4.3, kỹ thuật SMOTE mang lại sự đánh đổi rõ rệt:



Hình 4.12: Bảng thống kê kết quả thực nghiệm trên dữ liệu đã sử dụng SMOTE.

- **Cải thiện Recall:** SMOTE giúp giải quyết vấn đề mất cân bằng dữ liệu, đẩy Recall của **Hồi quy Logistic** lên tới 70.52%.
- **Giảm Precision:** Do nhiễu từ dữ liệu nhân tạo, Precision của các mô hình đều giảm xuống dưới 50%.
- **Kết luận:** Hồi quy Logistic và SVM là hai thuật toán xử lý tốt nhất với

dữ liệu sau khi sinh mẫu, trong khi các mô hình phi tuyến như KNN hay Tree nhạy cảm với nhiễu và cho kết quả kém hơn.

Hiệu quả của lựa chọn đặc trưng (Feature Selection)

Dựa trên Bảng 4.6, việc áp dụng kỹ thuật lựa chọn đặc trưng cho mô hình Decision Tree mang lại cải thiện rõ rệt:

- **Tăng cường hiệu suất:** So với việc sử dụng toàn bộ 44 đặc trưng gốc (Bảng 4.1), mô hình sau khi chọn lọc đặc trưng đạt F1-Score cao hơn (0.4380) và cải thiện đáng kể chỉ số Recall (48.42%).
- **Ý nghĩa:** Điều này chứng minh rằng bộ dữ liệu ban đầu chứa nhiều đặc trưng nhiễu hoặc ít quan trọng. Việc loại bỏ chúng giúp Decision Tree giảm bớt độ phức tạp, tránh hiện tượng quá khớp (overfitting) và xây dựng được cấu trúc phân loại tổng quát hóa tốt hơn.

Kết luận chung

- **Mô hình tối ưu nhất:** Hồi quy Logistic là lựa chọn tốt nhất cho bài toán này. Nó đạt F1-Score cao nhất trên dữ liệu gốc và khả năng bao phủ (Recall) tốt nhất khi kết hợp với SMOTE.
- **Kỹ thuật giảm chiều:** LDA chứng minh sự vượt trội hoàn toàn so với PCA trong việc trích xuất đặc trưng phân lớp cho bộ dữ liệu nhân sự này.
- **Chiến lược đề xuất:** Nếu ưu tiên độ chính xác tổng thể, nên dùng Hồi quy Logistic trên dữ liệu gốc hoặc LDA. Nếu ưu tiên phát hiện người nghỉ việc (chấp nhận báo động giả), nên kết hợp Hồi quy Logistic với SMOTE.

4.5 So sánh và thảo luận thực nghiệm hồi quy

Dựa trên kết quả thực nghiệm từ Bảng 4.7 và Bảng 4.8, ta có bảng thống kê sau:

Hình 4.13: Bảng thống kê kết quả thực nghiệm hồi quy.

- Mô hình **SVM** (Linear Kernel, $C = 1000$) trên dữ liệu Onehot & Normalized đạt kết quả cao nhất toàn cục với $R^2 \approx 0.936$ và sai số RMSE thấp nhất (1181.51). Kết quả này vượt trội hơn so với cấu hình tốt nhất của MLP ($R^2 \approx 0.90$). Điều này gợi ý rằng dữ liệu có mối quan hệ tuyến tính mạnh mà SVM khai thác hiệu quả hơn.

- Đối với MLP, việc chuẩn hóa là yếu tố quyết định, giúp cải thiện R^2 từ 0.71 (dữ liệu gốc) lên 0.90. Bộ giải *Adam* cũng chứng minh sự vượt trội hoàn toàn so với *SGD* (không hội tụ, $R^2 < 0$).

- **Độ nhạy với giảm chiều dữ liệu (PCA):**

  – MLP duy trì độ ổn định khá tốt khi giảm chiều (R^2 giảm nhẹ từ 0.90

xuống 0.84), cho thấy khả năng học đặc trưng tốt của mạng nơ-ron ngay cả khi thông tin bị nén.

– Ngược lại, SVM bị suy giảm hiệu suất nghiêm trọng trên dữ liệu PCA (R^2 giảm sâu xuống 0.63). Điều này cho thấy SVM Linear phụ thuộc lớn vào không gian đặc trưng gốc để tìm ra siêu phẳng tối ưu.

Với tập dữ liệu này, SVM trên dữ liệu chuẩn hóa đầy đủ là lựa chọn tối ưu. Tuy nhiên, nếu buộc phải giảm chiều dữ liệu để tiết kiệm tài nguyên tính toán, MLP là lựa chọn phù hợp hơn.

Chương 5

Tổng kết

5.1 Kết luận

Đề tài đã hoàn thành mục tiêu cốt lõi là xây dựng mô hình dự đoán khả năng nghỉ việc của nhân viên tại công ty IBM. Quá trình nghiên cứu cơ sở lý thuyết và phân tích thực nghiệm đã cung cấp câu trả lời rõ ràng cho các câu hỏi nghiên cứu được đặt ra ban đầu.

5.2 Hạn chế của đề tài

5.2.1 Về phạm vi đặc trưng

Trong quá trình tiền xử lý, một số biến không cần thiết như Over18, EmployeeNumber, EmployeeCount, StandardHours sẽ bị loại bỏ do không mang lại lợi ích trong quá trình train mô hình. Hơn nữa, mô hình dự đoán chắc chắn còn bị ảnh hưởng bởi nhiều yếu tố phức tạp khác ngoài phạm vi bộ dữ liệu (ví dụ: yếu tố tâm lý cá nhân, văn hóa công ty, hoặc điều kiện thị trường lao động) mà nghiên cứu này chưa thể đo lường.

5.2.2 Về phạm vi khảo sát mô hình

Nghiên cứu này chủ yếu tập trung vào việc so sánh các phương pháp tiền xử lý dữ liệu và được thực nghiệm trên năm mô hình học máy (Naive Bayes, K Neighbors Nearest, và Logistic Regression, Decision Tree, SVM). Do đó, việc

đánh giá hiệu suất chưa mang tính toàn diện. Kết quả so sánh có thể chưa phản ánh đầy đủ hiệu quả của các thuật toán phức tạp và mạnh mẽ hơn (như các mô hình Ensemble,...) khi áp dụng trên cùng bộ dữ liệu.

5.2.3 Về vấn đề mất cân bằng dữ liệu

Bộ dữ liệu IBM HR Attrition là một bộ dữ liệu mất cân bằng (imbalanced) điển hình, với tỷ lệ nhân viên 'Nghỉ việc' (Yes) thấp hơn đáng kể so với 'Không nghỉ việc' (No). Nghiên cứu này chưa áp dụng các kỹ thuật chuyên biệt để xử lý vấn đề này. Điều này có thể khiến các mô hình dự đoán bị thiên vị (biased), có xu hướng dự đoán tốt cho lớp đa số (Không nghỉ việc) nhưng lại dự đoán kém chính xác ở lớp thiểu số (Nghỉ việc) – vốn là lớp quan trọng nhất cần được nhận diện.

Tài liệu tham khảo

[1] Naive-Bayes-Classifier. https://machinelearningcoban.com/2017/08/08/nbc/

[2] KNN. https://machinelearningcoban.com/2017/01/08/knn/

[3] Logistic Regression. https://machinelearningcoban.com/2017/01/27/logisticregression/

[4] Decision-Tree. https://scikit-learn.org/stable/modules/tree.html

[5] Multi-layer Perceptron. https://machinelearningcoban.com/2017/02/24/mlp/

[6] Support Vector Machine. https://machinelearningcoban.com/2017/04/09/smv/

[7] Soft Margin Support Vector Machine. https://machinelearningcoban.com/2017/04/13/softmarginsmv/

[8] Kernel Support Vector Machine. https://machinelearningcoban.com/2017/04/22/kernelsmv/

[9] K-means. https://machinelearningcoban.com/2017/01/01/kmeans/

[10] DBSCAN. https://phamdinhkhanh.github.io/deepai-book/ch_ml/DBSCAN.html

[11] SMOTE for Imbalanced Classification with Python. https://www.geeksforgeeks.org/machine-learning/smote-for-imbalanced-classification-with-python/

[12] *IBM HR Analytics Employee Attrition & Performance*
`https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset`