



## **ĐỒ ÁN MÔN HỌC**

# **ỨNG DỤNG CÁC MÔ HÌNH HỒI QUY VÀ HỌC MÁY THỐNG KÊ TRONG DỰ BÁO VÀ PHÂN TÍCH DỮ LIỆU KINH TẾ**

Ngành: **KHOA HỌC DỮ LIỆU**

Môn học: **MÁY HỌC THỐNG KÊ**

Giảng viên hướng dẫn : THS. NGUYỄN QUANG PHÚC

Sinh viên thực hiện :

Huỳnh Thái Linh                  MSSV: 2286400015

Nguyễn Nhật Nam                  MSSV: 2286400019

Trương Minh Khoa                  MSSV: 2286400011

Hồ Gia Thành                  MSSV: 2286400029

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2024

## **ĐỒ ÁN MÔN HỌC**

# **ỨNG DỤNG CÁC MÔ HÌNH HỒI QUY VÀ HỌC MÁY THỐNG KÊ TRONG DỰ BÁO VÀ PHÂN TÍCH DỮ LIỆU KINH TẾ**

Ngành: **KHOA HỌC DỮ LIỆU**

Môn học: **MÁY HỌC THỐNG KÊ**

Giảng viên hướng dẫn : THS. NGUYỄN QUANG PHÚC

Sinh viên thực hiện :

Huỳnh Thái Linh              MSSV: 2286400015

Nguyễn Nhật Nam              MSSV: 2286400019

Trương Minh Khoa              MSSV: 2286400011

Hồ Gia Thành              MSSV: 2286400029

Lớp: 22DKHA1

## **LỜI CAM ĐOAN**

Chúng tôi, Huỳnh Thái Linh, Nguyễn Nhật Nam, Trương Minh Khoa và Hồ Gia Thành xin cam đoan rằng:

Mọi thông tin và nghiên cứu được trình bày trong bài báo cáo này là trung thực và khách quan được thu thập và phân tích một cách cẩn thận dựa trên các nguồn chính thống và đáng tin cậy.

Bất kỳ thông tin hoặc ý kiến nào được trích dẫn từ các nguồn khác đều được nêu rõ nguồn gốc và được trích dẫn theo đúng quy định. Chúng tôi xin cam đoan rằng không có bất kỳ sự sao chép hoặc sử dụng thông tin không đúng đắn nào từ các nguồn khác.

Bài báo cáo này là công trình nghiên cứu độc lập của chúng tôi chưa từng được công bố ở bất kỳ nơi nào khác. Tôi cam đoan rằng đã tuân thủ đầy đủ các quy tắc và quy định của môn học bao gồm cả việc tham khảo và sử dụng công cụ nghiên cứu.

Tôi hy vọng rằng bài báo cáo này sẽ cung cấp một cái nhìn tổng quan rõ ràng và toàn diện về chủ đề “Ứng dụng các mô hình hồi quy và học máy thống kê trong dự báo và phân tích dữ liệu kinh tế” và sẽ đóng góp một phần nhỏ vào lĩnh vực nghiên cứu này.

TP.HCM, Ngày.....tháng.....năm 2025

**Sinh viên**

Huỳnh Thái Linh

Nguyễn Nhật Nam

Trương Minh Khoa

Hồ Gia Thành

## NHẬN XÉT CỦA GIÁNG VIÊN HƯỚNG DẪN

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

TP. Hồ Chí Minh, ngày      tháng      năm 2025

**Giáo viên hướng dẫn**

(Ký tên, đóng dấu)

## MỤC LỤC

<b>LỜI CAM ĐOAN .....</b>	i
<b>NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN .....</b>	ii
<b>MỤC LỤC.....</b>	iii
<b>DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT VÀ TỪ KHÓA .....</b>	ix
<b>DANH MỤC HÌNH VẼ.....</b>	x
<b>CHƯƠNG 1. TỔNG QUAN .....</b>	1
<b>1.1. Giới thiệu đê tài .....</b>	1
<b>1.2. Nhiệm vụ của đồ án .....</b>	1
1.2.1. <i>Tính cáp thiết của đê tài .....</i>	1
2.2.1. <i>Ý nghĩa khoa học và thực tiễn của đê tài.....</i>	2
<b>1.3. Mục tiêu .....</b>	2
1.3.1. <i>Mục tiêu tổng quan.....</i>	2
1.3.1. <i>Mục tiêu cụ thể .....</i>	2
<b>1.4. Đối tượng và phạm vi .....</b>	3
1.4.1. <i>Đối tượng .....</i>	3
1.4.2. <i>Phạm vi .....</i>	3
<b>1.5. Phương pháp nghiên cứu .....</b>	3
1.5.1. <i>Phương pháp nghiên cứu sơ bộ .....</i>	3
1.5.2. <i>Phương pháp nghiên cứu tài liệu .....</i>	3
1.5.3. <i>Phương pháp nghiên cứu thống kê.....</i>	4
1.5.4. <i>Phương pháp thực nghiệm .....</i>	4
1.5.5. <i>Phương pháp đánh giá .....</i>	4
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT .....</b>	5

<b>2.1. Hồi quy tuyến tính.....</b>	<b>5</b>
2.1.1. Khái niệm về hồi quy tuyến tính.....	5
2.1.2. Ưu điểm và nhược điểm .....	7
2.1.3. Hàm giả thuyết trong hồi quy tuyến tính .....	7
2.1.4. Các loại Linear Regression. ....	8
2.1.5. Các số liệu đánh giá cho mô hình hồi quy tuyến tính .....	12
<b>2.2. Hồi quy Logistic.....</b>	<b>14</b>
2.2.1. Khái niệm về hồi quy Logistic.....	14
2.2.2. Hàm Sigmoid.....	15
2.2.3. Phân loại hồi quy Logistic .....	17
2.2.4. Giả định của hồi quy Logistic .....	17
2.2.5. Tham số trong hồi quy Logistic.....	19
2.2.6. Cách hoạt động của hồi quy logistic .....	19
2.2.7. Ưu điểm và hạn chế của hồi quy Logistic.....	20
2.2.8. Ứng dụng hồi qua Logistic .....	20
<b>2.3. Decision Tree .....</b>	<b>21</b>
2.3.1. Khái niệm về Decision Tree ( Cây quyết định ) .....	21
2.3.2. Các thuật toán .....	22
2.3.3. Phân loại cây quyết định .....	23
2.3.4. Cách hoạt động của cây quyết định .....	25
2.3.5. Đánh giá mô hình .....	26
2.3.6. Ưu điểm và nhược điểm .....	27
2.3.7. Ứng dụng của cây quyết định .....	27
<b>2.4. Random Forest .....</b>	<b>27</b>

2.4.1.	<i>Khái niệm về Random Forest</i> .....	28
2.4.2.	<i>Đặc điểm của Random Forest</i> .....	28
2.4.3.	<i>Nguyên lý hoạt động của Random Forest</i> .....	29
2.4.4.	<i>Ưu điểm và nhược điểm của Random Forest</i> .....	30
2.4.5.	<i>Ứng dụng của Random Forest</i> .....	31
2.4.6.	<i>So sánh Random Forest và Decision Tree</i> .....	31
<b>2.5.</b>	<b>Support Vector Machine (SVM)</b> .....	<b>32</b>
2.5.1.	<i>Khái niệm về SVM</i> .....	32
2.5.2.	<i>Ứng dụng của SVM</i> .....	33
2.5.3.	<i>Thuật ngữ trong SVM</i> .....	34
2.5.4.	<i>Nguyên lý hoạt động của SVM</i> .....	37
2.5.5.	<i>Dánh giá hiệu suất cho SVM</i> .....	39
2.5.6.	<i>Ưu điểm và nhược điểm của SVM</i> .....	40
<b>2.6.</b>	<b>K – Nearest Neighbors(K-NN)</b> .....	<b>40</b>
2.6.1	<i>Khái niệm về K – Nearest Neighbors (K-NN)</i> .....	40
2.6.2	<i>Nguyên lý hoạt động của K – NN</i> .....	41
2.6.1	<i>Cách chọn k trong K – NN</i> .....	42
2.6.4	<i>Tính khoảng cách trong K - NN</i> .....	43
2.6.5	<i>Ưu điểm và nhược điểm của K –NN</i> .....	44
2.6.6	<i>Ứng dụng của K –NN trong thực tế</i> .....	44
<b>2.7.</b>	<b>Thuật toán phân cụm K – Means</b> .....	<b>44</b>
2.7.1	<i>Khái niệm về K – Means</i> .....	44
2.7.2	<i>Cách hoạt động của K –Means</i> .....	45
2.7.3	<i>Lựa chọn K cho K – Means</i> .....	48

2.7.1	<i>Ưu điểm và nhược điểm của K – Means.....</i>	51
2.7.5	<i>Ứng dụng của K – Means trong thực tế.....</i>	51
<b>2.8.</b>	<b>Thuật toán phân cụm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) .....</b>	<b>51</b>
2.8.1	<i>Khái niệm về DBSCAN .....</i>	52
2.8.3	<i>Cách xác định tham số cho DBSCAN.....</i>	53
2.8.3	<i>Nguyên lý hoạt động của DBSCAN .....</i>	54
2.8.4	<i>Ưu điểm và nhược điểm của DBSCAN .....</i>	57
2.8.5	<i>So sánh DBSCAN &amp; K – Means .....</i>	57
2.8.6	<i>Ứng dụng trong thực tế của DBSCAN.....</i>	58
<b>2.9.</b>	<b>Neural Network (Mạng nơ-ron).....</b>	<b>59</b>
2.9.1.	<i>Khái niệm về mạng nơ-ron .....</i>	59
2.9.2.	<i>Quá trình hoạt động của mạng nơ-ron.....</i>	60
2.9.3.	<i>Ứng dụng của mạng nơ-ron.....</i>	61
<b>2.10.</b>	<b>ARIMA (Autoregressive Intergrated Moving Average) .....</b>	<b>62</b>
2.10.1.	<i>Khái niệm về ARIMA .....</i>	62
2.10.2.	<i>Các bước xây dựng mô hình .....</i>	64
2.10.3.	<i>Ưu điểm và nhược điểm .....</i>	65
2.10.4.	<i>Ứng dụng của ARIMA .....</i>	65
<b>CHƯƠNG 3.</b>	<b>KẾT QUẢ THỰC NGHIỆM .....</b>	<b>66</b>
<b>3.1.</b>	<b>Dataset 1: Diamond Price Prediction .....</b>	<b>66</b>
3.1.1.	<i>Giới thiệu về bộ dữ liệu .....</i>	66
3.1.2.	<i>Tiến xử lý dữ liệu .....</i>	67
3.1.3.	<i>Mô hình hóa và dự đoán giá kim cương.....</i>	76

3.1.4.	<i>Tổng kết</i>	82
<b>3.2.</b>	<b>Dataset 2: Fake Bills</b>	<b>84</b>
3.2.1.	<i>Giới thiệu về bộ dữ liệu</i>	84
3.2.2.	<i>Tiền xử lý dữ liệu</i>	85
3.2.3.	<i>Bài toán 1: Phân loại bằng K – Nearest Neighbors (K – NN)</i>	92
3.2.4.	<i>Bài toán 2: Phát hiện tiền giả bằng Hồi Quy Logistic</i>	94
3.2.5.	<i>Tổng kết mô hình</i>	97
<b>3.3.</b>	<b>Dataset 3: Credit Score Classification – Phân loại điểm tín dụng</b>	<b>98</b>
3.3.1.	<i>Giới thiệu về bộ dữ liệu</i>	98
3.3.2.	<i>Tiền xử lý dữ liệu</i>	100
3.3.3.	<i>Bài toán 1: Phân loại điểm tín dụng bằng SVM (Support Vector Machine)</i>	118
3.3.4.	<i>Bài toán 2: K – Nearest Neighbors (K - NN)</i>	122
3.3.5.	<i>Tổng kết mô hình</i>	124
<b>3.4.</b>	<b>Dataset 4: Phân loại cam tốt và cam xấu bằng m ağ nơ – ron nhân tạo</b>	<b>125</b>
3.4.1.	<i>Giới thiệu bài toán và dữ liệu</i>	125
3.4.2.	<i>Tiền xử lý ảnh</i>	126
3.4.3.	<i>Xây dựng mô hình CNN (Convolutional Neural Network)</i>	129
3.4.4.	<i>Dự đoán và xuất kết quả</i>	132
3.4.5.	<i>Kết luận</i>	137
<b>3.5.</b>	<b>Dataset 5: Bank Customer Segmentation - Phân cụm khách hàng ngân hàng</b>	<b>138</b>
3.5.1.	<i>Giới thiệu về bộ dữ liệu</i>	138

3.5.2.	<i>Tiền xử lý dữ liệu</i> .....	140
3.5.3.	<i>Bài toán 1: Phân cụm bằng K – Means</i> .....	164
3.5.4.	<i>Phân cụm khách hàng bằng K – Means sau khi giảm chiều bằng PCA</i> 172	
3.5.5.	<i>Bài toán 2: Phân cụm bằng DBSCAN</i> .....	179
3.5.6.	<i>Phân cụm khách hàng PCA sau khi giảm chiều bằng PCA</i> .....	184
3.5.7.	<i>So sánh K – Means và DBSCAN</i> .....	188
<b>3.6.</b>	<b>Dự báo giá cổ phiếu Uber bằng mô hình ARIMA</b> .....	<b>189</b>
3.6.1.	<i>Giới thiệu về bài toán và dữ liệu</i> .....	189
3.6.2.	<i>Tiền xử lý dữ liệu</i> .....	191
3.6.3.	<i>Kiểm tra tính dừng và chọn tham số</i> .....	196
3.6.4.	<i>Huấn luyện và dự đoán ARIMA</i> .....	201
3.6.5.	<i>Đánh giá và so sánh kết quả</i> .....	206
3.6.6.	<i>Kết luận</i> .....	206
<b>CHƯƠNG 4.</b>	<b>KẾT LUẬN VÀ KIẾN NGHỊ</b> .....	<b>208</b>
<b>4.1.</b>	<b>Kết luận</b> .....	<b>208</b>
<b>4.2.</b>	<b>Kiến nghị</b> .....	<b>208</b>
<b>TÀI LIỆU THAM KHẢO</b>	.....	<b>209</b>

## **DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT VÀ TỪ KHÓA**

## **DANH MỤC HÌNH VẼ**

Hình 2.1: Linear Regression .....	5
Hình 2.2: Linearity .....	9
Hình 2.3: Homoscedasticity .....	10
Hình 2.4: Hàm Sigmoid.....	16
Hình 2.5: Cây quyết định (Decsion Tree) .....	21
Hình 2.6: Random Forest.....	28
Hình 2.7: Nguyên lý hoạt động của Random Forest.....	30
Hình 2.8: So sánh giữa Random Forest và Decsion Tree .....	31
Hình 2.9: Mô tả thuật toán SVM.....	32
Hình 2.10: Support Values Machine .....	33
Hình 2.11: Biểu đồ K-NN.....	41
Hình 2.12: Minh họa về K - Means.....	45
Hình 2.13: Ví dụ minh họa về khởi tạo tâm cụm.....	46
Hình 2.14: Ví dụ về gán điểm dữ liệu cho cụm cần nhất .....	46
Hình 2.15: Sau khi đo khoảng cách tâm tới điểm dữ liệu .....	47
Hình 2.16: Sau khi khởi tạo lại tâm cụm .....	47
Hình 2.17: Hoàn thành quá trình phân cụm.....	48
Hình 2.18: Thuật toán DBSCAN .....	52
Hình 2.19: Chọn một cụm xung quanh điểm dữ liệu .....	54
Hình 2.20: Bốn cụm điểm dữ liệu .....	55

Hình 2.21: Xác định bốn điểm lõi màu tím và một điểm ngoại lệ màu vàng .....	55
Hình 2.22: Tất cả các điểm dữ liệu lõi có màu tím .....	56
Hình 2.23: Điểm lõi được chọn ngẫu nhiên màu xanh ngọc .....	56
Hình 2.24: kết thúc phân cụm .....	57
Hình 2.25: So sánh giữa DBSCAN và K – Means .....	58
Hình 2.26: Cấu tạo của mạng nơ-ron .....	60
Hình 2.27: Các bước xây dựng mô hình .....	64
Hình 3.1: Đọc dữ liệu từ Diamonds Prices 2022.csv bằng pandas .....	67
Hình 3.2: Thông tin cấu trúc dữ liệu kim cương .....	67
Hình 3.3: Thống kê mô tả dữ liệu số .....	68
Hình 3.4: Kiểm tra số dòng dữ liệu trùng lặp bằng .....	69
Hình 3.5: Biểu đồ boxplot cho biến carat .....	69
Hình 3.6: Phân phối của biến carat .....	70
Hình 3.7: Loại bỏ dữ liệu trùng lặp .....	71
Hình 3.8: Loại bỏ các dòng có giá trị 0 ở cột x, y, z .....	71
Hình 3.9: Mã hóa biến phân loại bằng LabelEncoder .....	72
Hình 3.10: Ma trận tương quan giữa các biến số .....	72
Hình 3.11: Hàm xử lý outlier bằng phương pháp IQR .....	73
Hình 3.12: Boxplot của biến carat sau xử lý outlier .....	74
Hình 3.13: Phân phối của biến carat sau xử lý outlier .....	75
Hình 3.14: Chỉ số VIF kiểm tra đa cộng tuyến giữa các biến .....	75
Hình 3.15: Chuẩn hóa dữ liệu bằng MinMaxScaler .....	76
Hình 3.16: So sánh hiệu quả Linear Regression với hai tập biến .....	77
Hình 3.17: So sánh hiệu quả Random Forest với hai tập biến .....	79
Hình 3.18: So sánh hiệu quả Decision Tree với hai tập biến .....	81
Hình 3.19: Kết quả của ba mô hình .....	82
Hình 3.20: Minh họa so sánh hiệu suất 3 mô hình dự báo giá kim cương .....	83
Hình 3.21: Đọc bộ dữ liệu fake_bills bằng pandas .....	85
Hình 3.22: Thông tin tổng quan về DataFrame sau khi đọc dữ liệu .....	85

Hình 3.23: Chuyển đổi biến is_genuine sang định dạng nhị phân .....	85
Hình 3.24: Điện giá trị khuyết cho biến margin_low bằng giá trị mode (giá trị xuất hiện nhiều nhất).....	86
Hình 3.25: Phân phối dữ liệu biến mục tiêu is_genuine trước khi xử lý mất cân bằng .....	86
Hình 3.26: Boxplot biểu diễn phân bố giá trị cột margin_low .....	87
Hình 3.27: Biểu đồ phân bố biến margin_low.....	88
Hình 3.28: Phân phối thuộc tính length.....	88
Hình 3.29: Ma trận tương quan giữa các biến trong tập dữ liệu Fake Bills .....	89
Hình 3.30: Chuẩn hóa dữ liệu với MinMaxScaler.....	90
Hình 3.31: Cân bằng dữ liệu bằng RandomOverSampler.....	91
Hình 3.32: Phân bố lớp sau khi áp dụng Random OverSampling .....	91
Hình 3.33: Chia dữ liệu thành tập huấn luyện và kiểm tra sau khi cân bằng lớp .....	92
Hình 3.34: Khởi tạo mô hình .....	92
Hình 3.35: Huấn luyện mô hình với best_k.....	93
Hình 3.36: Đánh giá mô hình.....	93
Hình 3.37: classification_report K – NN.....	93
Hình 3.38: Confusion_Matrix K - NN .....	94
Hình 3.39: Huấn luyện mô hình.....	95
Hình 3.40: classification_report Logistic Regression .....	95
Hình 3.41: Confusion_Matrix Logistic Regression .....	96
Hình 3.42: Đọc dữ liệu từ tập huấn luyện và kiểm tra bằng pandas .....	100
Hình 3.43: Phân bổ nhãn Credit_Score .....	100
Hình 3.44: Kích thước dữ liệu sau khi gộp.....	101
Hình 3.45: Tần suất giá trị trong biến Type_of_Loan (gồm cả NaN) .....	102
Hình 3.46: Hàm text_cleaning() chuẩn hóa văn bản, xoá ký tự đặc biệt và khoảng trắng thừa .....	102
Hình 3.47: Kết quả dữ liệu sau khi làm sạch – các ký tự rác và chuỗi không hợp lệ đã được thay thế bằng NaN .....	103

Hình 3.48: Các cột dạng object cần được chuyển đổi kiểu dữ liệu phù hợp.....	104
Hình 3.49: Thông tin dataset sau khi chuyển đổi kiểu dữ liệu phù hợp .....	105
Hình 3.50: Phân phôi giá trị trong cột Name (bao gồm cả NaN) .....	106
Hình 3.51: Danh sách các biến dạng object trong tập dữ liệu .....	106
Hình 3.52: Các giá trị duy nhất của biến mục tiêu Credit_Score .....	106
Hình 3.53: Hàm xử lý outlier theo phương pháp IQR – thay thế các giá trị bất thường bằng giá trị trung bình. ....	107
Hình 3.54: Boxplot biến Outstanding_Debt trước xử lý ngoại lệ.....	107
Hình 3.55: Thống kê số lượng giá trị thiêu sau khi xử lý các biến số .....	108
Hình 3.56: Kết quả sau khi điền giá trị thiêu cho các biến dạng object.....	109
Hình 3.57: Xử lý các giá trị âm không hợp lệ trong cột Num_Bank_Accounts bằng cách thay thế bằng 0. ....	109
Hình 3.58: Xử lý các giá trị không hợp lý trong cột Delay_from_due_date bằng cách thay thế các giá trị âm thành NaN .....	110
Hình 3.59: ử lý ngoại lệ cột Amount_invested_monthly và biểu đồ hộp sau khi loại bỏ các giá trị bất thường. ....	110
Hình 3.60: Hoàn tất xử lý giá trị thiêu ở các biến số, còn lại chỉ thiêu ở các cột mục tiêu (Name, Credit_Score). .....	111
Hình 3.61: Thông tin sau khi lọc dữ liệu có Credit_Score không bị thiêu (train set) .....	112
Hình 3.62: Lưu tập dữ liệu huấn luyện gồm 100.000 dòng có nhãn Credit_Score vào file clean_train.csv.....	112
Hình 3.63: Thông tin tập kiểm tra gồm 50.000 dòng không có nhãn Credit_Score, được sử dụng để dự đoán sau huấn luyện.....	113
Hình 3.64: Tách dữ liệu test từ các bản ghi không có nhãn Credit_Score và lưu thành file clean_test.csv để phục vụ dự đoán sau này. ....	113
Hình 3.65: Khởi tạo đối tượng GetDummies với các tham số cấu hình định dạng phân tách dữ liệu phân loại nhiều nhãn. ....	114

Hình 3.66: Hàm fit() xác định các cột nhiều nhãn và chuẩn bị prefix cho biến giả.....	114
Hình 3.67: Hàm transform() chuyển đổi dữ liệu đầu vào thành các biến giả theo định dạng đã học từ hàm fit().....	115
Hình 3.68: Hàm get_feature_names_out() trả về danh sách tên các đặc trưng sau khi đã biến đổi (dummies) .....	115
Hình 3.69: Tỷ lệ phân phối các nhãn trong biến mục tiêu Credit_Score .....	115
Hình 3.70: Áp dụng transformer GetDummies để tạo biến giả từ các biến phân loại có chứa nhiều giá trị .....	116
Hình 3.71:Các biến phân loại còn lại chưa được xử lý sau khi áp dụng GetDummies .....	116
Hình 3.72:Giá trị thiếu sau khi biến đổi dữ liệu với GetDummies .....	116
Hình 3.73: Ma trận tương quan giữa các biến đầu vào và biến mục tiêu Credit_Score .....	117
Hình 3.74: Tạo tập dữ liệu chính thức bằng cách loại bỏ các đặc trưng không cần thiết .....	117
Hình 3.75: Kết quả chia dữ liệu huấn luyện (Train), kiểm định (Validation) và tập kiểm tra (Test). ....	118
Hình 3.76: Chuẩn hóa dữ liệu X_train và X_val bằng MinMaxScaler về khoảng [0, 1].....	118
Hình 3.77: Khởi tạo model .....	119
Hình 3.78: Tối ưu hóa tham số .....	119
Hình 3.79: Huấn luyện mô hình tối ưu.....	119
Hình 3.80: Đánh giá mô hình.....	120
Hình 3.81: Confusion Matrix mô hình SVM.....	121
Hình 3.82: Khởi tạo và huấn luyện mô hình .....	122
Hình 3.83: Classification Report.....	122
Hình 3.84: Confusion Matrix .....	123

Hình 3.85: Hàm tạo tập huấn luyện: đọc ảnh, resize và chuẩn hóa dữ liệu đầu vào .....	126
Hình 3.86: Hàm xử lý tập test.....	127
Hình 3.87: Tách dữ liệu hình ảnh và nhãn tương ứng từ tập huấn luyện và tập kiểm tra.....	127
Hình 3.88: Chuyển đổi dữ liệu ảnh và nhãn về định dạng mảng NumPy .....	128
Hình 3.89: Tách dữ liệu huấn luyện và validation theo tỷ lệ lớp cân bằng.....	129
Hình 3.90: Cấu trúc mô hình CNN dùng để phân loại cam tốt và cam xấu.....	129
Hình 3.91: Huấn luyện CNN .....	130
Hình 3.92: Log quá trình huấn luyện mô hình CNN theo từng epoch.....	131
Hình 3.93: Dự đoán nhãn cam tốt hoặc cam xấu trên tập test bằng mô hình CNN. ....	132
Hình 3.94:Bảng kết quả dự đoán của mô hình CNN trên tập ảnh test.....	133
Hình 3.95: Độ chính xác của mô hình CNN trên tập kiểm tra .....	133
Hình 3.96: Ma trận nhầm lẫn của mô hình CNN trên tập kiểm tra .....	134
Hình 3.97: Tên các ảnh đại diện cho hai lớp trong tập kiểm tra.....	135
Hình 3.98: Tạo file submission từ dự đoán mô hình trên tập test.....	136
Hình 3.99: File submission .....	136
Hình 3.100: Biểu đồ so sánh accuracy và val_accuracy qua các epoch trong quá trình huấn luyện mô hình CNN. ....	137
Hình 3.101: Đọc dữ liệu bank_transacrions bằng pandas .....	140
Hình 3.102: Thống kê mô tả ba biến định lượng chính .....	140
Hình 3.103: Thống kê mô tả các biến định danh và phân loại trong bộ dữ liệu giao dịch ngân hàng .....	141
Hình 3.104:Kiểm tra dữ liệu trùng lặp .....	142
Hình 3.105: Kiểm tra số lượng giá trị thiếu theo từng cột .....	143
Hình 3.106: Biểu đồ Boxplot của biến CustAccountBalance .....	143
Hình 3.107: Biểu đồ Boxplot của biến TransactionTime\.....	144
Hình 3.108: Biểu đồ Boxplot của biến TransactionAmount (INR) .....	145

Hình 3.109: Phân tích biến CustGende .....	145
Hình 3.110: Chuyển đổi định dạng ngày tháng .....	146
Hình 3.111: Hàm tính tuổi khách hàng và phân bố giá trị Age .....	146
Hình 3.112: Phân bố độ tuổi của khách hàng sau khi tính từ ngày sinh .....	146
Hình 3.113: Biểu đồ Boxplot của biến Age.....	147
Hình 3.114: Loại bỏ khách hàng có tuổi không hợp lệ và xóa cột ngày sinh .....	147
Hình 3.115: Dữ liệu sau tiền xử lý .....	148
Hình 3.116: Kết quả tính chỉ số Monetary cho từng khách hàng .....	149
Hình 3.117: Số lần giao dịch (Frequency) của từng khách hàng.....	149
Hình 3.118:Số ngày kể từ giao dịch gần nhất (Recency) của từng khách hàng....	150
Hình 3.119: Bảng RFM hoàn chỉnh của một số khách hàng.....	150
Hình 3.120: Dữ liệu đã hoàn tất sau khi gộp RFM .....	151
Hình 3.121: Cấu trúc dữ liệu sau khi xử lý và tính toán chỉ số RFM .....	151
Hình 3.122: Thống kê mô tả các biến số trong dữ liệu .....	152
Hình 3.123: Boxplot các biến tài chính .....	153
Hình 3.124: Phân phối tuổi, thời gian giao dịch gần nhất và tần suất giao dịch của khách hàng .....	153
Hình 3.125: Chi tiêu trung bình theo địa điểm khách hàng .....	154
Hình 3.126: Số dư tài khoản trung bình theo địa điểm khách hàng .....	155
Hình 3.127: Tỷ lệ giới tính của khách hàng .....	155
Hình 3.128: So sánh hành vi tài chính trung bình theo giới tính.....	156
Hình 3.129: So sánh chi tiêu trung bình theo giới tính trong các ngày trong tuần.	156
Hình 3.130: So sánh theo tháng về chi tiêu và số dư giữa khách hàng nam và nữ	157
Hình 3.131: Phân bố độ tuổi theo giới tính .....	158
Hình 3.132: Mối quan hệ giữa số dư tài khoản và chi tiêu theo giới tính.....	158
Hình 3.133: Thông tin tổng quan bộ dữ liệu sau khi loại bỏ các cột không cần thiết.	
	159
Hình 3.134: . Giá trị phân vị 50% và 95% của các biến quan trọng.....	160
Hình 3.135: Hàm loại bỏ ngoại lệ bằng phương pháp IQR.....	160

Hình 3.136: Biểu đồ heatmap thể hiện hệ số tương quan giữa các biến số trong tập dữ liệu sau khi xử lý. ....	161
Hình 3.137: Số lượng giá trị bị thiếu sau khi loại bỏ outlier bằng phương pháp IQR. ....	161
Hình 3.138: Xử lý giá trị thiếu và tạo biến cờ is_new_customer .....	162
Hình 3.139: Kết quả sau xử lý .....	162
Hình 3.140: Ma trận tương quan giữa các biến đầu vào sau xử lý .....	163
Hình 3.141: Kết quả sau khi chuẩn hóa dữ liệu bằng StandardScaler .....	163
Hình 3.142: Biểu đồ Elbow và Silhouette Score theo số cụm (k) .....	164
Hình 3.143: Chỉ số Silhouette trung bình .....	165
Hình 3.144: Phân bố số lượng khách hàng theo cụm K-Means (k = 5).....	165
Hình 3.145: Dendrogram từ Hierarchical Clustering.....	166
Hình 3.146: Biểu đồ pairplot sau khi phân cụm KMeans .....	167
Hình 3.147: Biểu đồ radar mô tả cụm 0 theo 3 chỉ số RFM.....	168
Hình 3.148: Biểu đồ radar mô tả cụm 4 theo 3 chỉ số RFM .....	168
Hình 3.149: Biểu đồ radar mô tả cụm 3 theo 3 chỉ số RFM .....	169
Hình 3.150: Biểu đồ radar mô tả cụm 2 theo 3 chỉ số RFM .....	170
Hình 3.151: Biểu đồ radar mô tả cụm 1 theo 3 chỉ số RFM.....	170
Hình 3.152: Biểu đồ phân bố chỉ số Silhouette cho các điểm dữ liệu sau phân cụm bằng Kmeans .....	171
Hình 3.153: Đoạn mã thực hiện .....	172
Hình 3.154: Biểu đồ phương sai tích lũy theo số thành phần PCA .....	173
Hình 3.155: Biểu đồ Elbow và Silhouette Score sau PCA.....	174
Hình 3.156: Silhouette Score sau khi PCA.....	174
Hình 3.157: Dendrogram sau khi giảm chiều bằng PCA .....	175
Hình 3.158: Phân bố số lượng khách hàng trong từng cụm sau phân cụm bằng KMeans trên dữ liệu PCA.....	175
Hình 3.159: Biểu đồ 3D scatter các cụm khách hàng theo PCA. ....	176

Hình 3.160: Biểu đồ violin thể hiện phân bố các giá trị RFM theo từng cụm sau PCA-Kmeans .....	177
Hình 3.161: biểu đồ Silhouette (sau PCA + KMeans) .....	179
Hình 3.162: Đoạn mã thực hiện .....	180
Hình 3.163: Biểu đồ K-Distance trên dữ liệu gốc .....	180
Hình 3.164: Khởi tạo và huấn luyện mô hình DBSCAN trên dữ liệu gốc .....	180
Hình 3.165: Kết quả nhãn cụm từ mô hình DBSCAN .....	181
Hình 3.166: Kết quả xác định số cụm từ mô hình DBSCAN.....	181
Hình 3.167: Kết quả tính Silhouette Score sau khi phân cụm bằng DBSCAN.....	181
Hình 3.168: Phân bố số lượng điểm theo cụm trong mô hình DBSCAN .....	182
Hình 3.169: Dữ liệu sau PCA .....	184
Hình 3.170: Xác định eps .....	185
Hình 3.171: Biểu đồ K-Distance sau PCA .....	185
Hình 3.172: Cấu hình DBSCAN sau khi PCA .....	185
Hình 3.173:Nhãn cụm sau khi phân cụm DBSCAN trên dữ liệu PCA .....	186
Hình 3.174: Nhãn cụm được tạo bởi DBSCAN sau PCA.....	186
Hình 3.175: Silhouette Score sau phân cụm DBSCAN + PCA .....	186
Hình 3.176: Silhouette sau phân cụm bằng DBSCAN (PCA) .....	187
Hình 3.177: Đọc tập dữ liệu uber_stock_data bằng pandas .....	191
Hình 3.178: Thông tin tổng quan về cấu trúc dữ liệu Uber Stock.....	192
Hình 3.179: Thống kê mô tả dữ liệu giá cổ phiếu Uber.....	192
Hình 3.180: Diễn biến giá đóng cửa cổ phiếu Uber trong giai đoạn 2019–2025 ..	193
Hình 3.181: Giá trị trung bình theo năm của các chỉ số cổ phiếu Uber (2019–2025) .....	193
Hình 3.182: Biểu đồ giá đóng cửa điều chỉnh trung bình theo năm .....	194
Hình 3.183: So sánh giá điều chỉnh cổ phiếu Uber theo từng năm (dựa trên ngày trong năm).....	194
Hình 3.184: Log-transform và phân chia tập train/test .....	195
Hình 3.185: Phân chia tập huấn luyện và kiểm tra sau log-transform.....	195

Hình 3.186: Kiểm tra tính dừng bằng trung bình và độ lệch chuẩn trượt .....	196
Hình 3.187: Phân rã chuỗi thời gian log giá đóng cửa cổ phiếu Uber .....	196
Hình 3.188: Biểu đồ tự tương quan (ACF) của chuỗi log giá đóng cửa .....	197
Hình 3.189: Biểu đồ độ trễ (Lag plot) của chuỗi log giá đóng cửa .....	198
Hình 3.190: Biểu đồ tự tương quan riêng phần (PACF) của chuỗi log giá đóng cửa .....	198
Hình 3.191: Biểu đồ chuỗi gốc và chuỗi sai phân bậc 1 .....	199
Hình 3.192: Kết quả kiểm định ADF, KPSS và biểu đồ PACF sau khi sai phân bậc 1 .....	200
Hình 3.193: Tự động xác định tham số mô hình ARIMA bằng auto_arima() .....	200
Hình 3.194: Bảng kết quả huấn luyện mô hình ARIMA(2,1,2) .....	201
Hình 3.195: Biểu đồ chẩn đoán mô hình ARIMA – Kiểm tra phần dư .....	202
Hình 3.196: Kết quả huấn luyện mô hình ARIMA(3,1,2) có thành phần xu thế tuyến tính .....	203
Hình 3.197: Biểu đồ dự báo giá cổ phiếu bằng mô hình ARIMA(2,1,2) .....	204
Hình 3.198: Dự báo và đánh giá sai số của mô hình ARIMA(2,1,2) .....	205
Hình 3.199: Dự báo baseline sử dụng trung bình tập huấn luyện .....	205
Hình 3.200: So sánh sai số RMSE giữa mô hình ARIMA và baseline .....	206

# CHƯƠNG 1. TỔNG QUAN

## 1.1. Giới thiệu đề tài

Trong thời đại công nghệ 4.0 phát triển, việc áp dụng các mô hình học máy và hồi quy thống kê để khai thác, phân tích và dự báo đã trở thành một xu hướng tất yếu. Các phương pháp truyền thống không còn đáp ứng được yêu cầu xử lý khối lượng lớn dữ liệu và tính chất phi tuyến ngày càng rõ rệt trong các hiện tượng kinh tế. Do đó, việc nghiên cứu và triển khai các mô hình học máy như hồi quy tuyến tính, hồi quy logistic, cây quyết định, rừng ngẫu nhiên, SVM, KNN, mạng nơ-ron, K-Means, DBSCAN và ARIMA là cần thiết nhằm nâng cao hiệu quả phân tích và dự báo kinh tế.

## 1.2. Nhiệm vụ của đồ án

- Khảo sát lý thuyết các mô hình học máy thống kê và hồi quy phổ biến.
- Tiến hành xử lý, chuẩn hóa và phân tích dữ liệu kinh tế thực tế từ các nguồn mở.
- Áp dụng từng mô hình cho các bài toán cụ thể như: dự đoán giá, phân loại rủi ro, phát hiện gian lận, phân cụm khách hàng, dự báo chuỗi thời gian.
- So sánh hiệu quả các mô hình theo các tiêu chí đánh giá phù hợp để rút ra kết luận thực tiễn.

### 1.2.1. Tính cấp thiết của đề tài

Sự phát triển của công nghệ dữ liệu lớn (Big Data) đặt ra yêu cầu cấp thiết về việc khai thác tri thức ẩn trong dữ liệu kinh tế – tài chính. Các tổ chức tài chính, doanh nghiệp hay cơ quan quản lý đều có nhu cầu cao trong việc dự báo xu hướng, phân tích hành vi và tối ưu hóa quyết định dựa trên dữ liệu. Việc ứng dụng các mô hình

học máy thống kê giúp tăng độ chính xác, giảm thời gian xử lý và hỗ trợ ra quyết định hiệu quả hơn.

### 2.2.1. Ý nghĩa khoa học và thực tiễn của đề tài

**Ý nghĩa khoa học:** Đề tài góp phần hệ thống hóa và làm rõ vai trò của các mô hình hồi quy và kỹ thuật học máy thống kê trong lĩnh vực phân tích dữ liệu kinh tế. Thông qua việc áp dụng các thuật toán hiện đại vào bài toán thực tiễn, đề tài tạo nền tảng để mở rộng các hướng nghiên cứu chuyên sâu, đặc biệt trong bối cảnh khoa học dữ liệu đang ngày càng gắn bó chặt chẽ với kinh tế học và quản trị.

**Ý nghĩa thực tiễn:** việc triển khai các mô hình phân tích giúp nâng cao hiệu quả dự báo, hỗ trợ các tổ chức, doanh nghiệp và nhà hoạch định chính sách trong việc đưa ra quyết định dựa trên dữ liệu có cơ sở, thay vì cảm tính hay kinh nghiệm chủ quan. Đồng thời, kết quả từ đề tài có thể được ứng dụng linh hoạt trong nhiều lĩnh vực như tài chính, thương mại, lao động hay điều hành vĩ mô, góp phần nâng cao chất lượng quản lý và năng lực cạnh tranh trong môi trường kinh tế số.

## 1.3. Mục tiêu

### 1.3.1. Mục tiêu tổng quan

Xây dựng một báo cáo tổng hợp và ứng dụng thực nghiệm các mô hình hồi quy và học máy thống kê vào bài toán dự báo và phân tích dữ liệu kinh tế.

### 1.3.1. Mục tiêu cụ thể

- Tìm hiểu, trình bày lý thuyết về các mô hình: Linear Regression, Logistic Regression, Decision Tree, Random Forest, SVM, KNN, Neural Network, K-Means, DBSCAN, ARIMA.
- Thực nghiệm các mô hình trên các bộ dữ liệu kinh tế thực tế từ Kaggle.
- Đánh giá, so sánh hiệu quả các mô hình và rút ra bài học ứng dụng

## **1.4. Đối tượng và phạm vi**

### *1.4.1. Đối tượng*

Đối tượng Các mô hình học máy thống kê và hồi quy được áp dụng trong các bài toán phân tích, dự báo kinh tế dựa trên dữ liệu thực.

### *1.4.2. Phạm vi*

Dữ liệu được khai thác từ nền tảng Kaggle, chủ yếu trong các lĩnh vực: tài chính, thương mại, thị trường chứng khoán và hành vi người tiêu dùng.

## **1.5. Phương pháp nghiên cứu**

### *1.5.1. Phương pháp nghiên cứu sơ bộ*

Phương pháp này được sử dụng nhằm khảo sát tổng quan các tài liệu, công trình nghiên cứu liên quan đến ứng dụng hồi quy và học máy trong phân tích dữ liệu kinh tế. Việc này giúp định hình hướng tiếp cận, lựa chọn mô hình phù hợp và xây dựng cấu trúc nghiên cứu có hệ thống.

### *1.5.2. Phương pháp nghiên cứu tài liệu*

Để tài vận dụng phương pháp nghiên cứu tài liệu để thu thập, tổng hợp và phân tích các nguồn học thuật đáng tin cậy như sách chuyên ngành, bài báo khoa học, các báo cáo nghiên cứu và tài nguyên trực tuyến. Đây là cơ sở quan trọng để hình thành nền tảng lý thuyết vững chắc cho việc xây dựng và đánh giá mô hình.

### *1.5.3. Phương pháp nghiên cứu thống kê*

Các kỹ thuật thống kê được sử dụng để mô tả dữ liệu, phân tích mối quan hệ giữa các biến và hỗ trợ trong việc khám phá đặc điểm của tập dữ liệu. Ngoài ra, thống kê cũng đóng vai trò trong việc đánh giá kết quả mô hình thông qua các chỉ số định lượng cụ thể.

### *1.5.4. Phương pháp thực nghiệm*

Để tài triển khai mô hình hồi quy và học máy trên các bộ dữ liệu kinh tế thực tế từ Kaggle. Quá trình này bao gồm các bước như tiền xử lý dữ liệu, xây dựng mô hình, huấn luyện, kiểm thử và tinh chỉnh tham số để tối ưu hiệu quả dự báo.

### *1.5.5. Phương pháp đánh giá*

Hiệu suất của các mô hình được đánh giá thông qua các chỉ số định lượng như hệ số xác định ( $R^2$ ), sai số bình phương trung bình (RMSE), sai số tuyệt đối trung bình (MAE) và độ chính xác (Accuracy) tùy theo bài toán. Việc so sánh các mô hình dựa trên những chỉ số này giúp lựa chọn phương pháp tối ưu nhất cho từng trường hợp phân tích cụ thể.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

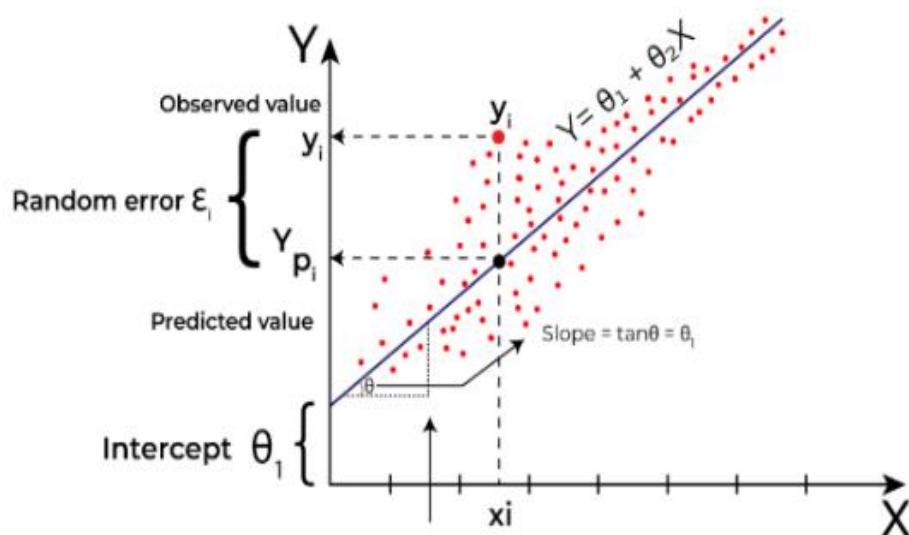
### 2.1. Hồi quy tuyến tính

#### 2.1.1. Khái niệm về hồi quy tuyến tính

Hồi quy tuyến tính là một phương pháp thống kê và thuật toán học máy có giám sát, dùng để mô hình hóa mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập. Mô hình này học từ dữ liệu gắn nhãn và tạo ra một hàm tuyến tính phù hợp nhất để dự đoán giá trị đầu ra liên tục.

Hồi quy tuyến tính được ưa chuộng nhờ tính đơn giản, dễ triển khai và khả năng giải thích cao. Các hệ số trong mô hình giúp xác định mức độ ảnh hưởng của từng biến đầu vào đến kết quả đầu ra, hỗ trợ phân tích và ra quyết định.

Mục tiêu chính của mô hình là tìm ra đường tuyến tính tốt nhất, tức là đường thẳng có sai số dự đoán nhỏ nhất. Độ dốc của đường thể hiện mức độ biến động của Y khi X thay đổi, từ đó giúp mô hình đưa ra dự đoán chính xác hơn.



Hình 2.1: Linear Regression

Trong hồi quy tuyến tính, Y được gọi là biến phụ thuộc hoặc biến mục tiêu, còn X là biến độc lập, hay còn gọi là biến dự báo của Y. Mô hình hồi quy có thể sử dụng nhiều loại hàm khác nhau, trong đó hàm tuyến tính là dạng đơn giản và phổ biến

nhất. Biến X có thể là một đặc trưng duy nhất hoặc tập hợp nhiều đặc trưng đại diện cho bài toán.

Hồi quy tuyến tính là mô hình dự đoán giá trị của Y dựa trên một hoặc nhiều giá trị X. Khi chỉ có một biến X, ta gọi là hồi quy tuyến tính đơn; còn khi có nhiều biến X, đó là hồi quy tuyến tính đa biến. Mục tiêu của mô hình là tìm ra mối quan hệ tuyến tính giữa X và Y, nghĩa là khi X thay đổi thì Y cũng thay đổi theo một quy luật nhất định.

Mô hình này rất hữu ích vì nó không chỉ giúp dự đoán kết quả, mà còn cho ta biết mức độ ảnh hưởng của từng yếu tố đầu vào (X) đến kết quả đầu ra (Y). Nhờ tính đơn giản và dễ áp dụng, hồi quy tuyến tính được sử dụng phổ biến trong các lĩnh vực như kinh tế, tài chính, y học và khoa học dữ liệu để phân tích, dự đoán và hỗ trợ ra quyết định.

Không chỉ đóng vai trò là công cụ phân tích dữ liệu hiệu quả, hồi quy tuyến tính còn là “bước đệm” quan trọng giúp người học và nhà nghiên cứu làm quen với tư duy mô hình hóa trong khoa học dữ liệu. Khả năng diễn giải kết quả một cách trực quan qua các hệ số hồi quy giúp người dùng hiểu rõ mối quan hệ giữa các biến, từ đó tăng tính minh bạch và đáng tin cậy trong quá trình ra quyết định.

Bên cạnh đó, hồi quy tuyến tính còn thể hiện tính linh hoạt cao khi có thể mở rộng và điều chỉnh để phù hợp với các bài toán phức tạp hơn, như thêm yếu tố tương tác giữa các biến, xử lý dữ liệu không tuyến tính thông qua biến đổi đặc trưng, hay kết hợp với các kỹ thuật chính quy hóa nhằm giảm thiểu hiện tượng đa cộng tuyến. Nhờ vào sự cân bằng giữa tính đơn giản và hiệu quả, mô hình này tiếp tục được sử dụng làm chuẩn so sánh cho nhiều thuật toán nâng cao hơn trong lĩnh vực học máy và thống kê.

Trong thực tiễn, hồi quy tuyến tính được ứng dụng rộng rãi trong các lĩnh vực như dự báo kinh tế, phân tích rủi ro tài chính, tối ưu hóa chi phí sản xuất, và thậm chí là phân tích xu hướng hành vi người tiêu dùng. Với những ưu điểm nổi bật cả về lý thuyết lẫn thực hành, hồi quy tuyến tính không chỉ là nền tảng học thuật vững chắc mà còn là công cụ hữu ích trong giải quyết các vấn đề thực tế.[1], [2]

### *2.1.2. Ưu điểm và nhược điểm*

**Ưu điểm:** Mô hình hồi quy tuyến tính có tính đơn giản và dễ áp dụng, phù hợp với người mới bắt đầu trong phân tích dữ liệu. Nhờ cấu trúc dễ hiểu, mô hình có thể được triển khai trên nhiều nền tảng như Excel, Python hoặc R mà không cần kiến thức chuyên sâu. Một điểm mạnh nổi bật là khả năng giải thích rõ ràng thông qua các hệ số, giúp xác định mức độ ảnh hưởng của các biến đầu vào. Mô hình cũng cho phép sử dụng đồng thời nhiều biến độc lập, từ đó đánh giá tổng hợp các yếu tố tác động đến biến mục tiêu. Ngoài ra, nó có thể được kết hợp linh hoạt với các phương pháp như chọn biến, kiểm định thống kê hoặc kỹ thuật chính quy để nâng cao hiệu quả.

**Nhược điểm:** Mô hình này chỉ phù hợp trong trường hợp mối quan hệ giữa biến độc lập và phụ thuộc mang tính tuyến tính. Khi dữ liệu có tính phi tuyến hoặc cấu trúc phức tạp, mô hình dễ đưa ra kết quả sai lệch. Hồi quy tuyến tính cũng giả định rằng các quan sát là độc lập và phần dư tuân theo phân phối chuẩn điều này không phải lúc nào cũng đúng trong thực tế, đặc biệt với dữ liệu chuỗi thời gian. Mô hình còn dễ bị ảnh hưởng bởi giá trị ngoại lệ hoặc sự tương quan cao giữa các biến đầu vào, khiến kết quả thiếu ổn định. Với các bài toán có tính phi tuyến rõ rệt, hiệu quả của hồi quy tuyến tính thường thấp hơn so với các mô hình hiện đại như Random Forest hay mạng nơ-ron sâu. [3]

### *2.1.3. Hàm giả thuyết trong hồi quy tuyến tính*

Khi xây dựng mô hình hồi quy tuyến tính, chúng ta đưa ra một số giả định cơ bản nhằm đảm bảo kết quả mô hình có ý nghĩa và đáng tin cậy:

- Tính tuyến tính: Giả định rằng tồn tại mối quan hệ tuyến tính giữa biến độc lập ( $X$ ) và biến phụ thuộc ( $Y$ ). Nghĩa là, khi  $X$  thay đổi thì  $Y$  cũng thay đổi theo một tỷ lệ tương ứng.
- Tính độc lập: Các quan sát trong dữ liệu phải độc lập với nhau, tức là sai số từ một quan sát không được ảnh hưởng đến sai số của các quan sát khác.

Công thức hồi quy tuyến tính được biểu diễn như sau:

$$\hat{Y} = \theta_1 + \theta_2 X$$

Hoặc với từng quan sát:

$$\hat{y}_i = \theta_1 + \theta_2 x_i$$

Trong đó:

- $y_i$  Giá trị thực tế tương ứng với quan sát thứ  $i$ .
- $x_i$  Giá trị đầu vào (biến độc lập).
- $\theta_1$  Hệ số chặn (intercept).
- $\theta_2$  Hệ số góc (hệ số của biến  $X$ ).

Mô hình có được đường thẳng hồi quy tốt nhất khi tìm được các giá trị  $\theta_1$  và  $\theta_2$  tốt nhất:

- $\theta_0$  là hệ số chặn (intercept).
- $\theta_1$  là hệ số của biến độc lập (coefficient) của  $X$ .

Để tìm được đường hồi quy tốt nhất, mô hình cần điều chỉnh các hệ số  $\theta_1$  và  $\theta_2$  sao cho khoảng cách giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất.

Mục tiêu là tối thiểu hóa sai số bằng hàm mất mát, cụ thể là hàm sai số bình phương trung bình (Mean Squared Error - MSE):

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

Khi sai số này đạt giá trị nhỏ nhất, mô hình sẽ có độ chính xác cao và khả năng dự báo tốt hơn trên dữ liệu mới.

#### 2.1.4. Các loại Linear Regression.

Khi mô hình chỉ bao gồm một biến độc lập, ta gọi đó là hồi quy tuyến tính đơn hoặc hồi quy tuyến tính một biến. Ngược lại, nếu có từ hai biến độc lập trở lên, mô hình được gọi là hồi quy tuyến tính bội hay hồi quy tuyến tính đa biến. Sự phân loại

này giúp xác định mức độ phức tạp và khả năng mô hình hóa của mô hình hồi quy đối với dữ liệu đầu vào.

### Hồi quy tuyến tính đơn biến (Simple Linear Regression):

Phương trình hồi quy tuyến tính đơn biến thể hiện mối quan hệ tuyến tính giữa một biến độc lập X và một biến phụ thuộc Y, được biểu diễn dưới dạng:

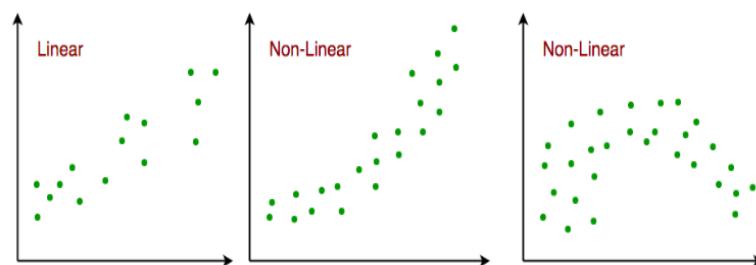
$$Y = \beta_0 + \beta_1 X$$

Với:

- Y là biến phụ thuộc (giá trị cần dự đoán).
- X là biến độc lập (biến đầu vào).
- $\beta_0$  hệ số chặn (Giá trị của Y khi X = 0).
- $\beta_1$  hệ số góc (Thể hiện mức độ thay đổi của Y khi X tăng lên một đơn vị).

Giả định về hồi quy tuyến tính đơn biến:

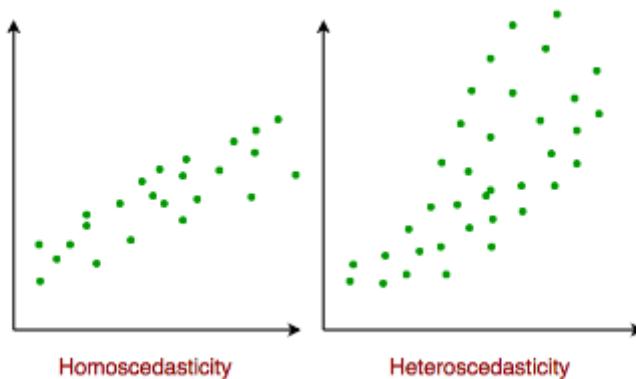
- Tính tuyến tính (Linearity): Mọi quan hệ giữa biến độc lập và biến phụ thuộc phải là tuyến tính. Nghĩa là khi X thay đổi, Y cũng thay đổi theo một tỷ lệ nhất định. Dữ liệu nên có xu hướng nằm gần một đường thẳng.



Hình 2.2: Linearity

- Tính độc lập (Independence): Các quan sát trong tập dữ liệu phải độc lập với nhau. Giá trị của biến phụ thuộc trong một quan sát không được phụ thuộc vào các quan sát khác.

- Tính đồng nhất phương sai (Homoscedasticity): Sai số (residuals) phải có phương sai đồng đều ở mọi giá trị của X. Nếu sai số có phương sai thay đổi theo X, mô hình có thể bị sai lệch.



Hình 2.3: Homoscedasticity

- Phân phối chuẩn của sai số: Các sai số (phản dư) cần được phân phối chuẩn, tức là có hình chuông đối xứng quanh trung bình. Điều này quan trọng để các kiểm định thống kê liên quan đến mô hình được thực hiện chính xác.

### Hồi quy tuyến tính đa biến (Multiple Linear Regression):

Hồi quy tuyến tính bội là phương pháp mở rộng của hồi quy tuyến tính đơn, trong đó mô hình bao gồm **một biến phụ thuộc** và **nhiều biến độc lập**. Phương trình tổng quát được biểu diễn như sau:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Trong đó:

- Y: biến phụ thuộc (kết quả cần dự đoán).
- X<sub>1</sub>, X<sub>2</sub>, ..., X<sub>n</sub>: các biến độc lập (đặc trưng đầu vào).
- β<sub>0</sub>: hệ số chẵn.
- β<sub>1</sub>, β<sub>2</sub>, ..., β<sub>n</sub>: hệ số góc, biểu thị mức độ ảnh hưởng của từng biến độc lập đến Y

Mục tiêu của mô hình là tìm ra phương trình dự đoán tối ưu, cho phép ước lượng giá trị Y dựa trên nhiều biến X đầu vào. Mô hình học từ dữ liệu huấn luyện có chứa các cặp giá trị X và Y, từ đó tạo ra hàm dự đoán có khả năng tổng quát hóa cho dữ liệu mới.

Giả định trong hồi quy đa biến: Ngoài bốn giả định cơ bản giống với hồi quy tuyến tính đơn (tuyến tính, độc lập, đồng phương sai và phân phối chuẩn của sai số), mô hình hồi quy bội còn cần thỏa mãn các điều kiện sau:

- Không có đa cộng tuyến: Các biến độc lập không được có mối tương quan cao với nhau. Nếu xảy ra đa cộng tuyến (multicollinearity), mô hình sẽ khó xác định được ảnh hưởng riêng của từng biến, làm giảm độ tin cậy của kết quả.
- Tính cộng gộp (Additivity): Tác động của từng biến độc lập lên biến phụ thuộc là độc lập với các biến khác. Điều này nghĩa là không có sự tương tác giữa các biến trong ảnh hưởng của chúng đến Y.
- Lựa chọn biến phù hợp (Feature Selection): Cần lựa chọn kỹ các biến đầu vào. Việc đưa vào các biến dư thừa hoặc không liên quan có thể gây ra hiện tượng quá khớp (overfitting), làm mô hình phức tạp và kém hiệu quả khi áp dụng vào dữ liệu mới.
- Tránh overfitting: Khi mô hình học quá sát với dữ liệu huấn luyện, nó có thể “ghi nhớ” nhiều thay vì học được quy luật thực sự, từ đó dự đoán kém trên dữ liệu chưa từng thấy.

### **Đa cộng tuyến trong hồi quy tuyến tính đa biến:**

Đa cộng tuyến xảy ra khi hai hoặc nhiều biến độc lập trong mô hình có mối tương quan cao, khiến việc đánh giá chính xác ảnh hưởng riêng của từng biến trở nên khó khăn.

Cách phát hiện đa cộng tuyến:

- Ma trận tương quan: Hệ số tương quan cao ( $\geq \pm 1$ ) giữa các biến độc lập cho thấy nguy cơ đa cộng tuyến.

- Hệ số phỏng đại phương sai (VIF):  $VIF > 10$  là dấu hiệu rõ rệt cho thấy biến độc lập đang gây ra đa cộng tuyến nghiêm trọng.

Ứng dụng của hồi quy đa biến:

- Dự đoán giá bất động sản: Xác định giá trị nhà ở dựa trên vị trí, diện tích, số phòng,..
- Phân tích tài chính: Dự báo giá cổ phiếu hoặc chỉ số kinh tế dựa trên lãi suất, lạm phát, xu hướng thị trường,...
- Ước tính năng suất nông nghiệp: Dựa trên yếu tố khí hậu, đất đai, phân bón để dự đoán sản lượng.
- Phân tích doanh số thương mại điện tử: Đánh giá tác động của giá bán, chương trình khuyến mãi và xu hướng mùa vụ đến doanh số.

#### 2.1.5. Các số liệu đánh giá cho mô hình hồi quy tuyến tính

##### Mean Squared Error (MSE)

MSE là chỉ số đánh giá độ chính xác của mô hình, tính bằng trung bình bình phương sai số giữa giá trị thực tế và giá trị dự đoán.

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

Trong đó:

- n: Số lượng điểm dữ liệu.
- $y_i$ : Giá trị thực tế.
- $\hat{y}_i$ : giá trị dự đoán.

MSE giúp đo lường mức độ sai lệch của mô hình. Nó nhạy cảm với ngoại lệ, vì sai số lớn sẽ làm tăng đáng kể giá trị MSE.

##### Mean Absolute Error (MAE)

MAE là một chỉ số đánh giá độ chính xác của mô hình hồi quy, bằng trung bình sai số tuyệt đối giữa giá trị dự đoán và giá trị thực tế:

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{Y}_i - Y_i|$$

Trong đó:

- n: Số lượng quan sát.
- $y_i$ : Giá trị thực tế.
- $\hat{y}_i$ : giá trị dự đoán.

MAE cho biết mức sai lệch trung bình của mô hình. Giá trị MAE càng thấp thì mô hình càng chính xác. Khác với MSE, MAE ít bị ảnh hưởng bởi ngoại lệ do sử dụng giá trị tuyệt đối thay vì bình phương sai số.

### **Root Mean Squared Error (RMSE)**

RMSE là một chỉ số đánh giá mức độ sai lệch giữa giá trị dự đoán và giá trị thực tế, được tính bằng căn bậc hai của sai số bình phương trung bình (MSE). Nó thể hiện mức độ "khớp" giữa mô hình và dữ liệu quan sát.

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

- n: Số lượng quan sát.
- $y_i$ : Giá trị thực tế.
- $\hat{y}_i$ : giá trị dự đoán.

RMSE càng nhỏ chứng tỏ mô hình càng chính xác. Tuy nhiên, RMSE phụ thuộc vào đơn vị của biến, nên không phải là chỉ số chuẩn hóa như  $R^2$ .

### **Residual Standard Error (RSE)**

RSE là phiên bản hiệu chỉnh của RMSE, nhằm tránh thiên lệch khi ước lượng sai số từ mẫu. Thay vì chia cho toàn bộ số quan sát n, RSE chia cho bậc tự do của mô hình ( $n-2$ ):

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

RSE được sử dụng phổ biến trong thống kê để đánh giá chất lượng mô hình hồi quy với điều chỉnh phù hợp hơn khi làm việc với dữ liệu mẫu nhỏ.

### Hệ số xác định (R-squared)

R-squared ( $R^2$ ) là chỉ số thống kê thể hiện mức độ mà mô hình giải thích được phương sai của biến phụ thuộc. Giá trị của  $R^2$  nằm trong khoảng từ 0 đến 1, càng gần 1 thì mô hình càng phù hợp với dữ liệu.

Công thức tính:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Trong đó:

- RSS (Residual Sum of Squares): Thể hiện tổng các bình phương sai lệch giữa giá trị thực và giá trị dự báo.

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- TSS (Total Sum of Squares): Tổng bình phương sai số giữa giá trị thực tế và giá trị trung bình.

$$TSS = \sum_{i=1}^n (y_i - \bar{y})^2$$

$R^2$  phản ánh tỷ lệ biến thiên của biến phụ thuộc được giải thích bởi các biến độc lập trong mô hình. Giá trị  $R^2$  cao cho thấy mô hình có khả năng dự đoán tốt hơn.[4]

## 2.2. Hồi quy Logistic

### 2.2.1. Khái niệm về hồi quy Logistic

Hồi quy logistic là một thuật toán học máy có giám sát, được sử dụng phổ biến trong các bài toán phân loại nhị phân. Mục tiêu của thuật toán là dự đoán xác suất

xảy ra của một sự kiện, kết quả hoặc quan sát, từ đó đưa ra quyết định phân loại đầu ra vào một trong ba nhóm: Ví dụ như có/không, 0/1, hoặc đúng/sai

Mô hình phân tích mối quan hệ giữa một hoặc nhiều biến độc lập với một biến phụ thuộc rời rạc và ước lượng xác suất để một quan sát thuộc về một danh mục cụ thể. Kết quả đầu ra là một giá trị xác suất (từ 0 đến 1), sau đó được ngưỡng hóa để đưa ra phân loại cuối cùng.[5]

Những điểm chính:

- Hồi quy logistic được sử dụng để dự đoán đầu ra của một biến phân loại, nghĩa là kết quả đầu ra phải là giá trị rời rạc, chẳng hạn như: Có/Không, Đúng/Sai, hoặc 0/1.
- Mặc dù mục tiêu là phân loại đầu ra thành hai nhóm, mô hình không đưa ra kết quả tuyệt đối là 0 hoặc 1, mà thay vào đó dự đoán một xác suất nằm trong khoảng từ 0 đến 1. Dựa trên xác suất này, một ngưỡng (thường là 0.5) sẽ được sử dụng để quyết định đầu ra cuối cùng.
- Thay vì dùng đường hồi quy tuyến tính, hồi quy logistic sử dụng hàm sigmoid (hàm logistic) có dạng đường cong hình chữ S để mô hình hóa xác suất. Hàm này giúp "nén" giá trị đầu ra về khoảng (0, 1), rất phù hợp với các bài toán phân loại nhị phân.[6]

### 2.2.2. Hàm Sigmoid

Trong hồi quy logistic, đầu ra của mô hình cần biểu diễn xác suất, tức là một giá trị nằm trong khoảng từ 0 đến 1. Để làm được điều này, ta sử dụng một hàm toán học đặc biệt có tên là hàm sigmoid.

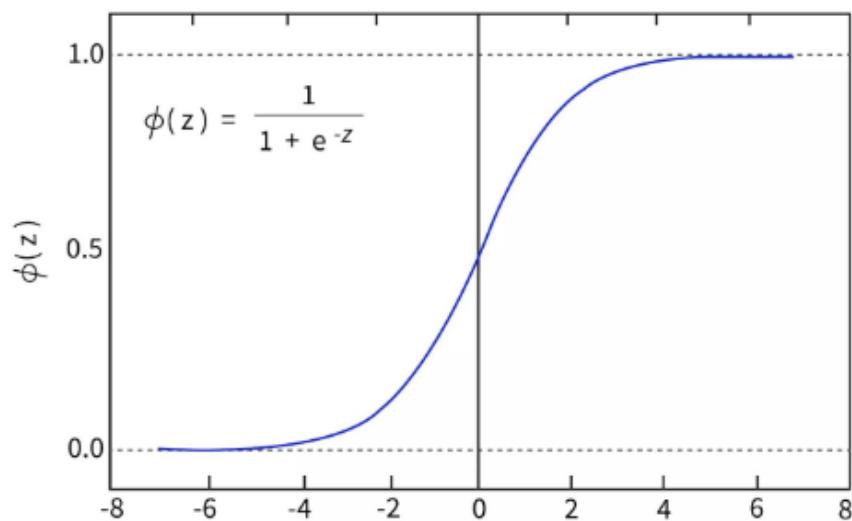
Hàm sigmoid (còn gọi là hàm logistic) là một hàm số có dạng đường cong hình chữ S, giúp "nén" mọi giá trị thực đầu vào (âm hoặc dương) về một giá trị trong khoảng (0, 1). Đây chính là lý do nó được sử dụng để chuyển đổi đầu ra của mô hình thành xác suất. [7]

Hàm sigmoid có công thức như sau:

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

Trong đó  $z$  là tổ hợp tuyến tính của các đặc điểm đầu vào. Hàm này ánh xạ bất kỳ giá trị đầu vào nào vào một phạm vi giữa 0 và 1, sau đó có thể được hiểu là xác suất của biến mục tiêu thuộc về một lớp cụ thể.

Biểu diễn đường cong của sigmoid:



Hình 2.4: Hàm Sigmoid

Ý nghĩa của hàm sigmoid:

- Chuyển đầu ra của mô hình thành xác suất – giá trị luôn nằm trong khoảng (0, 1).
- Giúp ra quyết định phân loại nhị phân:
  - o Xác suất  $\geq 0.5 \rightarrow$  phân loại là lớp 1
  - o Xác suất  $< 0.5 \rightarrow$  phân loại là lớp 0
- Khi:
  - o  $z \rightarrow +\infty \rightarrow$  xác suất gần 1
  - o  $z \rightarrow -\infty \rightarrow$  xác suất gần 0
- Tại  $z = 0$ , hàm sigmoid cho kết quả 0.5  $\rightarrow$  mô hình "cân bằng", không nghiêng về lớp nào.

- Dạng cong chữ S mượt mà, giúp tối ưu dễ dàng khi huấn luyện mô hình bằng gradient descent.[8]

### 2.2.3. Phân loại hồi quy Logistic

Hồi quy logistic được chia thành ba loại chính: hồi quy nhị phân, hồi quy đa thức và hồi quy thứ tự. Mỗi loại được sử dụng tùy thuộc vào số lượng và đặc điểm của biến phụ thuộc:

- Hồi quy logistic nhị phân (Binary Logistic Regression): Loại phổ biến nhất, được sử dụng khi biến phụ thuộc chỉ có hai giá trị rời rạc, chẳng hạn như có/không, thành công/thất bại, hoặc 0/1. Ví dụ:
  - o Dự đoán khách hàng có được duyệt vay ngân hàng hay không  
→ Kết quả: Có / Không
  - o Chẩn đoán nguy cơ mắc ung thư → Kết quả: Cao / Thấp
- Hồi quy đa thức (Multinomial Logistic Regression): Được sử dụng khi biến phụ thuộc có từ ba lựa chọn phân loại trở lên (không có thứ tự). Kết quả là nhiều lớp rời rạc. Ví dụ:
  - o Dự đoán phương tiện giao thông phổ biến nhất vào năm 2040  
→ Kết quả: Ô tô điện / Tàu điện / Xe buýt điện / Xe đạp điện
- Hồi quy logistic thứ tự (Ordinal Logistic Regression): Sử dụng khi biến phụ thuộc có các mức xếp hạng theo thứ tự, ví dụ như đánh giá, mức độ hài lòng, cấp bậc,... Ví dụ:
  - o Phân loại kích cỡ áo → Kết quả: XS / S / M / L / XL [9]

### 2.2.4. Giá định của hồi quy Logistic

#### Quan sát độc lập

Mỗi quan sát trong dữ liệu phải độc lập với nhau, tức là không có mối liên hệ hay phụ thuộc giữa các mẫu.

Điều này đảm bảo rằng mô hình không bị thiên lệch do dữ liệu lặp hoặc có cấu trúc phụ thuộc (ví dụ: dữ liệu chuỗi thời gian, dữ liệu theo nhóm,...).

### **Biến phụ thuộc nhị phân**

Hồi quy logistic cõi điển yêu cầu biến phụ thuộc (output) phải là nhị phân, chỉ nhận hai giá trị (ví dụ: 0 hoặc 1, Có hoặc Không).

Trong trường hợp có nhiều hơn hai lớp, cần mở rộng sang hồi quy logistic đa thức (Multinomial Logistic Regression) sử dụng hàm Softmax.

### **Mối quan hệ tuyến tính giữa biến độc lập và logit**

Mô hình giả định rằng có mối quan hệ tuyến tính giữa các biến độc lập và logit (log odds) của xác suất xảy ra sự kiện.

Đây không phải tuyến tính với đầu ra, mà là tuyến tính với logit:

$$\log \frac{p}{1-p}$$

Với:

- $p$ : là xác suất xảy ra một sự kiện (ví dụ: xác suất khách hàng mua hàng).
- $\frac{p}{1-p}$ : gọi là tỷ lệ cược (odds) Ví dụ: nếu  $p=0.8$ , thì odds =  $0.8 / 0.2 = 4 \rightarrow$  xác suất xảy ra gấp 4 lần không xảy ra
- $\log\left(\frac{p}{1-p}\right)$ : là logarit tự nhiên của odds, gọi là logit

### **Không có giá trị ngoại lệ đáng kể**

Dữ liệu không nên chứa các outlier quá lớn, vì chúng có thể gây ảnh hưởng tiêu cực đến độ chính xác của mô hình.

Các điểm ngoại lệ nên được xử lý thông qua kiểm tra thống kê hoặc trực quan hóa trước khi huấn luyện.

### **Kích thước mẫu đủ lớn**

Hồi quy logistic yêu cầu mẫu đủ lớn để đảm bảo tính ổn định và độ tin cậy trong ước lượng các tham số.

Quy tắc phỏ biến: mỗi lớp của biến phụ thuộc nên có ít nhất 10–20 quan sát cho mỗi biến độc lập.[9]

### 2.2.5. Tham số trong hồi quy Logistic

Hệ số hồi quy (*coefficients*): Đây là các hệ số (hay còn gọi là trọng số) thể hiện mức độ ảnh hưởng của từng đặc trưng (feature) đến xác suất dự đoán của mô hình. Hệ số lớn hơn 0 cho thấy đặc trưng đó tăng xác suất xảy ra, và ngược lại.

Điểm chặn (*intercept*): Là hệ số hằng trong mô hình. Nó cho biết xác suất dự đoán khi tất cả các đặc trưng đều bằng 0, đóng vai trò như "mốc khởi đầu" của mô hình.

Hàm liên kết (*link function*): Trong hồi quy logistic, hàm sigmoid (hoặc logit) đóng vai trò là hàm liên kết, giúp chuyển đổi ra từ dạng tuyến tính thành xác suất trong khoảng (0, 1) để phục vụ cho phân loại.

### 2.2.6. Cách hoạt động của hồi quy logistic

Ước lượng hệ số mô hình ( $\beta$ ):

- Các hệ số  $\beta$  trong mô hình được ước lượng thông qua quá trình tối ưu hóa, phỏ biến nhất là phương pháp Maximum Likelihood Estimation (MLE) – ước lượng hợp lý cực đại.
- MLE tìm ra tập giá trị  $\beta$  sao cho xác suất quan sát dữ liệu huấn luyện là cao nhất có thể dưới mô hình.

Phân lớp (Classification):

- Mô hình logistic dự đoán xác suất một quan sát thuộc về lớp 1. Nếu xác suất  $P(y = 1)$  vượt qua một ngưỡng cụ thể (thường là 0.5), quan sát được phân loại là lớp 1 và ngược lại là lớp 0. Ví dụ:
  - Nếu mô hình trả về  $P(y = 1) = 0.7$  và ngưỡng là 0.5  $\rightarrow$  mẫu này sẽ được phân vào lớp 1

Đánh giá mô hình:

- Hiệu quả mô hình được đánh giá thông qua các chỉ số như:

- Accuracy (độ chính xác)
- Precision, Recall, F1-score
- ROC Curve và AUC – thể hiện khả năng phân biệt giữa hai lớp.[10]

#### *2.2.7. Ưu điểm và hạn chế của hồi quy Logistic*

Thuận lợi	Hạn chế
Khả năng diễn giải mô hình cao	Giả sử mối quan hệ tuyến tính
Hiệu quả tính toán	Khó khăn đối với các vấn đề phức tạp, phi tuyến tính
Xử lý nhiều loại tính năng	Nhạy cảm với đa cộng tuyến
Cung cấp ước tính xác suất	Không phù hợp trong các tập dữ liệu có chiều cao

#### *2.2.8. Ứng dụng hồi quy Logistic*

Hồi quy logistic là một thuật toán mạnh mẽ và linh hoạt, được ứng dụng rộng rãi nhờ khả năng dự đoán xác suất và phân loại nhị phân hiệu quả. Dưới đây là một số lĩnh vực tiêu biểu:

- Tài chính – Chấm điểm tín dụng: Giúp ngân hàng dự đoán khả năng vỡ nợ của khách hàng, hỗ trợ quyết định cho vay an toàn hơn và giảm thiểu rủi ro tài chính.
- Y tế – Dự đoán bệnh tật: Dự đoán nguy cơ mắc bệnh dựa trên dữ liệu sinh học (như microRNA), giúp phát hiện sớm và hỗ trợ chẩn đoán chính xác hơn.
- Kinh doanh – Dự báo rời bỏ khách hàng: Doanh nghiệp sử dụng hồi quy logistic để phân tích hành vi và dự đoán khách hàng có khả năng rời bỏ, từ đó tối ưu hóa chiến lược giữ chân.

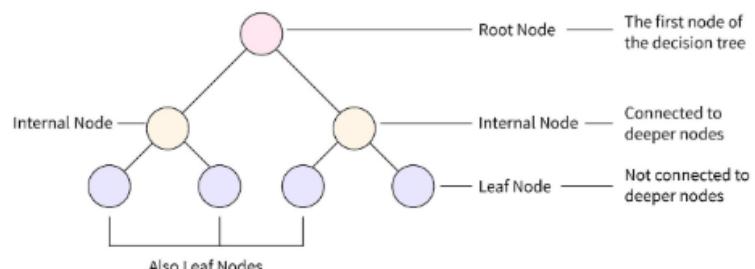
- NLP – Phân loại văn bản: Ứng dụng trong các tác vụ như lọc email spam, phát hiện nội dung độc hại, phân loại chủ đề văn bản và trích xuất thông tin từ PDF/OCR[11]

## 2.3. Decision Tree

### 2.3.1. Khái niệm về Decision Tree (Cây quyết định)

Cây quyết định (Decision Tree) là một thuật toán học máy có giám sát, dùng cho cả phân loại và hồi quy. Mô hình hoạt động như một chuỗi các câu hỏi có điều kiện, trong đó mỗi nút biểu diễn một phép kiểm tra thuộc tính, các nhánh là kết quả của phép kiểm tra, và nút lá chứa dự đoán cuối cùng.

Mục tiêu của cây quyết định là phân tách dữ liệu theo cách mà mỗi nhánh cuối cùng dẫn đến một nhóm dữ liệu đồng nhất nhất có thể. Nhờ tính trực quan và khả năng diễn giải cao, cây quyết định thường được ví như một sơ đồ tư duy ra quyết định, trong đó mỗi bước đi tương ứng với một câu hỏi, và quá trình đi xuống cây tương ứng với việc trả lời các câu hỏi đó cho đến khi đưa ra kết luận cuối cùng.



Hình 2.5: Cây quyết định (Decision Tree)

Thuật ngữ về cây quyết định:

- Root Node (Nút gốc): Là nút đầu tiên của cây, đại diện cho toàn bộ tập dữ liệu. Việc phân tách đầu tiên diễn ra tại đây, dựa trên thuộc tính quan trọng nhất.

- Leaf Node (Nút lá): Là các nút kết thúc của cây, nơi chứa kết quả đầu ra (phân loại hoặc giá trị dự đoán). Nút lá không bị phân tách tiếp.
- Splitting (Phân tách): Là quá trình chia một nút thành các nhánh con dựa trên một điều kiện hoặc thuộc tính cụ thể.
- Branch/Subtree (Nhánh hoặc cây con): Là đường dẫn từ một nút mẹ đến nút con hoặc nút lá, đại diện cho một phần nhỏ của dữ liệu sau khi phân tách.
- Decision Node (Nút quyết định): Là nút trung gian trong cây, nơi dữ liệu được tiếp tục phân tách dựa vào một thuộc tính nào đó.
- Pruning (Cắt tỉa): Là quá trình loại bỏ những nhánh không cần thiết hoặc không cải thiện độ chính xác, nhằm giảm overfitting và làm cho cây gọn hơn.[12]

### 2.3.2. Các thuật toán

#### ID3 (Iterative Dichotomiser 3):

- Tiêu chí phân chia: Information Gain (Lợi thông tin – dựa trên Entropy).
- Chỉ dùng cho: Biến phân loại (Classification).
- Giới hạn: Không hỗ trợ thuộc tính liên tục, dễ bị overfitting.
- Thích hợp: Bài toán nhỏ, dữ liệu sạch và rời rạc.

#### C4.5 (nâng cấp của ID3):

- Tiêu chí phân chia: Gain Ratio (khắc phục thiên lệch của Information Gain).
- Hỗ trợ: Thuộc tính liên tục & rời rạc, xử lý giá trị thiếu.
- Thêm tính năng: Có cơ chế cắt tỉa cây (pruning) để chống overfitting.
- Ứng dụng: Rất phổ biến trong thực tế.

## CART (Classification and Regression Tree)

- Tiêu chí phân chia:
  - o Gini Impurity cho phân loại.
  - o MSE cho hồi quy.
- Hỗ trợ: Cả Classification và Regression.
- Kết quả: Luôn sinh cây nhị phân (mỗi nút chỉ tách thành 2 nhánh).
- Thuật toán phổ biến nhất hiện nay (được dùng trong scikit-learn).[13]

### 2.3.3. Phân loại cây quyết định

Cây quyết định được chia thành hai loại chính, dựa trên bản chất của biến mục tiêu:

- Cây phân loại (Classification tree): Được sử dụng khi đầu ra là một giá trị rời rạc, nhằm phân loại dữ liệu thành các nhóm như “có/không”, “tốt/xấu” hoặc các nhãn lớp khác. Ví dụ, mô hình có thể dự đoán một email là thư rác hay không dựa trên đặc điểm của nội dung. Trong loại cây này các nút lá sẽ chứa nhãn lớp và tiêu chí phân chia phổ biến bao gồm:
  - o Gini Impurity (đo độ thuần nhất của dữ liệu):
$$Gini(S) = 1 - \sum_{i=1}^c p_i^2$$
    - Trong đó:
      - S: tập dữ liệu tại một nút.
      - c: số lượng lớp (categories).
      - $p_i$ : tỷ lệ mẫu thuộc lớp  $i$  trong tập S
    - $Gini = 0 \rightarrow$  dữ liệu thuần nhất hoàn toàn (tất cả cùng 1 lớp).

- Gini càng cao → dữ liệu càng hỗn tạp, càng cần được phân chia tiếp.
- Entropy (đo lường mức độ hỗn loạn của tập dữ liệu):
 
$$Entropy(S) = - \sum_{i=1}^c p_i \cdot \log_2(p_i)$$
- Trong đó:
  - $p_i$  là xác suất của lớp i
  - c là số lượng lớp
- Information Gain (Xác định hiệu quả của một thuộc tính trong việc tách dữ liệu):
 
$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$
- Trong đó:
  - S: tập dữ liệu ban đầu
  - A: thuộc tính đang xét
  - $S_v$ : tập con khi  $A=v$
- Cây hồi quy (Regression tree): Được dùng khi biến mục tiêu là một giá trị liên tục, như giá nhà, doanh thu hoặc điểm số thay vì phân loại, cây hồi quy đưa ra giá trị dự đoán tại các nút lá thường là giá trị trung bình của tập con dữ liệu. Các tiêu chí phân chia trong cây hồi quy thường là MSE (Mean Squared Error) và MAE (Mean Absolute Error). Nhằm tối thiểu sai số giữa dự đoán và giá trị thực. Mỗi loại cây được thiết kế để phù hợp với từng dạng bài toán, giúp mô hình hóa quá trình ra quyết định một cách trực quan và hiệu quả.
- MSE (Mean Squared Error): Tính tổng bình phương sai số giữa giá trị thực và trung bình nhóm.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$$

- MAE (Mean Absolute Error): Tính tổng sai số tuyệt đối.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \bar{y}|$$

- Variance Reduction (Giảm phương sai)): Được dùng để đánh giá hiệu quả của việc chia dữ liệu.

$$Reduction = Var(parent) - \left( \frac{n_1}{n} Var(child_1) + \frac{n_2}{n} Var(child_2) \right)$$

- Trong đó:

- Var(parent): Phương sai (độ biến động) của tập dữ liệu trước khi chia.
- Var(child1) & Var(child2): Phương sai của từng nhánh sau khi chia (tập con).
- n<sub>1</sub>, n<sub>2</sub>: Số lượng mẫu trong mỗi tập con.
- n: Tổng số mẫu tại nút cha (n = n<sub>1</sub>+n<sub>2</sub>).[14]

#### 2.3.4. Cách hoạt động của cây quyết định

Cây quyết định bắt đầu tại nút gốc, nơi mô hình chọn ra thuộc tính quan trọng nhất để thực hiện phép chia đầu tiên. Tại mỗi nút, một điều kiện (câu hỏi) được đặt ra nhằm phân tách dữ liệu thành các nhánh con. Mỗi nhánh đại diện cho một kết quả có thể có của điều kiện đó (ví dụ: “Có” hoặc “Không”).

Quá trình này tiếp tục một cách đệ quy: tại mỗi nhánh con, mô hình lại tiếp tục chọn thuộc tính tối ưu tiếp theo để phân chia dữ liệu, cho đến khi:

- Dữ liệu tại nút là đủ “thuần nhất”
- Hoặc đạt ngưỡng dừng (về độ sâu, số mẫu, tiêu chí dừng khác).

Cuối cùng, nút lá chứa đầu ra dự đoán: có thể là một nhãn lớp (trong phân loại) hoặc giá trị số (trong hồi quy).

#### 2.3.5. *Dánh giá mô hình*

Để đánh giá hiệu quả của mô hình cây quyết định, ta sử dụng các tiêu chí khác nhau tùy thuộc vào bài toán là phân loại hay hồi quy:

Đối với bài toán phân loại (Classification Tree):

- Độ chính xác (Accuracy): Là tỷ lệ phần trăm các dự đoán đúng trong tổng số mẫu quan sát.
- Ma trận nhầm lẫn (Confusion Matrix): Cho biết chi tiết số lần mô hình dự đoán đúng/sai theo từng lớp.
- Precision, Recall, F1-score:
  - o Precision: Độ chính xác trên từng lớp (tỷ lệ dự đoán đúng trong số các dự đoán cho lớp đó).
  - o Recall: Tỷ lệ phát hiện đúng trong tất cả mẫu thực sự thuộc lớp đó.
  - o F1-score: Là trung bình điều hòa của precision và recall, giúp cân bằng giữa độ chính xác và khả năng bao phủ trong mô hình phân loại.
- Đường cong ROC – AUC (nếu là bài toán nhị phân): Đánh giá khả năng phân biệt giữa 2 lớp.

Đối với bài toán hồi quy (Regression Tree):

- MSE (Mean Squared Error): Trung bình sai số bình phương giữa giá trị thực và dự đoán.
- MAE (Mean Absolute Error): Là chỉ số đo sai số dựa trên trung bình các giá trị tuyệt đối của sai lệch giữa dự đoán và thực tế.
- R<sup>2</sup> (R-squared): Đo lường mức độ mô hình giải thích được phương sai của dữ liệu.

Dánh giá chéo (Cross-validation):

- Để đảm bảo mô hình không bị overfitting hoặc phụ thuộc vào dữ liệu huấn luyện, phương pháp k-fold cross-validation thường được sử dụng. Mô hình được huấn luyện và kiểm thử trên nhiều tập con khác nhau của dữ liệu.

#### *2.3.6. Ưu điểm và nhược điểm*

**Ưu điểm:** Cây quyết định dễ hiểu nhờ cấu trúc trực quan như sơ đồ nhánh, phù hợp với cả người không chuyên. Mô hình có thể áp dụng cho cả bài toán phân loại và hồi quy, không yêu cầu chuẩn hóa dữ liệu, đồng thời có khả năng phát hiện các quan hệ phi tuyến giữa các biến

**Nhược điểm:** Mô hình dễ bị quá khóp nếu cây quá phức tạp, đồng thời thiếu ổn định vì chỉ cần thay đổi nhẹ dữ liệu cũng có thể tạo ra cây hoàn toàn khác. Ngoài ra, cây có xu hướng ưu tiên chia theo thuộc tính có nhiều giá trị, dễ gây thiên lệch.

#### *2.3.7. Ứng dụng của cây quyết định*

Tài chính – Phê duyệt khoản vay: Ngân hàng sử dụng cây quyết định để đánh giá đơn vay dựa trên các yếu tố như thu nhập, điểm tín dụng và lịch sử tài chính, giúp đưa ra quyết định nhanh và đáng tin cậy.

Y tế – Hỗ trợ chẩn đoán bệnh: Dựa vào các chỉ số như glucose, BMI hay huyết áp, cây quyết định có thể phân loại bệnh nhân có nguy cơ mắc tiểu đường, hỗ trợ bác sĩ trong quá trình ra quyết định.

Giáo dục – Dự đoán kết quả học tập: Trường học có thể sử dụng mô hình để nhận diện học sinh có khả năng trượt, dựa trên thông tin như thời gian học, điểm kiểm tra, và mức độ chuyên cần, từ đó can thiệp kịp thời.

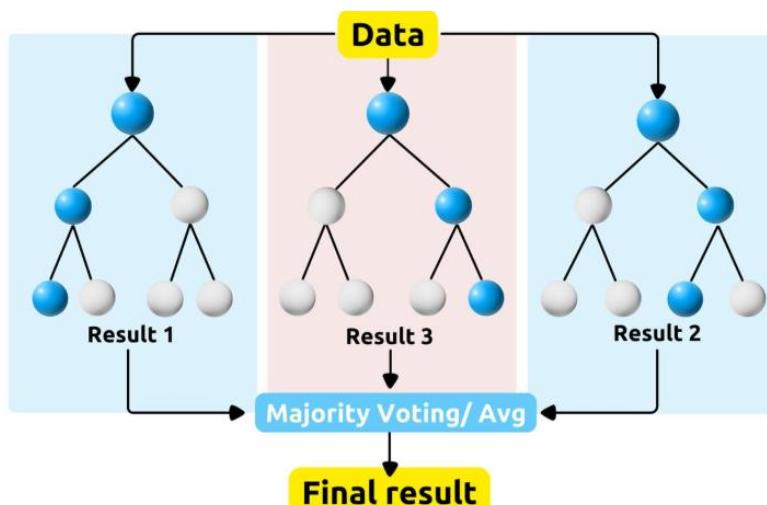
Cây quyết định cũng được sử dụng rộng rãi trong các lĩnh vực như học máy, khai phá dữ liệu và thống kê để xây dựng các mô hình dự đoán tự động.[15]

## **2.4. Random Forest**

#### 2.4.1. Khái niệm về Random Forest

Random Forest là một thuật toán học máy có giám sát, được sử dụng cho cả bài toán phân loại và hồi quy, tuy nhiên phổ biến nhất là trong phân loại. Mô hình này là một tập hợp của nhiều cây quyết định (Decision Trees), được xây dựng từ các tập dữ liệu con được chọn ngẫu nhiên theo phương pháp bootstrap.

Mỗi cây trong forest learning từ một phần dữ liệu và dự đoán độc lập. Khi đưa ra kết quả, Random Forest sẽ tổng hợp các dự đoán từ tất cả các cây bằng cách bỏ phiếu đa số trong phân loại, hoặc lấy trung bình trong hồi quy để đưa ra kết quả cuối cùng. Nhờ cơ chế tổ hợp này, Random Forest giảm thiểu nguy cơ overfitting, đồng thời cải thiện độ chính xác và tính ổn định so với việc chỉ dùng một cây quyết định đơn lẻ.



Hình 2.6: Random Forest

#### 2.4.2. Đặc điểm của Random Forest

Mô hình kết hợp mạnh mẽ: Random Forest là sự kết hợp của nhiều cây quyết định, hoạt động theo cơ chế bagging. Việc tổng hợp kết quả từ nhiều mô hình con giúp giảm thiểu sai số và nâng cao độ chính xác.

Lấy mẫu ngẫu nhiên có hoàn lại (Bootstrap Sampling): Mỗi cây trong rừng được huấn luyện trên một tập dữ liệu con được lấy ngẫu nhiên từ dữ liệu gốc. Kỹ thuật này tạo ra sự đa dạng giữa các cây, góp phần tăng khả năng khai thác hóa.

Tổng quát hóa tốt, ít overfitting: Nhờ học từ nhiều góc nhìn khác nhau của dữ liệu, Random Forest có khả năng dự đoán ổn định và chính xác hơn so với các mô hình đơn lẻ, đặc biệt là trên dữ liệu mới.

Chống nhiễu và ổn định cao: Ngay cả khi dữ liệu chứa nhiễu hoặc ngoại lệ, ảnh hưởng sẽ được trung hòa nhờ việc tổng hợp từ nhiều cây → giúp mô hình duy trì độ tin cậy trong thực tế.

Ít yêu cầu tinh chỉnh tham số: Random Forest thường cho hiệu suất tốt ngay cả với cấu hình mặc định, dễ triển khai hơn nhiều mô hình phức tạp khác.

Hỗ trợ cả phân loại và hồi quy: Đây là một mô hình đa năng, có thể áp dụng hiệu quả cho cả bài toán phân loại nhãn và dự đoán giá trị liên tục.

Phân tích mức độ quan trọng của đặc trưng: Random Forest cung cấp khả năng đánh giá feature importance, giúp hiểu được đặc trưng nào đóng vai trò lớn trong quá trình dự đoán rất hữu ích trong phân tích và lựa chọn đặc trưng.

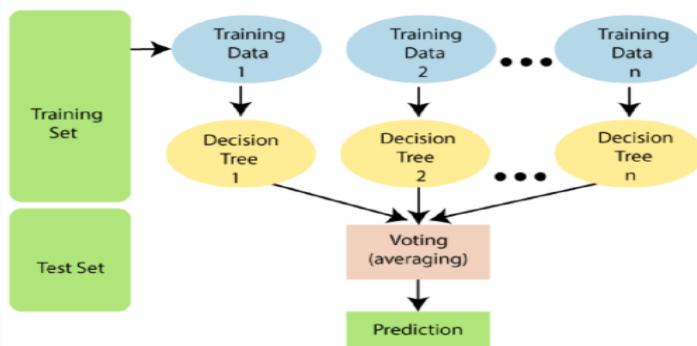
#### 2.4.3. Nguyên lý hoạt động của Random Forest

Random Forest là một thuật toán học máy theo hướng tổ hợp (ensemble), hoạt động bằng cách xây dựng nhiều cây quyết định và kết hợp kết quả dự đoán từ các cây này để đưa ra một dự đoán cuối cùng chính xác và ổn định hơn. Quá trình hoạt động của Random Forest bao gồm các bước chính sau:

- Bước 1: Tạo tập dữ liệu ngẫu nhiên (Bootstrap Sampling).
  - o Từ tập dữ liệu huấn luyện ban đầu, Random Forest tạo ra nhiều tập con bằng cách lấy mẫu ngẫu nhiên có hoàn lại (bootstrap). Điều này có nghĩa là một số mẫu có thể xuất hiện nhiều lần trong một tập con, trong khi một số mẫu khác có thể không xuất hiện.
- Bước 2: Xây dựng nhiều cây quyết định độc lập.
  - o Mỗi tập dữ liệu con sẽ được sử dụng để huấn luyện một cây quyết định riêng biệt. Trong quá trình xây dựng cây, tại mỗi nút phân chia, thuật toán không sử dụng toàn bộ tập đặc

trung, mà chỉ chọn một tập con ngẫu nhiên của các đặc trưng để xét chia tách.

- Điều này giúp tăng tính đa dạng giữa các cây và giảm sự tương quan (correlation) giữa chúng – một yếu tố quan trọng để tăng hiệu quả tổng hợp sau cùng.
- Bước 3: Tổng hợp kết quả dự đoán
  - Khi mô hình đã được xây dựng, Random Forest đưa ra dự đoán bằng cách tổng hợp kết quả từ tất cả các cây trong rừng:
    - Với bài toán phân loại, mô hình sẽ thực hiện bỏ phiếu đa số (majority voting) – lớp được dự đoán nhiều nhất sẽ là kết quả cuối cùng.
    - Với bài toán hồi quy, mô hình sẽ tính trung bình tất cả các giá trị dự đoán từ các cây để đưa ra giá trị đầu ra cuối cùng.[16]



Hình 2.7: Nguyên lý hoạt động của Random Forest

#### 2.4.4. Ưu điểm và nhược điểm của Random Forest

**Ưu điểm:** Random Forest giúp giảm overfitting nhờ kết hợp nhiều cây độc lập, đồng thời có khả năng xử lý dữ liệu nhiều và mối quan hệ phi tuyến hiệu quả. Mô hình cung cấp thông tin về mức độ quan trọng của từng đặc trưng và có thể tùy chỉnh linh hoạt thông qua các siêu tham số. Ngoài ra, nó dễ mở rộng cho bài toán phức tạp và quy mô lớn.

Nhược điểm: So với một cây đơn, Random Forest tiêu tốn nhiều thời gian huấn luyện và tài nguyên tính toán hơn. Mô hình thiếu trực quan và khó giải thích do cấu trúc phức tạp. Với các bài toán hồi quy, kết quả đôi khi không mượt mà, và tốc độ dự đoán có thể chậm nếu yêu cầu phản hồi thời gian thực.

#### 2.4.5. *Ứng dụng của Random Forest*

Random Forest là thuật toán đa dụng, thường được áp dụng trong các bài toán phân loại và hồi quy phức tạp. Trong xử lý ảnh và văn bản, mô hình được dùng để phân loại đối tượng trên ảnh hoặc xác định nội dung văn bản như lọc email rác hay phân loại chủ đề tin tức. Trong lĩnh vực kinh doanh, nó hỗ trợ phân khúc khách hàng và dự đoán khả năng rời bỏ dịch vụ. Riêng trong tài chính – ngân hàng, Random Forest giúp dự đoán rủi ro tín dụng, phát hiện gian lận giao dịch, ước lượng giá cổ phiếu và tối ưu chiến lược tiếp thị theo từng nhóm khách hàng.[17]

#### 2.4.6. *So sánh Random Forest và Decision Tree*

Tiêu chí	Decision Tree	Random Forest
Độ chính xác	Thấp hơn, Dễ bị overfitting	Cao hơn, tổng quát tốt hơn nhờ tổ hợp nhiều cây
Khả năng kháng nhiễu	Nhạy cảm với dữ liệu nhiều	Ôn định hơn, kháng nhiều tốt
Điển giải mô hình	Dễ hiểu, trực quan	Khó giải thích do cấu trúc phức tạp
Tốc độ dự đoán	Nhanh chỉ cần 1 cây để dự đoán	Chậm hơn phải tổng hợp kết quả từ nhiều cây
Overfitting	Dễ xảy ra nếu không cắt tia	Hạn chế được nhờ tính đa dạng trong mô hình
Ứng dụng phù hợp	Bài toán đơn giản, cần mô hình dễ hiểu	Bài toán phức tạp, yêu cầu độ chính xác cao

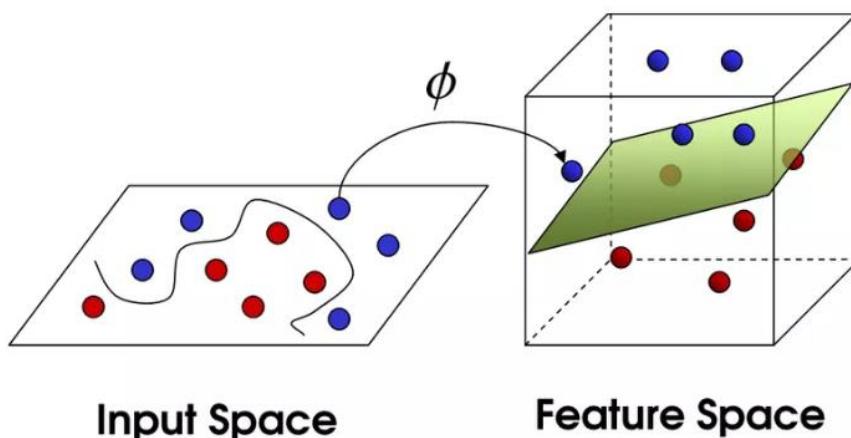
Hình 2.8: So sánh giữa Random Forest và Decision Tree

## 2.5. Support Vector Machine (SVM)

### 2.5.1. Khái niệm về SVM

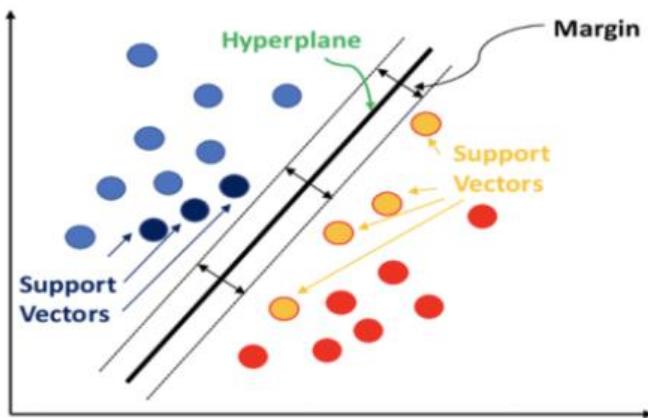
Support Vector Machine (SVM) là một thuật toán học máy có giám sát, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. SVM có nhiều ứng dụng thực tế như trong xử lý tín hiệu, y tế, nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên (NLP) và nhận diện giọng nói.

Mục tiêu chính của SVM là tìm một siêu phẳng (hyperplane) trong không gian đặc trưng, sao cho phân tách tối ưu hai lớp dữ liệu. Tính "tối ưu" ở đây được hiểu là tìm được siêu phẳng tạo ra khoảng cách lớn nhất (maximum margin) giữa hai nhóm dữ liệu qua đó giúp mô hình có khả năng tổng quát hóa tốt hơn khi áp dụng cho dữ liệu mới.



Hình 2.9: Mô tả thuật toán SVM

Trong Support Vector Machine, margin được hiểu là khoảng cách tối đa giữa siêu phẳng phân tách và các điểm dữ liệu gần nhất của mỗi lớp. Đây là vùng không chứa bất kỳ điểm dữ liệu nào, và nằm song song với siêu phẳng phân loại. Với các bài toán có thể phân tách tuyến tính, SVM sẽ cố gắng tìm ra siêu phẳng tối ưu sao cho margin là lớn nhất – từ đó tăng khả năng tổng quát hóa của mô hình.



Hình 2.10: Support Values Machine

Tuy nhiên, trong thực tế, dữ liệu thường không thể phân tách hoàn toàn. Vì vậy, SVM sử dụng một chiến lược linh hoạt hơn gọi là soft margin – cho phép một số điểm dữ liệu bị phân loại sai có kiểm soát, nhằm đạt được hiệu suất tổng thể tốt hơn và tránh overfitting. Mức độ cho phép sai lệch được điều chỉnh bằng tham số điều hòa (regularization parameter) C, cân bằng giữa việc tối đa hóa margin và giảm lỗi phân loại.[18]

### 2.5.2. Ứng dụng của SVM

Mặc dù SVM có thể được áp dụng cho nhiều loại bài toán, dưới đây là một số lĩnh vực ứng dụng phổ biến nhất trong thực tế nhờ vào khả năng hoạt động hiệu quả với dữ liệu có chiều cao và phân tách phức tạp.

- Phân loại văn bản (Text Classification): SVM được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên (NLP). Nhờ khả năng xử lý dữ liệu có nhiều đặc trưng (high-dimensional), SVM tỏ ra rất hiệu quả với văn bản như:
  - Phân tích cảm xúc.
  - Phát hiện thư rác.
  - Phân loại chủ đề tài liệu.
- Phân loại hình ảnh (Image Classification):
  - Nhận diện đối tượng.

- Truy xuất hình ảnh.
- Phân loại ảnh đã bị can thiệp hoặc chỉnh sửa.
- Tin sinh học và y học: SVM có khả năng phát hiện xu hướng tinh vi trong các tập dữ liệu sinh học phức tạp như:
  - Phân loại protein.
  - Phân tích biểu hiện gen.
  - Chẩn đoán bệnh, đặc biệt là trong nghiên cứu ung thư.
- Hệ thống tin địa lý (GIS): SVM hỗ trợ:
  - Phân tích cấu trúc địa vật lý dưới lòng đất.
  - Loại bỏ nhiễu trong dữ liệu điện từ.
  - Dự đoán hóa lỏng địa chấn của đất, ứng dụng trong kỹ thuật dân dụng và địa chất.[19]

### 2.5.3. Thuật ngữ trong SVM

Siêu phẳng (Hyperplane): Là ranh giới quyết định phân chia các lớp trong không gian đặc trưng. Trong trường hợp tuyến tính, siêu phẳng có dạng phương trình:

$$w * x + b = 0$$

- Trong đó:
  - w: Vector trọng số (weight vector) vuông góc với siêu phẳng, xác định hướng phân chia.
  - x: Vector đặc trưng (input vector) đại diện cho một điểm dữ liệu đầu vào.
  - b: Bias – điều chỉnh vị trí của siêu phẳng so với gốc tọa độ.
  - \*: Tích vô hướng giữa 2 vector.
- Phương trình này xác định siêu phẳng phân chia hai lớp dữ liệu trong không gian đặc trưng. Điểm nào nằm đúng trên siêu phẳng sẽ thỏa mãn phương trình  $w * x + b = 0$ .
- Nếu  $w * x + b > 0$ : Điểm nằm một phía của siêu phẳng.

- Nếu  $w \cdot x + b < 0$ : Điểm nằm phía còn lại.

Support Vector (Điểm hỗ trợ): Là những điểm dữ liệu gần siêu phẳng nhất.

Chúng đóng vai trò quyết định trong việc xác định vị trí và hướng của siêu phẳng cũng như khoảng cách margin.

Margin (Biên phân tách): Là khoảng cách giữa hai siêu phẳng hỗ trợ (lè), và SVM tìm siêu phẳng sao cho khoảng cách này lớn nhất.

- Công thức tính margin:

$$\text{Margin} \equiv \frac{2}{\|w\|}$$

- $\|w\|$ : Độ dài của vector w
- Nếu  $\|w\|$  lớn  $\rightarrow$  margin nhỏ  $\rightarrow$  mô hình dễ overfitting.
- Nếu  $\|w\|$  nhỏ  $\rightarrow$  margin lớn  $\rightarrow$  mô hình có khả năng tổng quát tốt hơn.
- Tối ưu hóa siêu phẳng phân loại sao cho margin là lớn nhất, đồng thời đảm bảo phân loại đúng (hoặc sai tối thiểu trong trường hợp soft margin).

Kernel (Hàm hạt nhân): Là một hàm ánh xạ dữ liệu từ không gian đầu vào sang không gian đặc trưng có chiều cao hơn, cho phép SVM xử lý dữ liệu không phân tách tuyến tính. Một số kernel phổ biến:[20]

- Linear:  $K(x_i, x_j) = x_i^T x_j$
- Polynomial:  $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$
- RBF (Gaussian):  $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
- Sigmoid:  $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Hard Margin (Biên cứng): Là siêu phẳng phân tách hoàn hảo hai lớp mà không cho phép điểm nào bị phân loại sai. Áp dụng khi dữ liệu hoàn toàn phân tách được.

- Công thức tính khoảng cách từ một điểm đến siêu phẳng

$$\text{Distance}(x_i) = \frac{|w \cdot x_i + b|}{\|w\|}$$

- Trong đó:

- $w$ : Vector trọng số.
- $b$ : bias là hằng điều chỉnh.
- $x_i$ : điểm dữ liệu bất kì.
- $\|w\|$ : Độ dài (norm) của vector trọng số.

- Bài toán được tối ưu:

$$\min_{w, b} \frac{1}{2} \|w\|^2$$

- Với ràng buộc:

$$y_i (w \cdot x_i + b) \geq 1 \quad \forall i$$

- $y_i \in \{-1, 1\}$ : nhãn của điểm dữ liệu  $x_i$ .

Soft Margin (Biên mềm): Áp dụng khi dữ liệu không phân tách tuyến tính hoàn hảo (do nhiễu, phân bố chồng lấn). SVM sử dụng biến slack  $\xi_i \geq 0$  để cho phép một số điểm vi phạm margin hoặc bị phân loại sai.

- Hàm mục tiêu:

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

- Với ràng buộc:

$$\begin{aligned} y_i (w \cdot x_i + b) &\geq 1 - \xi_i \quad \forall i \\ \xi_i &\geq 0 \quad \forall i \end{aligned}$$

- Trong đó:

- $\xi_i$ : độ vi phạm margin của điểm dữ liệu  $x_i$ .

Tham số C (Regularization Parameter): Là hệ số điều chỉnh giúp cân bằng giữa tối đa hóa margin và giảm thiểu lỗi phân loại.

- C lớn  $\rightarrow$  ít chấp nhận lỗi, dễ overfit.
- C nhỏ  $\rightarrow$  linh hoạt hơn, dễ tổng quát.[21]

Tối ưu hóa SVM bằng Lagrange Multipliers: rong quá trình huấn luyện mô hình SVM tuyến tính, mục tiêu là tìm ra một siêu phẳng phân tách hai lớp dữ liệu sao cho khoảng cách giữa chúng (margin) là lớn nhất, đồng thời các điểm dữ liệu được phân loại chính xác. Bài toán này là một ví dụ điển hình của bài toán tối ưu có ràng buộc, và được giải quyết bằng cách áp dụng phương pháp Lagrange Multipliers.

- Hàm lagrange gốc có dạng:

$$\min_{w, b} \frac{1}{2} |w|^2, \quad y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

- Để đưa các ràng buộc này vào hàm mục tiêu, ta sử dụng Lagrange Multipliers  $\lambda_i \geq 0$ , và xây dựng hàm Lagrange như sau:

$$L(w, b, \lambda) = \frac{1}{2} |w|^2 - \sum_{i=1}^n \lambda_i (y_i(w \cdot x_i + b) - 1)$$

- Trong đó:

- $\lambda_i$ : Hệ số Lagrange tương ứng với từng điểm dữ liệu.
- $x_i$ : điểm dữ liệu đầu vào.
- $y_i \in \{-1, 1\}$ : nhãn của điểm dữ liệu.
- Việc sử dụng hàm Lagrange giúp chuyển bài toán SVM gốc sang dạng đối ngẫu (dual form), giúp tối ưu hiệu quả hơn, đặc biệt với dữ liệu có nhiều chiều. Đây cũng là cơ sở để áp dụng kỹ thuật kernel trong SVM phi tuyến. Việc tối ưu hàm này theo các biến  $w$ ,  $b$  và  $\lambda$  cho phép tìm ra siêu phẳng tối ưu với margin lớn nhất.[22]

#### 2.5.4. Nguyên lý hoạt động của SVM

Thuật toán SVM phân loại dữ liệu thông qua một loạt bước cụ thể nhằm tìm ra ranh giới tối ưu giữa các nhóm (lớp) dữ liệu. Quy trình được thể hiện qua từng bước sau:

- Bước 1: Tiền xử lý dữ liệu đầu vào

- Trước khi huấn luyện mô hình, dữ liệu cần được chuẩn hóa về dạng số học và đưa về cùng thang đo (ví dụ: thông qua chuẩn hóa z-score hoặc Min-Max scaling).
- Việc này giúp SVM hoạt động hiệu quả hơn vì khoảng cách giữa các điểm dữ liệu là yếu tố quan trọng.
- **Bước2:** Tìm ranh giới phân chia (siêu phẳng)
  - SVM xác định một siêu phẳng có khả năng phân tách hai lớp dữ liệu với biên cách đều và lớn nhất có thể.
  - Những điểm dữ liệu nằm sát siêu phẳng ở hai phía được gọi là support vectors chúng quyết định vị trí và hướng của ranh giới.
- **Bước3:** Tối ưu hóa ranh giới
  - Thuật toán sử dụng kỹ thuật tối ưu (ví dụ như phương pháp Lagrange Multipliers) để tính toán các tham số  $w$  và  $b$  sao cho margin khoảng cách giữa siêu phẳng và các support vectors được mở rộng tối đa.
  - Mục tiêu là đảm bảo mô hình có thể phân biệt rõ ràng hai lớp, đồng thời tổng quát hóa tốt với dữ liệu chưa từng gặp.
- **Bước4:** Phân loại dữ liệu mới
  - Khi đã tìm được siêu phẳng tối ưu, mô hình có thể dự đoán nhãn cho dữ liệu mới bằng cách xem điểm đó nằm ở phía nào của siêu phẳng.
  - Nếu  $f(x) = w \cdot x + b > 0$ , dữ liệu được phân vào một lớp nếu nhỏ hơn 0, thuộc lớp còn lại.

Trường hợp dữ liệu không thể phân tách tuyến tính:

- Khi dữ liệu bị chồng lấn và không thể tách bằng một đường thẳng hoặc mặt phẳng, SVM sử dụng hàm kernel để biến đổi dữ liệu sang không gian khác có chiều cao hơn.

- Trong không gian mới này, dữ liệu có thể được phân tách tuyến tính, và mô hình vẫn sử dụng siêu phẳng để phân loại.

#### 2.5.5. Đánh giá hiệu suất cho SVM

Để đánh giá hiệu quả của mô hình SVM, có thể sử dụng một số chỉ số phổ biến sau:

- Accuracy (Độ chính xác tổng thể): Tỷ lệ phần trăm mẫu được phân loại đúng so với tổng số mẫu.

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

o Trong đó:

- TP – Dự đoán đúng lớp dương.
  - TN – đúng lớp âm.
  - FP – dương sai.
  - FN – âm sai.
- Precision (Độ chính xác theo lớp): Tỷ lệ các dự đoán đúng trong số các dự đoán thuộc một lớp cụ thể.

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

- Recall (Tỷ lệ phát hiện): Tỷ lệ các mẫu đúng được mô hình phát hiện trong tổng số mẫu thực tế thuộc lớp đó.

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

- F1-Score: Trung bình điều hòa của Precision và Recall, hữu ích khi cần cân bằng giữa hai yếu tố này.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Confusion Matrix (Ma trận nhầm lẩn): Bảng thể hiện số lượng dự đoán đúng/sai của từng lớp, giúp phân tích chi tiết lỗi mô hình.
- ROC Curve: Đường cong biểu diễn mối quan hệ giữa TPR (Recall) và FPR (dương tính sai) tại nhiều ngưỡng khác nhau.
- Precision-Recall Curve: Biểu đồ thể hiện sự đánh đổi giữa độ chính xác và khả năng phát hiện, đặc biệt hiệu quả với dữ liệu mất cân bằng.[23]

#### 2.5.6. Ưu điểm và nhược điểm của SVM

**Ưu điểm:** SVM hoạt động hiệu quả với dữ liệu có số chiều cao, đặc biệt trong các bài toán văn bản và hình ảnh. Nhờ kỹ thuật kernel, mô hình có thể xử lý tốt các bài toán phi tuyến. Việc tìm siêu phẳng tối ưu giúp SVM phân biệt rõ ràng giữa các lớp và giảm nguy cơ overfitting. Ngoài ra, mô hình có nghiệm duy nhất và vẫn giữ được độ ổn định khi dữ liệu có nhiễu nhẹ.

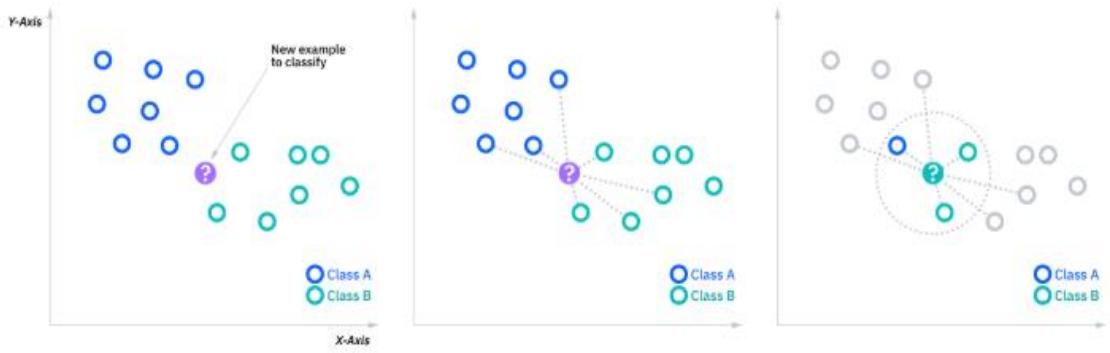
**Nhược điểm:** SVM đòi hỏi lựa chọn kernel phù hợp, nếu không sẽ ảnh hưởng đến hiệu quả mô hình. Với tập dữ liệu lớn, thời gian huấn luyện và chi phí tính toán cao. Hơn nữa, SVM xử lý phân loại đa lớp kém linh hoạt hơn so với các thuật toán khác và có thể bị ảnh hưởng khi dữ liệu chứa nhiều nhiễu.[24]

## 2.6. K – Nearest Neighbors(K-NN)

### 2.6.1 Khái niệm về K – Nearest Neighbors (K-NN)

K-Nearest Neighbors (K-NN) là một thuật toán học máy có giám sát, áp dụng cho cả bài toán phân loại và hồi quy, nhưng thường được sử dụng nhiều hơn trong các bài toán phân loại.

Thuật toán này thuộc nhóm không tham số (non-parametric) và học lười biếng (lazy learning), tức là K-NN không xây dựng mô hình huấn luyện cố định. Thay vào đó, nó lưu giữ toàn bộ dữ liệu huấn luyện và chỉ thực hiện tính toán khi có điểm dữ liệu mới cần dự đoán.[26]



Hình 2.11: Biểu đồ K-NN

### 2.6.2 Nguyên lý hoạt động của K – NN

Thuật toán K-Nearest Neighbors (K-NN) phân loại hoặc dự đoán giá trị cho một điểm dữ liệu mới bằng cách dựa vào các "hàng xóm" gần nhất trong tập dữ liệu đã biết. Cách thức hoạt động của nó có thể tóm gọn qua 4 bước sau:

Bước1: Chọn số lượng hàng xóm (k).

- Người dùng xác định số lượng hàng xóm gần nhất cần xem xét gọi là k. Giá trị này quyết định bao nhiêu điểm trong tập dữ liệu huấn luyện sẽ ảnh hưởng đến kết quả dự đoán của điểm mới.

Bước2: Đo khoảng cách gần giữa điểm cần dự đoán và các điểm trong tập dữ liệu.

- Thuật toán sẽ đo khoảng cách từ điểm mới đến tất cả các điểm đã biết trong tập huấn luyện. Khoảng cách càng nhỏ, điểm đó càng "gần" và có ảnh hưởng nhiều hơn đến kết quả dự đoán.

Bước3: Tìm ra k hàng xóm gần nhất.

- Sau khi tính khoảng cách, thuật toán sắp xếp các điểm dữ liệu theo độ gần và chọn ra k điểm gần nhất chính là các hàng xóm của điểm cần phân loại hoặc dự đoán.

Bước4: Dự đoán kết quả.

- Với bài toán phân loại: thuật toán sẽ “bỏ phiếu” – điểm mới sẽ được gán nhãn theo lớp phổ biến nhất trong số k hàng xóm.

- Với bài toán hồi quy: Thuật toán sẽ đưa ra dự đoán bằng cách tính trung bình giá trị của các hàng xóm gần nhất. [27]

### 2.6.1 Cách chọn k trong K – NN

Việc lựa chọn giá trị k số lượng hàng xóm gần nhất đóng vai trò cực kỳ quan trọng trong hiệu quả của mô hình K-Nearest Neighbors (KNN). Không có một giá trị k cố định cho mọi bài toán, nhưng có một số nguyên tắc giúp bạn chọn k phù hợp nhất:

Chọn k số lẻ như 3, 5, 7... đặc biệt hữu ích trong bài toán phân loại nhị phân, giúp tránh trường hợp hòa phiếu giữa các lớp.

Ước lượng nhanh bằng công thức: Đây là điểm khởi đầu hợp lý khi chưa có dữ liệu kiểm thử:

$$k = \sqrt{N}$$

- Trong đó:

- N là số lượng mẫu trong tập huấn luyện

Sử dụng kỹ thuật xác thực chéo (cross-validation) chẳng hạn như k-fold CV, để thử nghiệm nhiều giá trị k khác nhau và chọn ra giá trị mang lại kết quả chính xác cao nhất trên tập kiểm thử.

Đối với k quá nhỏ mô hình sẽ trở nên rất nhạy cảm với nhiễu trong dữ liệu, dễ bị ảnh hưởng bởi các điểm ngoại lai và có xu hướng overfitting. Do đó mô hình học quá sát dữ liệu huấn luyện và kém khả năng tổng quát.

Nếu chọn k quá lớn, mô hình có thể bỏ qua những đặc điểm quan trọng, khiến ranh giới phân lớp trở nên mờ nhạt và dẫn đến underfitting dẫn đến mô hình quá đơn giản và không thể học được đầy đủ từ dữ liệu.[27]

#### 2.6.4 Tính khoảng cách trong K - NN

Để tìm ra các "hàng xóm gần nhất" trong thuật toán K-Nearest Neighbors (KNN), điều quan trọng là chúng ta cần một cách định nghĩa thế nào là "gần nhất". Đó là lý do vì sao các phép đo khoảng cách được sử dụng để so sánh độ gần giữa một điểm truy vấn và tất cả các điểm còn lại trong tập dữ liệu huấn luyện:

- Khoảng cách Euclidean: Là loại khoảng cách phổ biến nhất, đo đường thẳng ngắn nhất giữa hai điểm trong không gian. Được tính bằng căn bậc hai của tổng bình phương hiệu tọa độ tương ứng giữa hai điểm → Phù hợp với dữ liệu số liên tục trong không gian vector.

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

- Khoảng cách Manhattan: Còn gọi là khoảng cách "taxi", đo tổng các giá trị tuyệt đối của hiệu tọa độ giữa hai điểm. Tưởng tượng như việc di chuyển trong một thành phố có các con đường vuông góc nhau → Phù hợp với hệ thống dữ liệu dạng lưới.

$$d(p, q) = \sum_{i=1}^n |p_i - q_i|$$

- Khoảng cách Minkowski: Là phiên bản tổng quát của Euclid và Manhattan, có thể điều chỉnh theo tham số p.
  - Khi  $p = 1 \rightarrow$  thành khoảng cách Manhattan.
  - Khi  $p = 2 \rightarrow$  thành khoảng cách Euclidean.[28]

$$d(p, q) = \left( \sum_{i=1}^n |p_i - q_i|^p \right)^{\frac{1}{p}}$$

### 2.6.5 Ưu điểm và nhược điểm của K –NN

Ưu điểm: K-NN là thuật toán đơn giản, dễ tiếp cận và không cần huấn luyện mô hình. Mô hình hoạt động tốt với cả bài toán phân loại lẩn hồi quy, hỗ trợ nhiều cách đo khoảng cách và dễ cập nhật khi có dữ liệu mới.

Nhược điểm: Với dữ liệu lớn, K-NN tiêu tốn nhiều tài nguyên do phải tính toán khoảng cách đến toàn bộ tập dữ liệu. Mô hình nhạy cảm với dữ liệu nhiễu và phụ thuộc nhiều vào việc lựa chọn và chuẩn hóa đặc trưng đầu vào.[29]

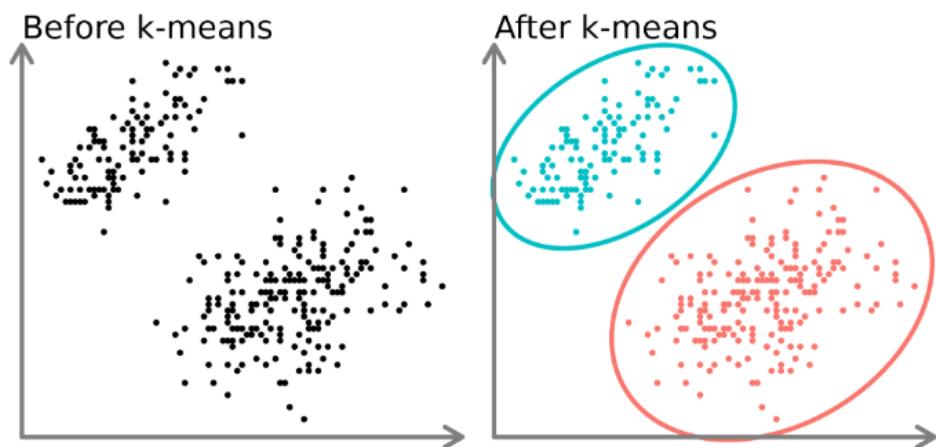
### 2.6.6 Ứng dụng của K –NN trong thực tế

K-NN được ứng dụng rộng rãi nhờ tính đơn giản, dễ triển khai và hiệu quả trong nhận dạng mẫu. Trong lĩnh vực tài chính, K-NN hỗ trợ chấm điểm tín dụng, dự đoán xu hướng thị trường và phát hiện gian lận. Trong y tế, mô hình được dùng để hỗ trợ chẩn đoán bệnh và phân tích gen nhờ khả năng so sánh với hồ sơ tương tự. Ngoài ra, K-NN còn là nền tảng cho các hệ thống gợi ý như đề xuất sản phẩm hoặc nội dung theo hành vi người dùng. Trong xử lý ảnh, K-NN được sử dụng để nhận diện khuôn mặt, phân loại ảnh và phát hiện đối tượng dựa trên đặc trưng hình học.[30]

## 2.7. Thuật toán phân cụm K – Means

### 2.7.1 Khái niệm về K – Means

Thuật toán K-means là một kỹ thuật học không giám sát nổi bật, được ứng dụng rộng rãi trong phân tích dữ liệu nhằm chia tập dữ liệu chưa gán nhãn thành các nhóm (cụm) sao cho các điểm dữ liệu trong cùng một cụm có mức độ tương đồng cao, trong khi sự khác biệt giữa các cụm là tối đa. Nguyên lý hoạt động của K-means dựa trên việc tối thiểu hóa tổng bình phương khoảng cách giữa các điểm dữ liệu và tâm cụm mà chúng được gán vào. Mỗi điểm sẽ được phân vào cụm có trung tâm gần nhất, và các tâm cụm được cập nhật liên tục cho đến khi đạt được trạng thái hội tụ.[31]



Hình 2.12: Minh họa về K - Means

Mục tiêu chính của K-Means là giảm thiểu tổng bình phương khoảng cách giữa các điểm dữ liệu và trung tâm cụm của chúng, được gọi là Within-Cluster Sum of Squares (WCSS):

$$WCSS = \sum_{i=1}^K \sum_{x \in C_i} \|x - \mu_i\|^2$$

Trong đó:

- K: Là số lượng cụm
- $C_i$ : Là tập hợp các điểm thuộc cụm thứ i
- $\mu_i$ : Trung tâm của cụm thứ i
- $\|x - \mu_i\|^2$  : Bình phương khoảng cách Euclidean giữa điểm x và trung tâm cụm  $\mu_i$

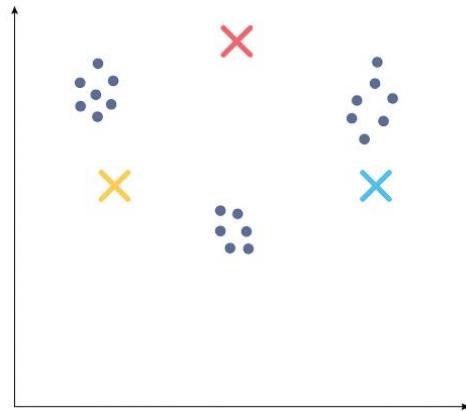
### 2.7.2 Cách hoạt động của K –Means

Bước 1: Chọn số lượng cụm

- Xác định số lượng cụm K mà ta sẽ nhóm dữ liệu. Ví dụ chọn K = 3

Bước 2: Khởi tạo tâm cụm ban đầu

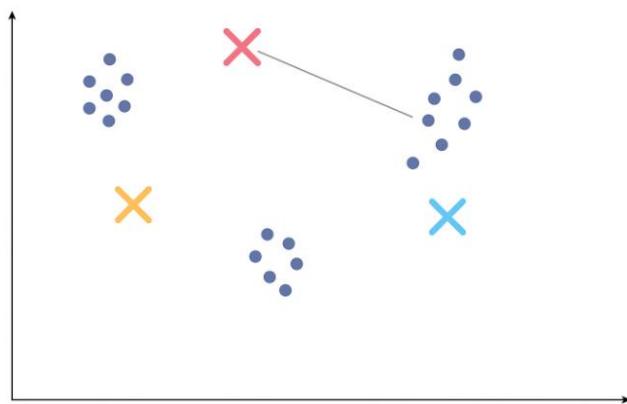
- Do vị trí chính xác của các tâm cụm chưa được biết, nên ở giai đoạn khởi tạo, thuật toán sẽ chọn ngẫu nhiên K điểm từ tập dữ liệu và coi đó là các tâm cụm ban đầu.



Hình 2.13: Ví dụ minh họa về khởi tạo tâm cụm

### Bước 3: Gán điểm dữ liệu cho cụm gần nhất

- Sau khi đã có tâm cụm ban đầu, mỗi điểm dữ liệu sẽ được gán vào cụm có tâm gần nhất. Khoảng cách giữa điểm dữ liệu và tâm cụm thường được đo bằng khoảng cách Euclidean. Điểm dữ liệu sẽ thuộc về cụm có tâm mà nó có khoảng cách Euclidean ngắn nhất.

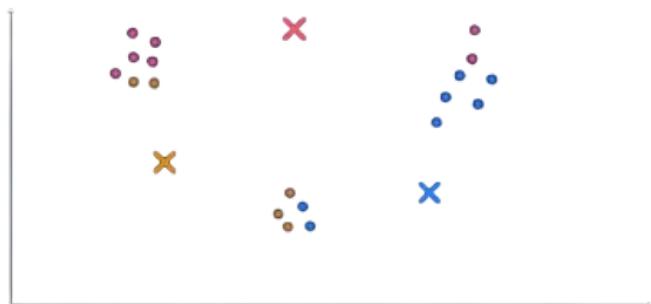


Hình 2.14: Ví dụ về gán điểm dữ liệu cho cụm gần nhất

- Đo khoảng cách từ các điểm tới tâm cụm bằng phép đo khoảng cách Euclidean.

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Sau đó chọn cụm cho dữ liệu có khoảng cách giữa các điểm dữ liệu và tâm nhỏ nhất.



Hình 2.15: Sau khi đo khoảng cách tâm tới điểm dữ liệu

#### Bước4: Khởi tạo lại tâm cụm

- Khởi tạo lại trọng tâm bằng cách tính toán giá trị trung bình của tất cả các điểm dữ liệu của cụm đó.

$$C_i = \frac{1}{|N_i|} \sum x_i$$

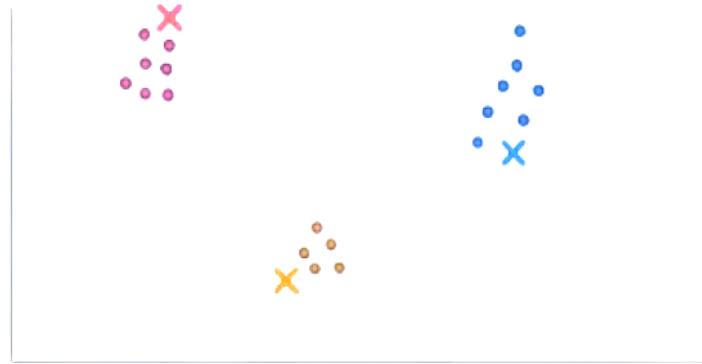
- Trong đó:
  - $C_i$ : là tâm cụm thứ i (centroid của cụm i)
  - $N_i$ : là tập các điểm thuộc cụm i
  - $|N_i|$ : là số lượng điểm trong cụm i (kích thước cụm)
  - $\sum x_i$ : là tổng các vector dữ liệu  $x_i$  trong cụm i



Hình 2.16: Sau khi khởi tạo lại tâm cụm

#### Bước5: Lặp lại

- Tiến hành lặp lại bước 3 và bước 4 cho đến khi có được trọng tâm tối ưu và việc chỉ định các điểm dữ liệu cho các cụm chính xác không còn thay đổi.[32]



Hình 2.17: Hoàn thành quá trình phân cụm

### 2.7.3 Lựa chọn K cho K – Means

Một trong những thách thức lớn nhất khi sử dụng K-Means là xác định số lượng cụm K phù hợp. Không có một giá trị K tối ưu duy nhất cho mọi bộ dữ liệu, và việc chọn K phụ thuộc vào đặc điểm của dữ liệu và mục tiêu phân tích. Dưới đây là những phương pháp chọn K cho tối ưu:

Phương pháp Elbow (Khuỷu tay): Tìm số lượng cụm sao cho tổng phuơng sai trong cụm (WSS – within-cluster sum of squares) là nhỏ nhất, nhưng không giảm quá nhiều khi tăng thêm cụm. Công thức WCSS:

$$WCSS = \sum_{i=1}^K \sum_{x \in C_i} \|x - \mu_i\|^2$$

- Cách thực hiện:
  - Thực hiện phân cụm với các giá trị K khác nhau (ví dụ từ 1 đến 10).
  - Tính tổng WSS cho mỗi K.
  - Vẽ biểu đồ WSS theo số cụm K.
  - Điểm "gấp khúc" (elbow) trên đồ thị thường là nơi phù hợp để chọn K.

Phương pháp Silhouette: Đánh giá mức độ phù hợp của từng điểm dữ liệu với cụm của nó. Giá trị Silhouette càng gần 1 cho thấy điểm đó nằm “đúng” trong cụm hơn

- Cách thực hiện:

- o Phân cụm dữ liệu với các giá trị khác nhau.
- o Tính độ rộng silhouette trung bình cho mỗi giá trị K.
- o Vẽ đồ thị silhouette theo K.
- o Chọn K tại vị trí có giá trị silhouette trung bình lớn nhất.

- Công thức:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- o a(i): Khoảng cách trung bình từ điểm i đến tất cả các điểm khác trong cùng cụm.
- o b(i): Khoảng cách trung bình nhỏ nhất từ điểm i đến các điểm trong cụm khác gần nhất.
- Kết quả trả về nằm trong khoảng [-1,1]
  - o Nếu gần 1: Điểm được phân cụm rất tốt.
  - o Nếu gần 0: Điểm dữ liệu nằm ở biên giữa hai cụm (không rõ ràng).
  - o Điểm gần -1: Điểm dữ liệu có thể bị gán vào cụm sai (rất tệ).
- Đánh giá chất lượng phân cụm theo điểm Silhouette trung bình:
  - o 0.71 – 1.00: Cấu trúc phân cụm mạnh.
  - o 0.51 – 0.70: Cấu trúc phân cụm hợp lý.
  - o 0.26 – 0.50: Cấu trúc phân cụm yếu, cần xem xét lại.
  - o ≤ 0.25: Không tìm thấy cấu trúc phân cụm đáng kể.

Phương pháp Gap Statistic: Gap Statistic là một kỹ thuật thống kê được đề xuất bởi Tibshirani và cộng sự (2001), dùng để xác định số cụm tối ưu trong phân cụm dữ liệu. Phương pháp này đánh giá sự khác biệt giữa độ phân tán của dữ liệu thực và dữ liệu ngẫu nhiên sinh ra từ phân phối đồng nhất.

- Cách thực hiện:

- Tạo dữ liệu tham chiếu bằng cách lấy mẫu ngẫu nhiên từ không gian của dữ liệu gốc.
- Tính và so sánh mức độ phân tán giữa dữ liệu thực và dữ liệu tham chiếu.
- Xác định số cụm tối ưu dựa trên mức độ chênh lệch lớn nhất giữa hai mức phân tán.

- Công thức Gap Statistic:

$$Gap(k) = E_n^*[\log(W_k)] - \log(W_k)$$

○ Trong đó:

- $W_k$ : Tổng phương sai trong cụm khi phân cụm dữ liệu thực với k cụm.
- $E_n^*[\log(W_k)]$ : Giá trị kỳ vọng  $\log(W_k)$  từ dữ liệu ngẫu nhiên sinh ra n lần.
- $n$ : Số lần ngẫu nhiên để được ước lượng giá trị kỳ vọng.

- Cách xác định số cụm K tối ưu:

- Tính Gap Statistic cho nhiều giá trị K.
- Chọn giá trị K nhỏ nhất thỏa mãn điều kiện:

$$Gap(k) \geq Gap(k+1) - s_{k+1}$$

○ Trong đó:  $s_{k+1}$  là sai số chuẩn của  $Gap(k+1)$ .

- Phân tích và đánh giá:

- Phạm vi: Không có giá trị cố định, thường là dương.
- Giá trị cao: Cho thấy cấu trúc phân cụm tốt.
- Chất lượng: K tối ưu là điểm mà Gap Statistic bắt đầu đạt ngưỡng ổn định hoặc giảm.[33]

### *2.7.1 Ưu điểm và nhược điểm của K – Means*

**Ưu điểm:** K-Means là thuật toán đơn giản, dễ triển khai và có tốc độ xử lý nhanh, đặc biệt phù hợp với dữ liệu lớn và số cụm nhỏ. Nhờ cấu trúc tính toán hiệu quả, K-Means dễ mở rộng và có thể áp dụng cho nhiều loại dữ liệu sau khi chuyển đổi phù hợp

**Nhược điểm:** Thuật toán yêu cầu xác định trước số cụm KKK, điều này có thể gây khó khăn khi chưa hiểu rõ dữ liệu. Kết quả phân cụm phụ thuộc nhiều vào cách khởi tạo tâm cụm ban đầu. Ngoài ra, K-Means giả định các cụm có hình dạng đều và dễ bị ảnh hưởng bởi điểm ngoại lai.[34]

### *2.7.5 Ứng dụng của K – Means trong thực tế*

K-Means là một trong những thuật toán phân cụm phổ biến, đặc biệt hiệu quả khi làm việc với dữ liệu số và liên tục. Mô hình được sử dụng rộng rãi trong nhiều lĩnh vực:

Phân khúc khách hàng: Hỗ trợ doanh nghiệp chia nhóm khách hàng theo hành vi hoặc đặc điểm tiêu dùng, từ đó cá nhân hóa chiến lược marketing và chăm sóc khách hàng.

Phát hiện gian lận: Giúp nhận diện các điểm dữ liệu bất thường so với phần lớn còn lại, ví dụ như giao dịch đáng ngờ hoặc truy cập trái phép.

Phân loại tài liệu: Áp dụng trong xử lý ngôn ngữ tự nhiên để nhóm tài liệu theo nội dung hoặc chủ đề tương tự, thông qua biểu diễn vector và đặc trưng từ vựng.

Phân tích dữ liệu không gian địa lý Hỗ trợ chia ảnh thành các vùng dựa trên màu sắc hoặc độ sáng, phục vụ các bài toán thị giác máy tính và y tế như chẩn đoán hình ảnh.[35]

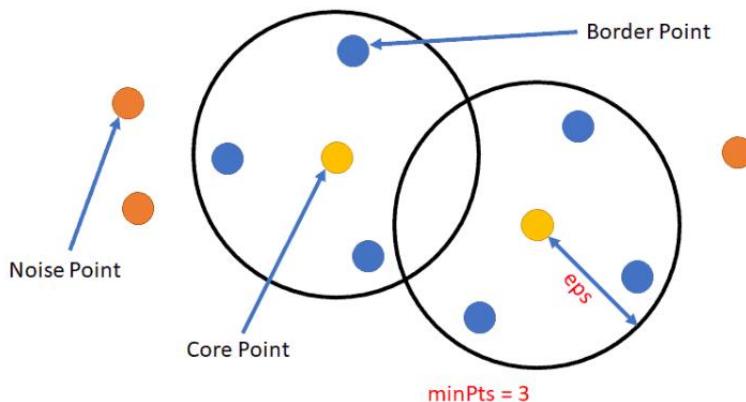
## **2.8. Thuật toán phân cụm Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

### 2.8.1 Khái niệm về DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán phân cụm dựa trên mật độ, được đề xuất bởi Ester, Kriegel, Sander và Xu vào năm 1996.

Thuật toán này phân nhóm các điểm dữ liệu nằm gần nhau trong vùng có mật độ cao, đồng thời nhận diện và loại bỏ các điểm nằm riêng lẻ ở vùng mật độ thấp như là ngoại lệ (nhiễu).

Khác với K-means, DBSCAN không yêu cầu xác định trước số cụm, và đặc biệt hiệu quả trong việc phát hiện các cụm có hình dạng phức tạp và chứa nhiễu. Đây là một trong những thuật toán phân cụm phổ biến và được sử dụng rộng rãi trong học máy không giám sát.[36]



Hình 2.18: Thuật toán DBSCAN

#### Các khái niệm quan trọng trong DBSCAN:

Epsilon ( $\varepsilon$ -neighborhood): Bán kính vùng lân cận của một điểm dữ liệu.

$$N_{\varepsilon}(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\}$$

- Trong đó:
  - $D$  là toàn bộ tập dữ liệu huấn luyện
  - $\text{dist}(p, q)$ : là khoảng cách giữa điểm  $p$  và  $q$  được tính bằng Euclidean.

MinPts: Số lượng điểm tối thiểu nằm trong bán kính  $\varepsilon$  để vùng được xem là có mật độ cao.

**Điểm lõi (Core Point):** Có ít nhất MinPts điểm nằm trong vùng lân cận  $\epsilon$  (kể cả chính nó).

**Điểm biên (Border Point):** Có ít hơn MinPts điểm trong vùng  $\epsilon$ , nhưng thuộc vùng lân cận của một điểm lõi.

**Điểm nhiễu (Noise Point / Outlier):** Không phải điểm lõi và cũng không nằm trong vùng lân cận của bất kỳ điểm lõi nào.[37]

### 2.8.3 Cách xác định tham số cho DBSCAN

Một trong những thách thức chính khi sử dụng DBSCAN là lựa chọn hai tham số cốt lõi là:  $\epsilon$  (epsilon) và MinPts.

Đối với K – Means thì thuật toán cần xác định trước số cụm K còn đối với DBSCAN cần yêu cầu xác định mức độ dày đặt cần thiết để phân thành cụm thông qua hai yếu tố trên

- $\epsilon$  (epsilon): Xác định bán kính vùng lân cận của mỗi điểm. Đây là yếu tố then chốt trong DBSCAN, nhưng đây cũng là tham số khó xác định nhất vì ảnh hưởng lớn đến kết quả của phân cụm.
  - o Để ước lượng giá trị  $\epsilon$  phù hợp nên sử dụng đồ thị khoảng cách k-distance graph:
    - Với mỗi điểm trong tập dữ liệu, tính khoảng cách điểm gần nhất trong k. Trong đó

$$k = \text{MinPts} - 1$$

- Sắp xếp khoảng cách theo thứ tự tăng dần.
- Vẽ biểu đồ khoảng cách này với (trục hoành là số điểm, trục tung là khoảng cách).
- Quan sát biểu đồ để tìm “điểm gãy” (elbow) nơi có độ dốc của đường cong thay đổi rõ rệt.
- Chọn  $\epsilon$  tương ứng với điểm gãy này làm giá trị phù hợp.

- MinPts: Dùng để xác định số điểm tối thiểu cần có trong vùng lân cận  $\epsilon$  để một điểm được xem là điểm lõi.

- o Gó ý chọn MinPts:

$$\text{MinPts} \geq D + 1$$

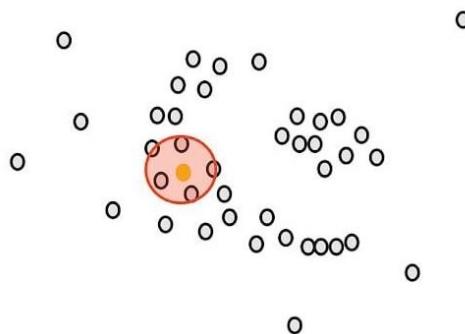
Với  $D$  là chiều của dữ liệu.

- Đối với dữ liệu 2 chiều: Thường chọn MinPts = 4.
    - Dữ liệu nhiều chiều hoặc nhiễu: Nên chọn MinPts = 5 đến 10.
    - o MinPts càng lớn thì thuật toán càng bền vững nhưng có thể bỏ qua những cụm nhỏ hoặc thừa.

### 2.8.3 Nguyên lý hoạt động của DBSCAN

Thuật toán DBSCAN thực hiện phân cụm theo mật độ qua các bước sau:

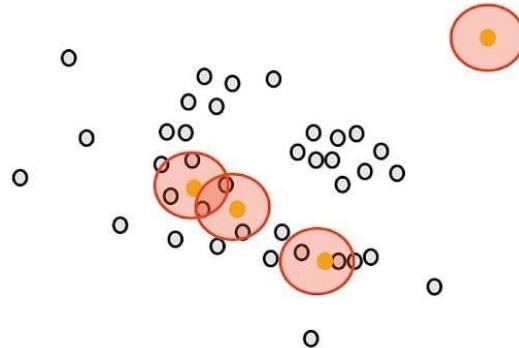
- Bước 1: Chọn một điểm bắt đầu
  - o Chưa được gán nhãn trên tập dữ liệu.
  - o Vẽ một vòng tròn xung quanh đó với bán kính  $\epsilon$  (epsilon) đây là vùng lân cận.



Hình 2.19: Chọn một cụm xung quanh điểm dữ liệu

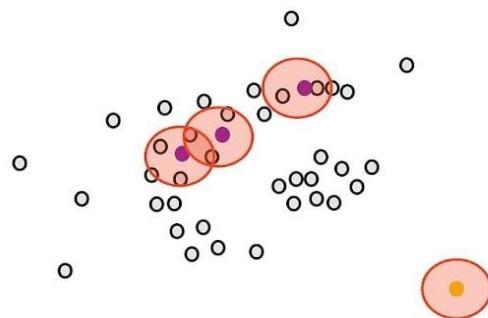
- Bước 2: Kiểm tra số lượng trong vùng  $\epsilon$ 
  - o Đếm số điểm nằm trong  $\epsilon$
  - o Nếu số điểm lớn hơn hoặc bằng MinPts thì được coi là điểm lõi.

- Nếu ít hơn điểm đó có thể là biên hoặc nhiều tùy thuộc và những bước tiếp theo.



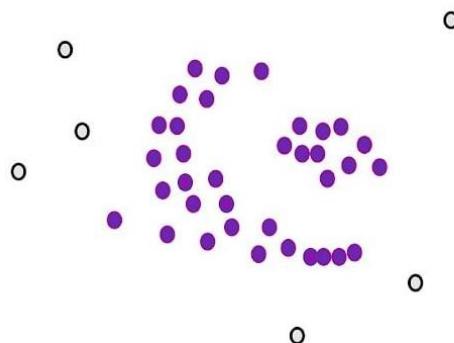
Hình 2.20: Bốn cụm điểm dữ liệu

- Bước 3: Xác định điểm lõi và mở rộng cụm
  - Với điểm lõi vừa tìm được bắt đầu tạo ra cụm mới.
  - Thêm tất cả các điểm trong vùng  $\epsilon$  của điểm lõi vào cụm đó.
  - Nếu trong số các điểm mới thêm vào có điểm nào cũng là điểm lõi thì tiếp tục mở rộng cụm từ điểm đó.



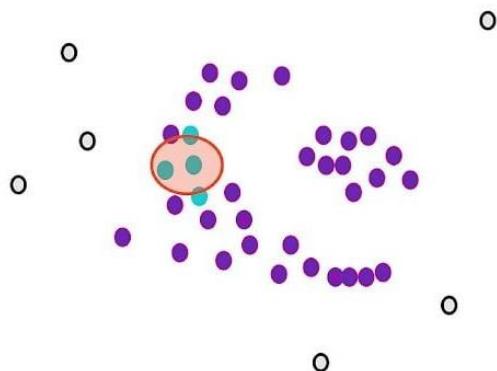
Hình 2.21: Xác định bốn điểm lõi màu tím và một điểm ngoại lề màu vàng

- Bước 4: Xử lý điểm biên và nhiều
  - Những điểm không đủ điều kiện làm điểm lõi nhưng nằm trong vùng lân cận của một điểm lõi sẽ được gán vào cụm đó được gọi là điểm biên.
  - Những điểm không nằm trong các vùng lân cận của bất kỳ điểm lõi nào được coi là điểm nhiều



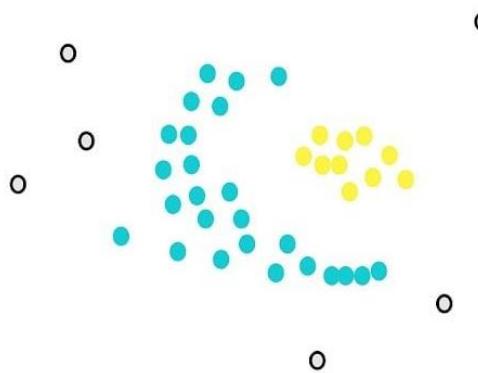
Hình 2.22: Tất cả các điểm dữ liệu lõi có màu tím

- Bước 5: Tiếp tục với các điểm chưa xét
  - o Chọn một điểm lõi khác chưa được gán cụm, lặp lại quá trình mở rộng cụm như các bước trên.
  - o Các cụm mới sẽ được hoàn thành từ các điểm lõi còn lại.



Hình 2.23: Điểm lõi được chọn ngẫu nhiên màu xanh ngọc

- Bước 6: Kết thúc phân cụm
  - o Sau khi xét hết các điểm:
    - Tất cả đều được gán cụm.
    - Các cụm được tô màu khác nhau
    - Điểm nhiễu không thay đổi



Hình 2.24: kết thúc phân cụm

#### 2.8.4 Ưu điểm và nhược điểm của DBSCAN

**Ưu điểm:** DBSCAN có khả năng phát hiện cụm với hình dạng bất kỳ, không bị giới hạn như K-Means. Thuật toán xử lý tốt dữ liệu nhiễu bằng cách tự động loại bỏ các điểm ngoài cụm. Ngoài ra, DBSCAN không yêu cầu xác định trước số lượng cụm, phù hợp với dữ liệu chưa rõ cấu trúc.

**Nhược điểm:** Kết quả phân cụm phụ thuộc nhiều vào lựa chọn tham số epsilon và min\_samples. DBSCAN gặp khó khăn khi dữ liệu có mật độ cụm không đồng đều và hoạt động kém hiệu quả với dữ liệu nhiều chiều do khoảng cách giữa các điểm trở nên không rõ ràng.[38]

#### 2.8.5 So sánh DBSCAN & K – Means

DBSCAN và K – Means đều là thuật toán phân cụm phổ biến trong học máy nhưng mỗi phương pháp lại phù hợp với các tình huống khác nhau do dựa trên nguyên lý hoạt động khác biệt.

Điểm khác biệt:

Tiêu chí	DBSCAN	K – Means
Nguyên lý hoạt động	Dựa trên mật độ điểm dữ liệu	Dựa trên khoảng cách đến các tâm cụm
Hình dạng cụm	Phát hiện cụm có hình dạng bất kì	Ưu chuộng có hình tròn , đồng đều
Yêu cầu số cụm trước	Không cần biết trước số lượng cụm	Phải xác định trước cụm K
Xử lý nhiễu (outliers)	Phát hiện và loại bỏ điểm nhiễu	Nhạy cảm với ngoại lệ
Nhạy cảm với khởi tạo	Ít bị ảnh hưởng	Có thể bị ảnh hưởng nhiều bởi cách chọn tâm ban đầu

Hình 2.25: So sánh giữa DBSCAN và K – Means

### Lựa chọn giữa DBSCAN và K – Means trong thực tế

Sử dụng DBSCAN khi:

- Dữ liệu phân bố phức tạp, không đồng đều hoặc chứa cụm không hình cầu.
- Không rõ trước số lượng cụm.
- Muốn xử lý tốt điểm nhiễu hoặc ngoại lệ.

Sử dụng K – Means khi:

- Dữ liệu có cụm rõ ràng.
- Biết trước số cụm hoặc có giả định hợp lý về số lượng cụm.
- Ưu tiên thuật toán đơn giản nhanh và dễ triển khai.[39]

#### 2.8.6 Ứng dụng trong thực tế của DBSCAN

DBSCAN được sử dụng hiệu quả trong các bài toán phân cụm phức tạp, đặc biệt khi dữ liệu không có cấu trúc rõ ràng hoặc chứa nhiều nhiễu:

Phân cụm không gian: DBSCAN phù hợp với dữ liệu địa lý như xác định khu vực có mật độ tội phạm cao, vùng có tỷ lệ bệnh bát thường hoặc điểm nóng tai nạn giao thông trong đô thị.

Phát hiện bát thường: Mô hình có thể cô lập các điểm lẻ không thuộc cụm, hỗ trợ phát hiện giao dịch gian lận, lỗi cảm biến trong công nghiệp hoặc dữ liệu bất thường trong hệ thống.

Phân khúc khách hàng: DBSCAN cho phép nhóm khách hàng theo hành vi tiêu dùng mà không cần xác định trước số nhóm, hữu ích cho phân tích mua sắm và cá nhân hóa chiến dịch tiếp thị.

Xử lý dữ liệu hình ảnh và văn bản: Có thể ứng dụng vào phân loại ảnh y khoa, thị giác máy tính và nhóm tài liệu theo chủ đề hoặc biểu hiện sinh học, hỗ trợ phân tích bệnh lý hoặc nghiên cứu chuyên sâu.[40]

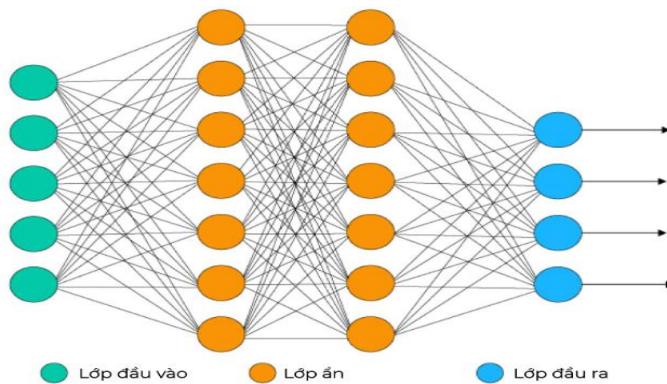
## 2.9. Neural Network (Mạng nơ-ron)

### 2.9.1. Khái niệm về mạng nơ-ron

Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN) là một thuật toán học máy mô phỏng cách hoạt động của hệ thần kinh con người. Mỗi mạng gồm nhiều đơn vị nhỏ gọi là nơ-ron nhân tạo, được kết nối với nhau qua các trọng số và có khả năng xử lý tín hiệu đầu vào để tạo ra kết quả đầu ra.

ANN là nền tảng của nhiều hệ thống học sâu (Deep Learning) hiện nay, và được ứng dụng rộng rãi trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói và phân tích dữ liệu

Cấu trúc của mạng thường gồm ba phần chính: lớp đầu vào, các lớp ẩn và lớp đầu ra. Tín hiệu được truyền qua các lớp này, sau khi được điều chỉnh bởi trong số và các hàm kích hoạt. Thông qua quá trình học, mạng có thể tự điều chỉnh để tối ưu hóa kết quả dự đoán.[41]



Hình 2.26: Cấu tạo của mạng nơ-ron

- Lớp đầu vào: Nhận các đặc trưng (biến) từ dữ liệu và truyền vào hệ thống dưới dạng các nút.
- Lớp ẩn: Xử lý và chuyển đổi dữ liệu đầu vào thông qua các phép toán tuyến tính và phi tuyến. Mỗi nơ-ron trong lớp ẩn kết nối với toàn bộ nơ-ron ở lớp trước và sau. Khi có từ hai lớp ẩn trở lên, mạng được xem là học sâu (Deep Learning).
- Lớp đầu ra: Tạo ra kết quả cuối cùng của mô hình. Với bài toán dự báo số, lớp đầu ra có một nơ-ron. Với phân loại, số lượng nơ-ron đầu ra là  $C-1$  hoặc  $C$ , trong đó  $C$  là số lớp.[42]

### 2.9.2. Quá trình hoạt động của mạng nơ-ron

#### Bước 1: Nhận dữ liệu đầu vào

- Các đặc trưng của dữ liệu (ví dụ: điểm số, hình ảnh, âm thanh...) được đưa vào lớp đầu vào dưới dạng các giá trị số, mỗi giá trị được gán cho một nơ-ron.

#### Bước 2: Truyền tín hiệu qua các lớp ẩn

- Mỗi nơ-ron trong lớp ẩn nhận tín hiệu từ các nơ-ron ở lớp trước, nhân với trọng số tương ứng.
- Sau đó, cộng thêm một giá trị bias, rồi đưa kết quả qua một hàm kích hoạt (activation function) để tạo ra đầu ra phi tuyến tính.

- Bias: Là một giá trị cố định giúp mô hình điều chỉnh đầu ra linh hoạt, tương tự hằng số trong hồi quy và không phụ thuộc vào dữ liệu đầu vào.
- Trọng số: Là một hệ số gán cho mỗi kết nối giữa các nơ-ron, thể hiện mức độ ảnh hưởng của một nơ-ron đến nơ-ron kế tiếp
- Tín hiệu này tiếp tục truyền qua các lớp ẩn tiếp theo (nếu có).

Bước3: Tạo ra kết quả ở lớp đầu ra

- Tín hiệu từ lớp ẩn cuối cùng được đưa đến lớp đầu ra để tạo ra dự đoán cuối cùng, chẳng hạn:

  - Một giá trị duy nhất trong bài toán hồi quy (dự đoán giá trị).
  - Hoặc xác suất của các lớp trong bài toán phân loại.[43], [44]

### Hàm kích hoạt (activation function)[45]

Sigmoid: Cho đầu ra từ 0 đến 1 → thường dùng cho bài toán phân loại nhị phân.

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

ReLU (Rectified Linear Unit): Cho đầu ra bằng 0 nếu đầu vào âm, còn lại giữ nguyên. Phổ biến vì đơn giản và hiệu quả.

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

Tanh: Giống sigmoid nhưng đầu ra nằm trong khoảng từ -1 đến 1.

#### 2.9.3. Ứng dụng của mạng nơ-ron

Mạng nơ-ron nhân tạo được ứng dụng rộng rãi trong nhiều lĩnh vực nhờ khả năng học từ dữ liệu và xử lý các mối quan hệ phi tuyến. Trong thị giác máy tính, mạng nơ-ron giúp nhận diện khuôn mặt, vật thể, chữ viết tay và hình ảnh y khoa. Trong xử lý ngôn ngữ, chúng hỗ trợ dịch tự động, chatbot, phân tích cảm xúc và nhận dạng giọng nói.

Ngoài ra, mạng nơ-ron còn được dùng trong dự báo tài chính, phát hiện gian lận, xe tự lái, hệ thống đề xuất (gợi ý phim, sản phẩm) và phân tích dữ liệu y tế. Đây là nền tảng cốt lõi trong nhiều ứng dụng trí tuệ nhân tạo hiện đại.[46]

## 2.10. ARIMA (Autoregressive Integrated Moving Average)

### 2.10.1. Khái niệm về ARIMA

ARIMA (Autoregressive Integrated Moving Average) là một mô hình thống kê được sử dụng phổ biến trong phân tích và dự báo chuỗi thời gian, đặc biệt hiệu quả khi dữ liệu có xu hướng thay đổi theo thời gian nhưng không thể hiện rõ yếu tố mùa vụ. Nhờ khả năng xử lý chuỗi không dừng thông qua quá trình sai phân, ARIMA phù hợp với nhiều loại dữ liệu thực tế trong lĩnh vực kinh tế, tài chính, vận hành và dự báo ngắn hạn.[47]

#### Giải mã bản chất của mô hình ARIMA

Mô hình ARIMA bao gồm ba thành phần chính, mỗi thành phần phản ánh một khía cạnh đặc trưng trong cấu trúc chuỗi thời gian:

- AR (Tự hồi quy – Autoregression): Thành phần này mô hình hóa mối quan hệ giữa giá trị hiện tại và các giá trị trong quá khứ (các độ trễ). Nói cách khác, dự báo được thực hiện dựa trên chính chuỗi dữ liệu của nó tại các thời điểm trước đó.
- I (Tích phân – Integrated): Nhằm biến một chuỗi không ổn định (non-stationary) thành ổn định (stationary), thành phần này sử dụng kỹ thuật sai phân. Quá trình sai phân thường được thực hiện bằng cách lấy hiệu giữa một giá trị và giá trị liền trước nó.
- MA (Trung bình trượt – Moving Average): Thành phần này xem xét mối quan hệ giữa giá trị hiện tại và sai số (residuals) của các quan sát trước đó trong mô hình trung bình trượt. Nó giúp điều chỉnh nhiều ngẫu nhiên trong dữ liệu.

Các thành phần trên được kết hợp trong mô hình theo dạng chuẩn ARIMA( $p, d, q$ ), trong đó:

- p: Độ tuổi của phần tự hồi quy, thể hiện số lượng quan sát trễ được sử dụng trong mô hình.
- d: Độ tuổi sai phân, chỉ số lần lặp chuỗi được sai phân để đạt tính ổn định.
- q: Độ tuổi của phần trung bình trượt, xác định số lượng độ trễ của sai số được đưa vào mô hình.[48]

$$Y_t = c + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

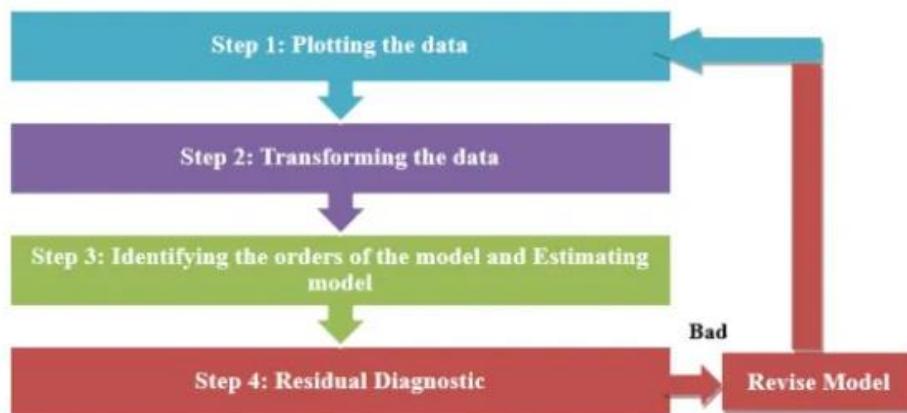
Trong đó:

- $Y_t$  : giá trị chuỗi thời gian tại thời điểm t.
- $\phi_i$  : tham số AR.
- $\theta_i$  : tham số MA.
- $\epsilon_t$  : nhiễu trắng (white noise).

### **Giả định của mô hình ARIMA**

- Chuỗi dừng: Chuỗi đầu vào cần có kỳ vọng, phương sai ổn định theo thời gian. Nếu không, cần sai phân để đạt tính dừng.
- Dữ liệu đơn biến: ARIMA là mô hình đơn biến; dự báo dựa vào chính chuỗi đó mà không có biến độc lập bên ngoài.
- Giới hạn tự hồi quy: Tổng hệ số AR tuyệt đối  $< 1$  để đảm bảo ổn định.
- Khả nghịch MA: Tổng hệ số MA tuyệt đối  $< 1$  để đảm bảo mô hình không nổ (explosion).

### 2.10.2.Các bước xây dựng mô hình



Hình 2.27: Các bước xây dựng mô hình

#### Bước 1: Vẽ biểu đồ dữ liệu

- Trước tiên, ta cần vẽ biểu đồ dữ liệu theo thời gian để quan sát đặc điểm chuỗi: xu hướng, tính mùa vụ, biến động bất thường hoặc outlier. Đây là bước quan trọng giúp hiểu bản chất chuỗi và định hướng bước xử lý tiếp theo.

#### Bước 2: Chuyển đổi dữ liệu (Transforming the data)).

- Nếu chuỗi không ổn định (non-stationary), ta cần chuyển đổi chuỗi về dạng ổn định bằng cách thực hiện phép sai phân (differencing) hoặc biến đổi logarit. Mục tiêu là loại bỏ xu hướng và làm giảm phương sai để chuỗi phù hợp với giả định của ARIMA.

#### Bước 3: Xác định và ước lượng mô hình (Identifying orders and Estimating model).

- Sau khi dữ liệu đã ổn định, ta tiến hành xác định các tham số của mô hình:
  - p: số bậc tự hồi quy (AR).
  - d: số lần sai phân (I).
  - q: số bậc trung bình trượt (MA)
- Việc chọn tham số thường dựa vào các biểu đồ ACF, PACF hoặc phương pháp tự động như auto\_arima. Sau đó, mô hình được ước lượng bằng các công cụ thống kê.

#### Bước 5: Kiểm định phần dư (Residual Diagnostics)

- Sau khi ước lượng xong mô hình, ta kiểm tra phần dư để xem mô hình có phù hợp không. Phần dư tốt cần là nhiễu trắng (white noise), tức không có tự tương quan.
- Nếu kết quả kiểm định phần dư không đạt, ta cần quay lại các bước trước để điều chỉnh mô hình (thay đổi p, d, q hoặc phương pháp xử lý dữ liệu).[49]

#### *2.10.3. Ưu điểm và nhược điểm*

ARIMA là mô hình chuỗi thời gian đơn biến, thường được sử dụng để dự báo các chuỗi có xu hướng. Mô hình kết hợp ba thành phần: hồi quy tự động (AR), sai phân (I), và trung bình trượt (MA), giúp xử lý tốt dữ liệu ổn định và không có mùa vụ. Với cấu trúc đơn giản, dễ triển khai và hiệu quả cao trong dự báo ngắn hạn, ARIMA được áp dụng rộng rãi trong tài chính, kinh tế và quản trị vận hành.

Nhưng trong đó ARIMA không phù hợp với dữ liệu có mùa vụ (trừ khi mở rộng thành SARIMA), dữ liệu phi tuyến, hoặc chuỗi nhiều biến. Ngoài ra, yêu cầu chuỗi phải dừng và khả năng dự báo dài hạn còn hạn chế khiến mô hình này kém linh hoạt hơn so với các phương pháp hiện đại như Prophet hay LSTM.[50]

#### *2.10.4. Ứng dụng của ARIMA*

Mô hình ARIMA được ứng dụng rộng rãi trong nhiều lĩnh vực cần dự báo chuỗi thời gian. Trong tài chính, ARIMA hỗ trợ dự báo giá cổ phiếu, lãi suất và tỷ giá. Trong kinh tế, mô hình giúp ước lượng các chỉ số như GDP, lạm phát hay thất nghiệp. Ngoài ra, ARIMA còn được sử dụng trong bán lẻ để dự báo doanh số, trong ngành năng lượng để ước tính nhu cầu tiêu thụ điện, cũng như trong y tế, môi trường và giao thông để theo dõi và dự báo xu hướng biến động theo thời gian. Nhờ khả năng xử lý tốt chuỗi dữ liệu có xu hướng, ARIMA là lựa chọn phổ biến trong các bài toán dự báo ngắn hạn.

## CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM

### 3.1. Dataset 1: Diamond Price Prediction

#### 3.1.1. Giới thiệu về bộ dữ liệu

Bộ dữ liệu “Diamond Price Prediction” được thu thập từ Kaggle tại địa chỉ: <https://www.kaggle.com/code/abdulrahmankhaled1/diamond-price-prediction>. Đây là một tập dữ liệu gồm hơn 53.000 mẫu dữ liệu về các viên kim cương, bao gồm cả đặc điểm vật lý, chất lượng và giá bán.

#### Mục tiêu phân tích:

- Tìm hiểu mối quan hệ giữa các đặc trưng vật lý và giá bán của viên kim cương.
- Phát triển mô hình hồi quy dự đoán giá dựa trên các thuộc tính đầu vào.
- Đề xuất mô hình có khả năng ứng dụng thực tế trong định giá sản phẩm.

#### Tổng quan về dữ liệu:

Tên biến	Giải thích	Kiểu dữ liệu
carat	Trọng lượng của viên kim cương	Số thực
cut	Chất lượng cắt (Fair, Good, Very Good, Premium, Ideal)	Phân loại
color	Mức độ màu (từ D – tốt nhất đến J – tệ nhất)	Phân loại
clarity	Độ trong của viên kim cương (I1, SI2, SI1,..., IF)	Phân loại
depth	Tỷ lệ chiều sâu so với chiều rộng trung bình (%)	Số thực
table	Tỷ lệ chiều rộng mặt trên với chiều rộng trung bình (%)	Số thực
x, y, z	Kích thước của viên kim cương theo ba chiều (mm)	Số thực

price	Giá của viên kim cương (đơn vị: USD)	Số nguyên
-------	--------------------------------------	-----------

### Đặc điểm của bài toán:

- Đây là một bài toán hồi quy: Đầu ra cần dự đoán (price) là một biến liên tục.
- Dữ liệu có sự pha trộn giữa các biến số thực và biến phân loại.
- Mục tiêu là tìm ra mối quan hệ giữa các thuộc tính của viên kim cương và giá bán của nó, từ đó xây dựng mô hình có khả năng dự đoán giá chính xác cho các viên kim cương mới.

#### 3.1.2. Tiết xử lý dữ liệu

##### Đọc dữ liệu:

```
diamond = pd.read_csv('D:\may_hoc_thong_ke\Diamonds Prices2022.csv', index_col=0)
```

Hình 3.1: Đọc dữ liệu từ Diamonds Prices 2022.csv bằng pandas

##### Khám phá dữ liệu:

```
# kiểm tra thông tin data
diamond.info()
```

<class 'pandas.core.frame.DataFrame'>  
Index: 53943 entries, 1 to 53943  
Data columns (total 10 columns):  
 # Column Non-Null Count Dtype   
--- -- ----- --  
 0 carat 53943 non-null float64  
 1 cut 53943 non-null object  
 2 color 53943 non-null object  
 3 clarity 53943 non-null object  
 4 depth 53943 non-null float64  
 5 table 53943 non-null float64  
 6 price 53943 non-null int64  
 7 x 53943 non-null float64  
 8 y 53943 non-null float64  
 9 z 53943 non-null float64  
dtypes: float64(6), int64(1), object(3)  
memory usage: 4.5+ MB

Python

Hình 3.2: Thông tin cấu trúc dữ liệu kim cương

Kiểm tra thông tin tổng quan cho ta thấy:

Số lượng mẫu: 53.943 dòng dữ liệu phù hợp để xây dựng mô hình học máy có tính ổn định.

Không có giá trị thiếu ở bất kỳ cột nào → đảm bảo độ đầy đủ, không cần xử lý missing values.

Cấu trúc biến:

- Biến số thực (float64): Gồm carat, depth, table, x, y, z thường đại diện cho kích thước, tỷ lệ hình học.
- Biến phân loại (object): Gồm cut, color, clarity cần chuyển sang dạng số trước khi đưa vào mô hình.
- Biến mục tiêu (price) ở dạng số nguyên (int64) phù hợp với bài toán hồi quy liên tục

#thông kê dữ liệu diamond.describe()							
	carat	depth	table	price	x	y	z
count	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000
mean	0.797935	61.749322	57.457251	3932.734294	5.731158	5.734526	3.538730
std	0.473999	1.432626	2.234549	3989.338447	1.121730	1.142103	0.705679
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.000000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

Hình 3.3: Thống kê mô tả dữ liệu số

Nhận xét:

Giá kim cương (price) dao động từ 326 đến 18.823 USD, trung bình khoảng 3.932 USD → cho thấy sự chênh lệch lớn giữa các mẫu.

Trọng lượng (carat) trung bình là 0.8, nhưng có mẫu đến 5.01 carat → cần kiểm tra outlier.

Các cột x, y, z có giá trị tối thiểu bằng 0 → đây là giá trị không hợp lệ, cần xử lý.

Các biến depth và table phân bố khá ổn định, không có dấu hiệu bất thường rõ rệt.

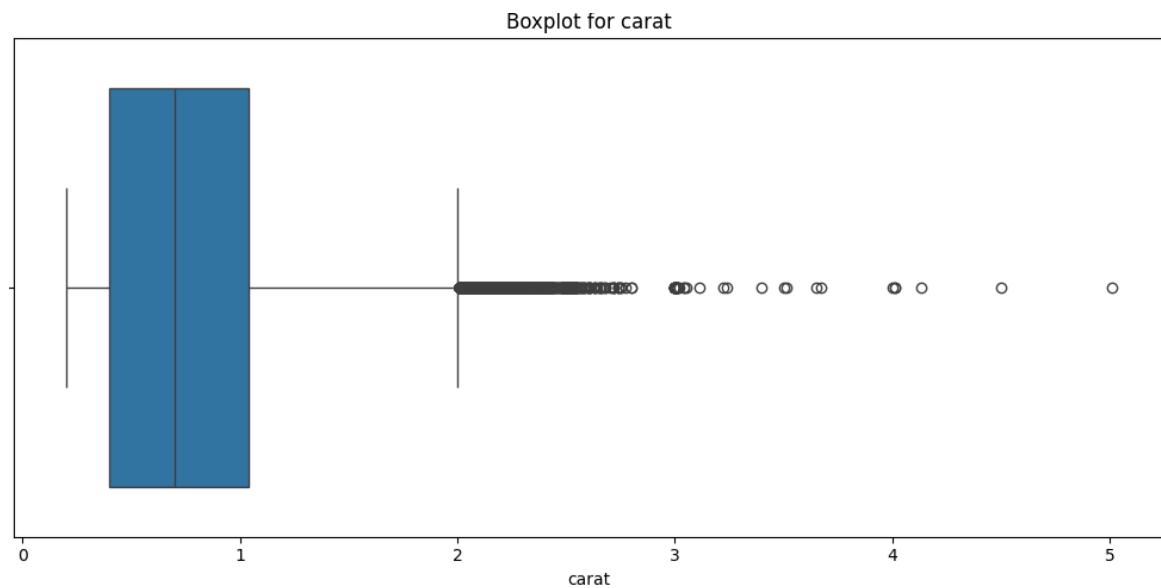
Dữ liệu cần chuẩn hóa do đơn vị và độ lớn các biến khác nhau.

```
#kiểm tra dữ liệu có bị trùng lặp không  
diamond.duplicated().sum()  
✓ 0.0s  
149
```

Hình 3.4: Kiểm tra số dòng dữ liệu trùng lặp bằng

Nhận xét:

Kết quả kiểm tra cho thấy có 149 dòng dữ liệu bị trùng lặp hoàn toàn. Đây là những mẫu lặp lại, không mang thêm thông tin mới, nếu giữ nguyên có thể làm mô hình học sai lệch (overfit). Cần loại bỏ để đảm bảo tính khách quan cho quá trình huấn luyện mô hình.



Hình 3.5: Biểu đồ boxplot cho biến carat

Nhận xét:

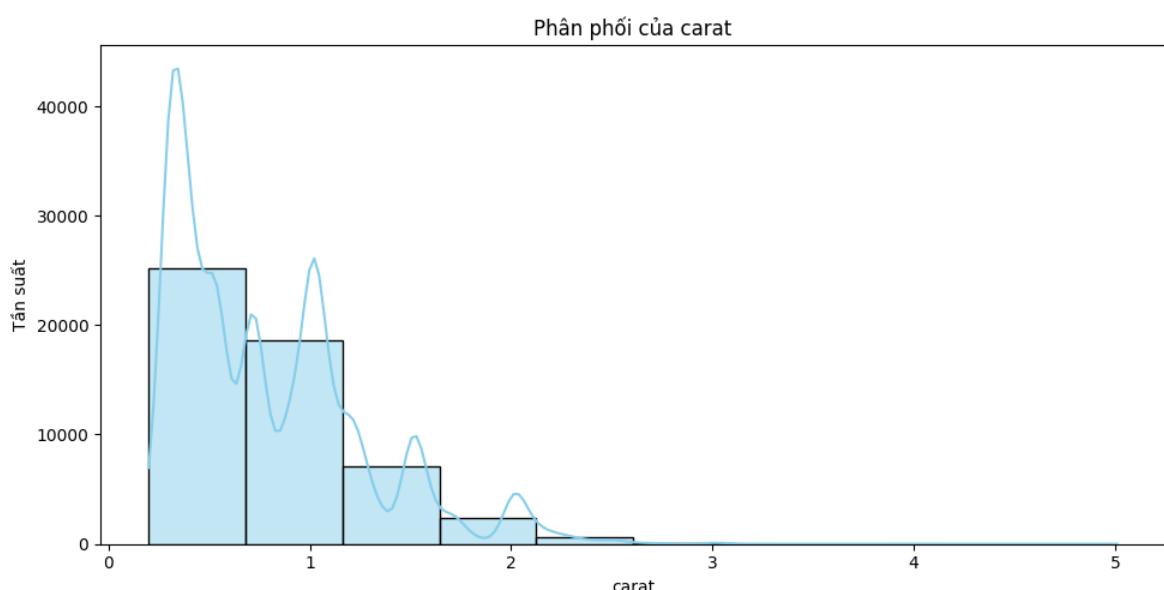
Phần lớn các mẫu có trọng lượng nằm trong khoảng 0.2 – 2 carat.

Tuy nhiên, tồn tại nhiều giá trị ngoại lệ vượt trội (outlier) phía trên, có mẫu đến 5 carat.

Đây là những giá trị hợp lý nhưng hiếm, cần được xử lý cẩn trọng để không gây lệch cho mô hình hồi quy tuyến tính.

### Kết luận:

- Biến carat có nhiều giá trị ngoại lệ phía lớn
- Dữ liệu đã được xử lý outlier bằng phương pháp IQR và thay thế bằng trung bình.
- Các biểu đồ còn lại được trình bày chi tiết trong Notebook đính kèm.



Hình 3.6: Phân phối của biến carat

Nhận xét:

Phân phối có dạng lệch phải rõ rệt, phần lớn dữ liệu tập trung trong khoảng 0.2 – 1.0 carat.

Có rất ít mẫu vượt quá 2 carat → điều này giải thích tại sao các mô hình dẽ bị ảnh hưởng nếu không xử lý outlier.

Đây là kiểu phân phối không chuẩn, nhưng phổ biến trong thực tế (vì kim cương lớn hiếm hơn).

### Kết luận:

- Các biến đầu vào và mục tiêu không tuân theo phân phối chuẩn đây là điều bình thường trong dữ liệu thực tế.

- Việc chuẩn hóa và/hoặc biến đổi log có thể được cân nhắc để cải thiện mô hình tuyến tính, nhưng không bắt buộc với các mô hình như Random Forest hoặc Decision Tree.
- Các biểu đồ còn lại được trình bày chi tiết trong Notebook đính kèm.

### Tiền xử lý dữ liệu:

Sau quá trình khám phá dữ liệu, nhận thấy dữ liệu có chất lượng tốt: Không thiếu giá trị, nhưng tồn tại một số vấn đề có thể ảnh hưởng đến hiệu quả mô hình nếu không được xử lý phù hợp. Do đó, các bước tiền xử lý được thực hiện nhằm đảm bảo mô hình học máy có thể khai thác tốt nhất thông tin từ dữ liệu.

```
#loại bỏ giá trị trùng lặp
diamond = diamond.drop_duplicates()

✓ 0.0s
```

Hình 3.7: Loại bỏ dữ liệu trùng lặp

Nhận xét:

Có tổng cộng 149 dòng dữ liệu bị trùng lặp được phát hiện trước đó.

Việc loại bỏ những dòng này giúp đảm bảo mô hình không học từ dữ liệu bị lặp, tránh hiện tượng overfitting và cải thiện tính khách quan cho quá trình huấn luyện.

```
#loại bỏ những hàng có giá trị 0 trong cột x,y,z
diamond = diamond[(diamond[['x', 'y', 'z']] != 0).all(axis=1)]
```

Hình 3.8: Loại bỏ các dòng có giá trị 0 ở cột x, y, z

Nhận xét:

Dữ liệu có một số dòng có giá trị bằng 0 ở kích thước x, y, z điều này không hợp lý với hình học thực tế của viên kim cương. Các dòng này đã được loại bỏ để đảm bảo độ tin cậy của dữ liệu đầu vào

```

#xử lý biến phân loại
#khởi tạo label encoder
LabelEncoder = LabelEncoder()

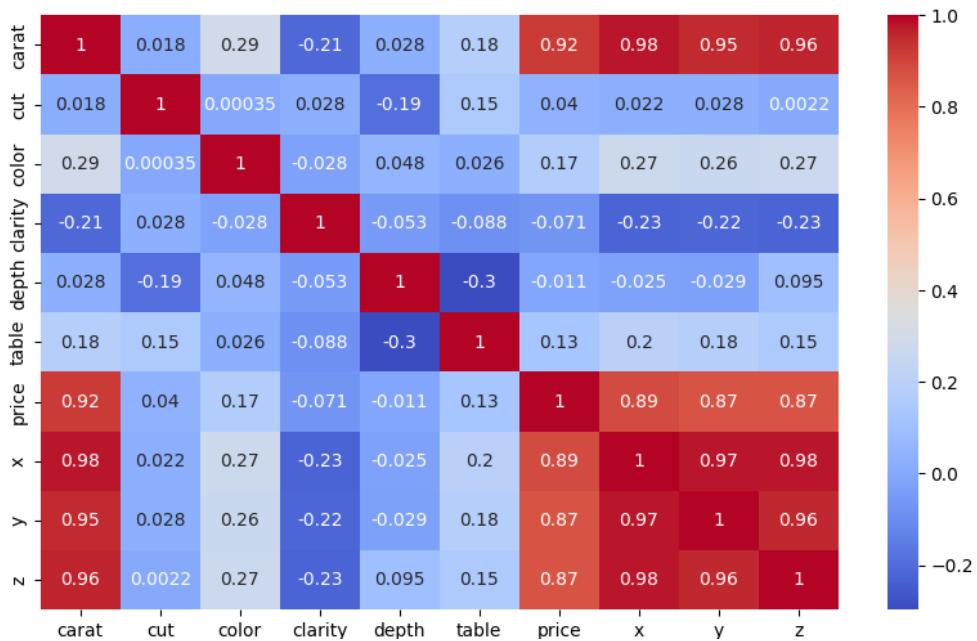
#mã hóa biến phân loại thành số
diamond['cut'] = LabelEncoder.fit_transform(diamond['cut'])
diamond['color'] = LabelEncoder.fit_transform(diamond['color'])
diamond['clarity'] = LabelEncoder.fit_transform(diamond['clarity'])

```

Hình 3.9: Mã hóa biến phân loại bằng LabelEncoder

Nhận xét:

Ba biến phân loại cut, color, clarity được mã hóa thành số bằng LabelEncoder. Việc này giúp mô hình học máy xử lý dữ liệu định tính dưới dạng số nguyên, đồng thời giữ được tính thứ tự tiềm ẩn trong một số biến.



Hình 3.10: Ma trận tương quan giữa các biến số

Nhận xét:

Biến carat có tương quan mạnh nhất với price (hệ số: 0.92) → là đặc trưng quan trọng hàng đầu.

Các biến x, y, z cũng tương quan rất cao với price (trên 0.87), nhưng lại tương quan rất mạnh với nhau → có thể gây đa cộng tuyến.

Đối các biến như depth, table, cut, clarity có tương quan yếu với price → ảnh hưởng ít hơn.

Việc vẽ heatmap giúp xác định rõ các biến quan trọng và hỗ trợ lựa chọn đặc trưng đầu vào hợp lý cho mô hình hồi quy.

```
#xử lý outlier
# 1. Hàm phát hiện outlier bằng IQR
def detect_outliers(column):
    """
    Phát hiện outlier dựa trên IQR.
    Trả về một mảng boolean, True nếu là outlier.
    """
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

# 2. Hàm chuyển outlier thành NaN
def replace_outliers_with_nan(df, column_name):
    """
    Chuyển các outlier trong một cột thành NaN.
    """
    outlier_mask = detect_outliers(df[column_name])
    df[column_name] = df[column_name].where(~outlier_mask, np.nan)

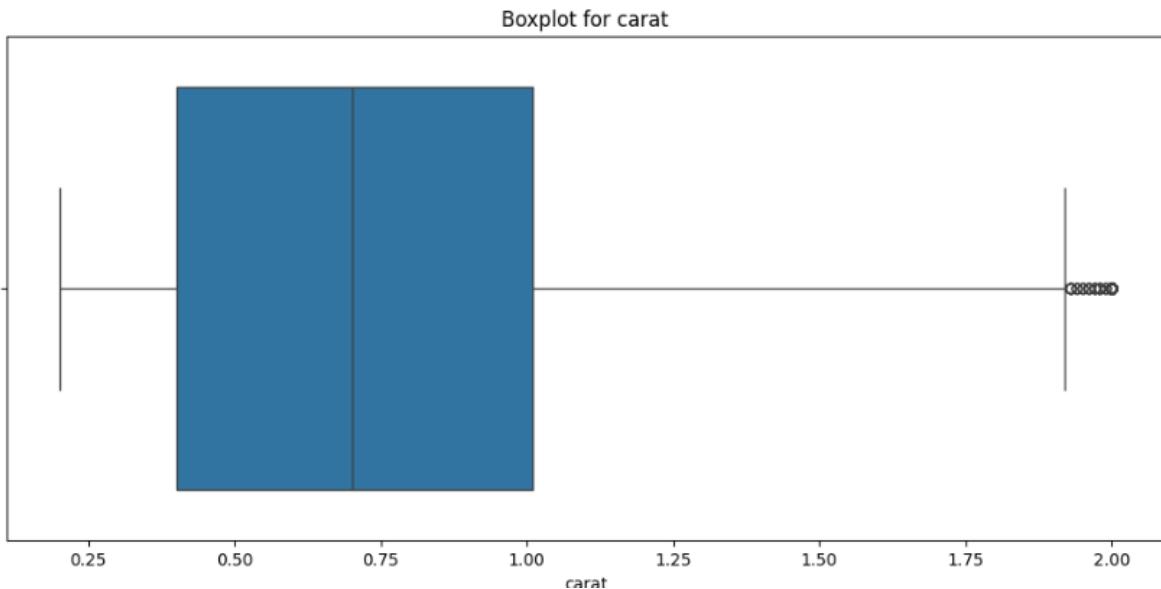
# 3. Hàm thay NaN bằng mean
def fill_nan_with_mean(df, column_name):
    """
    Thay các giá trị NaN trong cột bằng giá trị trung bình.
    """
    mean_value = df[column_name].mean()
    df[column_name] = df[column_name].fillna(mean_value)
```

Hình 3.11: Hàm xử lý outlier bằng phương pháp IQR

Nhận xét: Ba hàm xử lý outlier được xây dựng:

- Phát hiện giá trị ngoại lệ bằng khoảng từ phân vị (IQR).
- Thay thế outlier bằng NaN.
- Đèn lại bằng giá trị trung bình của từng cột.

Phương pháp này giúp dữ liệu sạch hơn mà vẫn giữ lại tính phân phối gần đúng ban đầu, phù hợp với các mô hình hồi quy nhạy cảm với outlier như Linear Regression.



Hình 3.12: Boxplot của biến carat sau xử lý outlier

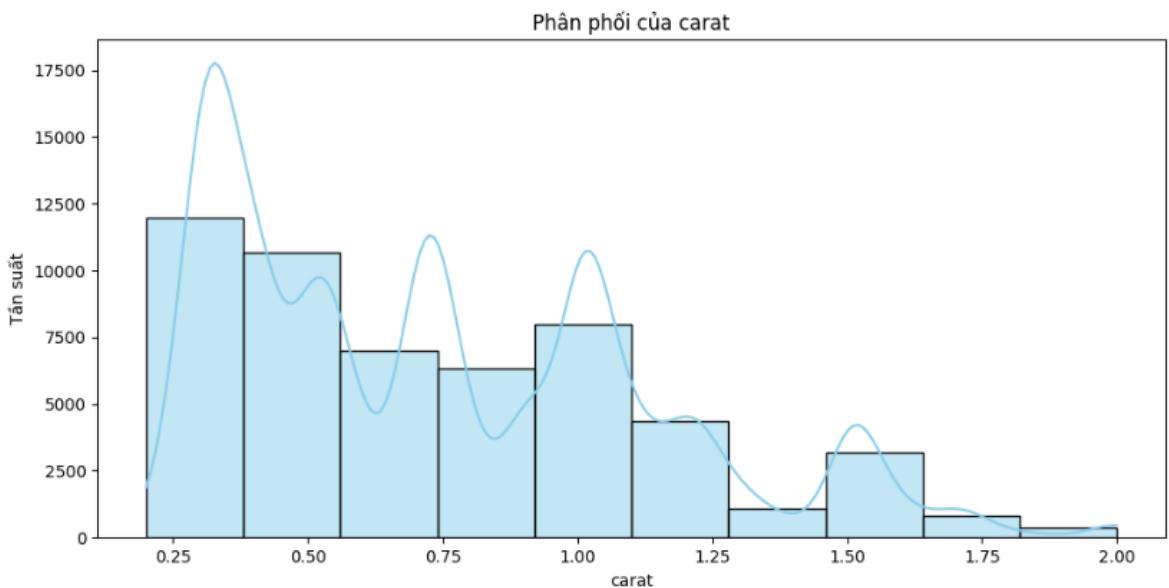
Nhận xét:

Sau khi áp dụng phương pháp IQR và thay thế outlier bằng giá trị trung bình, biến carat vẫn có một số giá trị cao hơn trung bình nhưng đã ổn định và không còn quá chênh lệch.

Phân phối tập trung trong khoảng 0.3 – 1.2 carat, phù hợp với thực tế thị trường kim cương.

Chú thích:

Biểu đồ minh họa sau khi xử lý outlier. Các biến còn lại được xử lý tương tự và được trình bày chi tiết trong notebook đính kèm.



Hình 3.13: Phân phối của biến carat sau xử lý outlier

Phân phối đã mượt và ổn định hơn. Dù vẫn lệch phải nhẹ, dữ liệu đã đủ tốt để sử dụng trong các mô hình học máy.

Chú thích: Các biến khác cũng có xu hướng phân phối tương tự và được trình bày trong notebook đính kèm.

```
# Tính VIF
X = diamond.drop(columns=["price"]) # Biến mục tiêu giá định là X4
X["Intercept"] = 1 # Thêm intercept cho VIF
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

Feature	VIF
carat	4.914938
cut	1.094065
color	1.084197
clarity	1.064935
depth	1.741394
table	1.176156
x	307.439443
y	318.492628
z	86.408929
Intercept	7072.639199

Hình 3.14: Chỉ số VIF kiểm tra đa cộng tuyến giữa các biến

Nhận xét:

Các biến x, y, z có chỉ số VIF rất cao (trên 80 đến hơn 300), phản ánh hiện tượng đa cộng tuyến rõ rệt giữa các biến kích thước không gian.

Tuy nhiên, do mục tiêu của bài toán là dự đoán giá kim cương với đầy đủ thông tin hình học, các biến này vẫn được giữ lại để đánh giá mức độ ảnh hưởng thực tế trong quá trình huấn luyện mô hình.

Các biến còn lại có VIF < 5, không có dấu hiệu đa cộng tuyến đáng kể nên được giữ nguyên.

```
# chuẩn hóa dữ liệu
scaler = MinMaxScaler()
diamond_scaler = diamond.drop(['price'],axis=1)
diamond_scaler_x = scaler.fit_transform(diamond_scaler)
diamond_scaler_y = scaler.fit_transform(diamond['price'].values.reshape(-1,1))
```

Hình 3.15: Chuẩn hóa dữ liệu bằng MinMaxScaler

Tất cả các biến đầu vào và biến mục tiêu (`price`) được chuẩn hóa về khoảng [0, 1] bằng MinMaxScaler. Việc này giúp loại bỏ sự khác biệt về đơn vị đo lường giữa các biến và đảm bảo mô hình hồi quy không bị ảnh hưởng bởi độ lớn khác nhau của dữ liệu.

### 3.1.3. Mô hình hóa và dự đoán giá kim cương

Sau khi tiền xử lý dữ liệu và chuẩn hóa hoàn tất, nhóm triển khai ba mô hình học máy để dự đoán giá kim cương (`price`) dựa trên các đặc trưng đầu vào:

- Linear Regression
- Random Forest
- Decision Tree

Mỗi mô hình được huấn luyện trên dữ liệu đã chuẩn hóa và đánh giá bằng:

- Train/Test Split: Chia dữ liệu thành tập huấn luyện và kiểm tra để đo lường sai số thực tế.
- K-Fold Cross Validation (k=5): Dữ liệu được chia thành 5 phần, mỗi phần lần lượt được dùng làm tập kiểm tra, trong khi 4 phần còn

kỹ thuật đánh giá phô biến, giúp giảm ảnh hưởng của cách chia dữ liệu và đảm bảo tính ổn định của kết quả. Với hơn 50.000 mẫu, chia 5 fold vẫn đảm bảo kích thước đủ lớn cho mỗi lần huấn luyện.

## Linear Regression

Linear Regression là mô hình tuyến tính cơ bản, được sử dụng như một chuẩn so sánh (baseline) trong các bài toán dự báo. Mô hình này giả định mối quan hệ tuyến tính giữa các biến đầu vào và giá trị mục tiêu price. Nhóm tiến hành huấn luyện mô hình trên hai tập biến đầu vào để đánh giá vai trò của các đặc trưng không gian (x, y, z) trong dự đoán giá kim cương

Trong dự án này, nhóm đã thử nghiệm mô hình Linear Regression trên hai tập biến đầu vào:

- Tập đầy đủ: carat, cut, color, clarity, depth, table, x, y, z
- Tập rút gọn (loại bỏ x, y, z): carat, cut, color, clarity, depth, table

Kết quả được đánh giá bằng hai phương pháp: Train/Test Split và K-Fold Cross Validation ( $k=5$ ).

Tập biến đầu vào	R <sup>2</sup> Score (Train/Test)	Mean R <sup>2</sup> (K-Fold)	MAE	RMSE
carat, cut, color, clarity, depth, table, x, y, z	0.8129	0.8147	0.0706	0.093
carat, cut, color, clarity, depth, table	0.587	0.5875	0.073	0.1381

Hình 3.16: So sánh hiệu quả Linear Regression với hai tập biến

## Phân tích chi tiết kết quả cho mô hình Linear Regression

- **Ý nghĩa các chỉ số đánh giá và mức độ đóng góp vào dự đoán**

Với tập biến đầy đủ, mô hình Linear Regression đạt  $R^2 = 0.8129$ , thể hiện rằng mô hình có thể giải thích được hơn 81% sự biến động của giá kim cương thông qua các biến đầu vào.

Đây là kết quả rất đáng kể đối với một mô hình tuyến tính, cho thấy dữ liệu có quan hệ khá rõ ràng với biến mục tiêu. Điều này giúp doanh nghiệp có thể dự báo giá kim cương tương đối chính xác chỉ từ thông tin đầu vào (như trọng lượng, màu sắc, độ trong...).

MAE (Mean Absolute Error):

- MAE đo sai số trung bình tuyệt đối (không quan tâm dấu)
- Cả hai mô hình có MAE tương đối gần nhau (0.0706 vs 0.0730) → mô hình rút gọn vẫn giữ sai số trung bình ở mức chấp nhận được.
- MAE không phản ánh rõ sự ảnh hưởng của các sai số lớn, vốn rất quan trọng trong các bài toán định giá.

RMSE (Root Mean Squared Error):

- RMSE phạt sai số lớn mạnh hơn MAE, do bình phương sai số trước khi lấy căn.
- RMSE mô hình đầy đủ = 0.0930 → thấp hơn nhiều so với RMSE mô hình rút gọn = 0.1381.
- Điều này cho thấy mô hình rút gọn dễ mắc sai số lớn hơn ở các mẫu đặc biệt — ví dụ như những viên kim cương có giá trị cao hoặc hình dạng bất thường, vốn rất nhạy cảm với thiếu hụt thông tin hình học.

- **Đóng góp của từng nhóm biến – đặc biệt là x, y, z**

Bộ ba  $x$ ,  $y$ ,  $z$  thể hiện kích thước thực tế của viên kim cương (chiều dài, rộng, cao) và bổ sung thông tin cho `carat` (trọng lượng), giúp mô hình hiểu rõ hơn về hình dạng thực của viên kim cương.

Khi loại bỏ ba biến này:

- $R^2$  giảm mạnh từ 0.8129 → 0.5870.
- RMSE tăng từ 0.0930 → 0.1381 → Điều này cho thấy mô hình mất đi khả năng dự đoán chính xác với các mẫu có đặc điểm hình học đặc biệt, ảnh hưởng đến độ chính xác tổng thể.

Do đó,  $x$ ,  $y$ ,  $z$  là những biến không thể thiếu nếu mục tiêu là tối ưu độ chính xác dự đoán giá trị trong thực tế.

## Random Forest

Random Forest là mô hình học máy thuộc nhóm ensemble (tập hợp), hoạt động dựa trên việc xây dựng nhiều cây quyết định (Decision Trees) và tổng hợp kết quả dự

đoán từ các cây này. Nhờ cơ chế bỏ phiếu (với bài toán phân loại) hoặc trung bình (với bài toán hồi quy), Random Forest có khả năng giảm overfitting, xử lý tốt dữ liệu phi tuyến và thường cho hiệu suất cao, ổn định

Trong dự án này, mô hình Random Forest được triển khai trên hai tập biến đầu vào:

- Tập đầy đủ: carat, cut, color, clarity, depth, table, x, y, z
- Tập rút gọn: carat, cut, color, clarity, depth, table

Mô hình được đánh giá bằng:

- Train/Test Split
- K-Fold Cross Validation ( $k = 5$ )

Tập biến đầu vào	R <sup>2</sup> Score (Train/Test)	Mean R <sup>2</sup> (K-Fold)	MAE	RMSE
carat, cut, color, clarity, depth, table, x, y, z	0.9784	0.98	0.0151	0.0318
carat, cut, color, clarity, depth, table	0.9774	0.9771	0.016	0.0325

Hình 3.17: So sánh hiệu quả Random Forest với hai tập biến

### Phân tích chi tiết kết quả mô hình Random Forest

- **Mức độ đóng góp vào dự đoán**

Với  $R^2 = 0.9784$ , Random Forest có khả năng giải thích gần 98% phương sai của giá kim cương, thể hiện hiệu quả vượt trội trong việc dự báo so với mô hình tuyến tính.

Kết quả này cho thấy mô hình có thể học được mối quan hệ phi tuyến phức tạp giữa các đặc trưng đầu vào và biến mục tiêu, điều mà Linear Regression không thể thực hiện được.

- **MAE và RMSE cực kỳ thấp**

MAE (0.0151) và RMSE (0.0318) của mô hình với tập biến đầy đủ đều ở mức rất thấp, cho thấy sai số dự đoán trung bình chỉ khoảng 1.5% giá trị sau khi chuẩn hóa.

So sánh với Linear Regression (MAE ~ 0.0706, RMSE ~ 0.0930), Random Forest giảm sai số hơn 75%, đặc biệt là với các mẫu có giá trị lớn.

- **Độ ồn định qua K-Fold Cross Validation**

$R^2$  trung bình qua 5 fold đạt 0.9800, gần bằng với kết quả Train/Test → chứng minh mô hình học tốt và không bị overfit.

MAE và RMSE qua từng fold ồn định, dao động nhỏ → mô hình có khả năng tổng quát hóa rất tốt trên dữ liệu chưa từng thấy.

- **So sánh hai tập biến đầu vào**

Khi loại bỏ x, y, z,  $R^2$  giảm nhẹ (0.9784 → 0.9774) và RMSE tăng không đáng kể (0.0318 → 0.0325).

Điều này cho thấy:

- Với Random Forest, mô hình vẫn hoạt động tốt ngay cả khi thiếu một số biến hình học, nhờ khả năng tự động chọn lọc và phân nhánh thông minh.
- Tuy nhiên, tập biến đầy đủ vẫn cho hiệu suất cao nhất, và nên được ưu tiên sử dụng trong thực tế nếu không có giới hạn về dữ liệu.

## Decision Tree

Decision Tree Regression là mô hình học máy đơn giản, trực quan, hoạt động dựa trên việc chia nhỏ không gian dữ liệu thành các vùng quyết định (decision regions) theo cấu trúc dạng cây. Mỗi nhánh đại diện cho một điều kiện kiểm tra, và mỗi lá chứa giá trị dự đoán.

Với khả năng xử lý linh hoạt các mối quan hệ phi tuyến và không yêu cầu chuẩn hóa dữ liệu, Decision Tree thường được sử dụng như một mô hình nền tảng có thể diễn giải được.

Mô hình Decision Tree được huấn luyện trên hai tập biến đầu vào:

- **Tập đầy đủ:** carat, cut, color, clarity, depth, table, x, y, z
- **Tập rút gọn:** carat, cut, color, clarity, depth, table (loại bỏ x, y, z)

Mô hình được đánh giá bằng phương pháp:

- Train/Test Split

Tập biến đầu vào	R <sup>2</sup> Score (Train/Test)	MAE	RMSE
carat, cut, color, clarity, depth, table, x, y, z	0.9616	0.0201	0.0423
carat, cut, color, clarity, depth, table	0.9641	0.0199	0.0409

Hình 3.18: So sánh hiệu quả Decision Tree với hai tập biến

## Phân tích chi tiết kết quả mô hình Decision Tree

- **Mức độ chính xác và ý nghĩa chỉ số đánh giá**

Cả hai mô hình (với và không có x, y, z) đều đạt  $R^2 > 0.96$ , thể hiện khả năng dự đoán rất cao.

MAE và RMSE đều rất thấp ( $\sim 0.02$  và  $\sim 0.04$ ), chỉ đứng sau Random Forest trong dự án này, và tốt hơn rõ rệt so với Linear Regression.

Như vậy, mô hình Decision Tree có khả năng dự đoán chính xác và ít sai sót ngay cả khi cấu trúc đơn giản.

- **Đặc điểm mô hình và vai trò trong thực tế**

Không cần chuẩn hóa dữ liệu → dễ tích hợp vào các hệ thống xử lý dữ liệu đầu vào trực tiếp.

Có thể diễn giải rõ ràng các ngưỡng phân chia, phù hợp cho các hệ thống yêu cầu giải thích.

Tuy nhiên, không có cơ chế tổng hợp như Random Forest, nên dễ bị overfit nếu dữ liệu có nhiều, đặc biệt khi không giới hạn độ sâu cây.

- **So sánh hai tập biến đầu vào**

Khác với Linear Regression, hiệu suất của Decision Tree không bị ảnh hưởng nhiều khi loại bỏ x, y, z ( $R^2$  còn tăng nhẹ).

Lý do có thể đến từ khả năng phân nhánh điều kiện rõ ràng theo carat, cut, clarity..., giúp mô hình tự tối ưu mà không cần thêm biến học.

Nhưng trong tập đầy đủ, các biến x, y, z vẫn góp phần làm giảm MAE/RMSE nhẹ trong một số tình huống → giữ lại chúng vẫn nên được cân nhắc nếu dữ liệu đầy đủ.

### 3.1.4. Tổng kết

Trong quá trình phân tích và xây dựng mô hình dự báo giá kim cương, nhóm đã triển khai và so sánh ba thuật toán hồi quy: Linear Regression, Random Forest, và Decision Tree, với các tập biến đầu vào khác nhau. Kết quả thực nghiệm cho thấy:

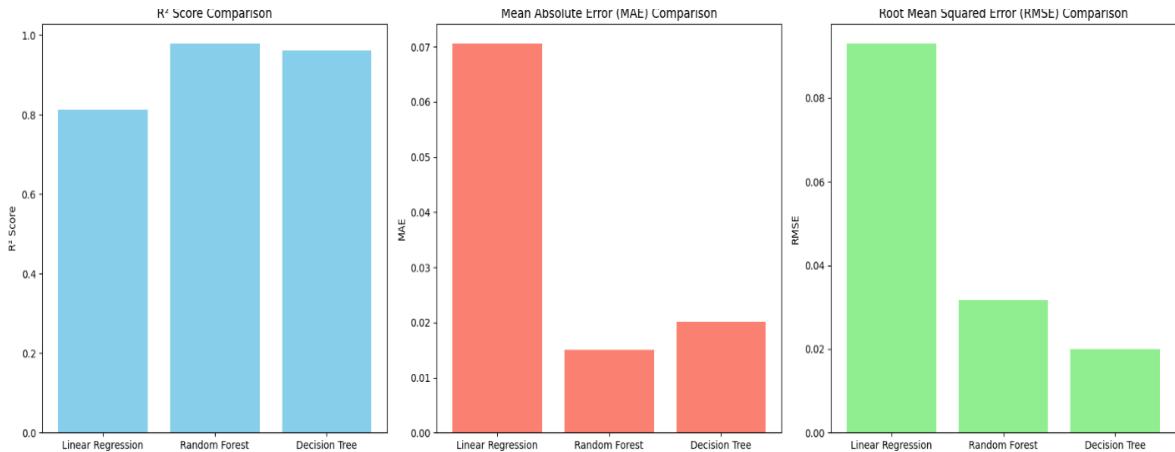
	Model	R <sup>2</sup> Score	MAE	RMSE
0	Linear Regression	0.812933	0.070559	0.092954
1	Random Forest	0.978358	0.015074	0.031773
2	Decision Tree	0.961574	0.020058	0.020058

Hình 3.19: Kết quả của ba mô hình

Linear Regression là mô hình tuyến tính cơ bản, có hiệu quả thấp nhất trong ba mô hình với  $R^2 = 0.8129$  và sai số cao hơn. Dù dễ cài đặt và giải thích, mô hình này không phù hợp cho dữ liệu có quan hệ phi tuyến như định giá kim cương.

Random Forest là mô hình có hiệu suất vượt trội nhất với  $R^2 = 0.9784$ , MAE = 0.0151 và RMSE = 0.0318. Mô hình này học được mối quan hệ phức tạp giữa các biến đầu vào và biến mục tiêu, đồng thời không được overfitting và tổng quát hóa tốt trên dữ liệu mới.

Decision Tree cho kết quả khá tốt với  $R^2 = 0.9616$ , RMSE = 0.0423 và MAE = 0.0201. Đây là mô hình có tính trực quan cao, phù hợp trong các tình huống cần mô hình dễ diễn giải và giải thích logic ra quyết định.



Hình 3.20: Minh họa so sánh hiệu suất 3 mô hình dự báo giá kim cương

Nhận xét:

- **R<sup>2</sup> Score (độ phù hợp):**

Random Forest đạt R<sup>2</sup> gần 0.98 → khả năng giải thích gần như toàn bộ sự biến động của giá.

Decision Tree theo sát ngay sau với R<sup>2</sup> ≈ 0.96.

Linear Regression thấp hơn rõ rệt (R<sup>2</sup> ≈ 0.81) → khó nắm bắt các mối quan hệ phức tạp trong dữ liệu.

- **MAE (Mean Absolute Error):**

Random Forest có sai số trung bình thấp nhất (~0.015), tiếp đến là Decision Tree (~0.020).

Linear Regression sai số cao nhất (~0.071), kém hiệu quả trong các mẫu ngoại lệ.

- **RMSE (Root Mean Squared Error):**

Chỉ số RMSE của Linear Regression cao gấp gần 3 lần so với Decision Tree, chứng tỏ mô hình này không ổn định khi gặp giá trị đặc biệt

Random Forest và Decision Tree có RMSE thấp hơn rõ rệt, dự đoán ổn định hơn.

### Kết luận:

Random Forest là mô hình nên ưu tiên triển khai trong thực tế.

Decision Tree phù hợp nếu yêu cầu minh bạch và dễ hiểu.

Linear Regression chỉ nên dùng làm chuẩn tham chiếu ban đầu.

### 3.2. Dataset 2: Fake Bills

#### 3.2.1. Giới thiệu về bộ dữ liệu

Bộ dữ liệu được trích xuất từ nguồn [Fake Bills - Kaggle](#), gồm thông tin về 1000 tờ tiền thật và 500 tờ tiền giả được đo bằng các tiêu chí liên quan đến hình dạng và tỷ lệ.

#### Mục tiêu phân tích:

Bài toán đặt ra là phân loại một tờ tiền là thật hay giả dựa trên các đặc trưng đo lường vật lý (physical measurements) của tờ tiền. Đây là bài toán phân loại nhị phân, đầu ra có hai lớp (0 = thật, 1 = giả), có tính ứng dụng cao trong lĩnh vực tài chính, ngân hàng và kiểm định tiền tệ.

#### Tổng quan về dữ liệu:

Tên biến	Ý nghĩa	Kiểu dữ liệu
diagonal	Độ dài đường chéo của tờ tiền (mm)	Số thực
height_left	Chiều cao phía bên trái (mm)	Số thực
height_right	Chiều cao phía bên phải (mm)	Số thực
Margin_low	Lề dưới của tờ tiền (mm)	Số thực
margin_up	Lề trên của tờ tiền (mm)	Số thực
length	Chiều dài toàn bộ tờ tiền (mm)	Số thực
is_genuine	Nhận mục tiêu (1 = giả, 0 = thật)	Nhị phân

#### Đặc điểm của bài toán:

- Tập dữ liệu có kích thước nhỏ (1000 mẫu), phù hợp để thử nghiệm các mô hình học máy như Logistic Regression, KNN.
- Các đặc trưng là số thực liên tục, không cần xử lý văn bản hay mã hóa nhãn.
- Bài toán cần tối ưu hiệu suất phân loại (độ chính xác, precision, recall) và xử lý mất cân bằng nếu có

### 3.2.2. Tiết xuât lý dữ liệu

#### Đọc dữ liệu

```
df = pd.read_csv(r"C:\Users\NAM\Downloads\fake_bills.csv",sep=';')
df
```

Hình 3.21: Đọc bộ dữ liệu fake\_bills bằng pandas

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   is_genuine  1500 non-null   bool   
 1   diagonal    1500 non-null   float64
 2   height_left 1500 non-null   float64
 3   height_right 1500 non-null   float64
 4   margin_low   1463 non-null   float64
 5   margin_up    1500 non-null   float64
 6   length       1500 non-null   float64
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

Hình 3.22: Thông tin tổng quan về DataFrame sau khi đọc dữ liệu

Nhận xét:

- Dataset có 1.500 dòng và 7 cột, trong đó:
  - Biến mục tiêu is\_genuine có kiểu bool (xác định tiền thật hay giả).
  - Các cột còn lại là biến số thực (float64) đại diện cho các đặc trưng hình học của tờ tiền như chiều cao, chiều dài, độ dài đường chéo, độ dài biên trên/dưới,..
- Cột margin\_low có thiếu 37 giá trị, cần xử lý trước khi huấn luyện mô hình.

```
### Tạo biến dummies
df['is_genuine'] = df['is_genuine'].replace(to_replace=[False,True],value= [0,1])
```

Hình 3.23: Chuyển đổi biến is\_genuine sang định dạng nhị phân

Nhận xét:

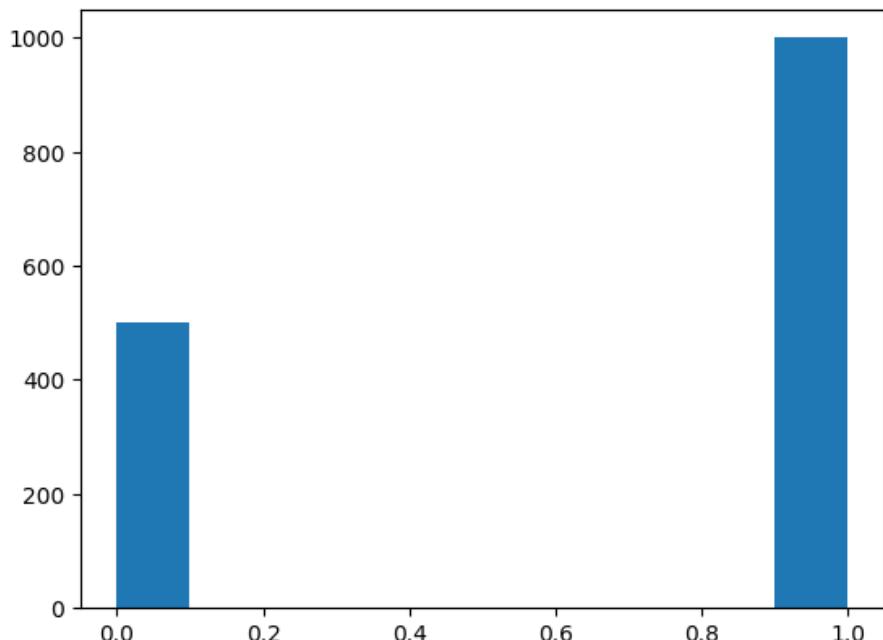
- Biến is\_genuine ban đầu có kiểu bool (True/False).
- Đoạn mã chuyển giá trị True thành 1 (tiền thật), False thành 0 (tiền giả).
- Việc chuẩn hóa này giúp các mô hình học máy (như KNN, Logistic Regression) xử lý tốt hơn vì chúng yêu cầu dữ liệu dạng số.

```
df['margin_low'] = df['margin_low'].fillna(df['margin_low'].mode()[0])
```

Hình 3.24: Điều trị khuyết cho biến margin\_low bằng giá trị mode (giá trị xuất hiện nhiều nhất)

Nhận xét:

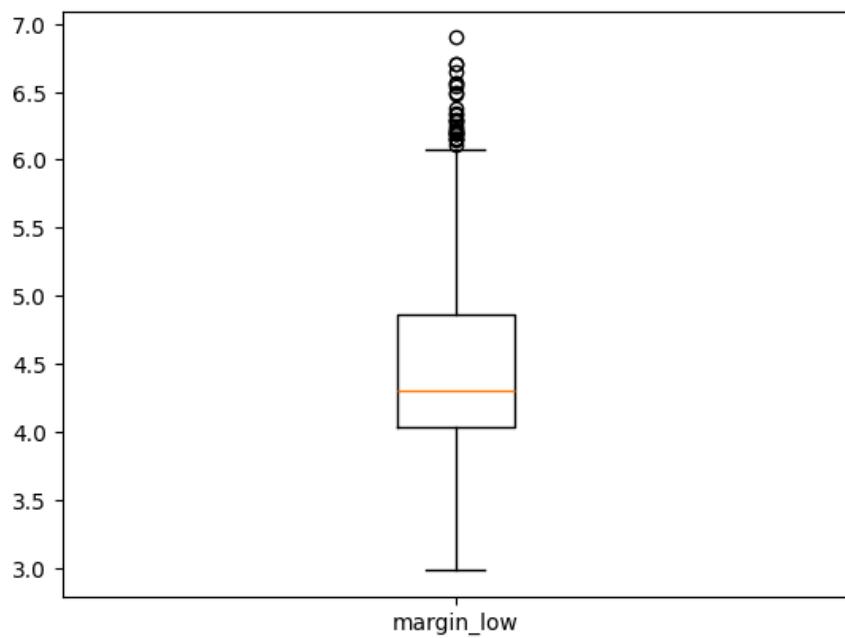
Giá trị khuyết trong cột margin\_low được xử lý bằng cách điền bằng mode (giá trị xuất hiện nhiều nhất). Cách này phù hợp khi dữ liệu dạng liên tục có xu hướng lặp lại một giá trị cụ thể và không có nhiều ngoại lệ. Việc thay thế bằng mode giúp đảm bảo giữ nguyên phân phối dữ liệu gốc và hạn chế làm lệch trung bình hoặc phương sai của tập dữ liệu.



Hình 3.25: Phân phối dữ liệu biến mục tiêu is\_genuine trước khi xử lý mất cân bằng

Biểu đồ cho thấy dữ liệu có sự mất cân bằng đáng kể giữa hai lớp: Khoảng 1.000 mẫu tiền thật và chỉ 500 mẫu tiền giả. Sự mất cân bằng này có thể làm sai lệch

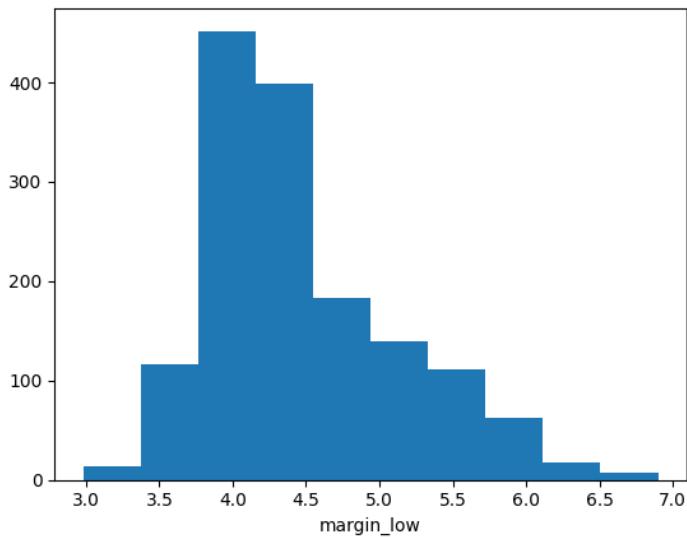
kết quả học của mô hình, nên sẽ cần xử lý bằng kỹ thuật Random OverSampling để đảm bảo mô hình học được tốt cả hai lớp.



Hình 3.26: Boxplot biểu diễn phân bố giá trị cột margin\_low

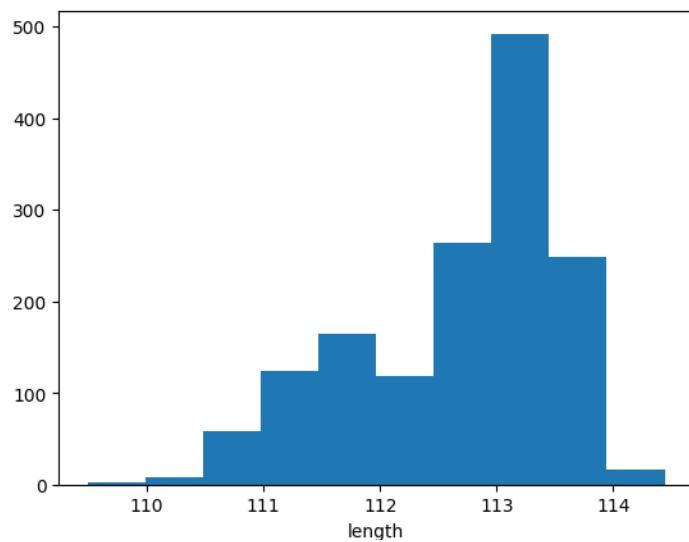
Biểu đồ cho thấy một số điểm ngoại lai (outliers) ở phía trên của biến margin\_low, phản ánh sự khác biệt rõ rệt về kích thước lề dưới giữa các tờ tiền. Tuy nhiên, do đặc điểm của bài toán là phân loại tiền thật – giả dựa trên kích thước vật lý, những điểm này không nên bị xem là giá trị sai lệch, mà có thể là dấu hiệu quan trọng giúp mô hình nhận diện tiền giả. Vì vậy, quyết định giữ lại outlier để bảo toàn thông tin phân biệt tiềm năng cho mô hình học máy.

Chú thích: Các biểu đồ còn lại được trình bày chi tiết trong Notebook đính kèm.



Hình 3.27: Biểu đồ phân bố biến margin\_low

- Phân phối có dạng lệch phải mạnh (right-skewed), với phần đuôi kéo dài về phía các giá trị lớn hơn.
- Có nhiều giá trị nhỏ gần 4.0, nhưng một số mẫu có margin\_low vượt mốc 6.0 và thậm chí gần 7.0.
- Đây có thể là các outlier, nhưng vẫn giữ lại vì trong bối cảnh đo lường giấy tờ/thước đo kỹ thuật số, điều này có thể phản ánh sai số sản xuất giữa tiền thật giả.



Hình 3.28: Phân phối thuộc tính length

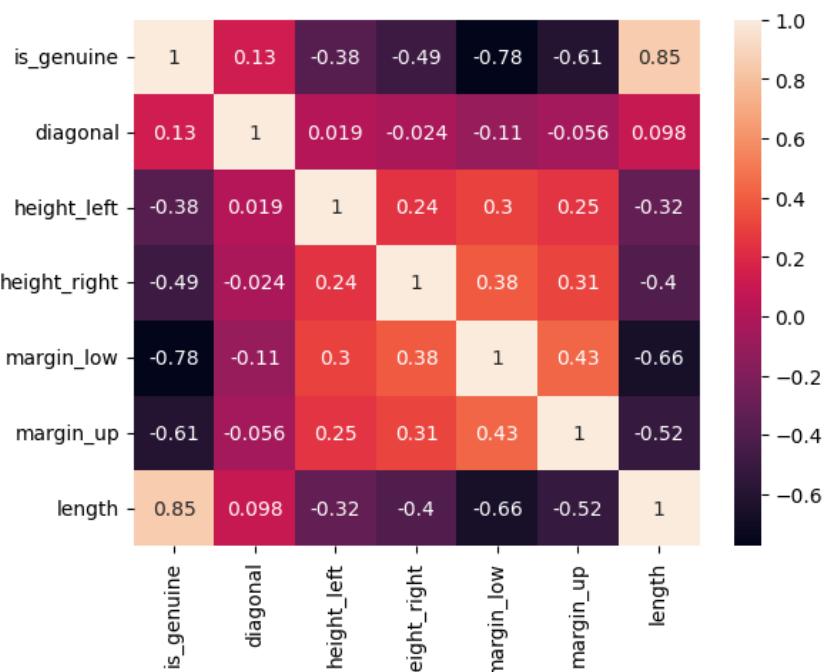
- Phân phối có dạng lệch trái nhẹ (left-skewed).

- Phần lớn các tờ tiền có chiều dài nằm trong khoảng 112–113.5 tức là không có sự dao động lớn về kích thước chiều dài.
- Không xuất hiện điểm bất thường (số liệu quá lớn hoặc âm).

### Kết luận:

- Không phát hiện giá trị bất thường như âm hoặc cực lớn không hợp lý.
- Các biến như length phân phối ổn định, trong khi margin\_low có thể giúp mô hình phân biệt tiền thật và giả hiệu quả hơn nhờ sự phân tán.
- Cần chuẩn hóa dữ liệu để giảm ảnh hưởng của sự khác biệt về thang đo giữa các biến

Chú thích: Các biểu đồ còn lại được trình bày chi tiết trong Notebook đính kèm.



Hình 3.29: Ma trận tương quan giữa các biến trong tập dữ liệu Fake Bills

### Nhận xét:

- Tương quan với biến mục tiêu is\_genuine:

- length có tương quan dương rất cao với is\_genuine (0.85) → đây là biến quan trọng nhất, cho thấy chiều dài tờ tiền là yếu tố then chốt để phân biệt tiền thật – giả.
  - margin\_low và margin\_up có tương quan âm khá mạnh lần lượt là -0.78 và -0.61 → các biến lề nhỏ hơn có xu hướng gắn với tiền giả.
  - height\_right và height\_left có tương quan âm trung bình (-0.49 và -0.38), có thể hỗ trợ phân loại nhưng không phải yếu tố chính.
  - diagonal chỉ có tương quan thấp (0.13) với nhãn phân loại → ít ảnh hưởng.
- Tương quan độc lập:
    - height\_right, height\_left, margin\_low và margin\_up có tương quan dương vừa phải với nhau (~0.3–0.4), thể hiện các yếu tố cấu trúc vật lý có xu hướng đồng biến.
    - Không có cặp nào có tương quan quá cao để gây lo ngại về đa cộng tuyến.

### Kết luận:

- Tất cả các biến đều có ít nhiều liên hệ với biến mục tiêu → có thể giữ lại để đưa vào mô hình.
- length, margin\_low và margin\_up là những đặc trưng quan trọng nhất trong việc phân biệt tiền thật – giả.
- Không cần loại bỏ biến nào do không phát hiện tương quan quá cao giữa các biến độc lập.

```
from sklearn.preprocessing import MinMaxScaler

x = df.drop('is_genuine', axis=1)
x.info()
y = df['is_genuine']
scaler = MinMaxScaler()
x = scaler.fit_transform(x)
```

Hình 3.30: Chuẩn hóa dữ liệu với MinMaxScaler

### Mô tả:

Hình trên minh họa bước chuẩn hóa dữ liệu bằng MinMaxScaler từ thư viện sklearn.preprocessing. Trước khi đưa vào mô hình học máy, tập đặc trưng (x) được tách khỏi biến mục tiêu (is\_genuine), sau đó tất cả các giá trị được chuẩn hóa về khoảng [0, 1].

### Mục đích:

Việc chuẩn hóa này là cần thiết bởi vì các mô hình như KNN và Logistic Regression rất nhạy cảm với thang đo của dữ liệu. Nếu không chuẩn hóa, các đặc trưng có giá trị lớn hơn có thể lấn át các đặc trưng khác, dẫn đến mô hình học lệch.

### Kết luận:

Việc áp dụng MinMaxScaler là một bước quan trọng giúp tăng độ chính xác của mô hình và đảm bảo các đặc trưng đóng vai trò công bằng trong quá trình học.

```
oversample = RandomOverSampler(sampling_strategy='minority')
x_over,y_over = oversample.fit_resample(x, y)
```

Hình 3.31: Cân bằng dữ liệu bằng RandomOverSampler

### Mục tiêu:

Dữ liệu gốc bị mất cân bằng (số lượng nhãn "1" lớn hơn nhãn "0"), điều này có thể khiến mô hình học lệch về phía nhãn chiếm đa số. Để khắc phục, ta áp dụng Oversampling để nhân bản các mẫu thuộc lớp thiểu số.

```
y_over.value_counts()
  ✓ 0.0s
  is_genuine
    1    1000
    0    1000
  Name: count, dtype: int64
```

Hình 3.32: Phân bố lớp sau khi áp dụng Random OverSampling

- Trước khi oversampling, lớp 0 (tiền giả) có số lượng thấp hơn lớp 1 (tiền thật).

- Sau khi áp dụng RandomOverSampler, cả hai lớp đã được cân bằng với 1.000 mẫu mỗi lớp
- Việc cân bằng này giúp mô hình học tốt hơn, tránh thiên lệch về lớp chiếm đa số, từ đó cải thiện độ chính xác cho bài toán phân loại.

```
x_train, x_test, y_train, y_test = train_test_split(x_over, y_over, test_size=0.2, random_state=42)
```

Hình 3.33: Chia dữ liệu thành tập huấn luyện và kiểm tra sau khi cân bằng lớp

- train\_test\_split: Chia tập dữ liệu thành 80% huấn luyện và 20% kiểm tra.
- x\_over, y\_over là dữ liệu đã được cân bằng lớp bằng RandomOverSampler.
- random\_state=42: Đảm bảo kết quả có thể lặp lại.

### 3.2.3. Bài toán 1: Phân loại bằng K – Nearest Neighbors (K – NN)

Phát hiện tiền giả dựa trên các đặc trưng vật lý của tờ tiền như chiều dài, chiều rộng, độ dày viền trên/dưới, chiều cao... bằng mô hình **K – Nearest Neighbors (K-NN)**.

#### Xây dựng mô hình K – Nearest Neighbors (K - NN)

- Khởi tạo mô hình:
  - Sử dụng KNeighborsClassifier() từ thư viện sklearn.neighbors
  - Áp dụng GridSearchCV để tìm số lượng hàng xóm (n\_neighbors) tối ưu từ 1 đến 20, với 5-fold cross-validation.

```
param_grid = {'n_neighbors': range(1, 21)}
knn = KNeighborsClassifier()
grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(x_train, y_train)
best_k = grid_search.best_params_['n_neighbors']
best_accuracy = grid_search.best_score_
```

Hình 3.34: Khởi tạo mô hình

```
best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(x_train, y_train)
```

Hình 3.35: Huấn luyện mô hình với best\_k

```
print(classification_report(y_pred=best_knn.predict(x_test),y_true=y_test))
```

Hình 3.36: Đánh giá mô hình

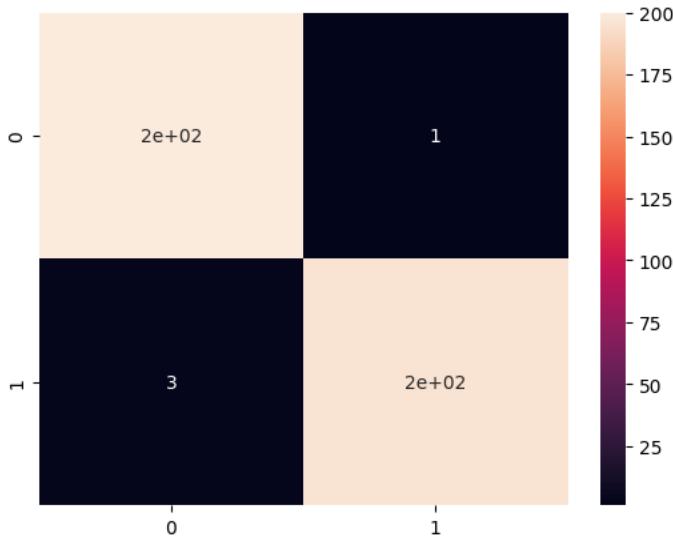
	precision	recall	f1-score	support
0	0.99	1.00	0.99	201
1	0.99	0.98	0.99	199
accuracy			0.99	400
macro avg	0.99	0.99	0.99	400
weighted avg	0.99	0.99	0.99	400

Hình 3.37: classification\_report K – NN

- Accuracy (Độ chính xác tổng thể): 99%
- Macro average F1-score: 0.99 → Hiệu suất cao và đồng đều ở cả hai lớp.
- Weighted average F1-score: 0.99 → Rất ổn định, phù hợp với dữ liệu cân bằng.

Nhận xét:

- Mô hình K-NN cho hiệu quả gần như tuyệt đối trên bài toán phân loại tiền thật giả.
- Với recall đạt 1.00 ở lớp tiền giả (0), mô hình không bỏ sót trường hợp nào là tiền giả, điều này cực kỳ quan trọng trong ứng dụng thực tế.
- F1-score đồng đều giữa hai lớp cho thấy mô hình không bị thiên lệch, xử lý tốt cả hai nhãn.
- Có thể tin tưởng sử dụng mô hình K-NN này trong hệ thống kiểm tra tiền thật giả tự động.



Hình 3.38: Confusion\_Matrix K - NN

Nhận xét:

- Dự đoán đúng:
  - 201 tờ tiền giả (class 0) được nhận diện đúng.
  - 196 tờ tiền thật (class 1) được nhận diện đúng.
- Dự đoán sai:
  - 1 tờ tiền giả bị nhầm thành tiền thật.
  - 3 tờ tiền thật bị nhầm thành tiền giả.

**Kết luận:**

- Mô hình rất chính xác và đáng tin cậy, với chỉ 4 dự đoán sai trên tổng số 400 mẫu.
- Việc chỉ có 1 false positive (nhầm tiền giả thành thật) và 3 false negatives là mức sai lệch cực thấp trong ứng dụng thực tế.
- Mô hình K-NN cho thấy hiệu quả đồng đều giữa các lớp và hoàn toàn có thể sử dụng trong hệ thống kiểm tra tiền.

#### 3.2.4. Bài toán 2: Phát hiện tiền giả bằng Hồi Quy Logistic

Sử dụng mô hình hồi quy Logistic để phân loại tiền thật (1) và giả (0) dựa trên các đặc trưng vật lý của tờ tiền, bao gồm: Kích thước, chiều dài, độ lệch mép,... Dữ liệu được lấy từ tập fake\_bills.csv

## Huấn luyện mô hình Logistic Regression

```
lr = LogisticRegression()
best_lr = GridSearchCV(lr,param_grid={'penalty': ['l1','l2','elasticnet','None'],
                                         'dual':[True,False],
                                         'C':[1,1.5,2,2.5,3,3.5,4,4.5,5],
                                         'solver':['lbfgs','liblinear','newton-cg','newton-cholesky','sag','saga'],
                                         'max_iter': [5,7]},cv=3,n_jobs=-1)
best_lr.fit(x_train,y_train)
```

Hình 3.39: Huấn luyện mô hình

- Sử dụng GridSearchCV để tìm siêu tham số tối ưu với các tham số:
  - Penalty: ['l1','l2','elasticnet', None]
  - Dual: [True, False]
  - C: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]
  - solver: ['lbfgs', 'liblinear', 'newton-cg', 'saga'...]
  - max\_iter: [5, 7]
- Mô hình được huấn luyện với các siêu tham số tốt nhất từ grid search và đánh giá bằng classification\_report

	precision	recall	f1-score	support
0	0.99	0.97	0.98	201
1	0.97	0.99	0.98	199
accuracy			0.98	400
macro avg	0.98	0.98	0.98	400
weighted avg	0.98	0.98	0.98	400

Hình 3.40: classification\_report Logistic Regression

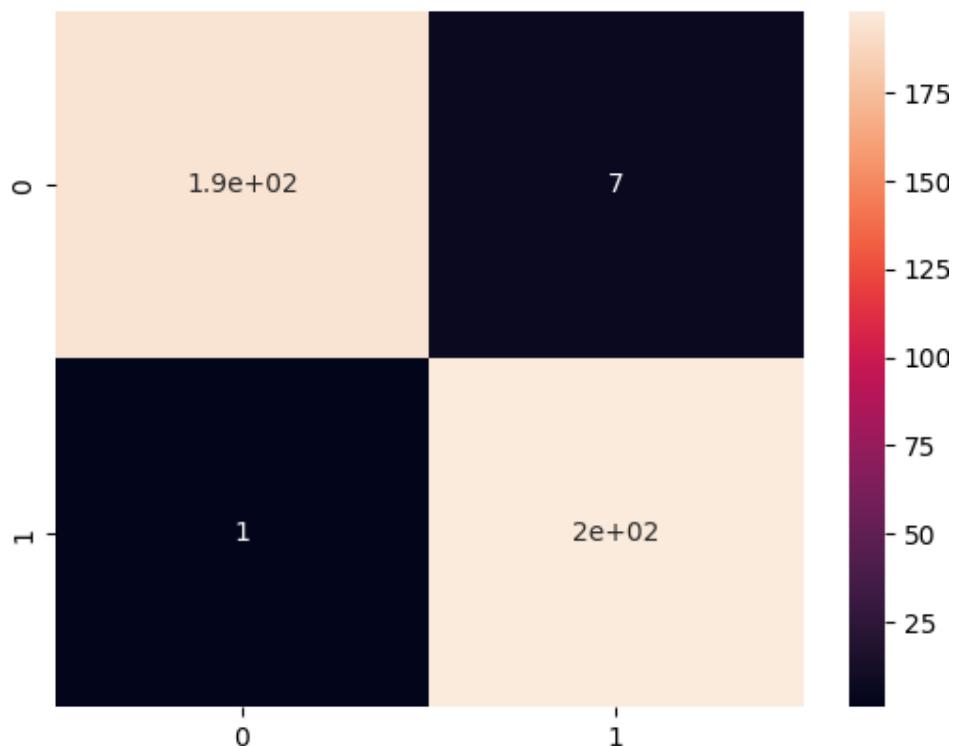
Nhận xét:

- Độ chính xác cao: Mô hình đạt accuracy 98%, rất tốt cho một bài toán phân loại nhị phân.
- F1-score đồng đều cho cả hai lớp (0 và 1), đạt 0.98, cho thấy khả năng phân biệt tốt cả tiền thật và tiền giả.
- Precision & recall cân bằng:

- Lớp 0 (giả): Precision cao (0.99) ít dự đoán nhầm tiền giả thành thật. Recall 0.97 có một vài trường hợp bị bỏ sót.
- Lớp 1 (thật): Recall rất cao (0.99) mô hình nhận diện gần như toàn bộ tiền thật. Precision 0.97 vẫn có chút nhầm lẫn.
- Macro và weighted avg đều là 0.98 → không có hiện tượng mất cân bằng giữa các lớp sau oversampling.

### Kết luận:

Mô hình Logistic Regression đạt hiệu suất rất cao (98%), phù hợp với bài toán phát hiện tiền giả. Đây là mô hình đơn giản, dễ triển khai, dễ giải thích, và hoàn toàn có thể ứng dụng trong thực tế.



Hình 3.41: Confusion\_Matrix Logistic Regression

### Nhận xét:

- Mô hình phân loại chính xác hầu hết các trường hợp, đặc biệt là lớp 1 (Tiền thật).

- Chỉ có 1 mẫu tiền thật bị nhầm thành tiền giả điều này rất tốt vì giúp hạn chế tối đa sai sót loại II (False Negative) vốn là rủi ro cao trong ngành ngân hàng.
- 7 mẫu tiền giả bị nhầm là thật, vẫn còn một chút rủi ro cần giảm thiểu.
- Tổng thể, Confusion Matrix thể hiện mô hình rất cân bằng, tỷ lệ sai thấp → mô hình hoạt động ổn định và đáng tin cậy.

### Kết luận:

Confusion matrix xác nhận rằng Logistic Regression là một mô hình hiệu quả trong bài toán phát hiện tiền giả, với khả năng phân loại chính xác cả hai lớp và tỷ lệ nhầm lẫn thấp.

#### 3.2.5. Tổng kết mô hình

Trong bài toán phát hiện tiền giả từ đặc điểm vật lý của tờ tiền (chiều cao, chiều rộng, đường chéo...), hai mô hình học máy đã được triển khai: K – Nearest Neighbors (KNN) và Hồi quy Logistic (Logistic Regression). Dữ liệu đã được tiền xử lý cẩn thận, bao gồm chuẩn hóa, xử lý thiếu giá trị, và cân bằng lại tập dữ liệu bằng kỹ thuật RandomOverSampler để đảm bảo mô hình học được tốt.

### Hiệu suất mô hình

Mô hình	Accuracy	Precision	Recall	F1-score
KNN	99%	0.99	0.99	0.99
Logistic Regression	98%	0.98	0.98	0.98

### Nhận xét:

- KNN đạt hiệu quả rất cao và ổn định, nhờ khả năng phân loại tốt các điểm gần nhau trong không gian đặc trưng.
- Logistic Regression cũng cho kết quả tốt, dễ giải thích, và có thời gian huấn luyện nhanh.
- Cả hai mô hình đều cho thấy khả năng phân biệt rõ ràng giữa tiền giả và tiền thật với độ chính xác cao.

## Kết luận:

- Cả hai mô hình đều phù hợp để áp dụng trong thực tế nhằm hỗ trợ kiểm tra tính xác thực của tiền giấy.
- Nếu yêu cầu giải thích mô hình rõ ràng hơn, Logistic Regression là lựa chọn tối ưu.
- Nếu ưu tiên độ chính xác cao nhất, mô hình KNN là lựa chọn nên cân nhắc

### 3.3. Dataset 3: Credit Score Classification – Phân loại điểm tín dụng

#### 3.3.1. Giới thiệu về bộ dữ liệu

Bộ dữ liệu được trích xuất từ nguồn Credit Score Classification – Kaggle, bao gồm hai file chính: train.csv và test.csv. Trong đó, tập train.csv chứa 100000 dòng dữ liệu huấn luyện và các biến liên quan.

#### Mục tiêu phân tích:

Mục tiêu của bài toán là phân loại điểm tín dụng (credit score) của một cá nhân dựa trên các thông tin tài chính, nhân khẩu học và hành vi chi tiêu. Đây là một bài toán phân loại đa lớp với 3 nhãn mục tiêu:

- Poor: Điểm tín dụng thấp.
- Standard: Điểm tín dụng trung bình.
- Good: Điểm tín dụng cao

Việc phân loại điểm tín dụng có ý nghĩa quan trọng trong hoạt động cho vay tài chính, thẩm định tín dụng, và ra quyết định của ngân hàng. Hệ thống phân loại chính xác giúp giảm rủi ro cho tổ chức tài chính và tối ưu hóa quy trình chấm điểm tín dụng.

#### Tổng quan về dữ liệu:

Tập huấn luyện (train.csv): gồm hơn 1000 bản ghi, mỗi bản ghi là thông tin về một khách hàng. Các biến trong tập train bao gồm

Biến	Mô tả	Kiểu dữ liệu
Age	Tuổi khách hàng	Số nguyên

Occupation	Nghề nghiệp (giáo viên, kỹ sư, quản lý...)	Phân loại
Annual_Income	Thu nhập hàng năm (USD)	Số thực
Monthly_Inhand_Salary	Thu nhập thực nhận hàng tháng	Số thực
Num_Bank_Accounts	Số tài khoản ngân hàng	Số nguyên
Num_Credit_Card	Số thẻ tín dụng khách hàng sở hữu	Số nguyên
Interest_Rate	Lãi suất áp dụng cho khách hàng (%)	Số nguyên
Num_of_Loan	Số khoản vay đã có	Số nguyên
Type_of_Loan	Danh sách loại vay đang có (vay cá nhân, vay học phí...)	Văn bản phân loại
Delay_from_due_date	Số ngày trễ hạn thanh toán	Số nguyên
Changed_Credit_Limit	Mức thay đổi hạn mức tín dụng	Số thực
Num_Credit_Inquiries	Số lần bị truy vấn tín dụng gần đây	Số nguyên
Credit_Mix	Loại hình tín dụng sử dụng (Bad / Standard / Good)	Phân loại
Outstanding_Debt	Tổng dư nợ	Số thực
Credit_Utilization_Ratio	Tỷ lệ sử dụng tín dụng (%)	Số thực
Credit_History_Age	Thời gian lịch sử tín dụng (ví dụ: 22 Years and 3 Months)	Văn bản
Payment_of_Min_Amount	Có thanh toán số tiền tối thiểu hay không	Phân loại (Yes/No/NM)
Payment_Behaviour	Hành vi thanh toán gần đây (ngày/tháng/kiểu)	Phân loại
Monthly_Balance	Số dư tài khoản hàng tháng	Số thực
Credit_Score	Biến mục tiêu: Poor, Standard, Good	Phân loại (3 lớp)

Tập kiểm tra (test.csv): Tập test.csv có cấu trúc tương tự tập train, gồm đầy đủ 19 biến đầu vào, không bao gồm biến mục tiêu Credit\_Score. Đây là tập dữ liệu được sử dụng để:

- Đánh giá mô hình trên dữ liệu chưa từng thấy
- Kiểm tra độ tổng quát hóa của mô hình
- Dự đoán thực tế sau huấn luyện

Tất cả các bước tiền xử lý và chuẩn hóa trên tập train đều phải được áp dụng đồng nhất lên tập test để đảm bảo tính nhất quán đầu vào cho mô hình.

### Đặc điểm bài toán:

Bài toán phân loại đa lớp, đòi hỏi mô hình phải phân biệt tốt giữa 3 nhóm người dùng với mức độ tín dụng khác nhau.

Dữ liệu gồm cả biến số, biến phân loại và biến văn bản (như Type\_of\_Loan, Payment\_Behaviour) → cần tiền xử lý cẩn thận.

Một số biến có thể cần mã hóa (encoding), tách chuỗi (chuan hóa Credit\_History\_Age) và xử lý thiếu (NM trong cột phân loại).

Có thể áp dụng các mô hình như SVM, KNN để đánh giá khả năng phân loại.

#### 3.3.2. Tiền xử lý dữ liệu

##### Đọc dữ liệu

```
df_train = pd.read_csv(r"C:\Users\Admin\Downloads\train.csv\train.csv")
df_test = pd.read_csv(r"C:\Users\Admin\Downloads\test.csv\test.csv")
```

Hình 3.42: Đọc dữ liệu từ tập huấn luyện và kiểm tra bằng pandas

##### Kiểm tra phân bố nhãn trong tập huấn luyện

```
df_train['Credit_Score'].value_counts()
```

Credit_Score	count
Standard	53174
Poor	28998
Good	17828
Name: count, dtype: int64	

Hình 3.43: Phân bố nhãn Credit\_Score

Nhận xét:

Lớp Standard chiếm đa số (hơn 50%), trong khi Good chỉ chiếm ~17%. Điều này có thể dẫn đến mô hình thiên lệch (bias) nếu không xử lý cân bằng dữ liệu, từ đó ảnh hưởng đến khả năng phân loại chính xác các nhóm khách hàng tốt hoặc kém.

### Gộp tập dữ liệu huấn luyện và kiểm tra

```
df = pd.concat([df_train, df_test], ignore_index=True)  
df.shape  
  
(150000, 28)
```

Hình 3.44: Kích thước dữ liệu sau khi gộp

Để thuận tiện cho quá trình xử lý dữ liệu đồng nhất, hai tập dữ liệu train và test được nối lại thành một DataFrame duy nhất bằng phương thức pd.concat(). Việc này đảm bảo các bước tiền xử lý như chuẩn hóa, mã hóa biến... sẽ nhất quán trên toàn bộ tập dữ liệu. Kết quả sau khi gộp: 150.000 dòng và 28 cột.

### Kiểm tra giá trị thiếu

Sau khi hợp nhất dữ liệu huấn luyện và kiểm tra, nhóm tiến hành thống kê số lượng giá trị thiếu trên toàn bộ dataset (150.000 dòng, 28 cột). Kết quả cho thấy một số cột có lượng missing đáng kể:

- Monthly\_Inhand\_Salary: 22500 giá trị thiếu
- Type\_of\_Loan: 17112 giá trị thiếu
- credit\_History\_Age: 13500 giá trị thiếu
- Name: 15000 giá trị thiếu
- Num\_of\_Delayed\_Payment: 10500 giá trị thiếu
- Amount\_invested\_monthly: 6750 giá trị thiếu
- Num\_Credit\_Inquiries: 3000 giá trị thiếu
- Monthly\_Balance: 1762 giá trị thiếu
- Credit\_Score: 50000 giá trị thiếu của tập test

Nhận xét:

Nhiều cột có tỷ lệ thiếu trên 10%, đặc biệt là các biến liên quan đến tài chính cá nhân.

Cột Credit\_Score bị thiếu toàn bộ trong tập kiểm tra (Test set), đây là nhãn mục tiêu cần được dự đoán.

Những cột như Name, Monthly\_Inhand\_Salary, Type\_of\_Loan có thể xem xét xử lý bằng cách:

- Điền giá trị trung bình/mode (nếu là số học hoặc phân loại rõ ràng)
- Loại bỏ hoặc giữ lại tùy theo vai trò của biến và tỷ lệ thiếu.

df['Type_of_Loan'].value_counts(dropna=False)	
✓	0.0s
Type_of_Loan	
NaN	17112
Not Specified	2112
Credit-Builder Loan	1920
Personal Loan	1908
Debt Consolidation Loan	1896
...	...
Not Specified, Mortgage Loan, Auto Loan, and Payday Loan	12
Payday Loan, Mortgage Loan, Debt Consolidation Loan, and Student Loan	12
Debt Consolidation Loan, Auto Loan, Personal Loan, Debt Consolidation Loan, Student Loan, and Credit-Builder Loan	12
Student Loan, Auto Loan, Student Loan, Credit-Builder Loan, Home Equity Loan, Debt Consolidation Loan, and Debt Consolidation Loan	12
Personal Loan, Auto Loan, Mortgage Loan, Student Loan, and Student Loan	12
Name: count, Length: 6261, dtype: int64	

Hình 3.45: Tần suất giá trị trong biến Type\_of\_Loan (gồm cả NaN)

- Thể hiện rõ cột Type\_of\_Loan có nhiều giá trị rỗng (ghép chuỗi, sai chuẩn).
- Có giá trị NaN (missing) → cần xử lý,
- Nhiều giá trị như "Debt Consolidation Loan, Student Loan, ...", "Not Specified"... → cho thấy dữ liệu cần được chuẩn hóa và tách riêng trước khi mô hình hóa.

```
def text_cleaning(data):
    if data is np.Nan or not isinstance(data, str):
        return data
    else:
        return str(data).strip('_ ,''')

✓ 0.0s

df = df.applymap(text_cleaning).replace([' ', 'nan', '!@9#%8', '#F%$D@*&8'], np.Nan)
```

Hình 3.46: Hàm text\_cleaning() chuẩn hóa văn bản, xoá ký tự đặc biệt và khoảng trắng thừa

ID	Customer ID	Month	Name	Age	SSN	Occupation
0	CUS_0x002	CUS_0x040	January	Aaron Maeshoh	23	821-00-0265
1	0x1003	CUS_0x040	February	Aaron Maeshoh	23	821-00-0265
2	0x1004	CUS_0x040	March	Aaron Maeshoh	>50	821-00-0265
3	0x1005	CUS_0x040	April	Aaron Maeshoh	23	821-00-0265
4	0x1006	CUS_0x040	May	Aaron Maeshoh	23	821-00-0265

Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	Type_of_Loan
19114.12	1824.8433333333328	3	4	3	4	Auto Loan, Credit-Builder Loan, Pers...
19114.12	Missing value	3	4	3	4	Auto Loan, Credit-Builder Loan, Pers...
19114.12	Missing value	3	4	3	4	Auto Loan, Credit-Builder Loan, Pers...
19114.12	Missing value	3	4	3	4	Auto Loan, Credit-Builder Loan, Pers...
19114.12	1824.8433333333328	3	4	3	4	Auto Loan, Credit-Builder Loan, Pers...

Delay_from_due_date	Num_of_Delayed_Payments	Changed_Credit_Limit	Num_Credit_Inquiries	Credit_Mix	Outstanding_Debt	Credit_Utilization_Ratio
3	7	11.27	4.0	Missing value	809.98	26.82261962699016
-1	Missing value	11.27	4.0	Good	809.98	31.944964005538421
3	7	Missing value	4.0	Good	809.98	28.60935202206993
5	4	6.27	4.0	Good	809.98	31.377861869582357
6	Missing value	11.27	4.0	Good	809.98	24.79734908846982

Credit_History_Age	Payment_of_Min_Amount	Total_EMIs_per_month	Amount_invested_month...	Payment_Behaviour	Monthly_Balance	Credit_Score
22 Years and 1 Months	No	49.574049021489417	80.4152954300253	High_spent_Small_value_payments	312.49403867943663	Good
Missing value	No	49.574049021489417	118.28022162236739	Low_spent_Large_value_payments	284.62916240607184	Good
22 Years and 3 Months	No	49.574049021489417	81.699521264648	Low_spent_Medium_value_payments	331.2096928537912	Good
22 Years and 4 Months	No	49.574049021489417	199.4507043910713	Low_spent_Small_value_payments	223.45130072736786	Good
22 Years and 5 Months	No	49.574049021489417	41.420153086217326	High_spent_Medium_value_payment	341.48923165222177	Good

Hình 3.47: Kết quả dữ liệu sau khi làm sạch – các ký tự rác và chuỗi không hợp lệ đã được thay thế bằng NaN

Bộ dữ liệu gồm 28 cột với thông tin đa dạng từ định danh khách hàng, đặc điểm nhân khẩu học, hành vi tài chính đến lịch sử tín dụng. Một số biến như Monthly\_Inhand\_Salary, Type\_of\_Loan, Credit\_History\_Age... có nhiều giá trị thiếu hoặc không chuẩn hóa. Điều này đòi hỏi bước xử lý dữ liệu kỹ lưỡng để đảm bảo độ tin cậy cho các mô hình phân loại tín dụng ở các bước sau

```

df.select_dtypes(include='O').info()
✓ 0.1s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               150000 non-null   object 
 1   Customer_ID     150000 non-null   object 
 2   Month            150000 non-null   object 
 3   Name              135000 non-null   object 
 4   Age               150000 non-null   object 
 5   SSN               141600 non-null   object 
 6   Occupation       139500 non-null   object 
 7   Annual_Income    150000 non-null   object 
 8   Num_of_Loan      150000 non-null   object 
 9   Type_of_Loan     132888 non-null   object 
 10  Num_of_Delayed_Payment 139500 non-null   object 
 11  Changed_Credit_Limit 146850 non-null   object 
 12  Credit_Mix       120000 non-null   object 
 13  Outstanding_Debt 150000 non-null   object 
 14  Credit_History_Age 136500 non-null   object 
 15  Payment_of_Min_Amount 150000 non-null   object 
 16  Amount_invested_monthly 143250 non-null   object 
 17  Payment_Behaviour   138600 non-null   object 
 18  Monthly_Balance    148238 non-null   object 
 19  Credit_Score        100000 non-null   object 

```

Hình 3.48: Các cột dạng object cần được chuyển đổi kiểu dữ liệu phù hợp

Trước khi xây dựng mô hình, nhóm tiến hành kiểm tra kiểu dữ liệu. Kết quả cho thấy có 20 cột đang ở định dạng object, bao gồm cả các trường dạng số (như Age, Annual\_Income, Credit\_History\_Age, v.v). Điều này yêu cầu phải chuyển đổi về định dạng số hoặc thời gian phù hợp để đảm bảo tính toán và mô hình hóa chính xác.

```

df.info()
✓ 0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               150000 non-null   int64  
 1   Customer_ID      150000 non-null   int64  
 2   Month            150000 non-null   int32  
 3   Name              135000 non-null   object  
 4   Age               150000 non-null   int32  
 5   SSN              141600 non-null   float64 
 6   Occupation        139500 non-null   object  
 7   Annual_Income    150000 non-null   float64 
 8   Monthly_Inhand_Salary 127500 non-null   float64 
 9   Num_Bank_Accounts 150000 non-null   int64  
 10  Num_Credit_Card   150000 non-null   int64  
 11  Interest_Rate     150000 non-null   int64  
 12  Num_of_Loan       150000 non-null   int32  
 13  Type_of_Loan      132888 non-null   object  
 14  Delay_from_due_date 150000 non-null   int64  
 15  Num_of_Delayed_Payment 139500 non-null   float64 
 16  Changed_Credit_Limit 146850 non-null   float64 
 17  Num_Credit_Inquiries 147000 non-null   float64 
 18  Credit_Mix        120000 non-null   object  
 19  Outstanding_Debt  150000 non-null   float64 
 ...
 26  Monthly_Balance   148238 non-null   float64 
 27  Credit_Score       100000 non-null   object  

```

Hình 3.49: Thông tin dataset sau khi chuyển đổi kiểu dữ liệu phù hợp

Sau khi kiểm tra và xác định các cột có kiểu dữ liệu không phù hợp (object), nhóm đã thực hiện chuyển đổi về kiểu số (int, float) hoặc thời gian (datetime) tùy theo ngữ nghĩa. Điều này giúp đảm bảo dữ liệu có thể được sử dụng hiệu quả trong quá trình phân tích và huấn luyện mô hình.

```

df['Name'].value_counts(dropna=False).head()
✓ 0.0s

Name
NaN      15000
Stevez      66
Langep      65
Jessicad      59
Johnc      58
Name: count, dtype: int64

```

Hình 3.50: Phân phối giá trị trong cột Name (bao gồm cả NaN)

Mặc dù tỷ lệ thiếu lớn, nhưng do nhóm không loại bỏ các dòng chứa NaN để tránh làm thay đổi tính chất tổng thể của dữ liệu, nên các giá trị này sẽ được giữ nguyên hoặc xử lý bằng kỹ thuật phù hợp trong bước tiền xử lý tiếp theo.

```

object_col = df.select_dtypes(include='O').columns
object_col
✓ 0.0s └─ Open 'object_col' in Data Wrangler

Index(['Name', 'Occupation', 'Type_of_Loan', 'Credit_Mix',
       'Payment_of_Min_Amount', 'Payment_Behaviour', 'Credit_Score'],
      dtype='object')

```

Hình 3.51: Danh sách các biến dạng object trong tập dữ liệu

Đây là các biến cần được xử lý riêng như mã hóa (Label Encoding, One-hot), hoặc xử lý NaN bằng mode (giá trị xuất hiện nhiều nhất)

```

nec_object_col = ['Occupation', 'Type_of_Loan', 'Credit_Mix', 'Payment_Behaviour']
df['Credit_Score'].unique()
✓ 0.0s

array(['Good', 'Standard', 'Poor', nan], dtype=object)

```

Hình 3.52: Các giá trị duy nhất của biến mục tiêu Credit\_Score

Biến mục tiêu Credit\_Score gồm ba nhóm phân loại: Good, Standard, Poor, kèm theo một số giá trị thiếu (NaN). Đây là biến phân loại đa lớp, làm cơ sở cho các thuật toán phân loại trong mô hình dự báo.

```

def get_iqr_lower_upper(df, columns, multiply=1.5):
    for col in columns:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        qir = q3 - q1

        lower_bound = q1 - 1.5 * qir
        upper_bound = q3 + 1.5 * qir

        meann = df[col].mean()

        df[col] = df[col].apply(lambda x: meann if x < lower_bound or x > upper_bound else x)

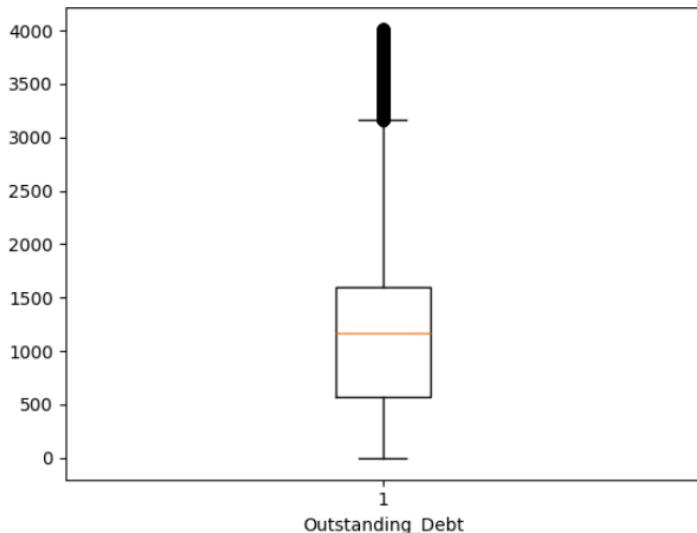
    return df

```

Hình 3.53: Hàm xử lý outlier theo phương pháp IQR – thay thế các giá trị bất thường bằng giá trị trung bình.

Để đảm bảo chất lượng dữ liệu và độ ổn định của mô hình, nhóm sử dụng phương pháp IQR để phát hiện và thay thế outlier (giá trị bất thường) bằng giá trị trung bình của biến.

Cụ thể, các giá trị nằm ngoài khoảng  $[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]$  sẽ được thay thế. Điều này giúp giảm nhiễu, tránh ảnh hưởng đến quá trình huấn luyện mô hình.



Hình 3.54: Boxplot biến Outstanding\_Debt trước xử lý ngoại lệ

Biểu đồ thể hiện sự phân bố của biến Outstanding\_Debt. Có thể thấy một số lượng lớn các điểm dữ liệu vượt xa khỏi ngưỡng phân vị trên (outliers). Những giá trị này nếu không được xử lý có thể gây nhiễu cho mô hình, làm sai lệch kết quả dự

đoán. Do đó, nhóm đã xử lý các giá trị ngoại lệ bằng phương pháp IQR và thay thế bằng giá trị trung bình của biến

Chú thích: Các biểu đồ còn lại được trình bày chi tiết trong Notebook đính kèm.

```
for i in numeric_col:  
    df_copy[i] = df_copy[i].fillna(df_copy[i].mean())  
    ✓ 0.0s  
  
df_copy.isna().sum()  
✓ 0.0s  
ID          0  
Customer_ID 0  
Month        0  
Name         15000  
Age          0  
SSN          0  
Occupation   10500  
Annual_Income 0  
Monthly_Inhand_Salary 0  
Num_Bank_Accounts 0  
Num_Credit_Card 0  
Interest_Rate 0  
Num_of_Loan   0  
Type_of_Loan  17112  
Delay_from_due_date 0  
Num_of_Delayed_Payment 0  
Changed_Credit_Limit 0  
Num_Credit_Inquiries 0  
Credit_Mix    30000  
Outstanding_Debt 0  
Credit_Utilization_Ratio 0  
Credit_History_Age 0  
Payment_of_Min_Amount 0  
Total_EMI_per_month 0  
Amount_invested_monthly 0  
Payment_Behaviour 11400  
Monthly_Balance 0  
Credit_Score    50000  
dtype: int64
```

Hình 3.55: Thống kê số lượng giá trị thiếu sau khi xử lý các biến số

Sau khi xử lý, các biến số không còn giá trị thiếu. Một số biến dạng object vẫn còn NaN và sẽ được xử lý ở bước tiếp theo.

```

for i in object_col:
    df_copy[i] = df_copy[i].fillna(df_copy[i].mode(dropna=True)[0])
✓ 0.0s

df_copy.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 28 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   ID                150000 non-null   int64  
 1   Customer_ID       150000 non-null   int64  
 2   Month             150000 non-null   int64  
 3   Name               135000 non-null   object  
 4   Age                150000 non-null   float64 
 5   SSN               150000 non-null   float64 
 6   Occupation         150000 non-null   object  
 7   Annual_Income     150000 non-null   float64 
 8   Monthly_Inhand_Salary 150000 non-null   float64 
 9   Num_Bank_Accounts 150000 non-null   float64 
 10  Num_Credit_Card   150000 non-null   float64 
 11  Interest_Rate     150000 non-null   float64 
 12  Num_of_Loan        150000 non-null   float64 
 13  Type_of_Loan       150000 non-null   object  
 14  Delay_from_due_date 150000 non-null   float64 
 15  Num_of_Delayed_Payment 150000 non-null   float64 
 16  Changed_Credit_Limit 150000 non-null   float64 
 17  Num_Credit_Inquiries 150000 non-null   float64 
 18  Credit_Mix         150000 non-null   object  
 19  Outstanding_Debt   150000 non-null   float64 
...
26  Monthly_Balance    150000 non-null   float64 
27  Credit_Score        100000 non-null   object  

```

Hình 3.56: Kết quả sau khi điền giá trị thiếu cho các biến dạng object

Tất cả các biến dạng object đã được xử lý giá trị thiếu bằng cách điền mode tức giá trị xuất hiện phổ biến nhất. Phương pháp này giúp duy trì tính phân bố đặc trưng của dữ liệu mà không làm thay đổi bản chất của các biến phân loại.

### Xử lý các trường hợp không hợp lệ

```
df_copy.loc[df_copy['Num_Bank_Accounts'] < 0, 'Num_Bank_Accounts'] = 0
```

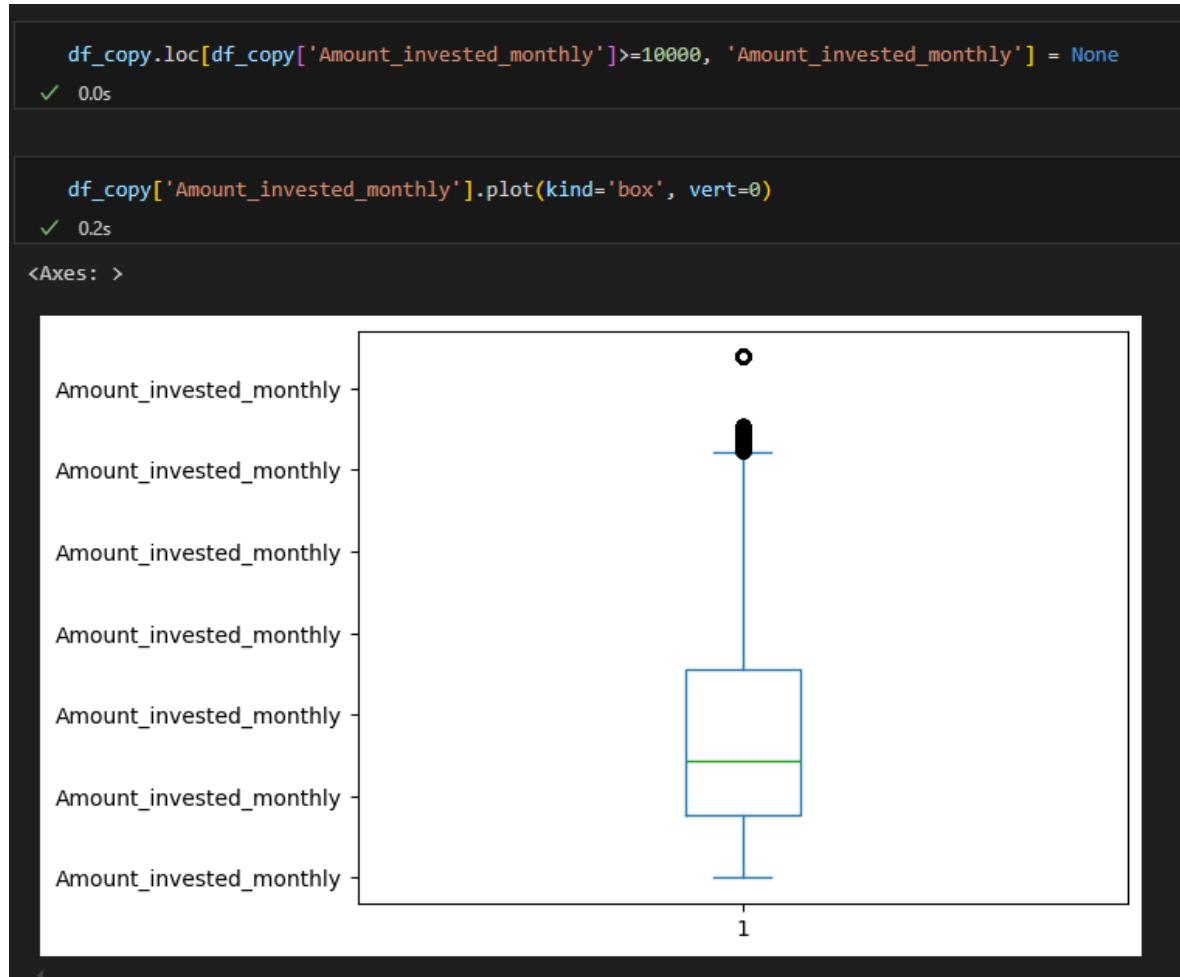
Hình 3.57: Xử lý các giá trị âm không hợp lệ trong cột Num\_Bank\_Accounts bằng cách thay thế bằng 0.

Một số dòng dữ liệu có giá trị âm trong số tài khoản ngân hàng, điều này không thực tế. Nhóm đã thay thế toàn bộ các giá trị này bằng 0 để đảm bảo tính hợp lý và chuẩn hóa dữ liệu đầu vào.

```
df_copy.loc[df_copy['Delay_from_due_date']<0, 'Delay_from_due_date'] = None
```

Hình 3.58: Xử lý các giá trị không hợp lý trong cột Delay\_from\_due\_date bằng cách thay thế các giá trị âm thành NaN

Một số bản ghi có giá trị âm không hợp lý ở cột Delay\_from\_due\_date. Để đảm bảo tính toàn vẹn của dữ liệu, nhóm đã thay thế những giá trị này bằng NaN nhằm xử lý sau bằng phương pháp phù hợp (ví dụ: điền trung bình, loại bỏ,...)



Hình 3.59: ứ lý ngoại lệ cột Amount\_invested\_monthly và biểu đồ hộp sau khi loại bỏ các giá trị bất thường.

Các giá trị từ 10,000 trở lên được xem là bất thường và được gán lại thành NaN. Biểu đồ hộp cho thấy sau xử lý, phần lớn dữ liệu phân bố trong khoảng hợp lý và chỉ còn một vài ngoại lệ nhỏ.

```

temp = ['Delay_from_due_date','Num_of_Delayed_Payment','Monthly_Balance']
for i in temp:
    df_copy[i] = df_copy[i].fillna(df_copy[i].mean())
✓ 0.0s

df_copy.isna().sum()
✓ 0.0s

ID          0
Customer_ID 0
Month        0
Name         15000
Age          0
SSN          0
Occupation   0
Annual_Income 0
Monthly_Inhand_Salary 0
Num_Bank_Accounts 0
Num_Credit_Card 0
Interest_Rate 0
Num_of_Loan   0
Type_of_Loan  0
Delay_from_due_date 0
Num_of_Delayed_Payment 0
Changed_Credit_Limit 0
Num_Credit_Inquiries 0
Credit_Mix    0
Outstanding_Debt 0
Credit_Utilization_Ratio 0
Credit_History_Age 0
Payment_of_Min_Amount 0
Total_EMI_per_month 0
Amount_invested_monthly 0
Payment_Behaviour 0
Monthly_Balance 0
Credit_Score   50000

```

Hình 3.60: Hoàn tất xử lý giá trị thiếu ở các biến số, còn lại chỉ thiếu ở các cột mục tiêu (Name, Credit\_Score).

Sau khi xử lý các trường hợp ngoại lệ và điền giá trị trung bình cho các biến số (Delay\_from\_due\_date, Num\_of\_Delayed\_Payment, Monthly\_Balance), tập dữ liệu hiện tại chỉ còn thiếu ở hai cột: Name và Credit\_Score. Điều này giúp đảm bảo tính toàn vẹn dữ liệu trước khi xây dựng mô hình dự báo.

```

df_copy[df_copy['Credit_Score'].notna()].info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
Index: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               100000 non-null   int64  
 1   Customer_ID      100000 non-null   int64  
 2   Month            100000 non-null   int64  
 3   Name              90015 non-null   object  
 4   Age               100000 non-null   float64 
 5   SSN              100000 non-null   float64 
 6   Occupation        100000 non-null   object  
 7   Annual_Income    100000 non-null   float64 
 8   Monthly_Inhand_Salary 100000 non-null   float64 
 9   Num_Bank_Accounts 100000 non-null   float64 
 10  Num_Credit_Card   100000 non-null   float64 
 11  Interest_Rate    100000 non-null   float64 
 12  Num_of_Loan       100000 non-null   float64 
 13  Type_of_Loan     100000 non-null   object  
 14  Delay_from_due_date 100000 non-null   float64 
 15  Num_of_Delayed_Payment 100000 non-null   float64 
 16  Changed_Credit_Limit 100000 non-null   float64 
 17  Num_Credit_Inquiries 100000 non-null   float64 
 18  Credit_Mix        100000 non-null   object  
 19  Outstanding_Debt  100000 non-null   float64 
 ...
 26  Monthly_Balance   100000 non-null   float64 
 27  Credit_Score       100000 non-null   object  

```

Hình 3.61: Thông tin sau khi lọc dữ liệu có Credit\_Score không bị thiếu (train set)

Tập train gồm 100.000 bản ghi có Credit\_Score không bị thiếu. Các cột quan trọng đều đã được xử lý và không còn giá trị khuyết (null), đảm bảo chất lượng đầu vào cho các mô hình học máy sau này

```
df_copy[df_copy['Credit_Score'].notna()].to_csv("clean_train.csv", index=False)
```

Hình 3.62: Lưu tập dữ liệu huấn luyện gồm 100.000 dòng có nhãn Credit\_Score vào file clean\_train.csv.

```

df_copy[df_copy['Credit_Score'].isna()].info()
  ✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
Index: 50000 entries, 100000 to 149999
Data columns (total 28 columns):
 #   Column           Non-Null Count Dtype  
--- 
 0   ID               50000 non-null  int64  
 1   Customer_ID      50000 non-null  int64  
 2   Month            50000 non-null  int64  
 3   Name              44985 non-null  object  
 4   Age               50000 non-null  float64 
 5   SSN              50000 non-null  float64 
 6   Occupation        50000 non-null  object  
 7   Annual_Income    50000 non-null  float64 
 8   Monthly_Inhand_Salary 50000 non-null  float64 
 9   Num_Bank_Accounts 50000 non-null  float64 
 10  Num_Credit_Card   50000 non-null  float64 
 11  Interest_Rate     50000 non-null  float64 
 12  Num_of_Loan       50000 non-null  float64 
 13  Type_of_Loan      50000 non-null  object  
 14  Delay_from_due_date 50000 non-null  float64 
 15  Num_of_Delayed_Payment 50000 non-null  float64 
 16  Changed_Credit_Limit 50000 non-null  float64 
 17  Num_Credit_Inquiries 50000 non-null  float64 
 18  Credit_Mix        50000 non-null  object  
 19  Outstanding_Debt   50000 non-null  float64 
 ...
 26  Monthly_Balance   50000 non-null  float64 
 27  Credit_Score       0 non-null      object  

```

Hình 3.63: Thông tin tập kiểm tra gồm 50.000 dòng không có nhãn Credit\_Score, được sử dụng để dự đoán sau huấn luyện.

```
df_copy[df_copy['Credit_Score'].isna()].drop(columns='Credit_Score').to_csv("clean_test.csv", index=False)
```

Hình 3.64: Tách dữ liệu test từ các bản ghi không có nhãn Credit\_Score và lưu thành file clean\_test.csv để phục vụ dự đoán sau này.

Tập dữ liệu được chia thành hai phần: tập train gồm 100.000 bản ghi có nhãn Credit\_Score, và tập test gồm 50.000 bản ghi chưa có nhãn. Các cột định danh và thông tin không cần thiết được loại bỏ để chuẩn hóa đầu vào cho mô hình.

### Xử lý dummies vs catagorical vs chọn Feature

Các cột dạng phân loại (categorical) được mã hóa thành biến nhị phân (one-hot encoding) để thuận tiện cho các mô hình học máy. Sau đó, loại bỏ các cột không cần thiết như ID, Customer\_ID, Name, Month, SSN vì không có giá trị dự đoán hoặc trùng lặp thông tin

```

class GetDummies(BaseEstimator, TransformerMixin):
    def __init__(self, data_sep=',', col_name_sep='_'):
        """
        Transformer that creates dummy variables from categorical columns with a separator.
        Parameters:
        - data_sep (str): Separator used to split categorical values into multiple dummy variables.
        - col_name_sep (str): Separator used to separate the column name from the prefix in the output column names.
        """
        self.data_sep = data_sep
        self.col_name_sep = col_name_sep

```

Hình 3.65: Khởi tạo đối tượng GetDummies với các tham số cấu hình định dạng phân tách dữ liệu phân loại nhiều nhãm.

Phương thức `__init__` cho phép tùy chỉnh ký tự phân cách giữa các giá trị (`data_sep`) và định dạng tiền tố cho tên cột (`col_name_sep`) khi tạo các biến giả từ cột phân loại chứa nhiều nhãm. Đây là bước tiền xử lý quan trọng giúp mô hình hóa dữ liệu phân loại phức tạp.

```

# Return self nothing else to do here
def fit(self, X, y = None):
    """
    Fit the transformer to the data.
    Parameters:
    - X (pandas.DataFrame): Input data with categorical columns.
    - y (array-like): Target variable (ignored).
    Returns:
    - self: Returns the transformer object.
    """
    object_cols = X.select_dtypes(include="object").columns
    self.dummy_cols = [col for col in object_cols if X[col].str.contains(self.data_sep, regex=True).any()]
    self.dummy_prefix = [''.join(map(lambda x: x[0], col.split(self.col_name_sep))) if self.col_name_sep in col else col[:2] for col in self.dummy_cols]

    for col, pre in zip(self.dummy_cols, self.dummy_prefix):
        dummy_X = X.join(X[col].str.get_dummies(sep=self.data_sep).add_prefix(pre+self.col_name_sep))

    dummy_X.drop(columns = self.dummy_cols, inplace=True)
    self.columns = dummy_X.columns
    return self

```

Hình 3.66: Hàm `fit()` xác định các cột nhiều nhãm và chuẩn bị prefix cho biến giả.

Hàm `fit()` giúp trích xuất danh sách các cột cần xử lý và chuẩn bị prefix cho tên các biến giả. Nó sử dụng kỹ thuật tách chuỗi dựa trên ký tự phân cách (`data_sep`) để phát hiện những cột chứa nhiều giá trị phân loại trong một ô, từ đó lưu cấu trúc để áp dụng trong bước `transform`. Đây là bước huấn luyện ban đầu, định hình cách biến đổi dữ liệu.

```

# Transformer method we wrote for this transformer
def transform(self, X, y = None):
    """
    Transform the input data by creating dummy variables.
    Parameters:
        - X (pandas.DataFrame): Input data with categorical columns.
        - y (array-like): Target variable (ignored).
    Returns:
        - X_transformed (pandas.DataFrame): Transformed data with dummy variables.
    """
    for col, pre in zip(self.dummy_cols, self.dummy_prefix):
        X_transformed = X.join(X[col].str.get_dummies(sep=self.data_sep).add_prefix(pre+self.col_name_sep))

    X_transformed = X_transformed.reindex(columns=self.columns, fill_value=0)
    return X_transformed

```

Hình 3.67: Hàm `transform()` chuyển đổi dữ liệu đầu vào thành các biến giả theo định dạng đã học từ hàm `fit()`.

Hàm `transform()` giúp chuẩn hóa các cột categorical phức hợp bằng cách tạo biến giả nhất quán theo định dạng đã học trước đó. Việc `reindex` đảm bảo mọi dữ liệu đầu vào đều có cùng định dạng đầu ra, một bước rất quan trọng trong pipeline xử lý dữ liệu cho mô hình học máy.

```

# to get feature names
def get_feature_names_out(self, input_features=None):
    """
    Get the names of the transformed features.
    Parameters:
        - input_features (array-like): Names of the input features (ignored).
    Returns:
        - output_features (list): Names of the transformed features.
    """
    return self.columns.tolist()

```

Hình 3.68: Hàm `get_feature_names_out()` trả về danh sách tên các đặc trưng sau khi đã biến đổi (dummies).

Hàm này rất hữu ích khi cần kiểm tra hoặc gán tên các đặc trưng trong pipeline, đặc biệt khi làm việc với `ColumnTransformer` hay xuất kết quả sau quá trình học máy.

```

df["Credit_Score"].value_counts(normalize=True).sort_index()

✓ 0.0s

Credit_Score
Good      0.17828
Poor      0.28998
Standard   0.53174
Name: proportion, dtype: float64

```

Hình 3.69: Tỷ lệ phân phối các nhãn trong biến mục tiêu `Credit_Score`

Tập dữ liệu mêt cân bằng nhẹ, trong đó nhóm Standard chiếm hơn 53%, trong khi Good chỉ khoảng 17.8%. Điều này cần được cân nhắc khi chọn mô hình và đánh giá độ chính xác.

```
dummy = GetDummies()
df_new = dummy.fit_transform(df)
```

Hình 3.70: Áp dụng transformer GetDummies để tạo biến giả từ các biến phân loại có chứa nhiều giá trị

```
final_obj = df_new.select_dtypes(include='O').columns.to_list()
final_obj
✓ 0.0s └─ Open 'final_obj' in Data Wrangler
['Occupation',
 'Credit_Mix',
 'Payment_of_Min_Amount',
 'Payment_Behaviour',
 'Credit_Score']
```

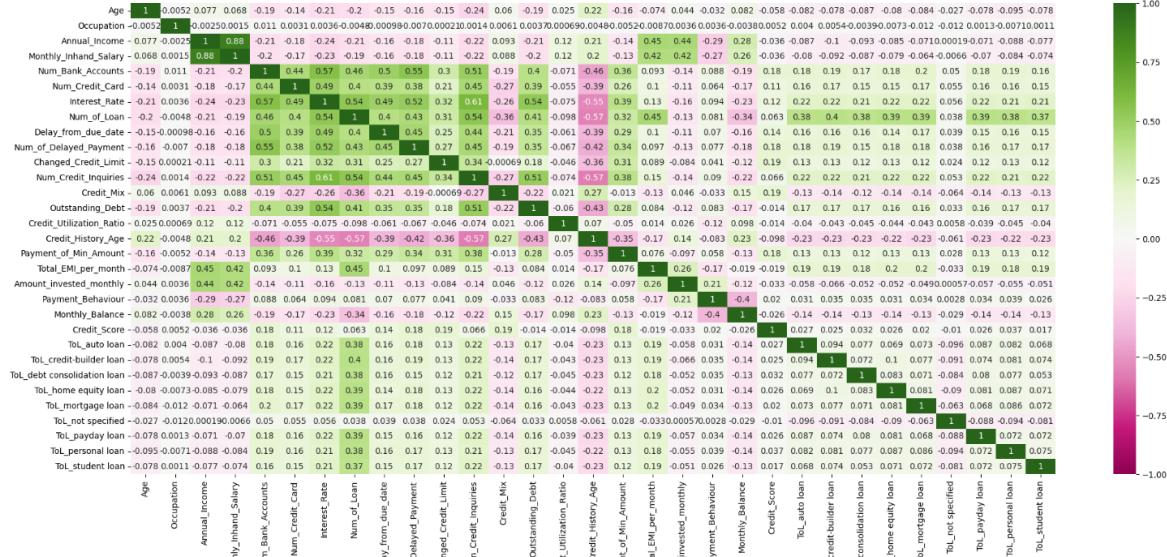
Hình 3.71: Các biến phân loại còn lại chưa được xử lý sau khi áp dụng GetDummies

Sau khi sử dụng GetDummies, vẫn còn một số biến phân loại đơn giá trị (ví dụ: 'Occupation', 'Credit\_Score') không chứa dấu phân cách, do đó không bị tách thành các biến giả.

```
df_new.isna().sum()
✓ 0.0s
Age          0
Occupation   0
Annual_Income 0
Monthly_Inhand_Salary 0
Num_Bank_Accounts 0
Num_Credit_Card 0
Interest_Rate 0
Num_of_Loan 0
Delay_from_due_date 0
Num_of_Delayed_Payment 0
Changed_Credit_Limit 0
Num_Credit_Inquiries 0
Credit_Mix 0
Outstanding_Debt 0
Credit_Utilization_Ratio 0
Credit_History_Age 0
Payment_of_Min_Amount 0
Total_EMIs_per_month 0
Amount_invested_monthly 0
Payment_Behaviour 0
Monthly_Balance 0
Credit_Score      50000
Tol_auto loan    0
Tol_credit-builder loan 0
Tol_debt consolidation loan 0
...
Tol_not specified 0
Tol_payday loan   0
Tol_personal loan 0
Tol_student loan   0
dtype: int64
```

Hình 3.72: Giá trị thiếu sau khi biến đổi dữ liệu với GetDummies

Sau khi áp dụng GetDummies, toàn bộ các biến dummies đều không còn giá trị thiêus. Duy chỉ còn cột Credit\_Score có 50.000 giá trị thiêus – đây là dữ liệu mục tiêu được tách ra phục vụ cho việc dự đoán, cần được xử lý riêng.



Hình 3.73: Ma trận tương quan giữa các biến đầu vào và biến mục tiêu Credit\_Score

Biến Credit\_Score có tương quan dương nhẹ với Annual\_Income, Monthly\_Inhand\_Salary, Num\_Bank\_Accounts, um\_Credit\_Card và Interest\_Rate.

Các biến Credit\_History\_Age, Outstanding\_Debt, Payment\_of\_Min\_Amount cũng có mối liên hệ rõ ràng với Credit\_Score.

Một số biến có tương quan cao với nhau như Annual\_Income và Monthly\_Inhand\_Salary (hệ số ~0.88) → có thể cân nhắc loại bỏ 1 trong 2 để giảm đa cộng tuyến.

Các dummy variable như Tol\_\* hầu hết không có tương quan mạnh → có thể giữ lại hoặc loại bỏ tùy vào kết quả mô hình hóa.

```

no_features = ['Tol_not specified', 'Occupation', 'Credit_Utilization_Ratio', 'Age']

0.0s

alt_df = df_new.copy()
official_df = alt_df.drop(no_features, axis=1)
official_df

```

Hình 3.74: Tạo tập dữ liệu chính thức bằng cách loại bỏ các đặc trưng không cần thiết

Việc loại bỏ các đặc trưng dư thừa giúp giảm nhiễu và tăng hiệu quả huấn luyện của mô hình. Tập offical\_df sau xử lý là tập được sử dụng cho các bước tiếp theo như chia tập train/test, huấn luyện mô hình và đánh giá.

```
x = offical_df.drop(columns="Credit_Score")
y = offical_df['Credit_Score']
df_test = pd.read_csv(r"C:\Users\Admin\Downloads\clean_test.csv")
df_test.drop(columns=['ID', 'Customer_ID', 'Month', 'Name', 'SSN'], inplace=True)
X_test = df_test

X_train, X_val, y_train, y_val = train_test_split(x, y, test_size=0.2, random_state=42)

X_train.shape, y_train.shape, X_val.shape, y_val.shape, X_test.shape,
((80000, 26), (80000,), (20000, 26), (20000,), (50000, 22))
```

Hình 3.75: Kết quả chia dữ liệu huấn luyện (Train), kiểm định (Validation) và tập kiểm tra (Test).

- Dữ liệu đã được chia đúng tỉ lệ (80% train, 20% val).
- Tập X\_test có 22 đặc trưng (khác train vì đã loại cột đích Credit\_Score).
- Việc chia dữ liệu như trên là điều kiện cần để huấn luyện mô hình học máy hiệu quả và đánh giá khách quan.

```
scaler = MinMaxScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_val_scaled = pd.DataFrame(scaler.fit_transform(X_val), columns=X_val.columns)
```

Hình 3.76: Chuẩn hóa dữ liệu X\_train và X\_val bằng MinMaxScaler về khoảng [0, 1]

Sau khi áp dụng GetDummies cho df\_test, cần đảm bảo xử lý đồng bộ các cột với tập huấn luyện.

Sử dụng MinMaxScaler giúp các đặc trưng được đưa về cùng khoảng giá trị, thường là [0, 1], hỗ trợ hiệu quả cho các thuật toán như KNN, SVM

### 3.3.3. Bài toán 1: Phân loại điểm tín dụng bằng SVM (Support Vector Machine)

Dự đoán phân loại điểm tín dụng của khách hàng theo ba nhóm *Poor*, *Standard*, *Good* dựa trên các đặc điểm hành vi tài chính và hồ sơ cá nhân, sử dụng mô hình SVM với tối ưu siêu tham số bằng GridSearchCV.

#### Quy trình thực hiện

```
model_svm = SVC()
```

Hình 3.77: Khởi tạo model

- Sử dụng mô hình SVC() từ thư viện sklearn.svm.

```
best_svm = GridSearchCV(model_svm,param_grid={'C':[1,1.5,2,2.5,3,3.5,4,4.5,5],  
                                              'kernel':['linear','poly','rbf'],  
                                              'degree': [3,4,5,6,7,8],  
                                              'max_iter': [5,8],  
                                              'decision_function_shape':['ovo','ovr']},cv=5,scoring='accuracy')  
  
best_svm.fit(X_train_scaled,y_train)
```

Hình 3.78: Tối ưu hóa tham số

- Dùng GridSearchCV với tập hợp các giá trị thử nghiệm cho các tham số:
  - C từ 1 đến 5.5
  - Kernel: linear, poly, rbf
  - Max\_iter:5,8
  - decision\_function\_shape: ovo, ovr

```
model_svm = SVC(**best_svm.best_params_)  
model_svm.fit(X_train_scaled,y_train)
```

```
SVC(C=2, decision_function_shape='ovo', degree=8, kernel='poly', max_iter=8)
```

Hình 3.79: Huấn luyện mô hình tối ưu

- Huấn luyện mô hình tối ưu: Sau khi tìm được siêu tham số tốt nhất, khởi tạo lại SVC với các tham số này và tiến hành huấn luyện trên tập huấn luyện.

## Đánh giá mô hình

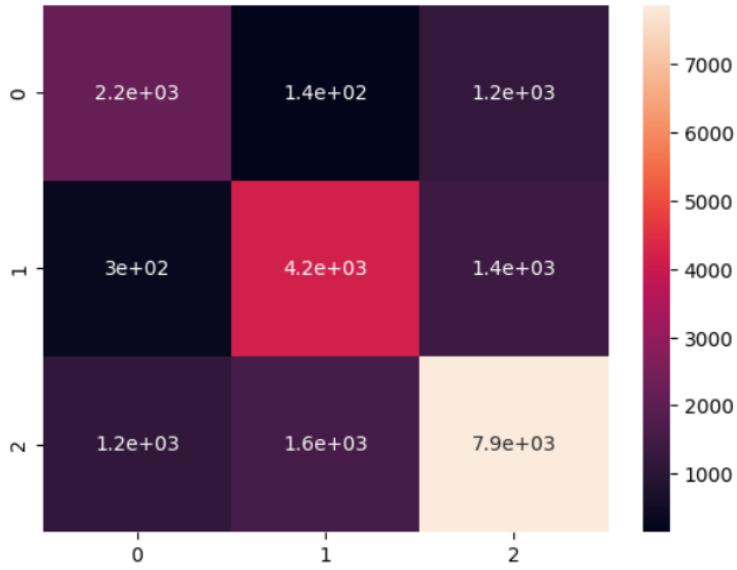
- Sử dụng classification\_report để đánh giá độ chính xác từng lớp

	<code>svm_pred = model_svm.predict(X_val_scaled)</code>
	<code>print(classification_report(y_pred=svm_pred,y_true=y_val))</code>
<hr/>	
	precision    recall    f1-score    support
0.0	0.03    0.01    0.02    3527
1.0	0.60    0.12    0.19    5874
2.0	0.54    0.90    0.68    10599
accuracy	0.51    20000
macro avg	0.39    0.34    0.30    20000
weighted avg	0.47    0.51    0.42    20000

Hình 3.80: Đánh giá mô hình

### Nhận xét:

- Accuracy (độ chính xác tổng thể):
  - Mô hình đạt 51% độ chính xác mức trung bình cho bài toán phân loại ba lớp. Tuy nhiên, vẫn còn nhiều không gian để cải thiện.
- F1-score theo từng lớp:
  - Lớp 0 (Poor): Precision và recall rất thấp (0.03 và 0.01), cho thấy mô hình hầu như không nhận diện được nhóm này.
  - Lớp 1 (Standard): Có recall khá thấp (0.12), dễ bị nhầm sang lớp khác. Precision 0.60 cho thấy có một phần dự đoán đúng nhưng chưa ổn định.
  - Lớp 2 (Good): Mô hình phân biệt tốt nhất nhóm này với precision 0.54, recall 0.90 và F1-score cao nhất 0.68.
- Macro vs Weighted average:
  - Macro avg:  $F1 = 0.30 \rightarrow$  mô hình thiên lệch giữa các lớp, đặc biệt là lớp nhỏ như *Poor* bị bỏ qua.
  - Weighted avg:  $F1 = 0.42 \rightarrow$  cao hơn do phụ thuộc vào lớp Good chiếm đa số.
- Dùng confusion\_matrix để trực quan hóa số lượng dự đoán đúng/sai cho từng nhãn



Hình 3.81: Confusion Matrix mô hình SVM

### Nhận xét:

- Điểm mạnh
  - Mô hình hoạt động tốt nhất ở lớp 2 (Good):
    - Dự đoán đúng 7.900 trong tổng 10.599 mẫu → Recall ~ 0.75
  - Lớp 1 (Standard) cũng đạt kết quả tương đối ổn với 4.200 dự đoán đúng trong hơn 5.874 mẫu
- Hạn chế:
  - Lớp 0 (Poor) có độ chính xác rất thấp:
    - 1.200 mẫu bị nhầm sang lớp 2, và 140 sang lớp 1 → dẫn đến recall chỉ ~0.01.
  - Mô hình thiên lệch dự đoán về lớp Good, do dữ liệu mất cân bằng.

### Tổng quan về hiệu xuất:

- Accuracy: 51% – chỉ vừa trên mức ngẫu nhiên (random).
- Precision & recall thấp với lớp 0 → mô hình khó phát hiện khách hàng có điểm tín dụng thấp, điều này không an toàn trong thực tế ngân hàng.

### 3.3.4. Bài toán 2: K – Nearest Neighbors (K - NN)

Dự đoán phân loại điểm tín dụng của khách hàng theo ba nhóm *Poor*, *Standard*, *Good* dựa trên các đặc điểm hành vi tài chính và hồ sơ cá nhân, sử dụng mô hình KNN.

#### Quy trình thực hiện

```
model_knn.fit(X_train_scaled,y_train)

+ KNeighborsClassifier ⓘ ?
KNeighborsClassifier()

knn_pred = model_knn.predict(X_val_scaled)
print(classification_report(y_pred=knn_pred,y_true=y_val))
```

Hình 3.82: Khởi tạo và huấn luyện mô hình

- Sử dụng KNeighborsClassifier() từ thư viện sklearn.neighbors.
- Mô hình được huấn luyện với dữ liệu đầu vào đã được scale bằng MinMaxScaler

#### Đánh giá mô hình

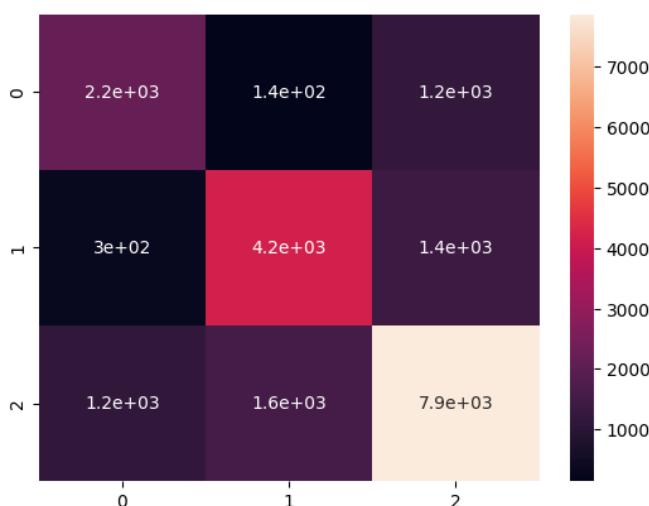
	precision	recall	f1-score	support
0.0	0.60	0.62	0.61	3527
1.0	0.71	0.71	0.71	5874
2.0	0.75	0.74	0.75	10599
accuracy			0.71	20000
macro avg	0.69	0.69	0.69	20000
weighted avg	0.71	0.71	0.71	20000

Hình 3.83: Classification Report

#### Nhận xét:

- Accuracy (Độ chính xác tổng thể):

- Mô hình đạt độ chính xác 71%, cao hơn rõ rệt so với mô hình SVM trước đó (51%), cho thấy KNN có khả năng phân biệt các nhóm tín dụng tốt hơn trong tập dữ liệu này.
- F1-score theo từng lớp:
  - Lớp 0 (Poor):
    - Precision = 0.60, Recall = 0.62, F1-score = 0.61 → Mô hình đã cải thiện rõ rệt khả năng nhận diện nhóm khách hàng rủi ro thấp so với SVM,
  - Lớp 1 (Standard):
    - Precision và Recall đều đạt 0.71 → Dự đoán ổn định và cân bằng.
  - Lớp 2 (Good):
    - Hiệu suất cao nhất với Recall = 0.75 và F1-score = 0.75 → Mô hình nhận diện tốt nhóm khách hàng uy tín.
  - Macro vs Weighted Average:
    - Macro avg F1 = 0.69 → Mô hình thể hiện hiệu suất đồng đều giữa các lớp.
    - Weighted avg F1 = 0.71 → Phản ánh mô hình tốt ngay cả với dữ liệu phân bố không đều



Hình 3.84: Confusion Matrix

### Nhận xét:

- Ưu điểm:
  - Phân loại cân bằng và hiệu quả giữa cả 3 lớp.
  - Đặc biệt, lớp Poor không còn bị bỏ sót như trong mô hình SVM trước đó.
- Hạn chế:
  - Một số nhầm lẫn giữa Standard và Good vẫn xảy ra → có thể cải thiện bằng cách tối ưu K hoặc dùng weights = 'distance'

### Tổng quan về hiệu suất mô hình

Mô hình KNN đạt độ chính xác 71% thể hiện khả năng phân loại tốt trên cả ba ba nhóm điểm tín dụng. Đặc biệt, F1-score của từng lớp đều dao động từ 0.61 đến 0.75, cho thấy độ ổn định và cân bằng. So với SVM, KNN xử lý tốt hơn nhóm Poor và Standard, là lựa chọn phù hợp hơn trong bối cảnh dữ liệu có sự mất cân bằng giữa các lớp

#### 3.3.5. Tổng kết mô hình

Trong phần 3.3, hai mô hình phân loại điểm tín dụng khách hàng đã được xây dựng và so sánh: Support Vector Machine (SVM) và K-Nearest Neighbors (KNN). Mục tiêu là phân loại khách hàng vào ba nhóm: *Poor*, *Standard*, *Good* dựa trên đặc điểm hành vi tài chính và hồ sơ cá nhân,

#### Kết quả so sánh

Tiêu chí	SVM	KNN
Accuracy	51%	71%
F1-score lớp <i>Poor</i> (0)	0.02	0.61
F1-score lớp <i>Standard</i> (1)	0.20	0.71
F1-score lớp <i>Good</i> (2)	0.68	0.75
Macro F1-score	0.30	0.69

Weighted F1-score	0.42	0.71
-------------------	------	------

### Nhận xét:

- Mô hình KNN vượt trội hơn rõ rệt về độ chính xác tổng thể và khả năng phân biệt giữa các nhóm tín dụng.
- SVM có độ chính xác thấp, đặc biệt kém hiệu quả trong việc phát hiện nhóm khách hàng Poor, tiềm ẩn rủi ro cao nếu áp dụng thực tế.
- Cả hai mô hình đều bị ảnh hưởng bởi sự mất cân bằng dữ liệu, tuy nhiên KNN vẫn thể hiện khả năng tổng quát tốt hơn.

### Đề xuất:

- Với dữ liệu hiện tại, KNN là mô hình phù hợp hơn để triển khai bước đầu.
- Có thể cân nhắc các mô hình nâng cao như Random Forest, XGBoost hoặc LightGBM để so sánh thêm.
- Nên áp dụng các kỹ thuật cân bằng dữ liệu (như SMOTE) và thử nghiệm thêm tối ưu siêu tham số để tăng hiệu suất mô hình

## 3.4. Dataset 4: Phân loại cam tốt và cam xấu bằng màng nơ – ron nhân tạo

### 3.4.1. Giới thiệu bài toán và dữ liệu

Bộ dữ liệu được trích xuất từ Cuộc Thi IT got talent, bao gồm hai file chính: train\_set và test\_set. Trong đó, tập train.csv chứa 2000+ hình ảnh huấn luyện và đa dạng tình trạng của trái cam bao gồm 1275 trái cam xấu và 926 trái cam tốt dưới dạng jpg.

### Mục tiêu phân tích:

Mục tiêu của bài toán là phân loại cam tốt và cam xấu dựa trên màu sắc và dấu hiệu bị hư. Đây là một bài toán phân loại nhị phân với 2 nhãn mục tiêu:

- Bad: Trái cam xấu
- Good: Trái cam tốt

Việc phân loại điểm tín dụng có ý nghĩa quan trọng trong hệ thống sản xuất cam nhằm giảm chi phí và đảm bảo ổn định chất lượng, loại bỏ lỗi do con người.

## Tổng quan về dữ liệu:

Tập huấn luyện (train\_set): gồm hơn 2000+ hình ảnh, phủ đa số các trường hợp bị hư, các tình trạng của trái cam và màu sắc, kích cỡ của chúng.

Tập kiểm tra (test\_set): Tập test\_set có cấu trúc tương tự tập train bao gồm 400 hình ảnh, gồm 2 loại trái cam tốt và trái cam xấu. Đây là tập dữ liệu được sử dụng để:

- Đánh giá mô hình trên dữ liệu chưa từng thấy.
- Kiểm tra độ tổng quát hóa của mô hình.
- Dự đoán thực tế sau huấn luyện.

Tất cả các bước tiền xử lý và chuẩn hóa trên tập train đều phải được áp dụng đồng nhất lên tập test để đảm bảo tính nhất quán đầu vào cho mô hình.

## Đặc điểm bài toán:

Bài toán phân loại nhị phân, đòi hỏi mô hình phải phân biệt tốt giữa 2 loại cam dựa trên tình trạng của nó. Có thể chuẩn hóa màu để đưa về dạng nhị phân 1 hoặc các giá trị gần về 0 nhất.

Ngoài ra do kích thước các ảnh khác nhau nên cần đưa về 1 kích thước để xử lý.

Cần phải lưu ý về các bước tiền xử lý để không bị mất các đặc trưng của 2 lớp như dấu hiệu bị dập, mốc, bị đốm đen,...

### 3.4.2. Tiền xử lý ảnh

```
category = ['Orange_bad', 'Orange_good']
train_set = []

def create_training_set():
    for j in category:
        label = category.index(j)
        path = os.path.join("C:\Users\NAM\Downloads\train_set\train_set",j)
        for img in os.listdir(path):
            try:
                # img_show = mpimg.imread(os.path.join(path,img))
                # gray_image = cv2.cvtColor(img_show, cv2.COLOR_BGR2GRAY)

                # chuyen grayscale va chinh kich co anh
                img_array=cv2.imread(os.path.join(path,img))
                new_array=cv2.resize(img_array,(32,32))
                #chuan hoa va chuyen sang 1D
                image_normalize = new_array /255.0
                train_set.append([image_normalize,label])
            except:
                pass
```

Hình 3.85: Hàm tạo tập huấn luyện: đọc ảnh, resize và chuẩn hóa dữ liệu đầu vào

Hàm `create_training_set()` có chức năng duyệt qua hai thư mục tương ứng với hai lớp "Orange\_bad" và "Orange\_good", chuyển ảnh thành ma trận số, resize về kích thước chuẩn (32x32), chuẩn hóa giá trị pixel về khoảng [0, 1] và lưu trữ kèm nhãn. Đây là bước quan trọng giúp đưa dữ liệu ảnh về dạng phù hợp để huấn luyện mô hình học sâu. Việc chuẩn hóa và đồng bộ kích thước ảnh giúp tăng tính ổn định khi training. Cách xử lý này đơn giản, hiệu quả và dễ mở rộng nếu có thêm nhiều lớp mới.

```
test_set = []

def create_testing_set():
    for j in catagory:
        label = catagory.index(j)
        path = os.path.join('C:/Users/Admin/Downloads/old_oranges_data/test_set/',j)
        for img in os.listdir(path):
            try:
                # img_show = mpimg.imread(os.path.join(path,img))
                # gray_image = cv2.cvtColor(img_show, cv2.COLOR_BGR2GRAY)

                # chuyen grayscale va chinh kich co anh
                img_array=cv2.imread(os.path.join(path,img))
                new_array=cv2.resize(img_array,(32,32))
                #chuan hoa va chuyen sang 1D
                image_normalize = new_array /255.0
                test_set.append([image_normalize,label])
            except:
                pass

create_testing_set()
```

Hình 3.86: Hàm xử lý tập test

Hàm `create_testing_set()` thực hiện xử lý tập kiểm tra tương tự như tập huấn luyện. Cụ thể, ảnh được resize về kích thước (32, 32), chuẩn hóa giá trị pixel về khoảng [0, 1], và gán nhãn phù hợp. Việc sử dụng cùng quy trình tiền xử lý cho cả hai tập đảm bảo tính nhất quán đầu vào, giúp mô hình đánh giá chính xác hơn trên dữ liệu chưa từng thấy. Đây là bước không thể thiếu để đảm bảo độ tổng quát của mô hình học sâu.

```
train_images = []
train_label = []
for block in train_set:
    train_images.append(block[0])
    train_label.append(block[1])

test_images = []
test_label = []
for block in test_set:
    test_images.append(block[0])
    test_label.append(block[1])
```

Hình 3.87: Tách dữ liệu hình ảnh và nhãn tương ứng từ tập huấn luyện và tập kiểm tra

Nhận xét:

Hai đoạn mã trong hình thể hiện quá trình tách dữ liệu từ danh sách train\_set và test\_set thành hai danh sách riêng biệt:

- train\_images, test\_images: chứa dữ liệu ảnh đã được resize và chuẩn hóa.
- train\_label, test\_label: chứa nhãn tương ứng (0 - cam xấu, 1 - cam tốt).

Việc tách riêng ảnh và nhãn như trên là bước cần thiết để định dạng dữ liệu đúng chuẩn đầu vào cho các mô hình học sâu (Deep Learning).

Cách làm này cũng giúp quá trình huấn luyện, đánh giá và dự đoán diễn ra hiệu quả và rõ ràng hơn.

```
lenoftrain = len(train_set)
train_images= np.array(train_images)
train_label = np.array(train_label)

lenoftest = len(test_set)
test_images= np.array(test_images)
test_label = np.array(test_label)
```

Hình 3.88: Chuyển đổi dữ liệu ảnh và nhãn về định dạng mảng NumPy

Nhận xét:

Đoạn mã này thực hiện việc:

- Tính số lượng phần tử trong train\_set và test\_set (lưu vào lenoftrain, lenoftest).
- Chuyển danh sách ảnh (train\_images, test\_images) và nhãn (train\_label, test\_label) từ Python list sang NumPy array

Việc chuyển đổi sang NumPy array là bắt buộc để đảm bảo định dạng phù hợp với yêu cầu đầu vào của mô hình học sâu Keras. Ngoài ra, NumPy còn giúp thao tác dữ liệu nhanh và hiệu quả hơn trong quá trình huấn luyện.

```

sss = StratifiedShuffleSplit(n_splits=2, test_size=0.2)
for train_index, val_index in sss.split(train_images, train_label):
    X_train_split, X_val = train_images[train_index], train_images[val_index]
    y_train_split, y_val = train_label[train_index], train_label[val_index]

```

Hình 3.89: Tách dữ liệu huấn luyện và validation theo tỷ lệ lớp cân bằng

Sử dụng StratifiedShuffleSplit để chia tập huấn luyện thành hai phần: 80% dùng huấn luyện và 20% dùng để validation. Phương pháp này đảm bảo tỷ lệ nhãn (cam tốt, cam xấu) được giữ nguyên giữa các tập, giúp quá trình huấn luyện và đánh giá ổn định và không bị lệch lớp.

### 3.4.3. Xây dựng mô hình CNN (Convolutional Neural Network)

Sau khi thực hiện tiền xử lý và chuẩn hóa dữ liệu ảnh, nhóm tiến hành xây dựng mô hình học sâu bằng mạng nơ-ron tích chập (CNN – Convolutional Neural Network), phù hợp với đặc thù của bài toán phân loại hình ảnh cam tốt và cam xấu.

#### Xây dựng mô hình

```

model2 = Sequential()

model2.add(layers.Conv2D(8, (2, 2), activation='relu', input_shape=(32, 32, 3)))
model2.add(layers.MaxPooling2D((3, 3)))
model2.add(layers.Conv2D(16, (2, 2), activation='relu'))
model2.add(layers.MaxPooling2D((3, 3)))
# Flatten the output and add dense layers
model2.add(layers.Flatten())
model2.add(layers.Dense(128, activation='relu'))
model2.add(layers.Dropout(0.5))
model2.add(layers.Dense(256, activation='relu'))
model2.add(layers.Dropout(0.5))
model2.add(layers.Dense(2, activation='sigmoid'))

```

Hình 3.90: Cấu trúc mô hình CNN dùng để phân loại cam tốt và cam xấu.

Sau bước tiền xử lý dữ liệu hình ảnh, nhóm tiến hành xây dựng mô hình phân loại sử dụng mạng nơ-ron tích chập (CNN) với cấu trúc như sau:

Mô hình được xây dựng theo kiến trúc tuần tự (Sequential) gồm:

- Lớp tích chập 1 (Conv2D):
  - Số kernel: 8.

- Kích thước kernel: (2, 2)
  - Hàm kích hoạt: ReLU
  - Input shape: (32, 32, 3) (tương ứng với ảnh RGB đã resize)
- Lớp pooling 1 (MaxPooling2D): kích thước (3, 3)
- Lớp tích chập 2: 16 kernel, kích thước (2, 2), activation = ReLU
- Lớp pooling 2: MaxPooling kích thước (3, 3)
- Flatten: Chuyển đặc trưng thành vector 1 chiều
- Dense 128 nodes + ReLU + Dropout 50%
- Dense 256 nodes + ReLU + Dropout 50%
- Lớp output (Dense): 2 node, hàm kích hoạt sigmoid dùng cho bài toán phân loại nhị phân (cam tốt và cam xấu)

#### Nhận xét:

Mô hình có chiều sâu vừa phải, phù hợp với kích thước ảnh nhỏ và bài toán nhị phân. Việc sử dụng hai lớp Dropout liên tiếp giúp giảm thiểu hiện tượng overfitting. Hàm kích hoạt sigmoid được lựa chọn đúng cho phân loại 2 lớp.

#### Biên dịch và huấn luyện mô hình

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model2.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(learning_rate=0.0075), metrics=['accuracy'])
history2 = model2.fit(X_train_split, y_train_split, epochs=50, batch_size=64, callbacks=[early_stopping], validation_data=(X_val,y_val))
```

Hình 3.91: Huấn luyện CNN

Sau khi xây dựng kiến trúc CNN, mô hình được biên dịch (compile) với các tham số như sau:

- Loss function: sparse\_categorical\_crossentropy – phù hợp cho bài toán phân loại nhiều lớp, trong đó nhãn đầu ra là số nguyên (0, 1, ...).
- Optimizer: Adam với learning rate = 0.0075, giúp tối ưu quá trình học với tốc độ phù hợp.
- Metric: accuracy – độ chính xác được sử dụng làm chỉ số chính để đánh giá mô hình.

Đồng thời nhóm áp dụng kỹ thuật EarlyStopping:

- Dừng sớm huấn luyện nếu val\_loss không cải thiện sau 10 epoch, tránh overfitting.
- Tự động khôi phục trọng số tốt nhất đã đạt được.

Mô hình được huấn luyện trên tập huấn luyện X\_train\_split, y\_train\_split trong tối đa 50 vòng lặp (epoch), với:

- batch size: 64
- validation set: X\_val, y\_val

```
28/28 - 1s 13ms/step - accuracy: 0.6279 - loss: 0.6328 - val_accuracy: 0.8957 - val_loss: 0.2664
Epoch 2/50
28/28 - 0s 6ms/step - accuracy: 0.8451 - loss: 0.3541 - val_accuracy: 0.9184 - val_loss: 0.1753
Epoch 3/50
28/28 - 0s 6ms/step - accuracy: 0.9323 - loss: 0.2041 - val_accuracy: 0.9116 - val_loss: 0.2205
Epoch 4/50
28/28 - 0s 7ms/step - accuracy: 0.9518 - loss: 0.1512 - val_accuracy: 0.9773 - val_loss: 0.0672
Epoch 5/50
28/28 - 0s 7ms/step - accuracy: 0.9686 - loss: 0.1099 - val_accuracy: 0.9841 - val_loss: 0.0414
Epoch 6/50
28/28 - 0s 6ms/step - accuracy: 0.9769 - loss: 0.0650 - val_accuracy: 0.9773 - val_loss: 0.0485
Epoch 7/50
28/28 - 0s 6ms/step - accuracy: 0.9774 - loss: 0.0701 - val_accuracy: 0.9796 - val_loss: 0.0713
Epoch 8/50
28/28 - 0s 6ms/step - accuracy: 0.9782 - loss: 0.0696 - val_accuracy: 0.9615 - val_loss: 0.1175
Epoch 9/50
28/28 - 0s 6ms/step - accuracy: 0.9605 - loss: 0.1006 - val_accuracy: 0.9615 - val_loss: 0.1093
Epoch 10/50
28/28 - 0s 6ms/step - accuracy: 0.9603 - loss: 0.1328 - val_accuracy: 0.9773 - val_loss: 0.0450
Epoch 11/50
28/28 - 0s 6ms/step - accuracy: 0.9805 - loss: 0.0700 - val_accuracy: 0.9932 - val_loss: 0.0277
Epoch 12/50
28/28 - 0s 6ms/step - accuracy: 0.9858 - loss: 0.0370 - val_accuracy: 0.9819 - val_loss: 0.0502
Epoch 13/50
28/28 - 0s 6ms/step - accuracy: 0.9860 - loss: 0.0393 - val_accuracy: 0.9841 - val_loss: 0.0355
...
Epoch 29/50
28/28 - 0s 6ms/step - accuracy: 0.9895 - loss: 0.0314 - val_accuracy: 0.9796 - val_loss: 0.0762
Epoch 30/50
28/28 - 0s 7ms/step - accuracy: 0.9777 - loss: 0.0582 - val_accuracy: 0.9796 - val_loss: 0.0534
```

Hình 3.92: Log quá trình huấn luyện mô hình CNN theo từng epoch.

Nhận xét:

Mô hình khởi đầu với độ chính xác thấp (accuracy = 62.79%) tại epoch 1, tuy nhiên nhanh chóng được cải thiện qua từng epoch:

- Epoch 2: accuracy tăng vọt lên 84.51%, val\_accuracy đạt 91.84%.
- Epoch 5: accuracy đạt 96.86%, val\_accuracy lên đến 98.41%.
- Từ epoch 5 trở đi, val\_accuracy duy trì ổn định trong khoảng 96.1% đến 99.3% thể hiện mô hình đã học tốt và không bị overfitting.

Giá trị loss và val\_loss giảm nhanh chóng:

- loss giảm từ 0.63 → 0.03 (epoch 30).
- val\_loss giảm mạnh từ 0.2664 (epoch 1) xuống dao động 0.04 – 0.07 sau epoch 5 → cho thấy quá trình học hiệu quả và ổn định.

Các mốc đáng chú ý:

- Epoch 12: val\_accuracy đạt đỉnh 99.32%, val\_loss rất thấp 0.0277.
- Epoch 30 (kết thúc): accuracy = 97.77%, val\_accuracy = 97.96%, val\_loss = 0.0534 vẫn rất tốt.

**Kết luận:** Mô hình CNN học nhanh, hội tụ tốt chỉ sau ~5 epoch đầu. Sau đó giữ vững hiệu suất cao, không dao động mạnh, val\_accuracy luôn > 96%. Đây là mô hình huấn luyện rất thành công, sẵn sàng áp dụng cho bước kiểm tra chính thức trên tập test

#### 3.4.4. Dự đoán và xuất kết quả

```
pre = model2.predict(test_images)

13/13 ━━━━━━━━━━━━ 0s 7ms/step

p2 = np.argmax(pre, axis=1)

testing = p2.tolist()
```

Hình 3.93: Dự đoán nhãn cam tốt hoặc cam xấu trên tập test bằng mô hình CNN.

Sau khi mô hình được huấn luyện, tiến hành dự đoán trên tập test chứa các hình ảnh cam chưa từng thấy.

Kết quả dự đoán được xử lý bằng np.argmax () để lấy chỉ số nhãn có xác suất cao nhất và chuyển sang dạng list để lưu trữ.

[0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
...
1,
1,
1,
1,
1]

Hình 3.94: Bảng kết quả dự đoán của mô hình CNN trên tập ảnh test.

Mỗi số trong danh sách biểu thị một nhãn dự đoán:

- 0 tương ứng với cam xấu (Orange\_bad)
- 1 tương ứng với cam tốt (Orange\_good)

Danh sách này tương ứng với số ảnh trong tập test (400 ảnh).

	precision	recall	f1-score	support
0	0.65	1.00	0.78	200
1	1.00	0.45	0.62	200
accuracy			0.72	400
macro avg	0.82	0.72	0.70	400
weighted avg	0.82	0.72	0.70	400

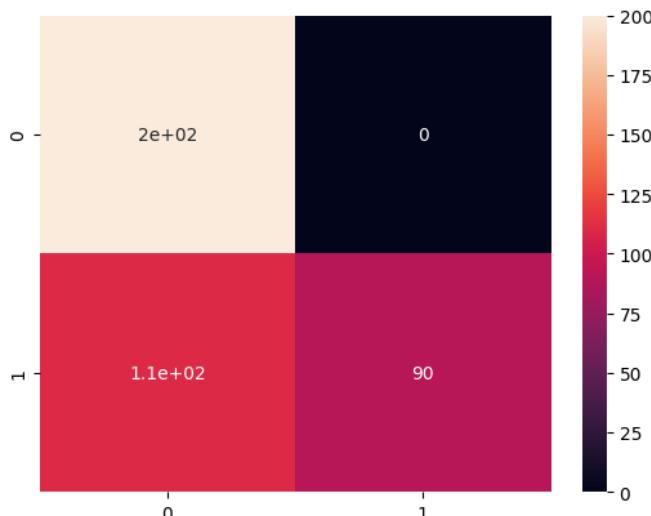
Hình 3.95: Độ chính xác của mô hình CNN trên tập kiểm tra

- Accuracy tổng thể đạt 72%, phù hợp với kết quả đã đánh giá trước đó.
- Precision và Recall giữa hai lớp có sự mất cân bằng rõ rệt:

- Lớp 0 (cam xấu): Precision = 0.65, Recall = 1.00 → mô hình nhận diện được hầu hết cam xấu, nhưng có thể đánh nhầm một số cam tốt thành cam xấu.
- Lớp 1 (cam tốt): Precision = 1.00, Recall = 0.45 → mô hình chỉ nhận diện được khoảng 45% số cam tốt, dù khi dự đoán là tốt thì gần như chắc chắn đúng.
- F1-score trung bình rơi vào khoảng 0.70, cho thấy mô hình chưa đạt hiệu năng đồng đều giữa các lớp.

Nhận xét:

- Mô hình thiên lệch về lớp cam xấu (class 0), ưu tiên phát hiện lỗi nhưng đánh đổi bằng việc bỏ sót nhiều mẫu cam tốt.
- Điều này có thể phù hợp với các ứng dụng yêu cầu phát hiện lỗi nghiêm ngặt, nhưng chưa tối ưu nếu muốn cân bằng cả hai lớp.
- Cần cân nhắc điều chỉnh loss function hoặc kỹ thuật xử lý mất cân bằng lớp (ví dụ: class weights, oversampling) để cải thiện kết quả cho lớp cam tốt (class 1)



Hình 3.96: Ma trận nhầm lẫn của mô hình CNN trên tập kiểm tra

Phân tích ma trận nhầm lẫn:

- Dự đoán đúng cam xấu (class 0): 200 ảnh
- Dự đoán nhầm cam tốt thành cam xấu: 110 ảnh

- Dự đoán đúng cam tốt (class 1): 90 ảnh.

- Dự đoán nhầm cam xấu thành cam tốt: 0 ảnh

→ Mô hình rất cẩn trọng với việc đánh nhầm cam xấu thành cam tốt (không xảy ra), nhưng bỏ sót khá nhiều cam tốt.

```
black_1276.jpg
black_1277.jpg
black_1278.jpg
black_1279.jpg
black_1280.jpg
black_1281.jpg
black_1282.jpg
black_1283.jpg
black_1284.jpg
black_1285.jpg
black_1286.jpg
black_1287.jpg
black_1288.jpg
black_1289.jpg
black_1290.jpg
black_1291.jpg
black_1292.jpg
black_1293.jpg
black_1294.jpg
black_1295.jpg
black_1296.jpg
black_1297.jpg
black_1298.jpg
black_1299.jpg
black_1300.jpg
...
fresh_996.jpg
fresh_997.jpg
fresh_998.jpg
fresh_999.jpg
```

Hình 3.97: Tên các ảnh đại diện cho hai lớp trong tập kiểm tra

- Hình ảnh liệt kê một phần tên file ảnh thuộc hai nhãn black và fresh, tương ứng với:
  - black\_\*.jpg: Cam xấu (hư, đốm đen, mốc...)
  - fresh\_\*.jpg: Cam tốt (tươi, không có dấu hiệu hư hỏng)
- Việc đặt tên ảnh theo nhãn giúp dễ dàng kiểm tra và đổi chiều nhãn mục tiêu trong quá trình huấn luyện và đánh giá mô hình.
- Quan sát thấy tập kiểm tra (test set) được tổ chức rõ ràng, bao gồm cả hai lớp và có độ đa dạng hình ảnh tương đối, phù hợp cho việc đánh giá hiệu quả mô hình phân loại.

```

submission = pd.DataFrame(columns=['image_name'])
for j in catagory:
    label = catagory.index(j)
    path = os.path.join('C:/Users/Admin/Downloads/old_oranges_data/test_set/',j)
    for img in os.listdir(path):
        try:
            df = pd.DataFrame({'image_name':[img]})
            submission = pd.concat([submission,df])
        except:
            pass

submission.insert(loc = len(submission.columns),column='label',value=testing)

```

Hình 3.98: Tạo file submission từ dự đoán mô hình trên tập test

Nhận xét:

- Đoạn code trên thực hiện tạo file kết quả dựa trên dự đoán testing của mô hình CNN cho tập kiểm tra.

Quá trình thực hiện:

- Duyệt qua từng thư mục (Orange\_bad, Orange\_good) chứa ảnh trong tập test.
- Tạo DataFrame chứa tên ảnh và nối vào submission.
- Thêm cột label ứng với kết quả dự đoán của mô hình

Kết quả cuối cùng sẽ là một file dạng .csv có hai cột: image\_name và label, dùng để nộp lên hệ thống chấm thi hoặc phân tích tiếp theo.

```

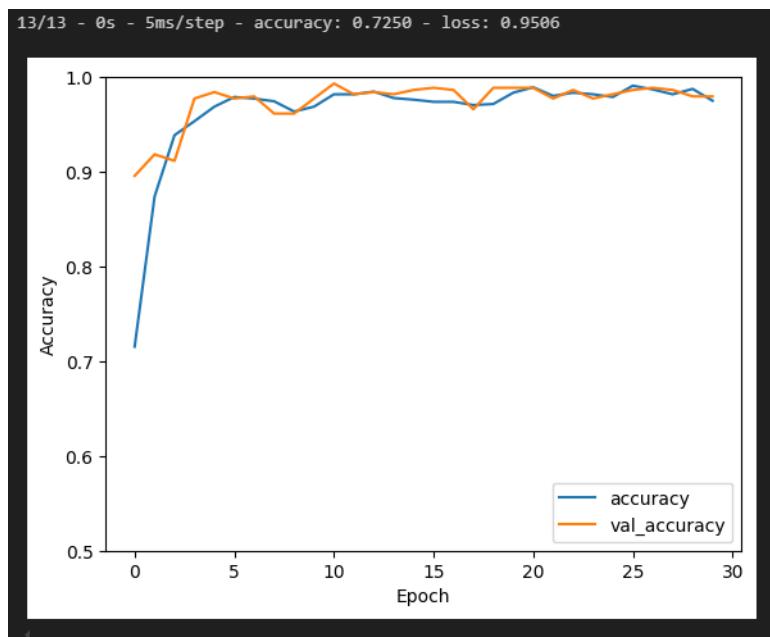
,image_name,label
0,black_1276.jpg,0
0,black_1277.jpg,0
0,black_1278.jpg,0
0,black_1279.jpg,0
0,black_1280.jpg,0
0,black_1281.jpg,0
0,black_1282.jpg,0
0,black_1283.jpg,0
0,black_1284.jpg,0
0,black_1285.jpg,0
0,black_1286.jpg,0
0,black_1287.jpg,0
0,black_1288.jpg,0
0,black_1289.jpg,0
0,black_1290.jpg,0
0,black_1291.jpg,0
0,black_1292.jpg,0
0,black_1293.jpg,0

```

Hình 3.99: File submission

- Cột image\_name chứa tên file ảnh, ví dụ: black\_1276.jpg

- Cột label là nhãn tương ứng (0 = cam xấu, 1 = cam tốt).
- Đây là cơ sở để xây dựng tập train/test, đưa vào mô hình học sâu



Hình 3.100: Biểu đồ so sánh accuracy và val\_accuracy qua các epoch trong quá trình huấn luyện mô hình CNN.

Nhận xét:

- Accuracy và validation accuracy đều đạt trên 95% sau khoảng 10 epoch, cho thấy mô hình học khá nhanh và hiệu quả trên tập huấn luyện.
- Mức chênh lệch giữa hai đường nhỏ, cho thấy mô hình không bị overfitting nghiêm trọng.
- Accuracy trên tập test chỉ đạt 72.5% → mô hình vẫn còn chưa tổng quát hóa tốt cho dữ liệu thực tế.

#### 3.4.5. Kết luận

Bài toán phân loại hình ảnh cam tốt và cam xấu đã được tiếp cận bằng mô hình học sâu CNN với quy trình tiền xử lý ảnh bài bản, từ việc resize, chuẩn hóa, đến chia dữ liệu huấn luyện và kiểm tra hợp lý. Mô hình CNN được thiết kế gồm các lớp tích

chập, pooling, fully connected kết hợp dropout để giảm overfitting, cùng cơ chế EarlyStopping nhằm tối ưu hóa quá trình huấn luyện.

#### Kết quả mô hình:

- Accuracy trên tập huấn luyện/validation đạt rất cao (>98%), chứng tỏ mô hình đã học tốt từ dữ liệu huấn luyện.
- Accuracy trên tập kiểm tra thực tế chỉ đạt 72.5%, thấp hơn đáng kể so với validation. Điều này phản ánh khả năng tổng quát hóa còn hạn chế, có thể do:
  - Mô hình đã học quá kỹ trên tập huấn luyện (overfitting nhẹ).
  - Dữ liệu test có sự khác biệt đáng kể về đặc trưng hình ảnh.
  - Cần áp dụng thêm kỹ thuật data augmentation, điều chỉnh dropout hoặc cải thiện kiến trúc mạng.

#### Tổng kết:

- Mô hình CNN đã thể hiện khả năng nhận diện tốt tình trạng của trái cam trên tập huấn luyện.
- Nhưng để ứng dụng vào thực tế, cần cải thiện khả năng dự đoán trên dữ liệu mới bằng cách tăng cường dữ liệu, tinh chỉnh mô hình và kết hợp thêm các kỹ thuật xử lý ảnh nâng cao.

### 3.5. Dataset 5: Bank Customer Segmentation - Phân cụm khách hàng ngân hàng

#### 3.5.1. Giới thiệu về bộ dữ liệu

Bộ dữ liệu “Bank Customer Segmentation” được cung cấp trên Kaggle bởi Shivam Bansal, chứa hơn 1 triệu giao dịch tài chính từ khách hàng ngân hàng. Mỗi bản ghi thể hiện một giao dịch, kèm theo thông tin nhân khẩu học như: ngày sinh, giới tính, vị trí và số dư tài khoản.

#### Mục tiêu phân tích

- Phân tích hành vi và đặc điểm khách hàng dựa trên dữ liệu giao dịch.
- Phân khúc khách hàng theo độ tuổi, giới tính, khu vực và mức chi tiêu.

- Làm tiền đề cho mô hình học máy như phân cụm, phát hiện bất thường hoặc dự báo chi tiêu.

### Tổng quan về dữ liệu

Tên biến	Ý nghĩa	Kiểu dữ liệu
TransactionID	Mã định danh giao dịch duy nhất	Chuỗi ký tự (string)
CustomerID	Mã định danh khách hàng	Chuỗi ký tự (string)
CustomerDOB	Ngày sinh của khách hàng (dùng để tính tuổi)	Chuỗi ký tự (string)
CustGender	Giới tính khách hàng (Male, Female)	Chuỗi ký tự (string)
CustLocation	Vị trí địa lý của khách hàng (thành phố hoặc vùng miền)	Chuỗi ký tự (string)
CustAccountBalance	Số dư tài khoản tại thời điểm giao dịch (đơn vị: INR – Rupee Ấn Độ)	Số thực (float)
TransactionDate	Ngày thực hiện giao dịch	Chuỗi ký tự (string)
TransactionTime	Thời gian giao dịch dạng UNIX timestamp	Số nguyên (int)
TransactionAmount	Số tiền giao dịch (đơn vị: INR)	Số thực (float)

### Đặc điểm của bài toán:

Bài toán đặt ra là phân khúc khách hàng ngân hàng dựa trên dữ liệu giao dịch thực tế và đặc điểm nhân khẩu học. Dữ liệu không có nhãn, nên thuộc dạng học không giám sát, trong đó mục tiêu là tìm ra các nhóm khách hàng có hành vi tương đồng nhằm phục vụ cho hoạt động marketing, chăm sóc khách hàng hoặc quản trị rủi ro.

Hai thuật toán được đề xuất:

- K-Means: Phù hợp với dữ liệu đã chuẩn hóa, giúp chia khách hàng thành các nhóm dựa trên các đặc trưng như tuổi, số dư, số tiền giao dịch và tần suất giao dịch. Kết quả dễ trực quan và giải thích.
- DBSCAN: Phân cụm theo mật độ, không cần biết trước số cụm và có khả năng phát hiện ngoại lệ (khách hàng có hành vi bất thường), phù hợp trong bối cảnh dữ liệu lớn, không đồng đều.

### 3.5.2. Tiền xử lý dữ liệu

#### Đọc dữ liệu

```
df=pd.read_csv(r"C:\Users\NAM\Downloads\bank_transactions.csv\bank_transactions.csv")
```

Hình 3.101: Đọc dữ liệu bank\_transactions bằng pandas

#### Khám phá dữ liệu

	# CustAccountBalance	# TransactionTime	# TransactionAmount (INR)
count	1046198.0	1048567.0	1048567.0
mean	115403.54005622261	157087.52939297154	1574.3350034570992
std	846485.3806006602	51261.85402232933	6574.742978454002
min	0.0	0.0	0.0
25%	4721.76	124030.0	161.0
50%	16792.18	164226.0	459.03
75%	57657.36	200010.0	1200.0
max	115035495.1	235959.0	1560034.99

Hình 3.102: Thống kê mô tả ba biến định lượng chính

- Biến CustAccountBalance – Số dư tài khoản:
  - Trung bình (mean): khoảng 115.403 INR, nhưng độ lệch chuẩn rất cao (846.485 INR), cho thấy sự phân tán lớn giữa các khách hàng.
  - Giá trị tối đa: hơn 11,5 triệu INR, rất cao so với giá trị trung vị (16.792 INR) → có thể có nhiều outlier (khách hàng VIP).
  - Giá trị tối thiểu = 0 → Có thể là tài khoản mới, không hoạt động, hoặc cần được kiểm tra kỹ hơn.
- Biến TransactionAmount (INR) – Số tiền giao dịch:

- Giá trị trung bình: khoảng 1.574 INR, trong khi trung vị chỉ 459 INR → phân phối lệch phai rõ rệt.
- Tối đa lên đến 1,56 triệu INR, trong khi 75% khách hàng giao dịch dưới 1.200 INR → cần xử lý hoặc phân tích riêng các giao dịch có giá trị rất cao.
- Biến TransactionTime:
  - Biến này là UNIX timestamp nên không có ý nghĩa thống kê trực tiếp về mặt nghiệp vụ, nhưng các giá trị từ min-max đều hợp lệ.
  - Sẽ cần chuyển đổi về thời gian thực (datetime) để phục vụ các phân tích như: giao dịch theo giờ, theo ngày,..

	⊕ TransactionID	⊕ CustomerID	⊕ CustomerDOB	⊕ CustGender	⊕ CustLocation	⊕ TransactionDate
count	1048567	1048567	1045170	1047467	1048416	1048567
unique	1048567	884265	17254	3	9355	55
top	T1	C5533885	1/1/1800	M	MUMBAI	7/8/16
freq	1	6	57339	765530	103595	27261

Hình 3.103: Thống kê mô tả các biến định danh và phân loại trong bộ dữ liệu giao dịch ngân hàng

- TransactionID và CustomerID:
  - TransactionID: có 1.048.567 giá trị duy nhất, không trùng lặp → phù hợp là khóa chính cho từng giao dịch.
  - CustomerID: có khoảng 884.265 khách hàng duy nhất, nhưng có khách hàng thực hiện tới 6 giao dịch → dữ liệu nhiều-một giữa giao dịch và khách hàng, rất phù hợp để phân tích theo cá nhân.
- CustomerDOB – Ngày sinh khách hàng:
  - Gồm 17.254 giá trị duy nhất, cho thấy có thể có trùng lặp (nhiều khách hàng sinh cùng ngày).
  - Giá trị phổ biến nhất là 1/1/1800, với 57.339 lần lặp → đây rất có thể là giá trị mặc định lỗi, cần xử lý (loại bỏ hoặc gán giá trị thiếu).
- CustGender – Giới tính khách hàng:
  - Có 3 giá trị phân biệt, nhưng chỉ nên có “M” (Male) và “F” (Female) → cần kiểm tra và loại bỏ giá trị không hợp lệ (có thể là “T” hoặc ký tự sai định dạng).

- Tỉ lệ giới tính lệch khá mạnh: 76% là nam → cần cân nhắc trong trực quan hóa và phân cụm.
- CustLocation – Khu vực địa lý:
  - Có tới 9.355 khu vực khác nhau, thành phố phô biến nhất là Mumbai, chiếm hơn 10% giao dịch.
  - Việc này cho thấy mức độ phân tán địa lý cao, nhưng có thể nhóm lại theo vùng/miền để phân tích cụ thể hơn (nếu có dữ liệu hỗ trợ).
- TransactionDate – Ngày giao dịch:
  - Chỉ có 55 ngày giao dịch khác nhau trong hơn 1 triệu bản ghi → có thể là dữ liệu thu thập theo batch/ngày, hoặc có giới hạn thời gian nghiên cứu.
  - Ngày phô biến nhất là 7/8/2016 với 27.261 giao dịch, cần kiểm tra xem đây có phải là đợt cao điểm nào không (lẽ, đợt ưu đãi,...)

```
# check duplicate
df.duplicated().sum()
✓ 1.2s
0
```

Hình 3.104: Kiểm tra dữ liệu trùng lặp

- Kết quả cho thấy không có bản ghi nào bị trùng lặp hoàn toàn trong tập dữ liệu.
- Dữ liệu được lưu trữ duy nhất và đúng chuẩn cho mỗi giao dịch.
- Không cần thực hiện bước loại bỏ trùng lặp, giúp tối ưu thời gian xử lý và giữ được toàn bộ dữ liệu đầu vào cho mô hình

```

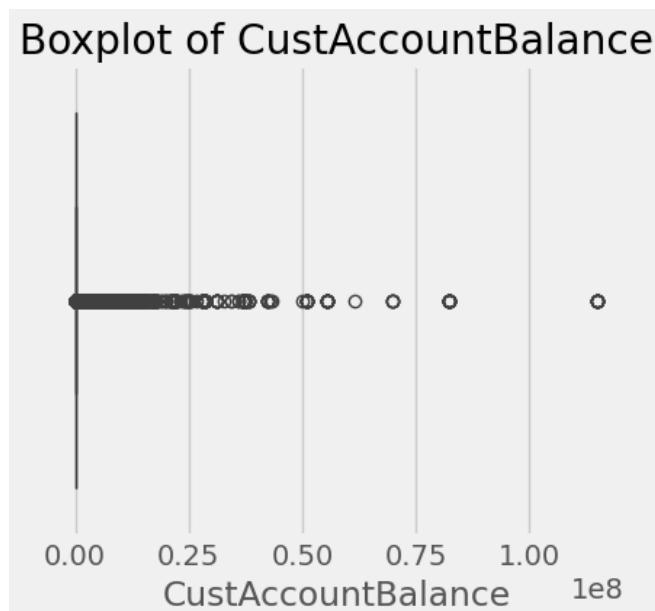
# check null values
df.isnull().sum()
✓ 0.1s

TransactionID          0
CustomerID             0
CustomerDOB            3397
CustGender              1100
CustLocation             151
CustAccountBalance      2369
TransactionDate          0
TransactionTime          0
TransactionAmount (INR)    0
dtype: int64

```

Hình 3.105: Kiểm tra số lượng giá trị thiếu theo từng cột

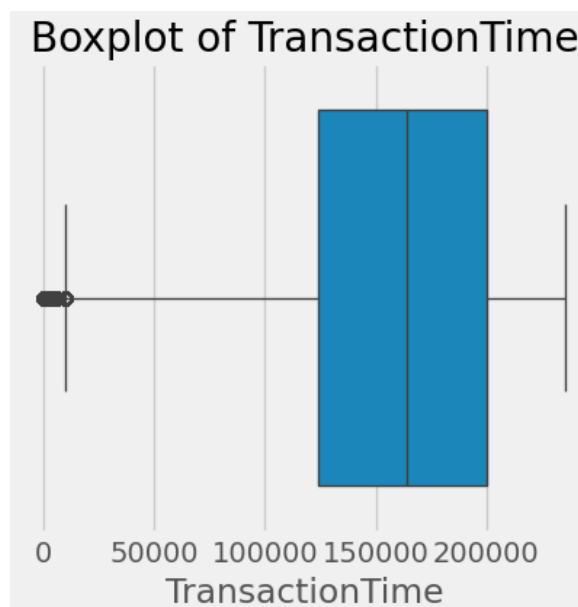
- CustomerDOB thiếu 3.397 dòng (0,32%) – là biến quan trọng để tính tuổi, nên các dòng thiếu sẽ được loại bỏ.
- CustGender thiếu 1.100 dòng (0,10%) – có thể loại bỏ trực tiếp.
- CustLocation thiếu 151 dòng (0,01%) – tỷ lệ rất nhỏ, có thể loại bỏ hoặc thay bằng "Unknown".
- CustAccountBalance thiếu 2.369 dòng (0,23%) – là biến định lượng quan trọng, khuyến nghị loại bỏ các dòng thiếu.
- Các cột còn lại không có giá trị thiếu.



Hình 3.106: Biểu đồ Boxplot của biến CustAccountBalance

Biểu đồ cho thấy sự phân phối lệch phải rõ rệt, với rất nhiều giá trị ngoại lệ nằm phía trên. Phần lớn khách hàng có số dư tài khoản thấp, trong khi một số ít có số dư vượt trội lên đến hàng chục triệu INR.

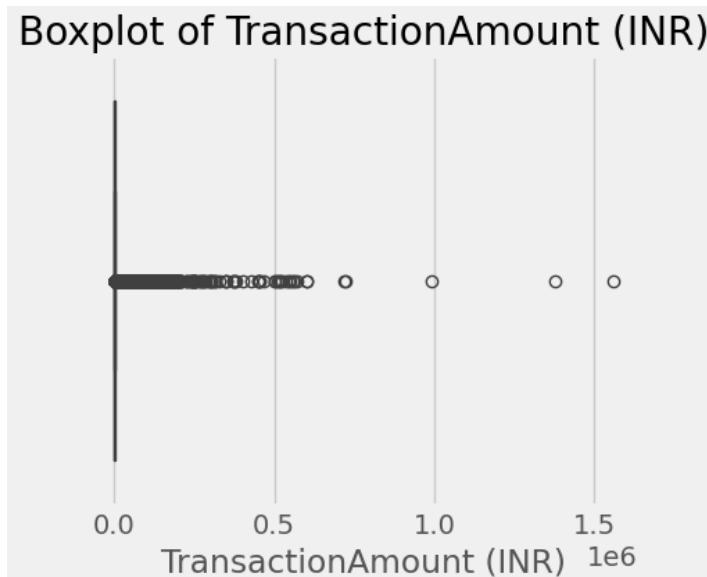
Điều này cho thấy sự khác biệt lớn về năng lực tài chính giữa các nhóm khách hàng, và cũng là cơ sở quan trọng để thực hiện phân cụm.



Hình 3.107: Biểu đồ Boxplot của biến TransactionTime

Biểu đồ cho thấy phần lớn giá trị TransactionTime tập trung trong một khoảng rõ ràng (từ khoảng 120.000 đến 200.000), phản ánh mức phân佈 thời gian giao dịch tương đối đồng đều trong khoảng thời gian ghi nhận.

Trong đó cũng xuất hiện một số giá trị nhỏ bất thường ở bên trái (gần 0), được xem là outlier – có thể là các giao dịch được ghi nhận sớm hoặc lỗi hệ thống.



Hình 3.108: Biểu đồ Boxplot của biến TransactionAmount (INR)

Biểu đồ cho thấy phân phối của số tiền giao dịch bị lệch phải nghiêm trọng, với nhiều giá trị ngoại lệ nằm ở phía cao.

Hầu hết các giao dịch có giá trị nhỏ hơn 1.200 INR, nhưng vẫn tồn tại các giao dịch vượt quá 1 triệu INR – cho thấy sự khác biệt lớn trong hành vi chi tiêu giữa các nhóm khách hàng.

Các điểm ngoại lệ có thể là giao dịch của khách hàng doanh nghiệp hoặc giao dịch bất thường, cần được cân nhắc kỹ trong phân cụm.

```
df['CustGender'].value_counts()
✓ 0.0s
CustGender
M    765530
F    281936
T        1
```

Hình 3.109: Phân tích biến CustGender

Tập dữ liệu có 76,5% khách hàng là nam (M) và 23,5% là nữ (F), thể hiện sự chênh lệch giới tính đáng kể trong hành vi sử dụng dịch vụ ngân hàng.

Đáng chú ý, có 1 giá trị không hợp lệ là 'T', Giá trị này đã được loại bỏ nhằm đảm bảo tính nhất quán khi xử lý và phân tích dữ liệu.

## Xử lý dữ liệu

```
# convert to datetime
df.CustomerDOB = pd.to_datetime(df.CustomerDOB, errors='coerce')
df['TransactionDate'] = pd.to_datetime(df['TransactionDate'], errors='coerce')
```

Hình 3.110: Chuyển đổi định dạng ngày tháng

Các cột ngày sinh khách hàng (CustomerDOB) và ngày giao dịch (TransactionDate) được chuyển sang định dạng datetime để phục vụ cho việc tính toán độ tuổi và phân tích thời gian.

```
# calculate age customer
from datetime import date
def age(birthdate):
    today = date.today()
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))
    return age
df['Age'] = df.CustomerDOB.apply(age)
df['Age'].value_counts()
```

Hình 3.111: Hàm tính tuổi khách hàng và phân bố giá trị Age

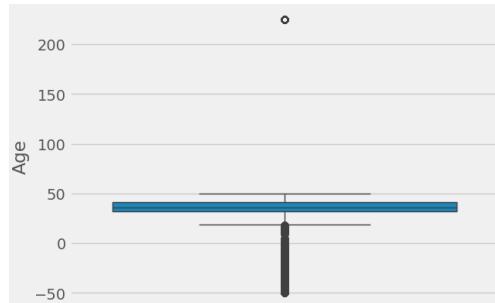
Độ tuổi khách hàng được tính từ biến CustomerDOB bằng cách so sánh với ngày hiện tại. Sau khi áp dụng hàm tính tuổi, biến mới Age được tạo ra để đưa vào phân tích.

Age	
34	72887
33	70648
35	70526
36	64312
32	62769
	...
10	4
-1	3
2	3
0	1
7	1

Hình 3.112: Phân bố độ tuổi của khách hàng sau khi tính từ ngày sinh

Biến Age được tạo từ ngày sinh CustomerDOB. Phân tích cho thấy phần lớn khách hàng nằm trong độ tuổi từ 32 đến 36, cho thấy tập khách hàng chủ yếu là người trưởng thành đang trong độ tuổi lao động.

Một số giá trị bất hợp lý như -1, 0, 2 hoặc 7 cho thấy có dữ liệu sai lệch về ngày sinh. Các bản ghi này đã được loại bỏ để đảm bảo độ tin cậy cho mô hình phân cụm.



Hình 3.113: Biểu đồ Boxplot của biến Age

Biểu đồ boxplot cho thấy độ tuổi khách hàng tập trung chủ yếu từ khoảng 18 đến 55 tuổi, với trung vị khoảng 40 tuổi.

Tồn tại nhiều giá trị ngoại lệ như độ tuổi âm, bằng 0, hoặc vượt quá 100, trong đó có giá trị gần 220 tuổi không hợp lý.

Những bản ghi này được xác định là outlier và đã được loại bỏ để đảm bảo tính chính xác cho mô hình phân cụm.

```
# age < 0 drop
df = df[df['Age']>0]
df.drop('CustomerDOB',axis=1,inplace=True)
df.shape
✓ 0.3s
(936822, 9)
```

Hình 3.114: Loại bỏ khách hàng có tuổi không hợp lý và xóa cột ngày sinh

Các bản ghi có giá trị Age nhỏ hơn hoặc bằng 0 được loại bỏ để đảm bảo dữ liệu hợp lý. Sau khi xử lý, số lượng bản ghi giảm từ hơn 1 triệu xuống còn 936.822 dòng.

Đồng thời, cột CustomerDOB được loại bỏ vì đã hoàn thành mục tiêu tính tuổi khách hàng.

$\Delta$ CustomerID	$\Delta$ CustGender	$\Delta$ CustLocation	# CustAccountBalance	$\Delta$ TransactionDate	# TransactionAmount (INR)	# Age
0 C5841053	F	JAMSHEDPUR	17819.05	2016-02-08 00:00:00	25.0	30
2 C4417068	F	MUMBAI	17874.44	2016-02-08 00:00:00	459.0	28
4 C9031234	F	NAVI MUMBAI	6714.43	2016-02-08 00:00:00	1762.5	37
6 C7126560	F	MUMBAI	973.46	2016-02-08 00:00:00	566.0	33
7 C1220223	M	MUMBAI	95075.54	2016-02-08 00:00:00	148.0	43

Hình 3.115: Dữ liệu sau tiền xử lý

Sau các bước làm sạch dữ liệu, bộ dữ liệu đầu vào đã được chuẩn hóa và chỉ giữ lại những biến có ý nghĩa phân tích: CustomerID, CustGender, CustLocation, CustAccountBalance, TransactionDate, TransactionAmount (INR) và biến mới Age được tính toán từ ngày sinh.

Các giá trị thiếu, không hợp lệ và ngoại lệ đã được xử lý triệt để. Các biến không mang giá trị phân tích như TransactionID, TransactionTime, CustomerDOB cũng đã được loại bỏ để tối ưu hóa tập dữ liệu đầu vào.

Dữ liệu sau xử lý có dạng gọn gàng, rõ ràng, đồng nhất và sẵn sàng cho các bước phân tích RFM, chuẩn hóa và phân cụm khách hàng trong các mục tiếp theo.

## RFM Analysis

- RFM Analysis là phương pháp phân tích hành vi khách hàng dựa trên 3 yếu tố:
  - Recency – Khách hàng giao dịch gần đây nhất cách hiện tại bao lâu (số ngày).
  - Frequency – Khách hàng đã giao dịch bao nhiêu lần.
  - Monetary – Tổng số tiền mà khách hàng đã chi tiêu.
- Phân tích RFM giúp xác định được nhóm khách hàng trung thành, khách hàng tiềm năng cũng như những khách hàng có dấu hiệu rời bỏ, từ đó hỗ trợ chiến lược marketing và phân cụm hành vi khách hàng hiệu quả hơn.

## Quy trình thực hiện RFM từ dữ liệu gốc

	<b>CustomerID</b>	<b># Monetary</b>
0	C1010011	5106.0
1	C1010012	1499.0
2	C1010014	1455.0
3	C1010018	30.0
4	C1010028	557.0
5	C1010031	1864.0
6	C1010035	750.0
7	C1010036	208.0
8	C1010037	19680.0
9	C1010038	100.0

Hình 3.116: Kết quả tính chỉ số Monetary cho từng khách hàng

Chỉ số Monetary thể hiện tổng số tiền chi tiêu của mỗi khách hàng trong toàn bộ lịch sử giao dịch.

Hình trên cho ta thấy có khách hàng chỉ chi tiêu 30 INR, trong khi có khách hàng chi tới gần 20.000 INR – cho thấy sự phân hóa lớn về giá trị tài chính, rất phù hợp để phân nhóm trong các bước tiếp theo.

	<b>CustomerID</b>	<b># Frequency</b>
0	C1010011	2
1	C1010012	1
2	C1010014	2
3	C1010018	1
4	C1010028	1

Hình 3.117: Số lần giao dịch (Frequency) của từng khách hàng

Chỉ số Frequency thể hiện số lần mỗi khách hàng đã phát sinh giao dịch trong khoảng thời gian quan sát.

Như hình 3.115 cho thấy, có khách hàng chỉ giao dịch 1 lần, trong khi một số khách hàng khác thực hiện 2 giao dịch hoặc nhiều hơn. Đây là cơ sở để đánh giá mức độ gắn bó của khách hàng với ngân hàng.

	CustomerID	# difference
0	C1010011	74
1	C1010012	117
2	C1010014	154
3	C1010018	85
4	C1010028	102

Hình 3.118: Số ngày kể từ giao dịch gần nhất (Recency) của từng khách hàng

Recency thể hiện khoảng cách thời gian (tính theo ngày) giữa giao dịch gần nhất của khách hàng và thời điểm quan sát cuối cùng trong dữ liệu.

Giá trị càng nhỏ thể hiện khách hàng vẫn còn hoạt động gần đây, giá trị càng lớn thì khả năng rời bỏ càng cao.

Ví dụ khách hàng C1010011 có Recency là 74 ngày – tức là vừa giao dịch cách đây khoảng hơn 2 tháng, trong khi khách hàng C1010014 đã 154 ngày không phát sinh giao dịch.

	CustomerID	# Monetary	# Frequency	# Recency
0	C1010011	5106.0	2	74
1	C1010012	1499.0	1	117
2	C1010014	1455.0	2	154
3	C1010018	30.0	1	85
4	C1010028	557.0	1	102

Hình 3.119: Bảng RFM hoàn chỉnh của một số khách hàng

- Sau khi tính toán các chỉ số riêng biệt, bảng RFM được hoàn thiện bằng cách gộp các chỉ số Monetary, Frequency, và Recency theo CustomerID.
- Bảng này là cơ sở để phân cụm khách hàng theo hành vi:
  - Monetary đo lường giá trị tài chính.
  - Frequency phản ánh mức độ gắn bó.
  - Recency thể hiện mức độ cập nhật của khách hàng.
- Những khách hàng có R thấp, F cao và M cao là khách hàng VIP cần được ưu tiên.

#	CustGender	CustLocation	# CustAccountBalance	# TransactionDate	# TransactionAmount (INR)
0	F	JAMSHEDPUR	17819.05	2016-02-08 00:00:00	25.0
1	F	MUMBAI	17874.44	2016-02-08 00:00:00	459.0
2	F	NAVI MUMBAI	6714.43	2016-02-08 00:00:00	1762.5
3	F	MUMBAI	973.46	2016-02-08 00:00:00	566.0
4	M	MUMBAI	95075.54	2016-02-08 00:00:00	148.0
#	Age	Monetary	Frequency	Recency	
	30	25.0		1	305
	28	459.0		1	305
	37	1762.5		1	305
	33	821.0		4	61
	43	148.0		1	305

Hình 3.120: Dữ liệu đã hoàn tất sau khi gộp RFM

Sau các bước xử lý, tính toán và gộp chỉ số RFM, dữ liệu đầu vào đã được chuẩn hóa và sẵn sàng cho phân cụm

Hình 3.118 thể hiện tập dữ liệu cuối cùng, bao gồm cả thông tin nhân khẩu học (Gender, Location, Age) và hành vi tài chính (Balance, TransactionAmount, RFM).

Các cột kỹ thuật không cần thiết như TransactionID, TransactionTime, CustomerDOB đã được loại bỏ.

```
<class 'pandas.core.frame.DataFrame'>
Index: 913947 entries, 0 to 936821
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CustGender        913947 non-null   object 
 1   CustLocation      913947 non-null   object 
 2   CustAccountBalance 913947 non-null   float64
 3   TransactionDate    913947 non-null   datetime64[ns]
 4   TransactionAmount (INR) 913947 non-null   float64
 5   Age                913947 non-null   int64  
 6   Monetary           913947 non-null   float64
 7   Frequency          913947 non-null   int64  
 8   Recency             913947 non-null   int64  
 9   TransactionMonth    913947 non-null   int32  
 10  TransactionMonthName 913947 non-null   object 
 11  TransactionDay     913947 non-null   int32  
 12  TransactionDayName 913947 non-null   object 
```

Hình 3.121: Cấu trúc dữ liệu sau khi xử lý và tính toán chỉ số RFM

Sau khi xử lý, bộ dữ liệu còn 913.947 bản ghi với 13 biến, gồm thông tin nhân khẩu học (Gender, Location, Age), hành vi tài chính (AccountBalance, TransactionAmount) và các chỉ số RFM (Monetary, Frequency, Recency).

Trong đó, từ biến ngày gốc TransactionDate, nhóm phân tích đã trích xuất thêm các trường liên quan đến tháng và ngày, dưới dạng số (TransactionMonth, TransactionDay) và tên tương ứng (TransactionMonthName, TransactionDayName).

Việc tách này giúp:

- Phân tích xu hướng chi tiêu theo ngày trong tuần hoặc tháng trong năm.
- Trực quan hóa hành vi khách hàng theo thời gian.

	# CustAccountBalance	⊕ TransactionDate	# TransactionAmount (INR)	# Age	# Monetary
count	913947.0	913947	913947.0	913947.0	913947.0
mean	89142.33409339929	2016-07-25 11:02:26.871974400	1390.3402733637727	48.15030412047963	1850.825315614581
min	0.0	2016-01-08 00:00:00	0.0	1.0	0.0
25%	4418.95	2016-06-09 00:00:00	150.0	33.0	205.77
50%	15405.55	2016-08-20 00:00:00	419.0	37.0	598.0
75%	50629.19	2016-09-09 00:00:00	1091.22	42.0	1598.0
max	55369688.16	2016-12-09 00:00:00	1560034.99	225.0	1560034.99
std	460546.80927779083	Missing value	5886.169754014142	44.71007826803003	6748.783169473995

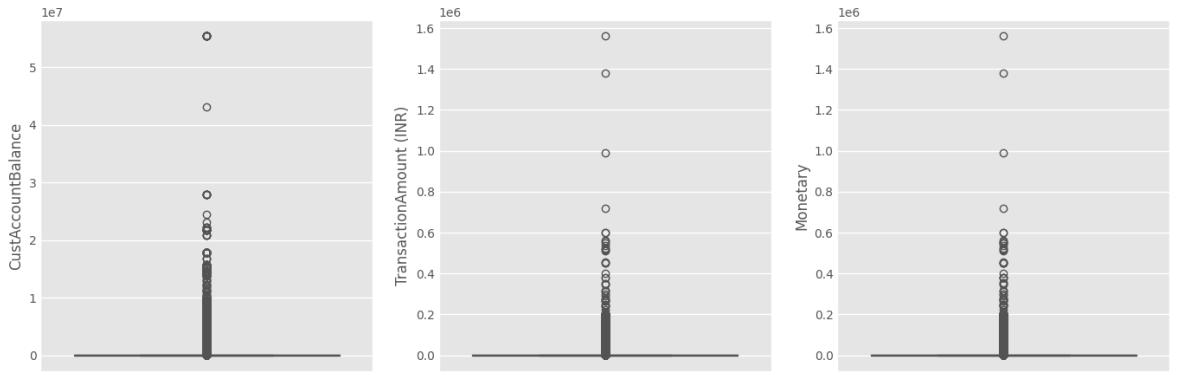
  

# Frequency	# Recency	# TransactionMonth	# TransactionDay
913947.0	913947.0	913947.0	913947.0
1.3276535729095889	123.70232956615646	7.343049432844574	14.402226824969063
1.0	0.0	1.0	8.0
1.0	85.0	6.0	8.0
1.0	107.0	8.0	9.0
2.0	123.0	9.0	20.0
6.0	336.0	12.0	31.0
0.582794762010825	78.25539716328795	2.6401489943807532	7.2952208270951155

Hình 3.122: Thống kê mô tả các biến số trong dữ liệu

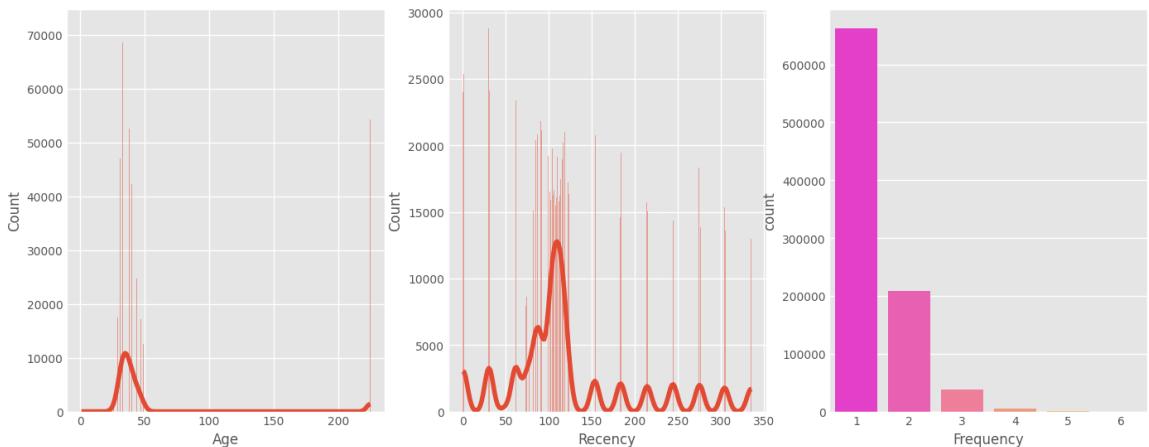
Nhận xét:

- Các biến tài chính như CustAccountBalance, TransactionAmount (INR), và Monetary có giá trị trung bình và độ lệch chuẩn rất lớn, đặc biệt Monetary có giá trị tối đa lên tới hơn 1.5 triệu INR, thể hiện sự chênh lệch rất cao giữa các khách hàng.
- Trong khi đó, các biến như Frequency (số lần giao dịch), TransactionMonth, và TransactionDay lại có giá trị nhỏ, nằm trong phạm vi hẹp.
- Chỉ số Recency cũng có độ phân tán lớn (từ 1 đến 336 ngày), cho thấy sự khác biệt lớn trong thời điểm giao dịch gần nhất giữa các khách hàng.



Hình 3.123: Boxplot các biến tài chính

- Thể hiện phân bố của ba biến tài chính quan trọng qua biểu đồ boxplot. Có thể thấy, cả ba biến đều xuất hiện nhiều giá trị ngoại lai (outliers), đặc biệt là Monetary và CustAccountBalance với các điểm vượt xa phạm vi giá trị trung bình.
- Điều này cho thấy dữ liệu tài chính có phân phối lệch, và việc xử lý outlier hoặc chuẩn hóa là rất cần thiết để mô hình phân cụm không bị ảnh hưởng bởi các giá trị bất thường.

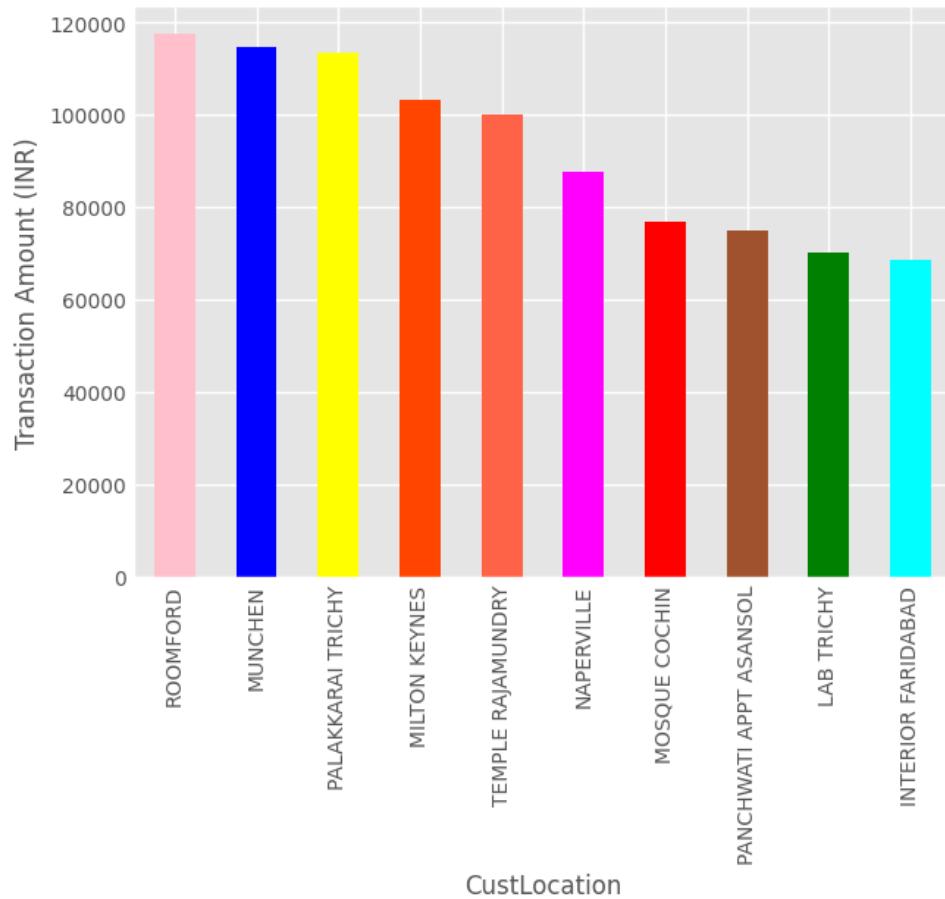


Hình 3.124: Phân phối tuổi, thời gian giao dịch gần nhất và tần suất giao dịch của khách hàng

Nhận xét:

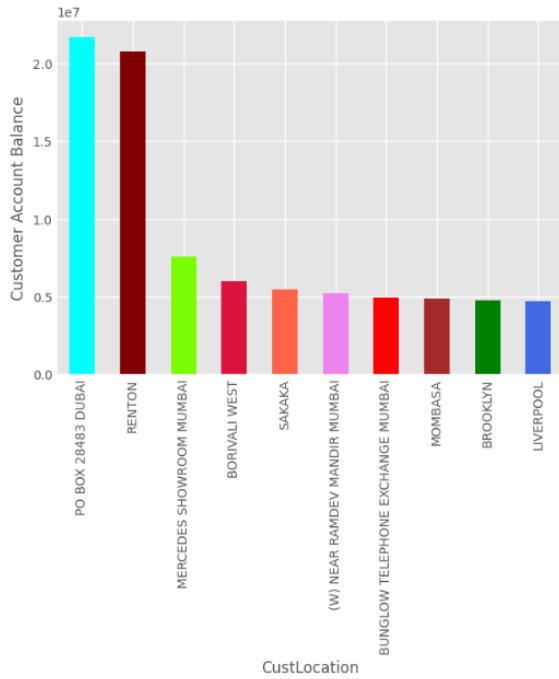
- Tuổi (Age) phân bố chủ yếu trong khoảng 30–50 tuổi, phản ánh nhóm khách hàng chính của ngân hàng. Tuy nhiên, có một số giá trị bất thường ở >200 tuổi nhiều khả năng là do lỗi ngày sinh hoặc giá trị mặc định.

- Recency (số ngày kể từ lần giao dịch gần nhất) dao động rộng, với phần lớn khách hàng thực hiện giao dịch cách đây khoảng 100 – 130 ngày.
- Frequency cho thấy phần lớn khách hàng chỉ có 1 hoặc 2 giao dịch, chứng tỏ tập khách hàng có hành vi tiêu dùng rải rác, không thường xuyên. Điều này sẽ ảnh hưởng mạnh đến phân cụm, khi cần phân biệt giữa khách hàng trung thành và khách hàng một lần



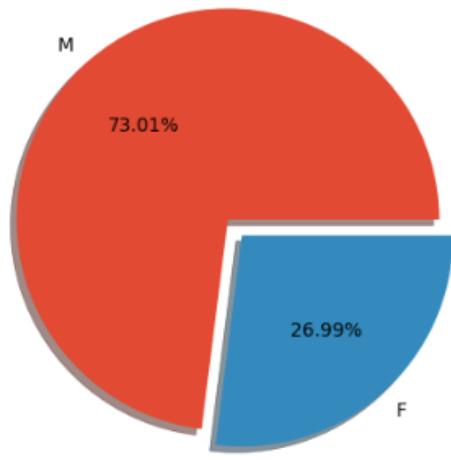
Hình 3.125: Chi tiêu trung bình theo địa điểm khách hàng

- Hình 3.123 cho thấy mức chi tiêu trung bình (tính theo TransactionAmount) của khách hàng tại 10 địa điểm cao nhất.
- Có thể thấy ROOMFORD, MUNCHEN và PALAKKARAI TRICHY là những khu vực có chi tiêu trung bình cao nhất, vượt mức 110.000 INR.
- Điều này cho thấy yếu tố vị trí địa lý có ảnh hưởng rõ rệt đến hành vi tài chính và có thể là một đặc trưng quan trọng trong phân cụm.



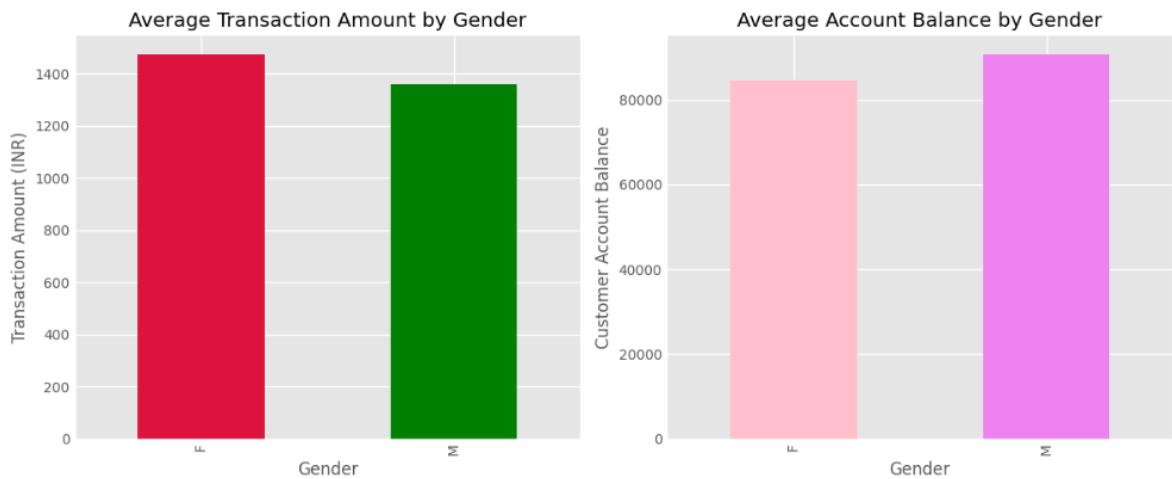
Hình 3.126: Số dư tài khoản trung bình theo địa điểm khách hàng

- Hình 3.124 thể hiện 10 địa điểm có số dư tài khoản trung bình cao nhất.
- Hai khu vực nổi bật là PO BOX 24883 DUBAI và RENTON, với số dư trung bình vượt quá 20 triệu INR – cao hơn hẳn so với các khu vực còn lại.
- Sự khác biệt rõ rệt giữa các địa điểm cho thấy CustLocation là một yếu tố ảnh hưởng mạnh đến tiềm lực tài chính của khách hàng.



Hình 3.127: Tỷ lệ giới tính của khách hàng

- Trong đó, nam giới chiếm 73,01%, còn nữ giới chiếm 26,99%.
- Sự chênh lệch này cho thấy khách hàng nam áp đảo, và yếu tố giới tính có thể là một chiều phân tích quan trọng khi phân cụm đặc biệt khi kết hợp với hành vi tài chính (chi tiêu, số dư, tần suất giao dịch).

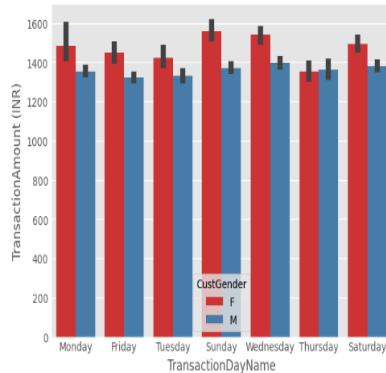


Hình 3.128: So sánh hành vi tài chính trung bình theo giới tính

Hình 3.126 trình bày so sánh giữa nam và nữ về hai khía cạnh tài chính:

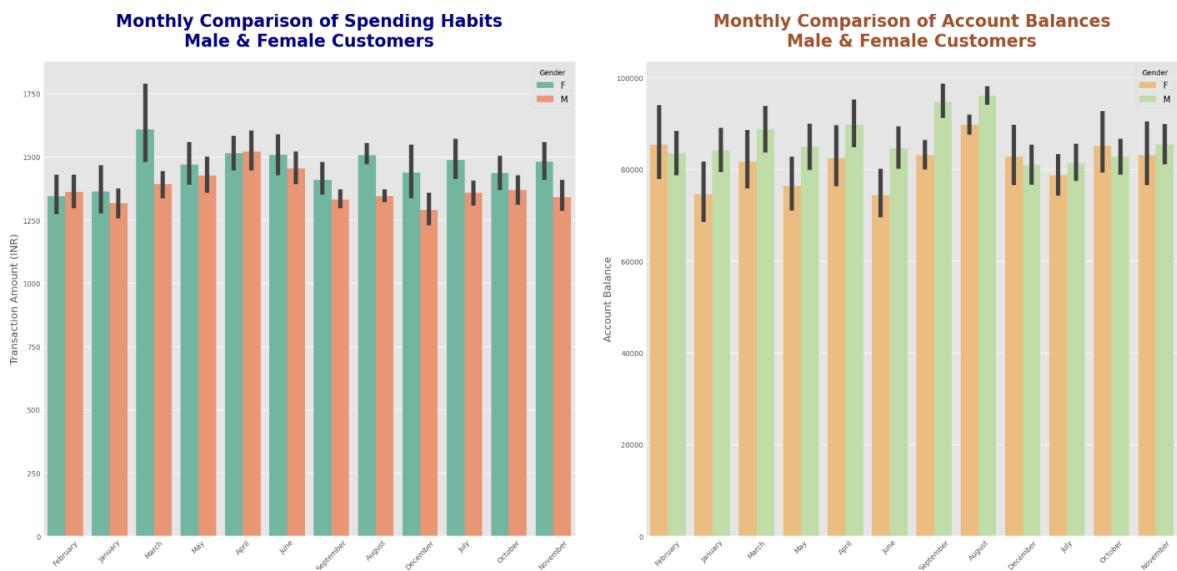
- Trái: Nữ giới có mức chi tiêu trung bình cao hơn nam ( $\approx$  1.470 INR so với 1.360 INR), cho thấy nữ có xu hướng chi mạnh hơn mỗi lần giao dịch.
- Phải: Nam giới lại có số dư tài khoản trung bình cao hơn nữ ( $\approx$  91.000 INR so với 85.000 INR), phản ánh năng lực tài chính tích lũy lớn hơn.

### Weekday-Wise Comparison of Spending Habits of Male & Female Customers



Hình 3.129: So sánh chi tiêu trung bình theo giới tính trong các ngày trong tuần

- Cả hai giới đều chi tiêu mạnh hơn vào cuối tuần, đặc biệt là Chủ Nhật (Sunday) và Thứ Tư (Wednesday),
- Trong suốt các ngày, nữ giới luôn có mức chi tiêu trung bình cao hơn nam giới, nhất quán với các phân tích trước đó.



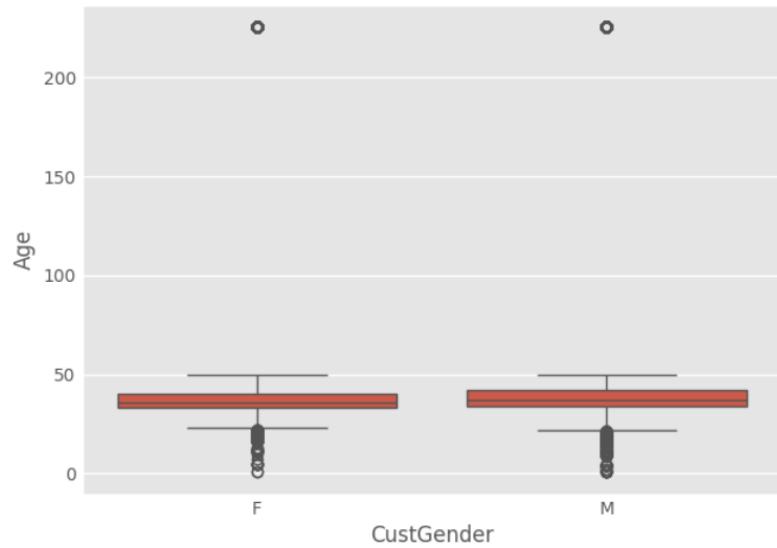
Hình 3.130: So sánh theo tháng về chi tiêu và số dư giữa khách hàng nam và nữ

Hình 3.128 minh họa hai biểu đồ:

- Bên trái: Chi tiêu trung bình (TransactionAmount) của khách hàng theo từng tháng.
- Bên phải: Số dư tài khoản trung bình (Account Balance) theo tháng.

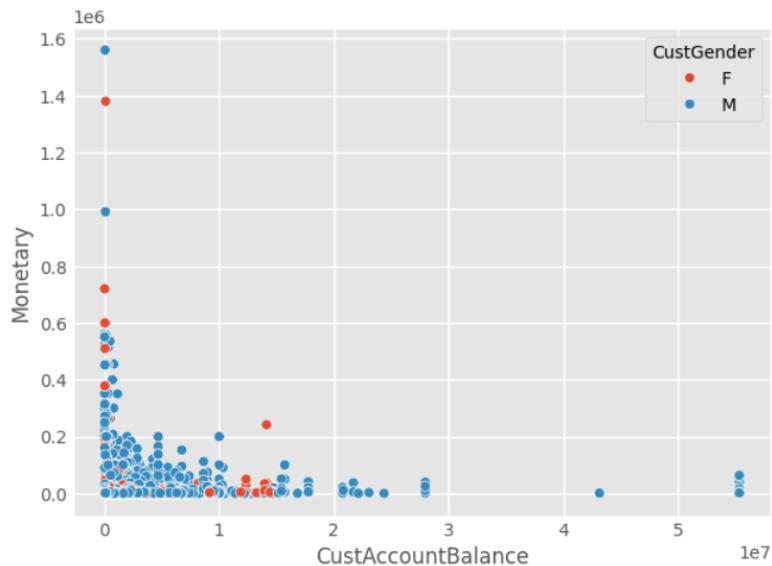
Nhìn chung:

- Nữ giới chi tiêu cao hơn nam giới ở hầu hết các tháng trong đó đặc biệt là vào tháng 3 và tháng 5.
- Nam giới duy trì số dư tài khoản cao hơn nữ giới ở hầu hết các thời điểm. Rõ nhất trên biểu đồ là vào tháng 7 – 9.



Hình 3.131: Phân bố độ tuổi theo giới tính

- Cả hai nhóm nam và nữ có độ tuổi tập trung quanh khoảng 35–40 tuổi, với độ lệch và phân tán tương đương nhau.
- Nhưng xuất hiện một vài outlier rất cao, với độ tuổi lên đến hơn 200, nhiều khả năng là do sai lệch dữ liệu đầu vào (ví dụ năm sinh = 1800,...).



Hình 3.132: Mối quan hệ giữa số dư tài khoản và chi tiêu theo giới tính

Nhận xét:

- Phần lớn khách hàng có số dư và chi tiêu ở mức thấp tập trung tại góc trái dưới

- Có một số khách hàng có số dư cao bất thường nhưng chi tiêu thấp, thể hiện hành vi giữ tiền ít tiêu dùng, thường gặp ở các khách hàng lớn tuổi hoặc doanh nghiệp.
- Một vài cá nhân có Monetary rất cao nhưng CustAccountBalance rất thấp, cho thấy xu hướng chi tiêu mạnh mẽ, có thể là nhóm khách hàng tiềm năng cần chú trọng.
- Cả hai giới đều có sự phân bố tương đồng, tuy nhiên khách hàng nam xuất hiện nhiều hơn ở các vùng ngoại lệ (outlier).

## Data Preprocessing

```
<class 'pandas.core.frame.DataFrame'>
Index: 913947 entries, 0 to 936821
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CustGender       913947 non-null   object 
 1   CustAccountBalance 913947 non-null   float64
 2   TransactionAmount (INR) 913947 non-null   float64
 3   Age              913947 non-null   int64  
 4   Monetary         913947 non-null   float64
 5   Frequency        913947 non-null   int64  
 6   Recency          913947 non-null   int64  
dtypes: float64(3), int64(3), object(1)
memory usage: 55.8+ MB
```

Hình 3.133: Thông tin tổng quan bộ dữ liệu sau khi loại bỏ các cột không cần thiết.

- Sau quá trình xử lý, bộ dữ liệu được rút gọn còn 7 biến quan trọng, phục vụ cho phân cụm khách hàng:
  - CustGender: Giới tính khách hàng
  - CustAccountBalance: Số dư tài khoản
  - TransactionAmount (INR): Số tiền giao dịch
  - Age: Tuổi
  - Monetary, Frequency, Recency: 3 chỉ số RFM đại diện cho hành vi tiêu dùng.

- Các biến mang tính mô tả hoặc không đóng góp nhiều vào mô hình như CustLocation, TransactionDate, TransactionMonth, TransactionMonthName, TransactionDay, và TransactionDayName đã được loại bỏ.
- Việc rút gọn này giúp giảm chiều dữ liệu, loại bỏ thông tin dư thừa và tránh gây nhiễu khi phân cụm.

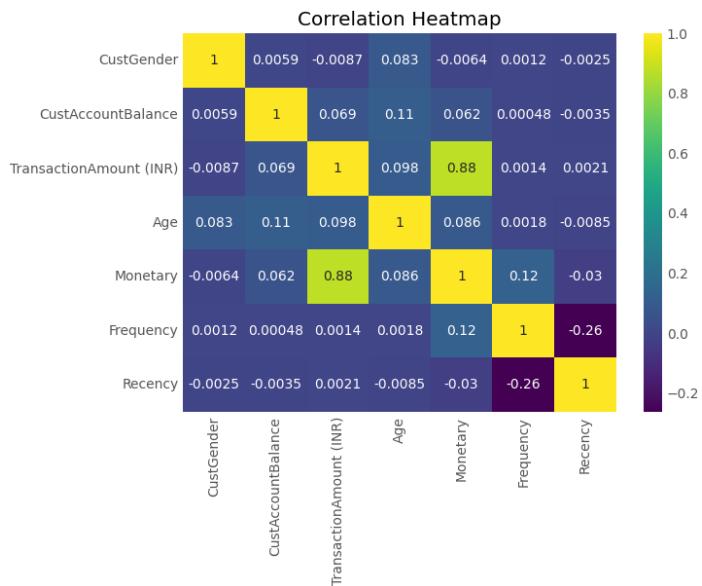
# CustAccountBalance	# TransactionAmount (INR)	# Age	# Monetary	# Frequency	# Recency
0.5	15405.55	419.0	37.0	598.0	1.0
0.95	340680.59	4750.0	225.0	6300.0	2.0

Hình 3.134: . Giá trị phân vị 50% và 95% của các biến quan trọng

Nhìn vào phân vị 50% và 95%, có thể thấy các biến như CustAccountBalance, TransactionAmount (INR) và Monetary có độ lệch lớn giữa trung vị và giá trị 95%, cho thấy phân phối lệch phải và tồn tại nhiều giá trị ngoại lệ lớn. Biến Age cũng xuất hiện giá trị cao bất thường (225 tuổi), cần được xử lý. Các biến Frequency và Recency phân bố chặt hơn, ít bị ảnh hưởng bởi outlier.

```
# func for removing the outliers
def remove_outliers(dataframe, col):
    q1 = dataframe[col].quantile(0.25)
    q3 = dataframe[col].quantile(0.75)
    iqr = q3 - q1
    min_val = q1 - 1.5 * iqr
    max_val = q3 + 1.5 * iqr
    dataframe=dataframe[(dataframe[col]< max_val) & (dataframe[col]> min_val)]
    return dataframe
```

Hình 3.135: Hàm loại bỏ ngoại lệ bằng phương pháp IQR.



Hình 3.136: Biểu đồ heatmap thể hiện hệ số tương quan giữa các biến số trong tập dữ liệu sau khi xử lý.

- TransactionAmount (INR) và Monetary có tương quan rất cao (0.88), hợp lý vì Monetary được tính từ tổng các giao dịch.
- Frequency và Monetary có tương quan dương nhẹ (0.12), cho thấy khách hàng giao dịch nhiều thì giá trị tổng chi tiêu có xu hướng tăng.
- Recency có tương quan âm với Frequency (-0.26), ngụ ý rằng khách hàng giao dịch gần đây thì thường giao dịch thường xuyên hơn,
- Các biến còn lại có hệ số tương quan rất nhỏ, cho thấy mối quan hệ yếu hoặc không đáng kể.

<b>CustGender</b>	0
<b>CustAccountBalance</b>	118851
<b>TransactionAmount (INR)</b>	96438
<b>Age</b>	54525
<b>Monetary</b>	90776
<b>Frequency</b>	5138
<b>Recency</b>	236166
<b>dtype:</b>	<b>int64</b>

Hình 3.137: Số lượng giá trị bị thiếu sau khi loại bỏ outlier bằng phương pháp IQR.

Việc loại bỏ outlier bằng phương pháp IQR dẫn đến xuất hiện nhiều giá trị thiếu ở các biến quan trọng như Recency, CustAccountBalance, TransactionAmount

(INR) và Monetary. Để đảm bảo chất lượng dữ liệu, các dòng chứa giá trị thiếu này sẽ được loại bỏ hoàn toàn trong bước xử lý tiếp theo.

```
# Impute từng cột quan trọng thay vì drop toàn bộ
han_out['CustAccountBalance'].fillna(han_out['CustAccountBalance'].median(), inplace=True)
han_out['TransactionAmount (INR)'] = han_out.groupby('CustGender')['TransactionAmount (INR)'].transform(lambda x: x.fillna(x.median()))
han_out['Age'].fillna(han_out['Age'].median(), inplace=True)

median_monetary = han_out['Monetary'].median()
han_out['Monetary'].fillna(median_monetary, inplace=True)
# Với Recency: Tạo cột flag và điền giá trị đặc biệt (ví dụ: -1)
han_out['is_new_customer'] = han_out['Recency'].isnull().astype(int)
han_out['Recency'].fillna(-1, inplace=True)

# Drop chỉ với Frequency (do tỷ lệ null rất thấp)
han_out.dropna(subset=['Frequency'], inplace=True)
```

Hình 3.138: Xử lý giá trị thiếu và tạo biến cờ is\_new\_customer

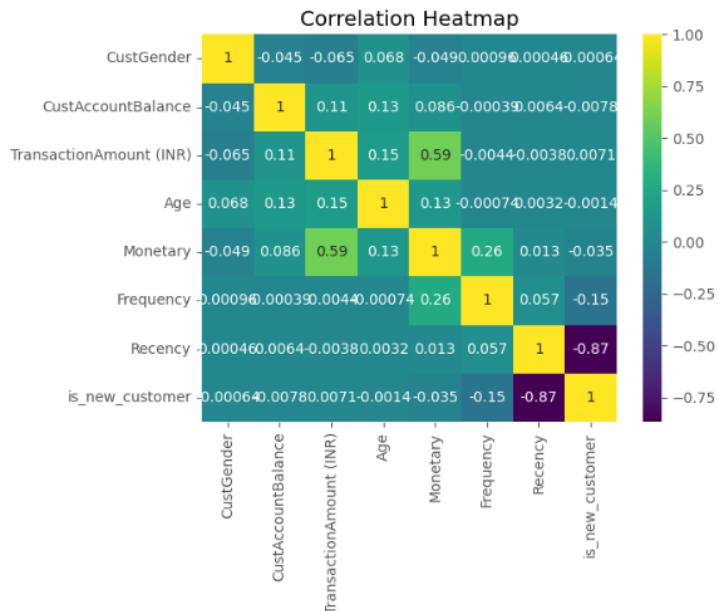
Nhận xét:

- Cột CustAccountBalance, Age và Monetary được điền giá trị thiếu bằng trung vị toàn cục nhằm giữ lại thông tin và tránh mất dữ liệu.
- Cột TransactionAmount (INR) được xử lý nâng cao hơn khi điền median theo từng nhóm giới tính (CustGender), cho phép bảo toàn sự khác biệt hành vi giữa các nhóm.
- Với Recency, nhóm tạo thêm biến cờ is\_new\_customer để đánh dấu khách hàng mới (Recency bị thiếu), sau đó điền giá trị bằng 1.
- Riêng cột Frequency do tỷ lệ thiếu rất nhỏ nên được loại bỏ trực tiếp các dòng thiếu.

CustGender	0
CustAccountBalance	0
TransactionAmount (INR)	0
Age	0
Monetary	0
Frequency	0
Recency	0
is_new_customer	0
dtype:	int64

Hình 3.139: Kết quả sau xử lý

Tất cả các cột quan trọng đều đã được xử lý khuyết dữ liệu hoàn toàn (0 giá trị thiếu), sẵn sàng cho bước chuẩn hóa và phân cụm.



Hình 3.140: Ma trận tương quan giữa các biến đầu vào sau xử lý

Biểu đồ thể hiện mức độ tương quan giữa các biến số. Các giá trị tương quan đều nhỏ hơn 0.6, cho thấy không có hiện tượng đa cộng tuyến nghiêm trọng. Đáng chú ý, biến Recency có tương quan nghịch khá mạnh với biến cờ is\_new\_customer (-0.87), phù hợp với kỳ vọng khi khách hàng mới thường có Recency cao (tức mới phát sinh giao dịch gần đây). Điều này cũng xác nhận ý nghĩa đúng đắn của biến đặc trưng mới được tạo ra.

```
scaler = StandardScaler()
df_scaled= scaler.fit_transform(df_cleaned)
df_scaled.shape
✓ 0.3s
(908809, 8)
```

Hình 3.141: Kết quả sau khi chuẩn hóa dữ liệu bằng StandardScaler

Tập dữ liệu sau khi làm sạch đã được chuẩn hóa với StandardScaler, đảm bảo các biến đầu vào có cùng phân phối chuẩn với trung bình bằng 0 và độ lệch chuẩn bằng 1. Kết quả trả về 908.809 quan sát với 8 biến đặc trưng, sẵn sàng để đưa vào các mô hình phân cụm như K-Means hoặc DBSCAN. Việc chuẩn hóa giúp cải thiện hiệu quả học và tránh hiện tượng các biến có đơn vị lớn chi phối kết quả phân cụm.

### 3.5.3. Bài toán 1: Phân cụm bằng K – Means

#### Mục tiêu:

Phân nhóm khách hàng dựa trên hành vi giao dịch và đặc điểm nhân khẩu học, từ đó giúp doanh nghiệp xây dựng các chiến lược chăm sóc và tiếp thị phù hợp cho từng phân khúc.

#### Phương pháp:

Sau khi hoàn tất quá trình tiền xử lý và chuẩn hóa dữ liệu, nhóm tiến hành phân cụm bằng thuật toán K-Means dựa trên các đặc trưng chính sau:

- Thông tin nhân khẩu học: Giới tính (CustGender), tuổi (Age).
- Hành vi tài chính: Số dư tài khoản (CustAccountBalance), giá trị giao dịch (TransactionAmount), và bộ 3 chỉ số RFM gồm Monetary, Frequency, Recency.

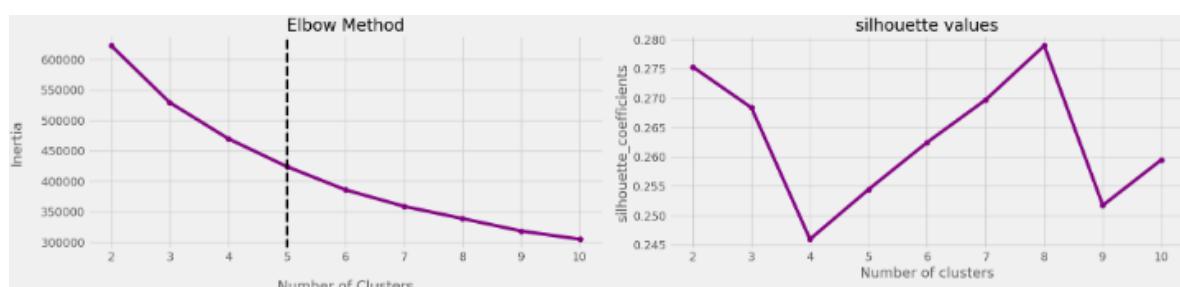
Để đảm bảo hiệu năng và tính ổn định khi huấn luyện mô hình trên tập dữ liệu lớn, nhóm đã lấy mẫu ngẫu nhiên 100.000 bản ghi từ dữ liệu gốc.

Mô hình K - Means được cấu hình với:

- Khởi tạo ngẫu nhiên (init='random').
- Số lần lặp tối đa (max\_iter=100)
- random\_state=42 để tái lập kết quả

#### Xác định số cụm (k) tối ưu:

- Elbow Method – phân tích độ “gãy” trong đường cong inertia.
- Silhouette Score – đo lường độ tách biệt giữa các cụm.



Hình 3.142: Biểu đồ Elbow và Silhouette Score theo số cụm (k)

## Kết quả:

- Biểu đồ Elbow cho thấy điểm gấp khúc rõ rệt tại  $k = 5$  đây là số cụm được lựa chọn.
- Silhouette Score đạt mức cao tại  $k = 2, 3, 5$  và  $8$ , tuy nhiên  $k = 5$  là điểm cân bằng tốt giữa độ tách biệt và cấu trúc cụm rõ ràng.
- Chọn  $k = 5$  để tiến hành phân cụm bằng K-Means.

```
silhouette_avg = silhouette_score(df_scaled_kmeans, labels)
print(f"Chỉ số Silhouette trung bình: {silhouette_avg}")
✓ 3m 7.7s
Chỉ số Silhouette trung bình: 0.2544491699012627
```

Hình 3.143: Chỉ số Silhouette trung bình

- Silhouette trung bình =  $0.254 \rightarrow$  mức độ tách biệt cụm ở mức chấp nhận được.
- Điều này cho thấy các cụm có độ phân biệt vừa phải, có thể sử dụng để phân tích nhóm khách hàng nhưng cần kết hợp trực quan hóa để hiểu rõ hơn hành vi từng cụm.

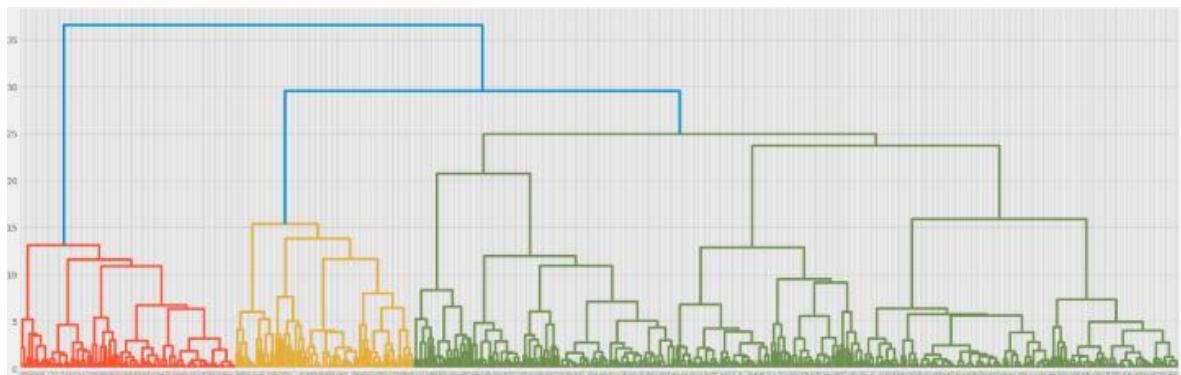
```
df_scaled_kmeans['cluster'].value_counts()
✓ 0.0s
cluster
3    30668
2    23743
0    16077
1    15828
4    13684
Name: count, dtype: int64
```

Hình 3.144: Phân bổ số lượng khách hàng theo cụm K-Means ( $k = 5$ )

- Cụm 3 là nhóm chiếm tỷ lệ lớn nhất (hơn 30 nghìn khách hàng), có thể là nhóm khách hàng phô biến với hành vi trung bình.
- Các cụm còn lại có quy mô từ  $\sim 13.000$  đến  $\sim 23.000$  khách, giúp đảm bảo phân bố cụm tương đối cân bằng, không quá lệch.
- Việc có cụm đông đảo hơn cũng có thể là cơ sở để phân tích sâu, cá nhân hóa chăm sóc hoặc upsale cho nhóm này

## Kiểm chứng cấu trúc cụm

### Dendrogram kiểm tra trực quan cấu trúc phân nhánh tự nhiên của dữ liệu

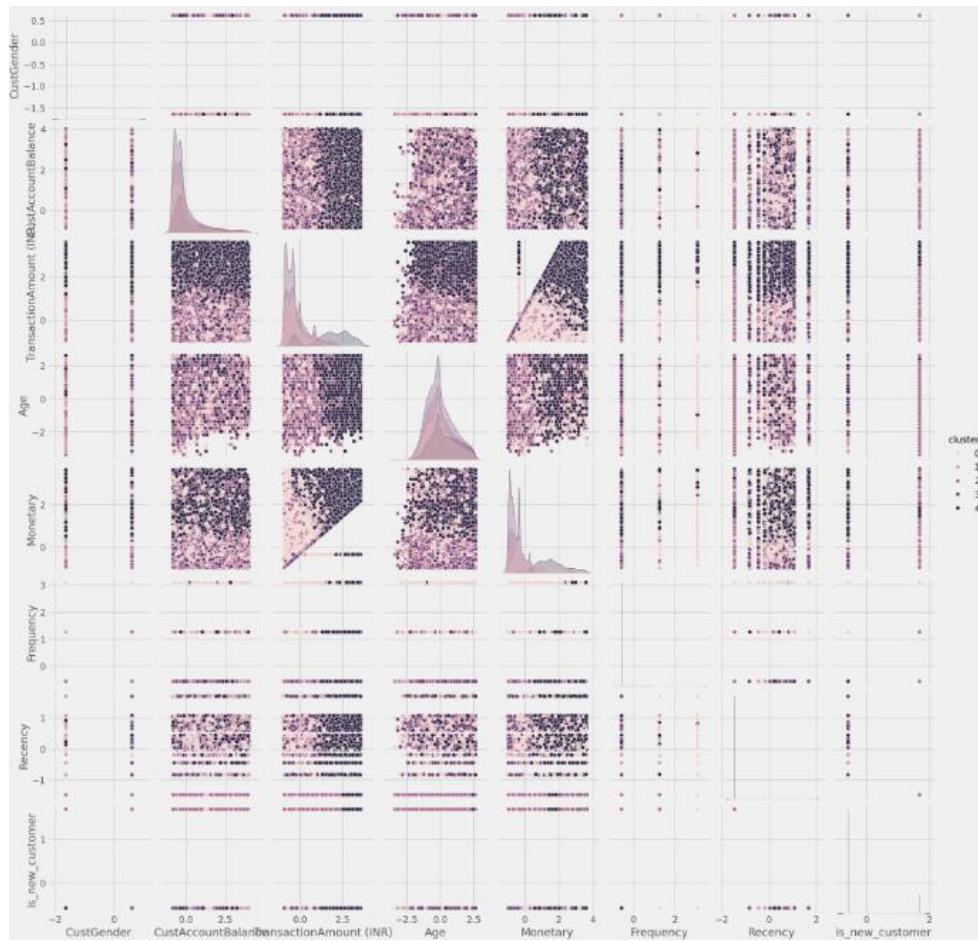


Hình 3.145: Dendrogram từ Hierarchical Clustering

#### Nhận xét:

- Dendrogram cho thấy cấu trúc phân nhánh tự nhiên trong dữ liệu khách hàng.
- Khi cắt cây tại độ cao khoảng 20–22, có thể quan sát thấy 4–5 cụm chính được hình thành.
- Khoảng cách giữa các nhánh chính rộng, cho thấy mức độ khác biệt rõ rệt giữa các nhóm khách hàng.
- Dendrogram cũng cỗ lựa chọn  $k = 5$  từ mô hình K-Means, cho thấy kết quả phân cụm là hợp lý và có tính trực quan.

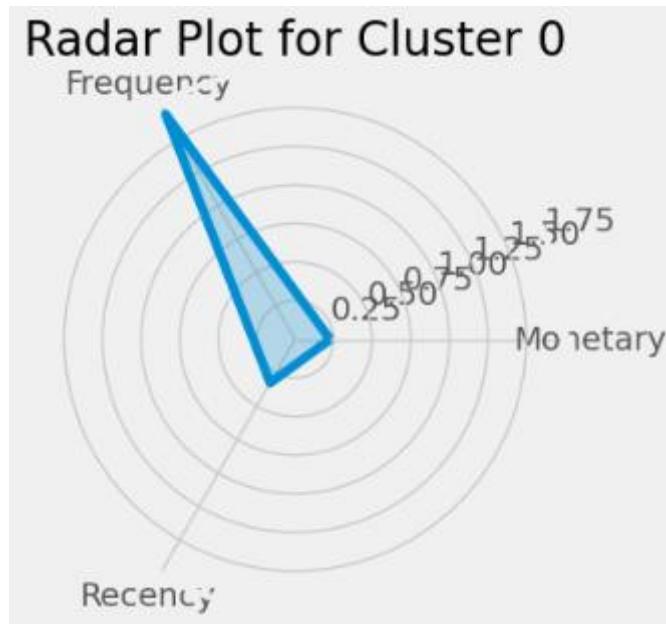
### Pairplot trực quan hóa sự phân tách cụm sau khi gán nhãn từ K – Means



Hình 3.146: Biểu đồ pairplot sau khi phân cụm KMeans

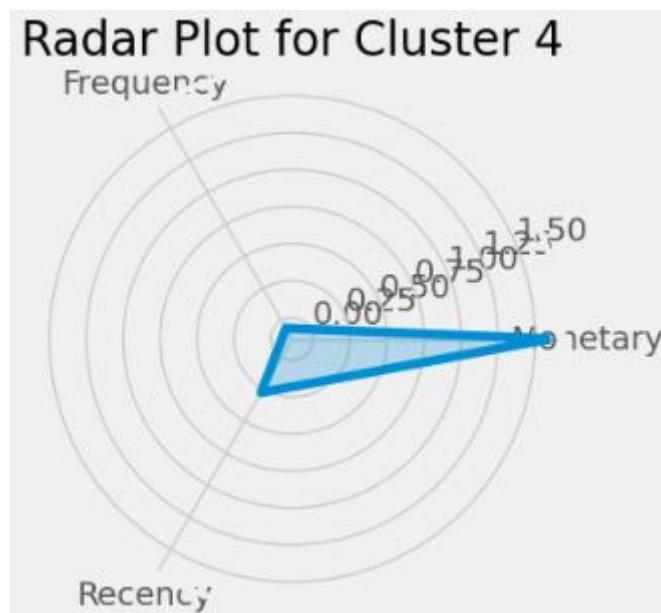
### Nhận xét:

- Một số cặp biến thể hiện sự phân tách rõ giữa các cụm như:
  - Monetary - Recency, Age - Frequency, CustAccountBalance - TransactionAmount
- Các cụm khách hàng có xu hướng tách biệt theo các trục quan trọng, hỗ trợ mô hình KMeans hoạt động hiệu quả.
- Một số chiều như CustGender hay is\_new\_customer có độ phân tách thấp hơn nhưng vẫn góp phần phân cụm bổ sung.
- Mặc dù vẫn có hiện tượng chồng lấn nhẹ giữa các cụm, điều này là bình thường trong các bài toán dữ liệu hành vi liên tục.



Hình 3.147: Biểu đồ radar mô tả cụm 0 theo 3 chỉ số RFM

- Cluster 0 có Frequency rất cao, trong khi Monetary và Recency ở mức thấp, cho thấy nhóm khách hàng này thường xuyên giao dịch, gần đây, nhưng với giá trị nhỏ.
  - Đây có thể là nhóm khách hàng trung thành nhưng chi tiêu thấp.
- Do đó, có thể thúc đẩy nhóm này bằng các chương trình upsell, combo, hoặc nâng hạng tài khoản để tăng giá trị giao dịch.



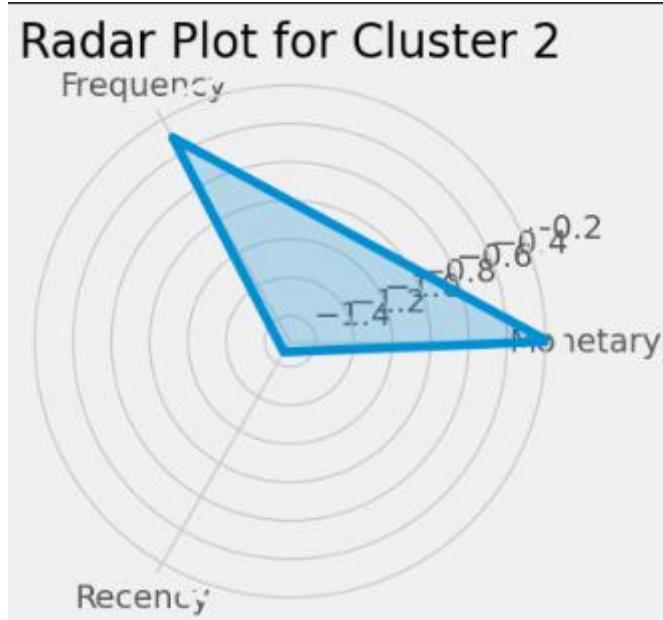
Hình 3.148: Biểu đồ radar mô tả cụm 4 theo 3 chỉ số RFM

- Cluster 4 có Monetary rất cao, nhưng Frequency và Recency thấp, cho thấy đây là khách hàng chi tiêu lớn nhưng giao dịch không thường xuyên và không quay lại gần đây.
  - Nhóm này có thể là những VIP cũ hoặc giao dịch một lần với giá trị lớn
- Đề xuất doanh nghiệp tái kích hoạt nhóm này bằng các chương trình tri ân riêng, ưu đãi độc quyền, hoặc liên hệ cá nhân hóa để khuyến khích quay lại



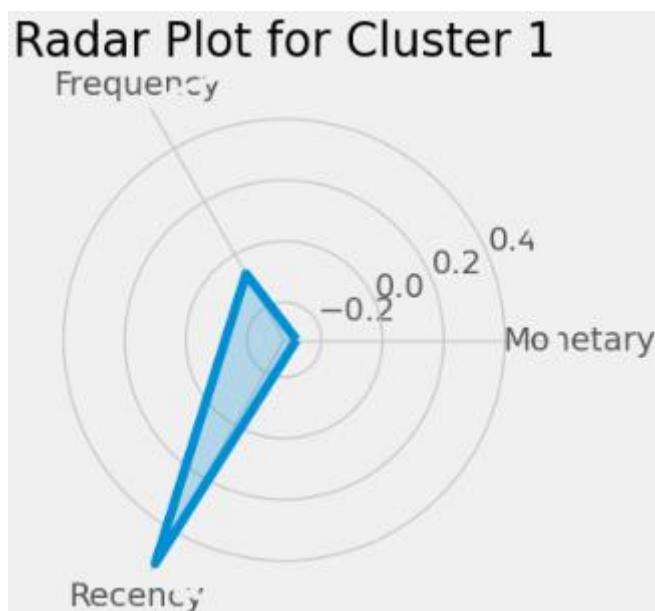
Hình 3.149: Biểu đồ radar mô tả cụm 3 theo 3 chỉ số RFM

- Cluster 3 có Recency rất thấp (tức là mua hàng rất lâu rồi), cùng với Frequency và Monetary đều thấp, cho thấy đây là nhóm khách hàng đã ngừng giao dịch từ lâu và có giá trị thấp.
  - Đây là nhóm không nên ưu tiên trong chiến dịch marketing vì khả năng quay lại thấp, ROI không cao.
- Doanh nghiệp nên loại khỏi các chiến dịch ưu đãi chính, hoặc chỉ tiếp cận lại nếu có chương trình rộng



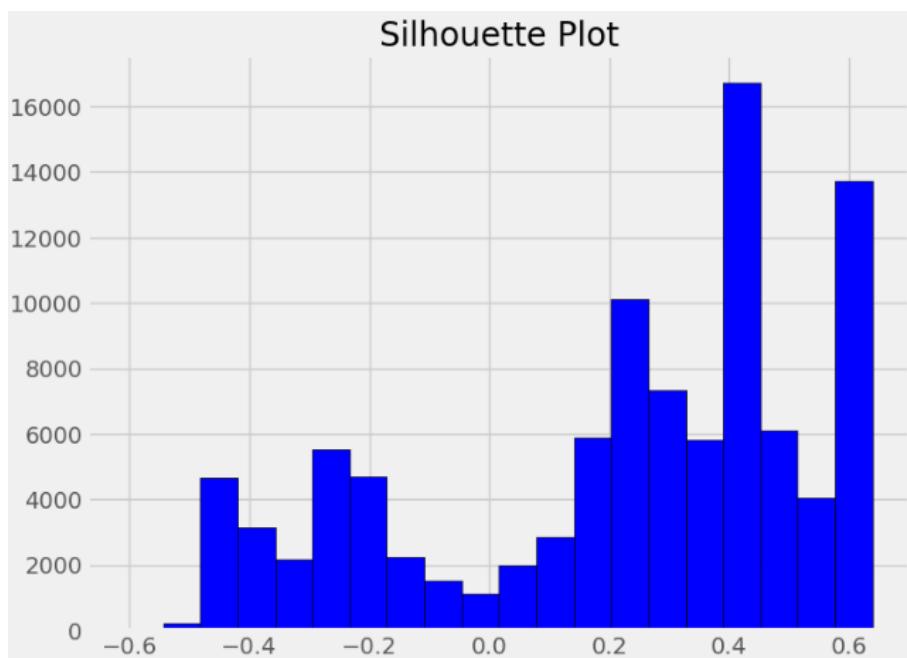
Hình 3.150: Biểu đồ radar mô tả cụm 2 theo 3 chỉ số RFM

- Cluster 2 có Monetary và Frequency đều cao, trong khi Recency rất thấp → đây là khách hàng cũ có giá trị cao, nhưng đã lâu không quay lại mua hàng.
  - Nhóm này từng trung thành và chi tiêu nhiều, cần chiến dịch tái kích hoạt như khuyến mãi riêng, email cá nhân hóa.
- Rất đáng đầu tư để giữ chân hoặc thu hút quay lại, vì họ từng mang lại doanh thu lớn.



Hình 3.151: Biểu đồ radar mô tả cụm 1 theo 3 chỉ số RFM

- Cluster 1 có Recency cao, Frequency và Monetary rất thấp → đây là nhóm khách hàng mới, vừa mới phát sinh giao dịch đầu tiên và giá trị giao dịch còn nhỏ,
  - Tiềm năng phát triển cao nếu được chăm sóc tốt.
- Nên đầu tư vào chiến dịch chào mừng, tặng voucher lần sau, tư vấn hỗ trợ để upsale và tăng tần suất mua.



Hình 3.152: Biểu đồ phân bố chỉ số Silhouette cho các điểm dữ liệu sau phân cụm bằng Kmeans

- Đa phần các điểm có giá trị Silhouette dương, tập trung nhiều ở khoảng 0.2 đến 0.6, cho thấy sự phân cụm là hợp lý và rõ ràng đối với phần lớn dữ liệu.
- Một số điểm có Silhouette âm, tức là bị phân cụm chưa tốt, có thể nằm gần biên giữa các cụm hoặc không thuộc cụm rõ ràng nào.
- Tổng thể mô hình KMeans đạt chất lượng khá, với khả năng tách biệt tương đối tốt giữa các nhóm khách hàng. Có thể cải thiện thêm bằng cách tinh chỉnh đặc trưng hoặc tăng số lượng cụm.

### 3.5.4. Phân cụm khách hàng bằng K – Means sau khi giảm chiều bằng PCA

#### Mục tiêu

Để tăng hiệu quả phân cụm và giảm nhiễu trong dữ liệu gốc, nhóm áp dụng kỹ thuật PCA (Principal Component Analysis) trước khi huấn luyện mô hình K - Means. PCA giúp:

- Tóm tắt thông tin dữ liệu vào một không gian mới ít chiều hơn.
- Loại bỏ biến dư thừa và nhiễu.
- Hỗ trợ mô hình hoạt động ổn định hơn và tăng khả năng trực quan hóa kết quả.

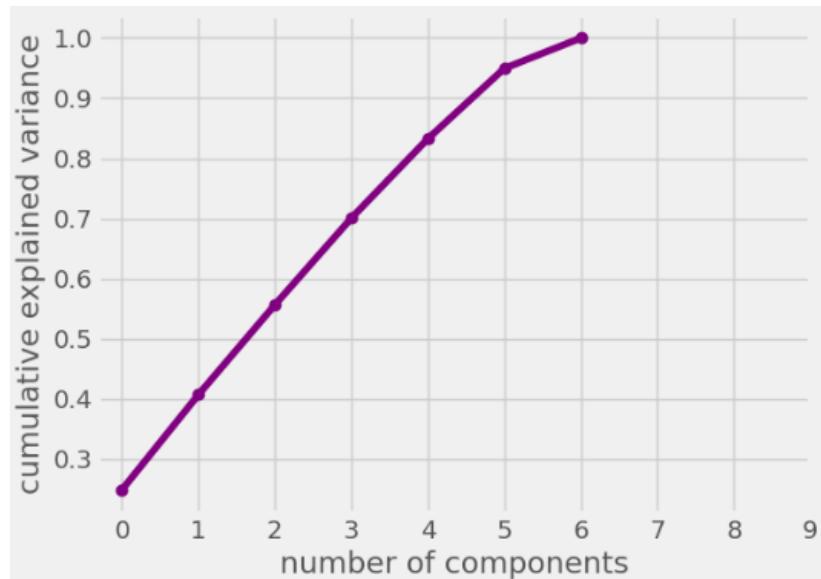
#### Phương pháp

- Giảm chiều dữ liệu bằng PCA
  - Sau bước chuẩn hóa, nhóm tiến hành áp dụng PCA để xác định số thành phần chính cần giữ lại. Đoạn mã sau minh họa quy trình xác định số chiều cần thiết để giải thích  $\geq 90\%$  phuông sai của dữ liệu.

```
# here we want to check how many component we need to explain 90% of the variance
from sklearn.decomposition import PCA

plt.style.use("fivethirtyeight")
pca = PCA().fit(df_pca_kmeans.iloc[:, :-1])
plt.plot(np.cumsum(pca.explained_variance_ratio_), color='purple', marker='o') #EX: cumsum([4,2,3,1,6]) ==>[4,6,9,10,16]
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
# x ticks
plt.xticks(np.arange(0, 10, 1))
plt.show();
```

Hình 3.153: Đoạn mã thực hiện



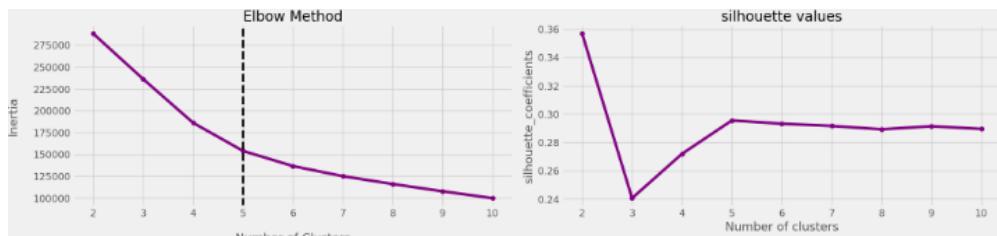
Hình 3.154: Biểu đồ phuơng sai tích lũy theo số thành phần PCA

Nhận xét:

- 6 thành phần chính đầu tiên đã giải thích 100% phuơng sai của dữ liệu, trong đó:
  - 3 thành phần đầu tiên đã giải thích ~85–90%, cho thấy phần lớn thông tin đã được giữ lại chỉ với vài thành phần.
- Điều này cho thấy PCA là công cụ hiệu quả để giảm chiều dữ liệu, giúp tăng tốc độ huấn luyện, trực quan hóa tốt hơn mà không mất nhiều thông tin.
- Sử dụng 3 thành phần đầu tiên là hợp lý cho việc phân cụm bằng KMeans ở bước tiếp theo.

### **Huấn luyện lại K – Means trên dữ liệu PCA**

- Áp dụng PCA để giảm số chiều xuống 3 thành phần chính, giữ lại phần lớn phuơng sai dữ liệu (trên 90%).
- Elbow method và Silhouette Score tiếp tục được sử dụng để đánh giá số cụm phù hợp



Hình 3.155: Biểu đồ Elbow và Silhouette Score sau PCA

Nhận xét:

- Biểu đồ Elbow và Silhouette Score (Hình 3.155) tiếp tục xác nhận số cụm tối ưu là  $k = 5$ , tương tự kết quả trước khi giảm chiều.

Nhận định:

- PCA giúp giảm nhiễu và làm rõ cấu trúc cụm trong dữ liệu, đồng thời cải thiện chất lượng phân nhóm.
- Việc giảm chiều xuống 3 thành phần còn tạo điều kiện thuận lợi cho việc trực quan hóa 3D các cụm khách hàng.

## Kết quả

```
silhouette_avg = silhouette_score(pca_data_kmeans, labels)
print(f"Chỉ số Silhouette trung bình: {silhouette_avg}")

số Silhouette trung bình: 0.29568889550548055
```

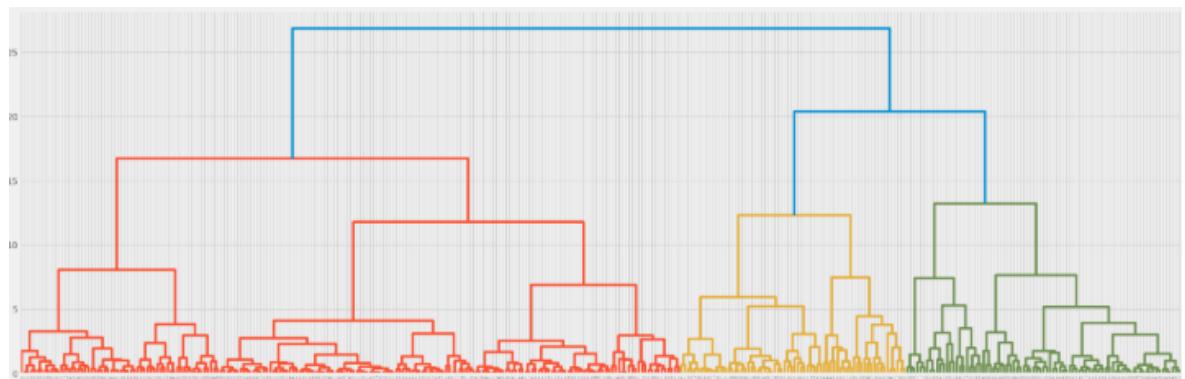
Hình 3.156: Silhouette Score sau khi PCA

- Chỉ số Silhouette trung bình đạt 0.296, cao hơn so với KMeans gốc (0.254) → cho thấy mô hình phân cụm sau PCA có sự phân tách tốt hơn giữa các nhóm khách hàng.
- Việc giảm chiều bằng PCA không những không làm mất thông tin mà còn giúp tăng độ chính xác và hiệu quả tính toán của mô hình.

## Kiểm chứng cấu trúc cụm sau PCA

Để kiểm tra tính tự nhiên của việc phân nhóm khách hàng trong không gian PCA, nhóm tiếp tục áp dụng phương pháp phân cụm phân cấp (Hierarchical Clustering) và trực quan hóa kết quả bằng biểu đồ dendrogram. Việc này giúp đánh

giá mức độ phân tách giữa các nhóm dữ liệu ngay cả khi chưa áp dụng mô hình phân cụm chính thức.



Hình 3.157: Dendrogram sau khi giảm chiều bằng PCA

Nhận xét:

- Biểu đồ dendrogram cho thấy dữ liệu sau PCA vẫn duy trì cấu trúc phân nhánh rõ ràng, phản ánh sự tồn tại của các phân khúc khách hàng có đặc trưng tương đồng.
- Khi cắt cây ở độ cao khoảng 15–18 trên trục tung, có thể quan sát được khoảng 4–5 cụm chính hình thành → cung cấp thêm cho việc chọn  $k = 5$  là hợp lý.
- Các nhánh chính có độ cao phân tách đáng kể, cho thấy mức độ khác biệt cao giữa các cụm.
- Các nhánh chính có độ cao phân tách đáng kể, cho thấy mức độ khác biệt cao giữa các cụm.

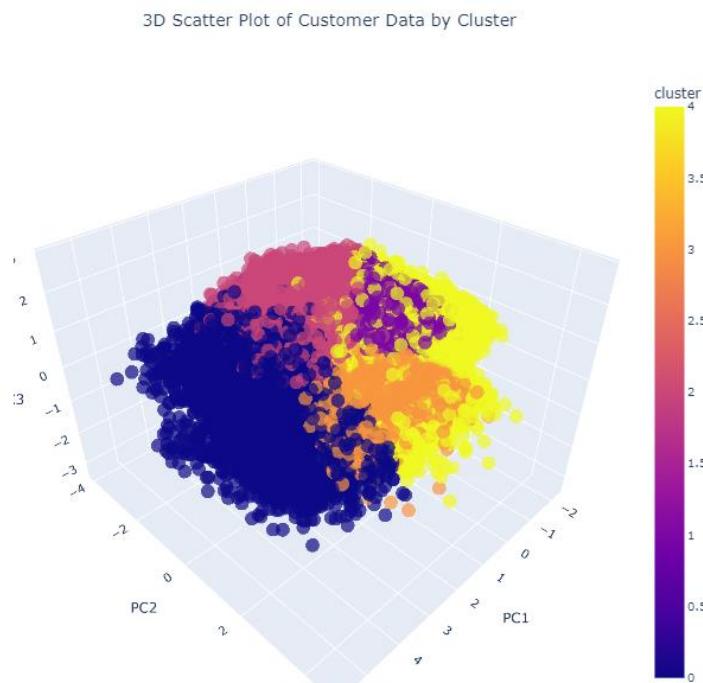
**Kết luận:** Dendrogram là bằng chứng bổ sung quan trọng, xác nhận rằng dữ liệu sau PCA vẫn giữ được cấu trúc phân nhóm rõ ràng, giúp mô hình KMeans hoạt động hiệu quả và có ý nghĩa thực tiễn.

cluster
4 31761
1 20413
3 18365
2 16863
0 12598
Name: count, dtype: int64

Hình 3.158: Phân bố số lượng khách hàng trong từng cụm sau phân cụm bằng KMeans trên dữ liệu PCA.

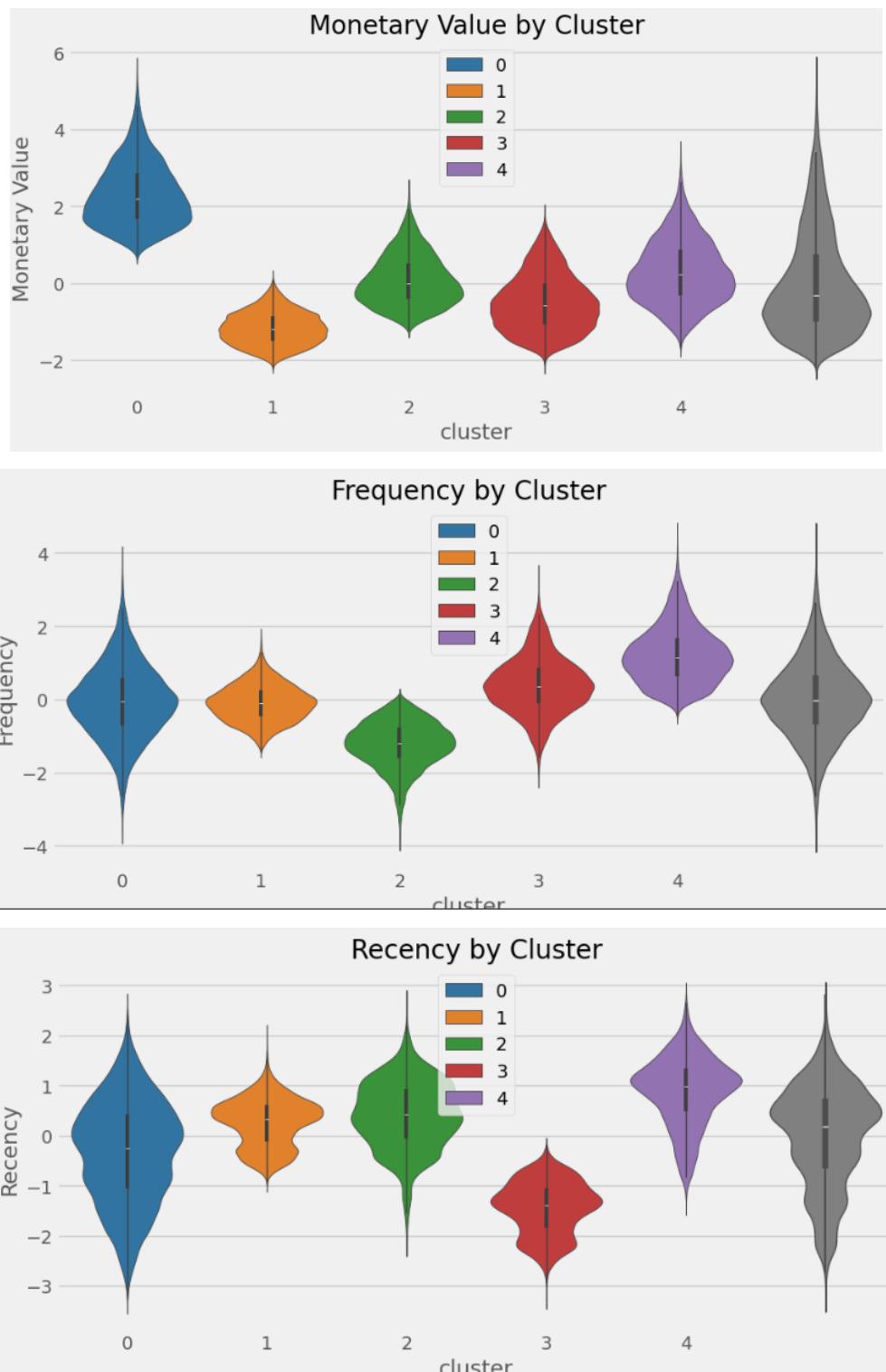
Nhận xét:

- Cụm 4 là cụm lớn nhất, chiếm gần 1/3 dữ liệu, phản ánh đây là nhóm khách hàng phổ biến nhất theo các đặc trưng đã chọn.
- Các cụm còn lại phân bố tương đối đều nhau, cho thấy dữ liệu không bị mất cân bằng quá mức sau khi giảm chiều và phân cụm.
- Sự đa dạng trong kích thước cụm giúp doanh nghiệp có thể xây dựng các chiến lược tiếp cận khác nhau cho từng nhóm, ví dụ:
  - Cụm lớn → nhóm đại trà, cần tối ưu hóa hiệu suất phục vụ.
  - Cụm nhỏ hơn → nhóm tiềm năng/đặc biệt, cần ưu đãi hoặc cá nhân hóa.



Hình 3.159: Biểu đồ 3D scatter các cụm khách hàng theo PCA.

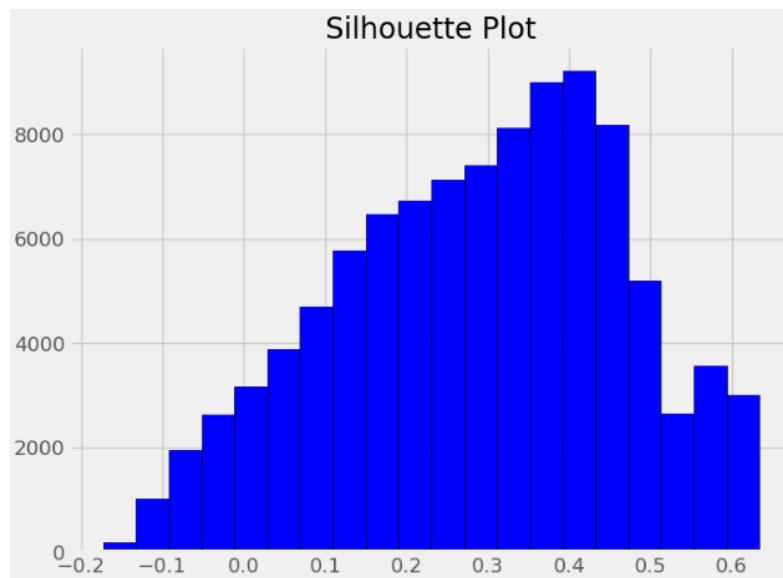
Biểu đồ 3D cho thấy các cụm được phân tách rõ ràng trong không gian PCA, xác nhận hiệu quả của mô hình KMeans sau khi giảm chiều.



Hình 3.160: Biểu đồ violin thể hiện phân bố các giá trị RFM theo từng cụm sau PCA-Kmeans

- Cụm 0 (xanh dương): Monetary cao, Frequency rất cao, Recency thấp.
  - Nhóm khách hàng trung thành, chi tiêu nhiều và vừa giao dịch gần đây.

- Đè xuất: Ưu tiên giữ chân, xây dựng chương trình khách hàng thân thiết, ưu đãi dài hạn
- Cụm 1 (màu cam): Monetary cao, Frequency rất cao, Recency thấp
  - Nhóm khách hàng trung thành, chi tiêu nhiều và vừa giao dịch gần đây
    - Đè xuất: Ưu tiên giữ chân, xây dựng chương trình khách hàng thân thiết, ưu đãi dài hạn.
- Cụm 2 (màu xanh lá cây): Monetary và Frequency thấp, Recency cao.
  - Tương tự cụm 1, nhưng có thể là khách hàng từng trải nghiệm dịch vụ rồi ngừng.
    - Đè xuất: Tiếp cận lại với ưu đãi trải nghiệm lại dịch vụ/sản phẩm.
- Cụm 3 (màu đỏ): Monetary và Frequency trung bình, Recency rất thấp.
  - Khách hàng vừa mới giao dịch gần đây nhưng chi tiêu và tần suất ở mức trung bình.
    - Đè xuất: Tiềm năng upsell nếu chăm sóc đúng lúc.
- Cụm 4 (màu đen): Monetary cao, Frequency và Recency ở mức trung bình.
  - Nhóm khách có giá trị cao nhưng không giao dịch thường xuyên và không quá gần đây.
    - Đè xuất: Cá nhân hóa ưu đãi, mời tham gia chương trình đặc biệt để tăng gắn bó.



Hình 3.161: biểu đồ Silhouette (sau PCA + KMeans)

Nhận xét:

- Phân lớn giá trị Silhouette nằm trong khoảng 0.2 đến 0.45, với đỉnh khoảng 0.4, cho thấy đa số điểm được phân cụm tương đối tốt.
- Một số điểm có Silhouette âm ( $< 0$ ), nghĩa là bị phân cụm sai hoặc nằm ở ranh giới giữa các cụm.

### 3.5.5. Bài toán 2: Phân cụm bằng DBSCAN

#### Mục tiêu

Nhằm đánh giá khả năng phân cụm khách hàng bằng các thuật toán không cần xác định trước số cụm, nhóm tiến hành triển khai mô hình DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN có ưu điểm vượt trội khi làm việc với dữ liệu có phân bố không đều, có khả năng phát hiện điểm nhiễu (noise) và tự động xác định số cụm.

#### Phương pháp

##### Xác định tham số eps bằng K – Distance Graph

- Sử dụng  $k = 28$  để tính khoảng cách hàng xóm gần nhất (theo kinh nghiệm:  $k \approx 2 * \text{số chiều}$ )

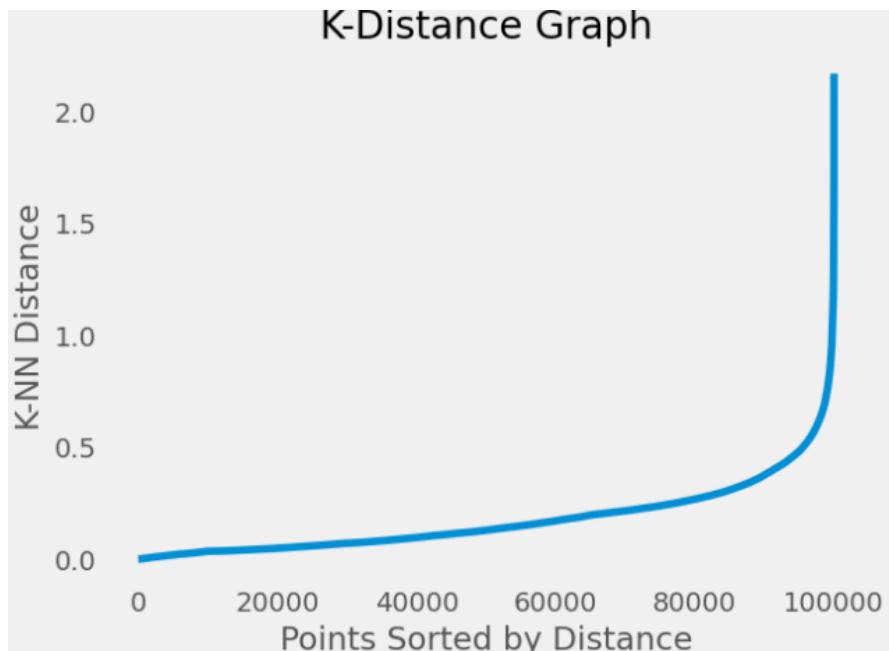
```

k=28
knn = NearestNeighbors(n_neighbors = k)
model = knn.fit(df_scaled_dbSCAN)
distances, indices = knn.kneighbors(df_scaled_dbSCAN)
distances = np.sort(distances, axis=0)
distances = distances[:,1]

```

Hình 3.162: Đoạn mã thực hiện

- Vẽ biểu đồ K-Distance để xác định điểm “gãy”, là ngưỡng eps lý tưởng



Hình 3.163: Biểu đồ K-Distance trên dữ liệu gốc

Quan sát biểu đồ, nhóm xác định điểm gãy ở khoảng  $\text{eps} = 0.7$ , đây là giá trị được lựa chọn.

### Cấu hình và huấn luyện DBSCAN

Sau khi xác định  $\text{eps} = 0.7$  và  $\text{min\_samples} = 250$ , nhóm áp dụng mô hình DBSCAN:

```

dbSCAN = DBSCAN(eps=0.7, min_samples=250)
labels = dbSCAN.fit_predict(df_scaled_dbSCAN)

```

Hình 3.164: Khởi tạo và huấn luyện mô hình DBSCAN trên dữ liệu gốc

DBSCAN tự động gán nhãn cụm, trong đó:

- Các điểm trong cụm được gán nhãn từ 0 trở lên
- Các điểm nhiễu được gán nhãn -1

```
Labels: [ 0 -1 -1 ... -1 -1 0]
```

Hình 3.165: Kết quả nhãn cụm từ mô hình DBSCAN

Các điểm dữ liệu được gán nhãn cụm tự động, trong đó -1 biểu thị các điểm nhiễu (outliers).

```
unique_labels = np.unique(labels)
num_clusters = len(unique_labels) - (1 if -1 in unique_labels else 0)
✓ 0.0s

# Hiển thị kết quả
print(f"Số lượng cụm: {num_clusters}")
print(f"Các nhãn cụm: {unique_labels}")
✓ 0.0s

Số lượng cụm: 8
Các nhãn cụm: [-1  0  1  2  3  4  5  6  7]
```

Hình 3.166: Kết quả xác định số cụm từ mô hình DBSCAN

Sau khi huấn luyện mô hình DBSCAN, nhóm sử dụng hàm np.unique () để xác định tất cả các nhãn cụm được gán. Trong đó, nhãn -1 đại diện cho các điểm bị coi là nhiễu. Khi loại bỏ nhãn -1, số cụm hợp lệ được phát hiện là 8 cụm.

```
filtered_labels = labels[labels != -1]
filtered_data = df_scaled_dbSCAN[labels != -1]

# Tính chỉ số Silhouette
score = silhouette_score(filtered_data, filtered_labels)

# In kết quả
print(f"Silhouette Score: {score}")

Silhouette Score: 0.36079959394262595
```

Hình 3.167: Kết quả tính Silhouette Score sau khi phân cụm bằng DBSCAN

- Chỉ số Silhouette sau khi loại bỏ điểm nhiễu (label = -1) là 0.361 ( $\approx 0.36$ ).

Ý nghĩa:

- Đây là chỉ số khá tốt, cao hơn so với KMeans (~0.25–0.30), cho thấy DBSCAN phân nhóm rõ hơn và tự nhiên hơn, đặc biệt với dữ liệu có hình dạng không cầu.
- Việc lọc bỏ điểm nhiễu giúp cải thiện chất lượng cụm, vì DBSCAN không bắt buộc mọi điểm đều phải thuộc về một cụm.

cluster
-1 39941
1 38902
2 10121
0 6860
3 2066
4 993
5 415
7 366
6 336
Name: count, dtype: int64

Hình 3.168: Phân bố số lượng điểm theo cụm trong mô hình DBSCAN

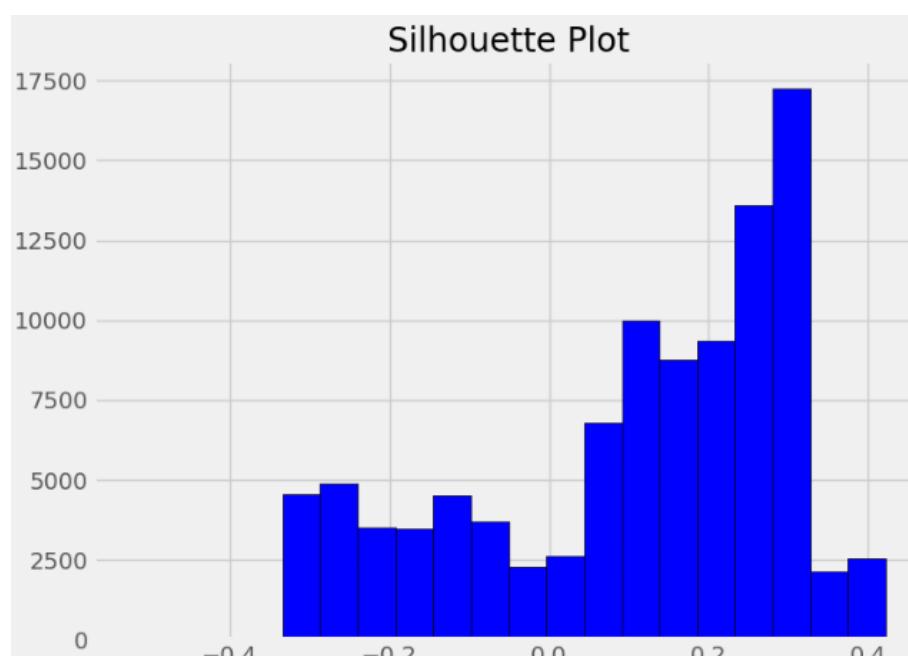
Nhận xét:

- Tỷ lệ nhiễu rất cao (~28%), phản ánh đặc điểm dữ liệu có nhiều khách hàng không phù hợp với bất kỳ nhóm nào – có thể là khách hàng có hành vi bất thường hoặc mới
- Cụm 1 là cụm chiếm ưu thế – đây có thể là phân khúc khách hàng phổ biến nhất về hành vi hoặc độ tuổi.
- Các cụm nhỏ như 4–7 có thể là nhóm khách hàng đặc biệt hoặc ngoại lệ – cần xem xét kỹ hơn để cá nhân hóa chính sách tiếp cận.
- Điều này cho thấy DBSCAN phân chia cụm không đều, nhưng bù lại phát hiện tốt điểm nhiễu và phân khúc đặc biệt

### Kết quả phân cụm DCSCAN

- Sau khi áp dụng mô hình DBSCAN với các tham số  $\text{eps} = 0.7$ ,  $\text{min\_samples} = 250$ , mô hình đã phân loại 100.000 điểm dữ liệu vào các cụm như sau:
- Tổng số cụm được phát hiện: 8 cụm (không bao gồm điểm nhiễu).

- Số lượng điểm bị coi là nhiễu (label = -1): 39.941 điểm, chiếm khoảng 39.9% tổng số dữ liệu.
- Kết quả này cho thấy DBSCAN hoạt động tốt trong việc phát hiện cấu trúc phân cụm không đồng nhất và loại bỏ các điểm không thuộc về cụm nào rõ ràng. Tuy nhiên, tỷ lệ điểm nhiễu cao cũng là một yếu tố cần được cân nhắc trong việc đánh giá hiệu quả mô hình



Đánh giá hiệu quả phân cụm qua chỉ số Silhouette:

- Chỉ số Silhouette trung bình đạt khoảng 0.36, cho thấy mức độ phân tách giữa các cụm là khá tốt và có thể chấp nhận được trong bối cảnh dữ liệu thực tế.
- Phân bố Silhouette:
  - Đa số điểm dữ liệu có giá trị Silhouette dương (từ 0.1 đến 0.4), phản ánh việc phân cụm là hợp lý và các điểm nằm tương đối gần tâm cụm của mình hơn là các cụm khác.
  - Tuy nhiên, vẫn còn một tỷ lệ nhất định các điểm có giá trị Silhouette âm, cho thấy một số điểm có thể bị gán sai cụm hoặc nằm ở vùng ranh giới giữa các cụm.

- Kết luận: Mặc dù DBSCAN phát hiện được các cụm tự nhiên trong dữ liệu và xử lý tốt các điểm ngoại lai, vẫn cần tinh chỉnh thêm thông số hoặc kết hợp với các bước giảm chiều/phân tích chuyên sâu để nâng cao độ đồng nhất nội cụm

### 3.5.6. Phân cụm khách bằng PCA sau khi giảm chiều bằng PCA

#### Mục tiêu

Trong phần trước, mô hình DBSCAN được áp dụng trực tiếp trên dữ liệu chuẩn hóa và cho kết quả bước đầu khả quan. Tuy nhiên, dữ liệu ban đầu có nhiều chiều và chứa nhiều nhiễu, dẫn đến chỉ số Silhouette không cao và tỷ lệ điểm nhiễu lớn.

Để cải thiện hiệu quả phân cụm, nhóm tiếp tục áp dụng kỹ thuật phân tích thành phần chính (PCA) trước khi sử dụng DBSCAN.

#### Quy trình thực hiện

##### Giảm chiều dữ liệu bằng PCA

PCA được áp dụng để rút trích các thành phần chính, giữ lại tối đa 4 thành phần đầu tiên, đảm bảo giải thích trên 90% phương sai của toàn bộ dữ liệu.

```
print(f"Dữ liệu sau PCA (shape): {pca_data_dbSCAN.shape}")

Dữ liệu sau PCA (shape): (100000, 4)
```

Hình 3.169: Dữ liệu sau PCA

##### Nhận xét:

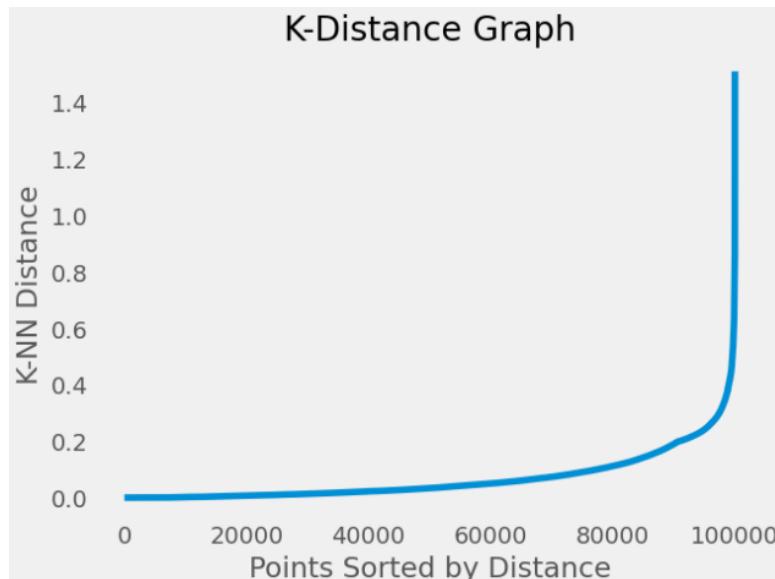
Dữ liệu ban đầu gồm nhiều đặc trưng đã được giảm xuống còn 4 thành phần chính (principal components) bằng kỹ thuật PCA, với 100.000 quan sát được giữ lại. Điều này giúp đơn giản hóa không gian dữ liệu, giảm nhiễu và cải thiện hiệu quả của thuật toán DBSCAN trong việc phát hiện cấu trúc cụm tự nhiên

```

# tìm eps cho DBSCAN
knn = NearestNeighbors(n_neighbors = k)
model = knn.fit(pca_data_dbSCAN)
distances, indices = knn.kneighbors(pca_data_dbSCAN)
distances = np.sort(distances, axis=0)
distances = distances[:,1]

```

Hình 3.170: Xác định  $\text{eps}$



Hình 3.171: Biểu đồ K-Distance sau PCA

Quan sát đồ thị cho thấy điểm gãy tại  $\text{eps} = 0.3$ , được sử dụng làm tham số đầu vào cho mô hình.

### Cáu hình huấn luyện DBSCAN trên dữ liệu PCA

```

#thực hiện DBSCAN
eps=0.3
minPts=50
dbSCAN_pca = DBSCAN(eps=eps, min_samples=minPts)
labels_pca = dbSCAN_pca.fit_predict(pca_data_dbSCAN)

```

Hình 3.172: Cáu hình DBSCAN sau khi PCA

Mô hình DBSCAN được áp dụng trên dữ liệu đã giảm chiều (5 thành phần chính), với tham số  $\text{eps} = 0.3$  và  $\text{min\_samples} = 50$ , giúp tăng độ tách biệt cụm và giảm nhiễu. Đây là bước quan trọng để phát hiện các cụm tự nhiên trong không gian PCA tối ưu hơn

```

# chuyển thành dữ liệu về DataFrame và thêm label
df_pca = pd.DataFrame(pca_data_dbSCAN, columns=[f'PC{i+1}' for i in range(n_components)])
df_pca['cluster'] = labels_pca

# hiển thị và đếm số lượng cụm
unique_labels = set(labels_pca)
print(f"Unique cluster labels: {unique_labels}")
print(f"Number of clusters (including noise): {len(unique_labels)}")

Unique cluster labels: {0, 1, 2, 3, 4, -1}
Number of clusters (including noise): 6

```

Hình 3.173: Nhãn cụm sau khi phân cụm DBSCAN trên dữ liệu PCA

Dữ liệu đã được giảm chiều còn 4 thành phần chính bằng PCA trước khi áp dụng DBSCAN. Sau khi huấn luyện mô hình, DBSCAN đã phát hiện ra tổng cộng **6** cụm bao gồm cả cụm nhiễu (label = -1).

cluster	count
2	49007
-1	19770
1	15171
0	12366
3	2980
4	706

Name: count, dtype: int64

Hình 3.174: Nhãn cụm được tạo bởi DBSCAN sau PCA

- Silhouette Score (loại điểm nhiễu): - 0.123

```

# tính silhouette score trừ các điểm nhiễu
filtered_labels = labels[labels != -1]
filtered_data = df_pca[labels != -1]

# Tính chỉ số Silhouette
score = silhouette_score(filtered_data, filtered_labels)

# In kết quả
print(f"Silhouette Score: {score}")

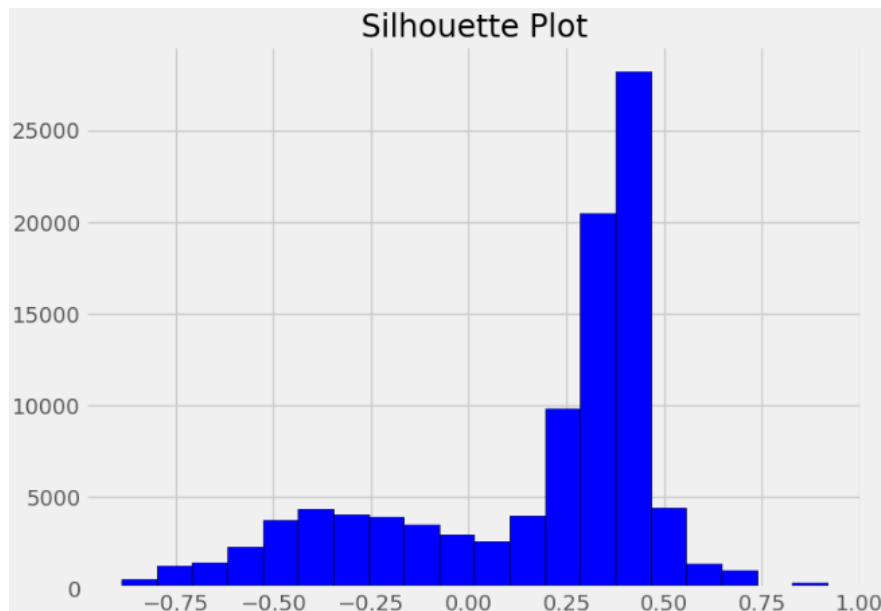
Silhouette Score: 0.3464155361905604

```

Hình 3.175: Silhouette Score sau phân cụm DBSCAN + PCA

### Nhận xét:

- Sau khi loại bỏ các điểm nhiễu (label = -1), mô hình DBSCAN đạt chỉ số Silhouette trung bình là 0.3464..
- Giá trị này cho thấy mức độ phân tách giữa các cụm là tốt, các cụm có độ đồng nhất tương đối cao.
- So với kết quả phân cụm KMeans (Silhouette ~0.2957 sau PCA), DBSCAN thể hiện hiệu quả phân cụm cao hơn, đặc biệt là trong việc phát hiện cấu trúc phân nhóm không cầu phương.



Hình 3.176: Silhouette sau phân cụm bằng DBSCAN (PCA)

Biểu đồ cho thấy phần lớn các điểm dữ liệu có hệ số Silhouette nằm trong khoảng 0.25 đến 0.5, với đỉnh tại gần 0.4, điều này cho thấy:

- Đa số các điểm dữ liệu nằm trong các cụm có độ đồng nhất cao và được phân tách rõ ràng khỏi các cụm khác.
- Một số điểm có giá trị Silhouette âm hoặc gần 0, phản ánh một phần nhỏ các điểm nằm gần ranh giới cụm hoặc được gán chưa tối ưu.

Tổng thể, phân cụm DBSCAN kết hợp PCA đã tạo ra các cụm khá rõ ràng, có tính gắn kết nội cụm tốt. Đây là dấu hiệu tích cực cho việc sử dụng mô hình này trong phân khúc khách hàng thực tế.

### 3.5.7. So sánh K – Means và DBSCAN

Sau khi áp dụng hai thuật toán phân cụm phổ biến là K-Means và DBSCAN trên cùng một tập dữ liệu đã được xử lý và chuẩn hóa, nhóm tiến hành đánh giá và so sánh dựa trên các tiêu chí về hiệu quả phân cụm, khả năng phát hiện nhiễu, độ linh hoạt và khả năng ứng dụng thực tiễn.

Tiêu chí	K-Means (gốc)	K-Means + PCA	DBSCAN (gốc)	DBSCAN + PCA
Loại mô hình	Centroid-based	Centroid + PCA	Density-based	Density + PCA
Xác định số cụm trước	Có ( $k = 5$ )	Có ( $k = 5$ )	Không	Không
Số cụm phát hiện	5 cụm	5 cụm	8 cụm	7 cụm + nhiễu
Phát hiện nhiễu	Không	Không	Có (39.93%)	Có (34.8%)
Silhouette Score	0.2544	0.2956	0.3608	0.3464
Tốc độ, tính ổn định	Cao	Cao	Trung bình	Trung bình thấp
Tính trực quan, dễ triển khai	Cao	Cao	Trung bình	Thấp

### Nhận xét tổng quan:

- K-Means + PCA là mô hình hiệu quả và ổn định nhất:
  - Có Silhouette Score cao nhất trong các mô hình K-Means ( $\approx 0.296$ ).
  - Nhờ PCA giảm chiều, mô hình học nhanh hơn và ít bị nhiễu.
  - Các cụm tách biệt rõ ràng trên biểu đồ 3D scatter và radar chart.
- K-Means gốc vẫn hoạt động tốt (Silhouette  $\approx 0.254$ ), phù hợp nếu không muốn giảm chiều. Tuy nhiên, độ phân tách cụm kém hơn khi chưa kết hợp PCA
- DBSCAN (gốc) phát hiện tới 8 cụm + 39.9% điểm nhiễu, cho thấy khả năng phát hiện outlier tốt. Tuy nhiên, độ ổn định thấp hơn, nhạy với tham số eps, min\_samples
- DBSCAN + PCA giúp giảm nhiễu còn 34.8%, số cụm giảm còn 7.
  - Tuy nhiên, độ tách biệt cụm (Silhouette  $\approx 0.3464$ ) vẫn thấp hơn so với K-Means + PCA.
  - Một phần do các cụm bị chồng lấn trên radar plot và không gian PCA

### Kết luận:

- Nếu mục tiêu là phân khúc khách hàng rõ ràng, dễ triển khai trong marketing, K-Means + PCA là lựa chọn tối ưu: nhanh, chính xác, dễ áp dụng thực tiễn.
- DBSCAN phù hợp với bài toán phát hiện khách hàng bất thường hoặc nhóm nhỏ lẻ, nhưng yêu cầu tinh chỉnh tham số kỹ càng và đánh đổi hiệu năng.
- Kết hợp PCA với cả hai thuật toán giúp giảm chiều, cải thiện tốc độ và trực quan hóa, đặc biệt hữu ích với tập dữ liệu lớn như trong bài toán này.

## 3.6. Dự báo giá cổ phiếu Uber bằng mô hình ARIMA

### 3.6.1. Giới thiệu về bài toán và dữ liệu

Bộ dữ liệu “Uber Stocks Dataset 2025” được cung cấp trên nền tảng Kaggle bởi tác giả M Hassan Saboor, ghi lại dữ liệu giá cổ phiếu của công ty Uber trong nhiều năm (từ 2019 đến 2025). Mỗi dòng dữ liệu đại diện cho thông tin giao dịch trong một ngày, bao gồm giá mở cửa, đóng cửa, giá cao nhất, thấp nhất, khối lượng giao dịch và giá điều chỉnh.

## Mục tiêu phân tích

- Phân tích xu hướng giá cổ phiếu Uber theo thời gian.
- Xác định tính dừng của chuỗi và các yếu tố ảnh hưởng theo mùa.
- Xây dựng mô hình ARIMA để dự báo giá cổ phiếu trong tương lai.
- So sánh hiệu quả mô hình dự báo với baseline (trung bình).

## Tổng quan về dữ liệu

Tên biến	Ý nghĩa	Kiểu dữ liệu
Date	Ngày giao dịch cổ phiếu	Chuỗi ký tự (string)
Open	Giá mở cửa trong ngày	Số thực (float)
High	Giá cao nhất trong ngày	Số thực (float)
Low	Giá thấp nhất trong ngày	Số thực (float)
Close	Giá đóng cửa trong ngày	Số thực (float)

Adj Close	Giá đóng cửa điều chỉnh (đã tính cổ tức, chia tách...)	Số thực (float)
Volume	Khối lượng cổ phiếu giao dịch trong ngày	Số nguyên (int)

### Đặc điểm của bài toán

Bài toán đặt ra là dự báo giá cổ phiếu Uber trong tương lai gần, dựa trên chuỗi thời gian giá đóng cửa lịch sử. Dữ liệu dạng chuỗi thời gian liên tục, không có nhãn phân loại nên thuộc dạng bài toán học có giám sát, sử dụng mô hình dự báo chuỗi thời gian cổ điển.

Thuật toán được lựa chọn:

- ARIMA (Autoregressive Integrated Moving Average): Là mô hình kinh điển trong dự báo chuỗi thời gian, đặc biệt hiệu quả với các chuỗi có tính dừng sau khi lấy sai phân. Mô hình này kết hợp ba thành phần: hồi quy tự động (AR), sai phân tích hợp (I) và trung bình trượt (MA).
- Mô hình baseline: Là giá trị trung bình của tập huấn luyện, được sử dụng để so sánh mức độ cải thiện của ARIMA.

#### 3.6.2. Tiền xử lý dữ liệu

##### Đọc dữ liệu

```
df = pd.read_csv(r"C:\Users\NAM\Downloads\uber_stock_data.csv")
df
```

Hình 3.177: Đọc tập dữ liệu *uber\_stock\_data* bằng pandas

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1444 entries, 0 to 1443
Data columns (total 7 columns):
 #   Column      Non-Null Count Dtype  
--- 
 0   Date        1444 non-null    datetime64[ns]
 1   Adj Close   1444 non-null    float64 
 2   Close       1444 non-null    float64 
 3   High        1444 non-null    float64 
 4   Low         1444 non-null    float64 
 5   Open        1444 non-null    float64 
 6   Volume      1444 non-null    int64  
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 79.1 KB

```

Hình 3.178: Thông tin tổng quan về cấu trúc dữ liệu Uber Stock

Nhận xét:

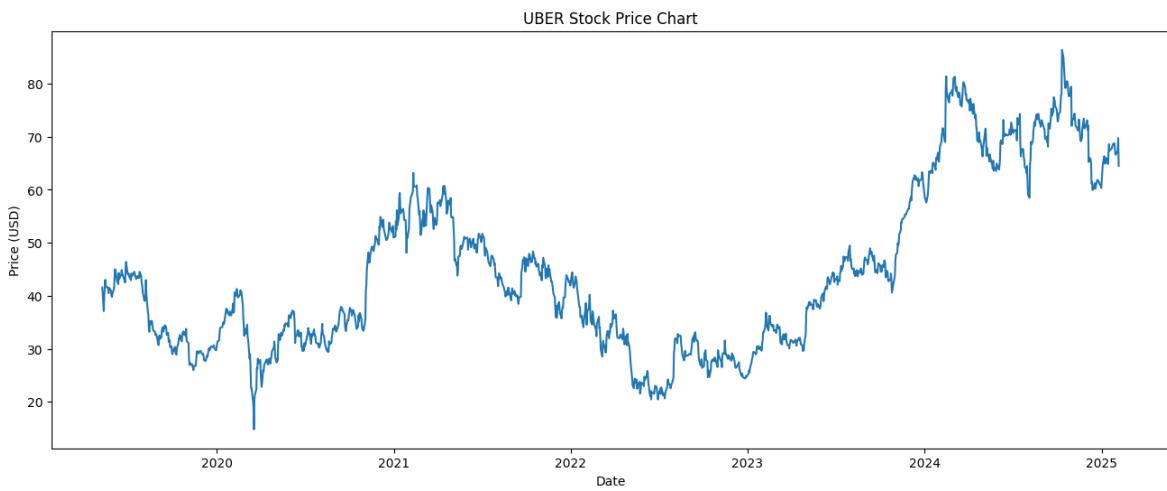
Bộ dữ liệu gồm 1444 dòng và 7 cột, không có giá trị thiếu. Các cột số liệu tài chính như Open, High, Low, Close, Adj Close đều ở định dạng float64, phù hợp để xử lý số liệu liên tục. Cột Volume có kiểu int64, biểu thị khối lượng cổ phiếu giao dịch mỗi ngày.

Đáng chú ý, cột Date đã được chuyển đổi về kiểu datetime64[ns], giúp hỗ trợ các thao tác chuỗi thời gian như phân nhóm theo năm, tính rolling mean, hoặc tách dữ liệu theo mốc thời gian. Việc chuyển đổi này là một bước quan trọng trong tiền xử lý để đảm bảo tính chính xác khi xây dựng mô hình ARIMA.

	# Date	# Adj Close	# Close	# High	# Low	# Open	# Volume
count	1444	1444.0	1444.0	1444.0	1444.0	1444.0	1444.0
mean	2022-03-22 02:49:31.745152256	44.46537397701524	44.46537397701524	45.28751138704986	43.64330121879502	44.493022126738225	24298002.880193904
min	2019-05-10 00:00:00	14.81999969	14.81999969	17.79999924	13.71000004	15.96000004	3380000.0
25%	2020-10-13 18:00:00	31.93000031	31.93000031	32.6537494675	31.1812496225	31.86749935	14989050.0
50%	2022-03-21 22:00:00	41.35500145	41.35500145	41.90999885	40.51849375000005	41.21500015	20369650.0
75%	2023-08-28 06:00:00	54.71999931000006	54.71999931000006	55.635001185	53.59750175499996	54.6649990075	28432800.0
max	2025-02-05 00:00:00	86.33999634	86.33999634	87.0	84.18000031	85.63999939	364231800.0
std	Missing value	15.59426150249535	15.740824515676335	15.482652472608617	15.643966544446947	17740835.254176002	

Hình 3.179: Thống kê mô tả dữ liệu giá cổ phiếu Uber

Bộ dữ liệu bao gồm các phiên giao dịch từ năm 2019 đến đầu năm 2025, với giá cổ phiếu trung bình khoảng 44–45 USD và dao động mạnh (giá thấp nhất ~13.7 USD, cao nhất ~87 USD). Khối lượng giao dịch trung bình hơn 24 triệu cổ phiếu/ngày, cho thấy tính thanh khoản cao. Các thống kê phản ánh sự biến động lớn về giá và khối lượng trong giai đoạn 6 năm, phù hợp để áp dụng mô hình chuỗi thời gian như ARIMA.



Hình 3.180: Diễn biến giá đóng cửa cổ phiếu Uber trong giai đoạn 2019–2025

Nhận xét:

Biểu đồ cho thấy giá cổ phiếu Uber biến động mạnh qua các năm 2019–2025.

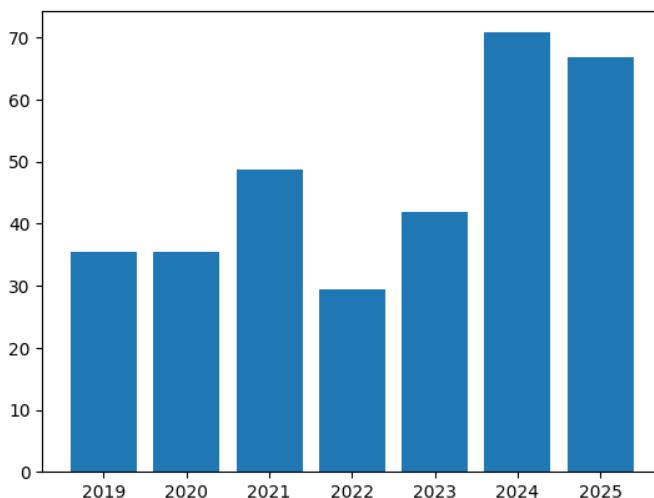
Cụ thể:

- Giai đoạn 2020 có sự giảm sâu, khả năng do ảnh hưởng của đại dịch COVID-19.
- Sau đó, giá phục hồi và tăng mạnh trong năm 2024, đạt đỉnh trên 85 USD.
- Nhìn chung, giá có xu hướng tăng trưởng về dài hạn nhưng kèm theo nhiều đợt biến động ngắn hạn, phù hợp với đặc điểm chuỗi tài chính thực tế.

Date	# Adj Close	# Close	# High	# Low	# Open	# Volume
2019-12-31 00:00:00	35.537668719509206	35.537668719509206	36.2141103806135	34.888012283190186	35.62572991736196	18798274.233128835
2020-12-31 00:00:00	35.5116601801581	35.5116601801581	36.296320255691704	34.63344657675889	35.46980236268775	27679798.418972332
2021-12-31 00:00:00	48.61003962003968	48.61003962003968	49.58869057607143	47.709346311508	48.733789776904764	23352206.34920635
2022-12-31 00:00:00	29.35203180625498	29.35203180625498	30.122422252151395	28.63255777378486	29.39109566673307	30004296.015936255
2023-12-31 00:00:00	41.780120002720004	41.780120002720004	42.35486016104	41.082435943680004	41.62381587208	24506541.2
2024-12-31 00:00:00	70.76682558527777	70.76682558527777	71.8383214888492	69.7702459152809	70.91456731150794	19200381.34920635
2025-12-31 00:00:00	66.76304261565217	66.76304261565217	67.83682615869564	65.73056561	66.63326130739131	27749685.173913043

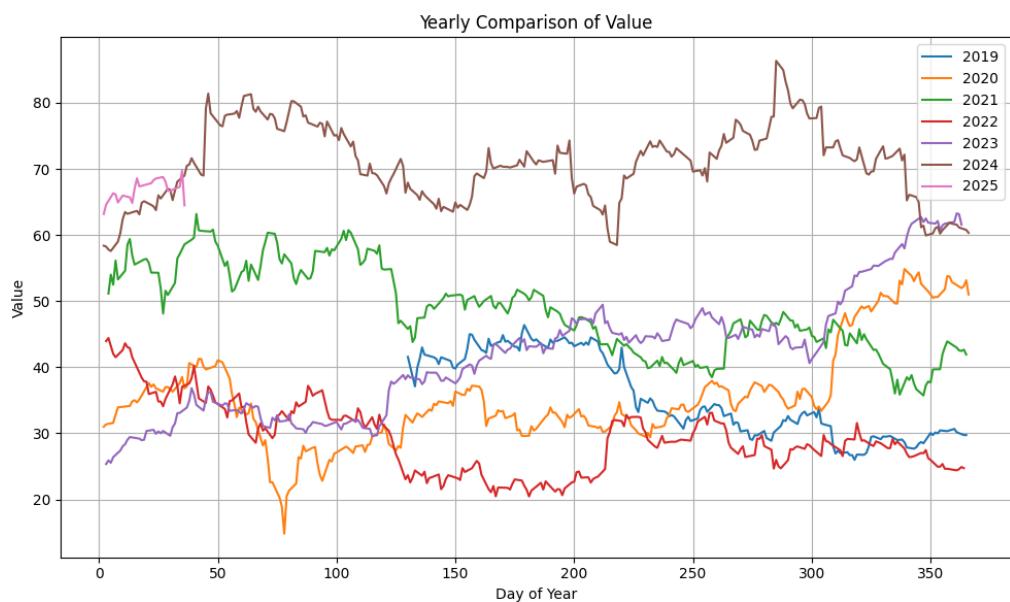
Hình 3.181: Giá trị trung bình theo năm của các chỉ số cổ phiếu Uber (2019–2025)

Giá đóng cửa và giá điều chỉnh gần như bằng nhau cho thấy Uber không thực hiện các thay đổi cấu trúc như chia tách cổ phiếu. Trong giai đoạn 2020–2022, giá cổ phiếu giảm mạnh, đặc biệt năm 2022 chỉ còn ~29 USD. Từ 2023 trở đi, giá phục hồi nhanh và đạt đỉnh vào năm 2024 (~70 USD). Khối lượng giao dịch biến động lớn, phản ánh sự thay đổi trong tâm lý thị trường.



Hình 3.182: Biểu đồ giá đóng cửa điều chỉnh trung bình theo năm

Giá cổ phiếu Uber giảm mạnh vào năm 2022, sau đó phục hồi nhanh từ 2023 và đạt đỉnh vào 2024 (~71 USD). Biểu đồ cho thấy xu hướng tăng trưởng rõ rệt giai đoạn hậu COVID.



Hình 3.183: So sánh giá điều chỉnh cổ phiếu Uber theo từng năm (dựa trên ngày trong năm)

Biểu đồ cho thấy xu hướng giá cổ phiếu Uber trong từng năm từ 2019 đến 2025.

Nhìn chung:

- Quý I của hầu hết các năm có xu hướng tăng nhẹ.
- Năm 2022 ghi nhận mức giá thấp và biến động mạnh, phù hợp với bối cảnh thị trường khó khăn.

- Năm 2024 nổi bật với mức giá cao vượt trội trong suốt cả năm.
- Giá trị của các năm sau 2022 có xu hướng phục hồi và tăng dần, đặc biệt là năm 2025 duy trì mức giá cao ổn định.

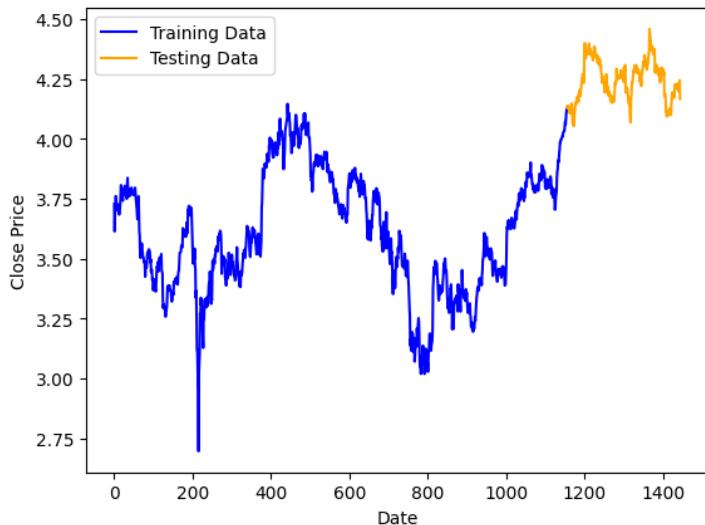
```
log_data = np.log(df['Close'])
train_data, test_data = log_data[:int(len(log_data)*0.8)], log_data[int(len(log_data)*0.8):]
```

Hình 3.184: Log-transform và phân chia tập train/test

Dữ liệu giá đóng cửa được **biến đổi logarit** để làm mượt và ổn định phương sai, giúp mô hình ARIMA hoạt động hiệu quả hơn. Sau đó, dữ liệu được chia thành hai phần:

- 80% đầu tiên: Tập huấn luyện (train\_data)
- 20% còn lại: Tập kiểm tra (test\_data)

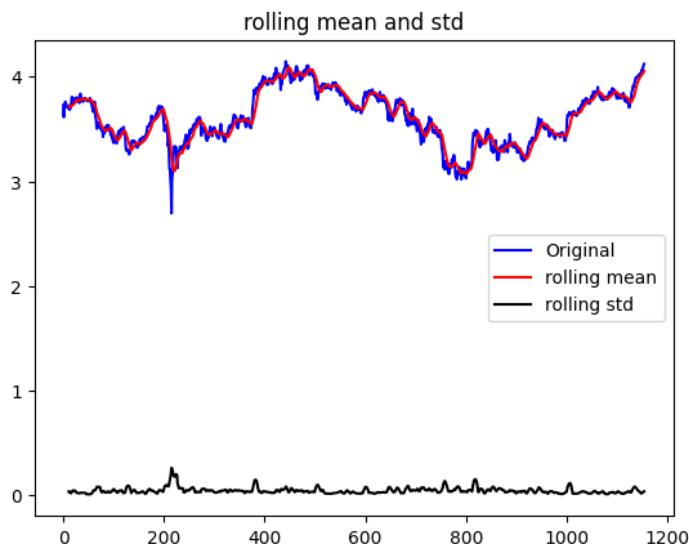
Việc chia theo thời gian giữ nguyên thứ tự chuỗi, phù hợp với bản chất của bài toán dự báo.



Hình 3.185: Phân chia tập huấn luyện và kiểm tra sau log-transform

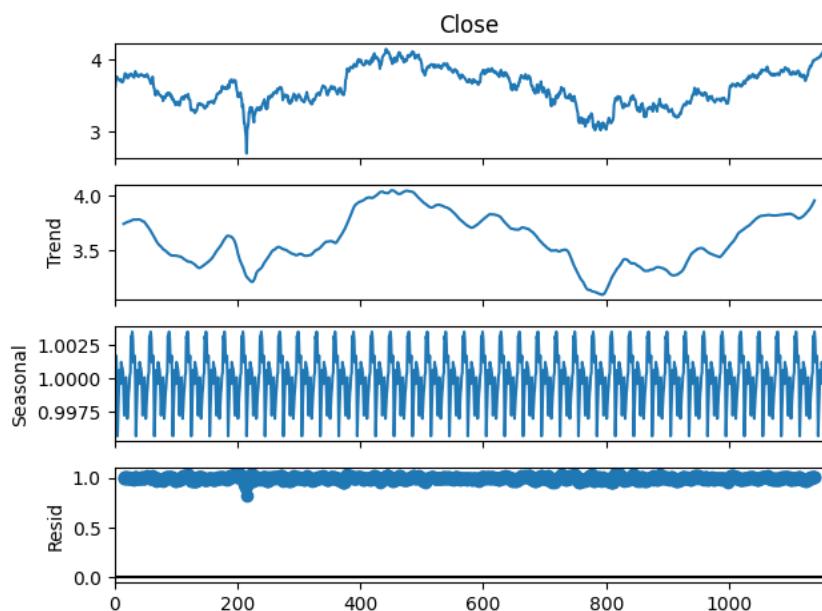
Biểu đồ thể hiện rõ ranh giới giữa dữ liệu huấn luyện (màu xanh) và dữ liệu kiểm tra (màu cam) sau khi thực hiện biến đổi logarit giá đóng cửa. Việc chia theo thứ tự thời gian giúp mô hình học được xu hướng quá khứ và đánh giá khả năng dự báo tương lai. Giá trị log biến động mượt hơn, cho thấy tính ổn định cao hơn so với chuỗi gốc.

### 3.6.3. Kiểm tra tính dừng và chọn tham số



Hình 3.186: Kiểm tra tính dừng bằng trung bình và độ lệch chuẩn trượt

Biểu đồ thể hiện đường trung bình trượt (màu đỏ) và độ lệch chuẩn trượt (màu đen) của chuỗi log giá đóng cửa. Kết quả cho thấy chuỗi chưa ổn định rõ ràng, vì giá trị trung bình và phương sai vẫn dao động theo thời gian. Do đó, cần thực hiện thêm bước sai phân (differencing) để đưa chuỗi về trạng thái dừng một yếu cầu cần thiết để áp dụng mô hình ARIMA.



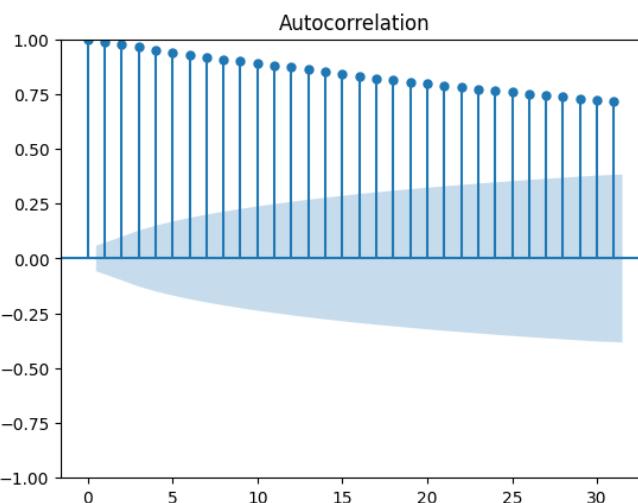
Hình 3.187: Phân rã chuỗi thời gian log giá đóng cửa cổ phiếu Uber

Nhận xét:

Chuỗi thời gian được phân rã thành 4 thành phần:

- Observed (Close): Biến động giá log ban đầu
- Trend: Thể hiện xu hướng tăng/giảm dài hạn và rõ ràng nhất trong các năm gần đây.
- Seasonal: Biến động lặp lại đều đặn theo chu kỳ ~30 ngày, cho thấy tính mùa vụ rõ rệt.
- Residual: Nhiều không có quy luật rõ, phần còn lại sau khi loại bỏ xu hướng và mùa vụ.

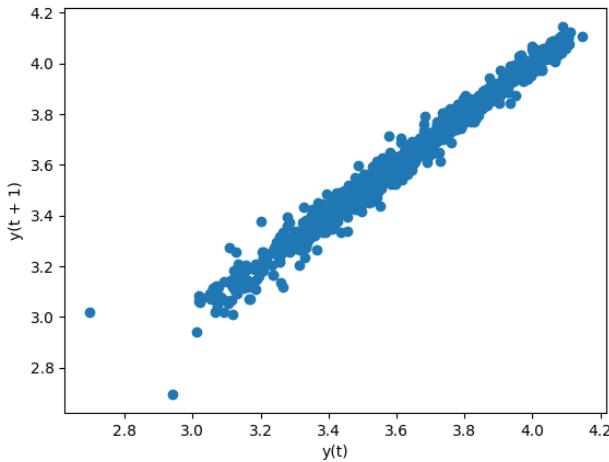
Kết quả này chứng minh chuỗi thời gian có đầy đủ đặc điểm cần thiết để áp dụng mô hình ARIMA.



Hình 3.188: Biểu đồ tự tương quan (ACF) của chuỗi log giá đóng cửa

Biểu đồ ACF cho thấy các giá trị tự tương quan đều rất cao và giảm dần chậm, kéo dài qua nhiều độ trễ. Đa số các giá trị nằm ngoài khoảng tin cậy, đặc biệt từ lag 1 đến khoảng lag 30.

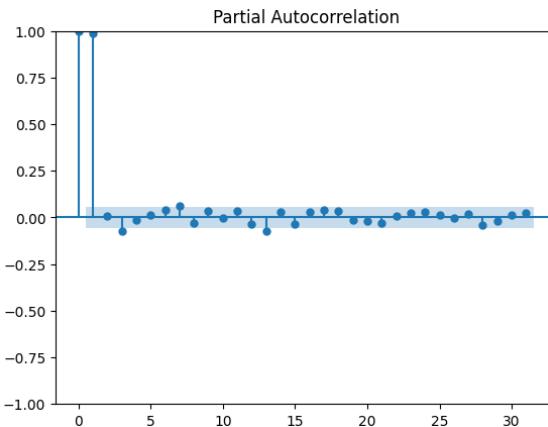
Điều này cho thấy chuỗi thời gian chưa dừng, vì một chuỗi dừng sẽ có ACF cắt về 0 sau vài độ trễ đầu tiên. Kết luận rút ra là chuỗi cần được sao phán ít nhất một lần ( $d = 1$ ) để ổn định trung bình và phù hợp với yêu cầu của mô hình ARIMA.



Hình 3.189: Biểu đồ độ trễ (Lag plot) của chuỗi log giá đóng cửa

Biểu đồ cho thấy các điểm dữ liệu phân bố rất sát thành một đường thẳng tuyến tính, chứng tỏ mối quan hệ giữa giá trị tại thời điểm  $y(t)$  và  $y(t+1)$  là rất mạnh và gần như tuyến tính hoàn toàn.

Đây là đặc điểm điển hình của một chuỗi thời gian có tính phụ thuộc mạnh, rất phù hợp để áp dụng các mô hình tự hồi quy như ARIMA. Ngoài ra, điều này cũng cung cấp nhận định rằng dữ liệu cần được xử lý cẩn thận (sai phân, chọn tham số) để mô hình có thể học được cấu trúc lặp lại theo thời gian.



Hình 3.190: Biểu đồ tự tương quan riêng phần (PACF) của chuỗi log giá đóng cửa

Biểu đồ PACF cho thấy:

- Độ trễ 1 và 2 vượt rõ ràng khỏi ngưỡng tin cậy, sau đó các độ trễ tiếp theo đều nằm trong khoảng tin cậy.

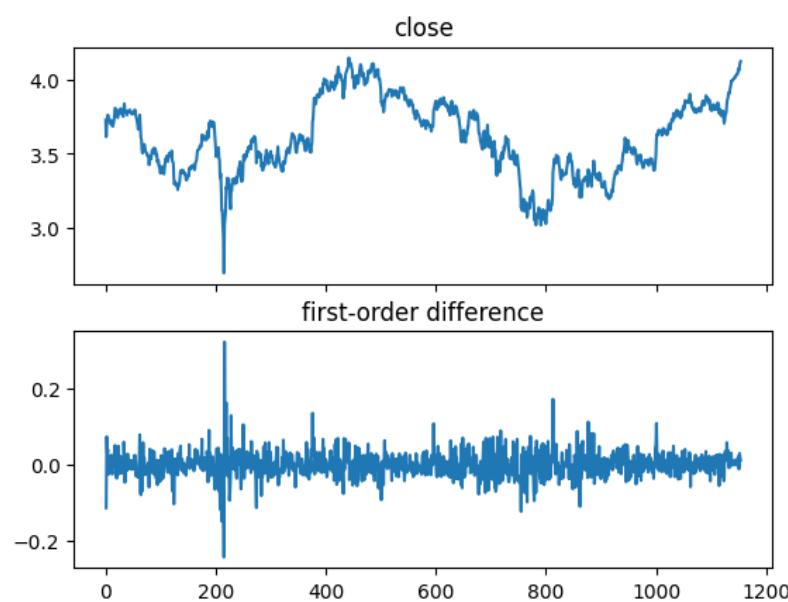
- Điều này cho thấy chuỗi có cấu trúc tự hồi quy bậc thấp, gợi ý tham số AR nên chọn là  $p = 2$ .

PACF giúp xác định số lượng độ trễ cần thiết để mô hình ARIMA có thể học được phần tự hồi quy mà không bị dư thừa hoặc overfitting.

### Tổng hợp quá trình chọn tham số cho mô hình ARIMA

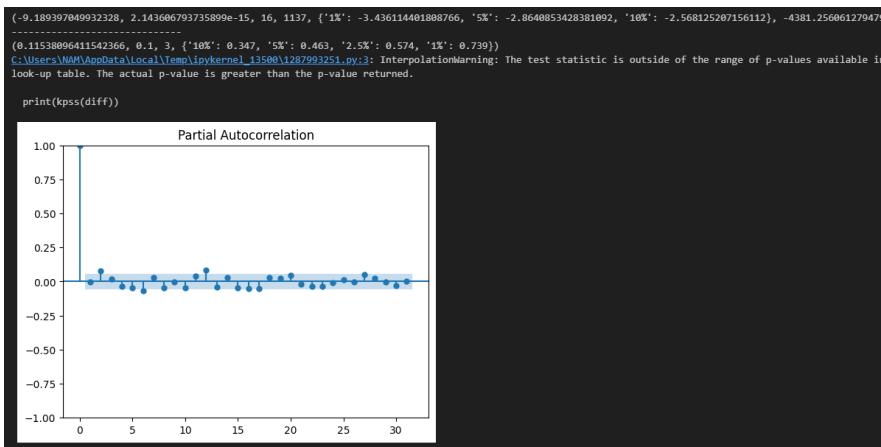
Thành phần	Dấu hiệu từ biểu đồ	Tham số đề xuất
d (sai phân)	Biểu đồ ACF giảm chậm và chuỗi chưa dừng rõ ràng	$d = 1$
p (thành phần AR)	Biểu đồ PACF cắt rõ tại độ trễ thứ 2	$p = 2$
q (thành phần MA)	Dựa vào kết quả tối ưu từ auto_arima() hoặc biểu đồ ACF	$q \approx 1 - 2$

Tổ hợp tham số khởi điểm hợp lý cho mô hình có thể là ARIMA (2, 1, 2). Sau đó, tham số sẽ được tinh chỉnh tự động bằng auto\_arima() để tìm ra cấu hình tối ưu nhất.



Hình 3.191: Biểu đồ chuỗi gốc và chuỗi sai phân bậc 1

- Biểu đồ trên hiển thị chuỗi log giá đóng cửa ban đầu (Close) với nhiều biến động rõ rệt, thể hiện xu hướng tăng/giảm không ổn định → chuỗi chưa dùng.
- Biểu đồ dưới thể hiện kết quả sau khi thực hiện sai phân bậc 1, ta thấy chuỗi dao động quanh giá trị trung bình (0), không còn xu hướng rõ rệt, phương sai ổn định hơn → chuỗi gần như đã dùng.



Hình 3.192: Kết quả kiểm định ADF, KPSS và biểu đồ PACF sau khi sai phân bậc 1

- Kiểm định ADF cho giá trị thống kê =  $-9.19$  và p-value =  $2.14e-15 \approx 0$ , nhỏ hơn mọi mức ý nghĩa thông thường ( $1\%$ ,  $5\%$ ,  $10\%$ ). Do đó, bác bỏ giả thuyết  $H_0$ , khẳng định chuỗi đã dùng.
- Kiểm định KPSS cho giá trị thống kê =  $0.115$ , nhỏ hơn các ngưỡng tối hạn tại mức ý nghĩa  $1\%$  đến  $10\%$ , nên không bác bỏ giả thuyết  $H_0$ , cũng khẳng định chuỗi đã dùng.
- Cả hai kiểm định đều thống nhất rằng chuỗi log giá đóng cửa đã đạt tính dùng sau sai phân bậc 1 → lựa chọn  $d = 1$  là phù hợp.
- Biểu đồ PACF sau khi sai phân cho thấy chỉ có độ trễ 1 vượt ngưỡng tin cậy, từ đó gợi ý tham số AR = 1 (tức  $p = 1$ ) cho mô hình ARIMA.

```

# xác định tham số p,d,q cho mô hình Arima
stepwise_fit=auto_arima(train_data,trace=True,
                        suppress_warnings=True)
print(stepwise_fit.summary())
stepwise_fit.plot_diagnostics(figsize=(15,8))

```

Hình 3.193: Tự động xác định tham số mô hình ARIMA bằng auto\_arima()

Đoạn mã sử dụng hàm auto\_arima () từ thư viện pmdarima để tự động lựa chọn các tham số tối ưu (p, d, q) cho mô hình ARIMA, dựa trên các tiêu chí như AIC, BIC. Việc bật trace=True giúp theo dõi quá trình thử nghiệm các tổ hợp tham số, còn suppress\_warnings=True để tránh in ra cảnh báo không cần thiết.

### 3.6.4. Huấn luyện và dự đoán ARIMA

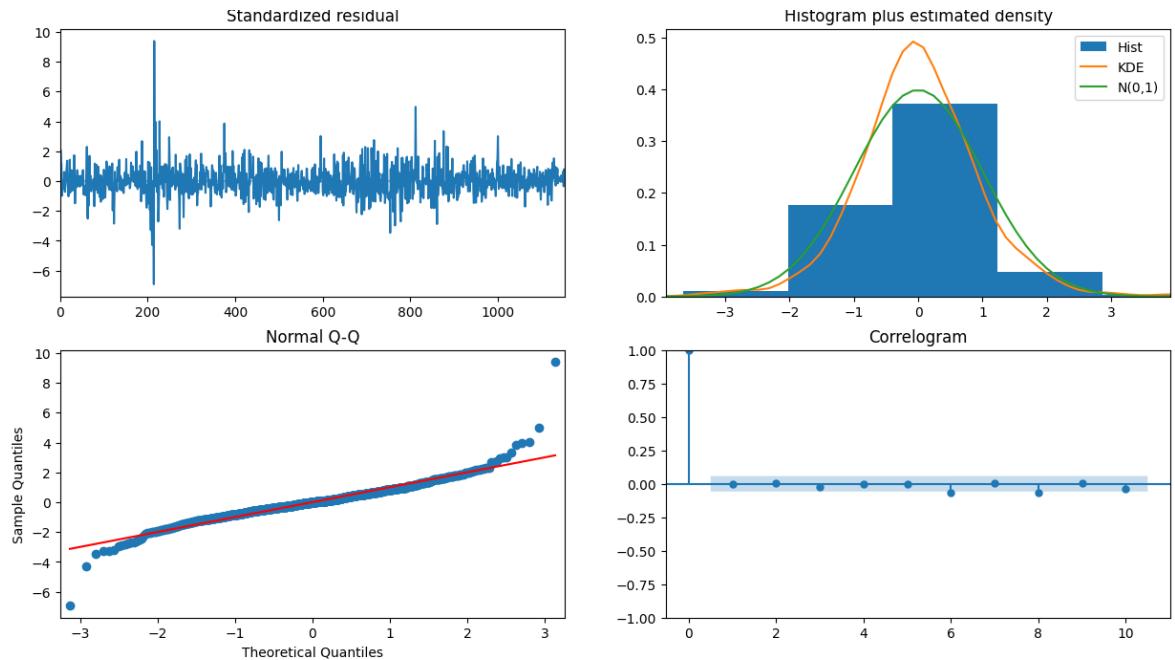
SARIMAX Results						
Dep. Variable:	y	No. Observations:	1155			
Model:	SARIMAX(2, 1, 2)	Log Likelihood:	2237.296			
Date:	Tue, 08 Apr 2025	AIC:	-4464.593			
Time:	09:31:54	BIC:	-4439.338			
Sample:	0 - 1155	HQIC:	-4455.061			
Covariance Type:	opg					
coef	std err	z	P> z	[0.025	0.975]	
ar.L1	0.5244	0.146	3.581	0.000	0.237	0.811
ar.L2	-0.5894	0.159	-3.706	0.000	-0.901	-0.278
ma.L1	-0.5276	0.142	-3.724	0.000	-0.805	-0.250
ma.L2	0.6743	0.149	4.515	0.000	0.380	0.963
sigma2	0.0012	2.52e-05	48.104	0.000	0.001	0.001
Ljung-Box (L1) (Q):	0.03	Jarque-Bera (JB):	4670.27			
Prob(Q):	0.87	Prob(JB):	0.00			
Heteroskedasticity (H):	0.55	Skew:	0.55			

Hình 3.194: Bảng kết quả huấn luyện mô hình ARIMA(2,1,2)

Nhận xét:

- Mô hình được chọn là ARIMA (2,1,2) theo kết quả từ auto\_arima () .
- Các hệ số trong mô hình (ar.L1, ar.L2, ma.L1, ma.L2) đều có p-value < 0.05, cho thấy các thành phần này có ý nghĩa thống kê trong mô hình.
- Sai số chuẩn (std err) thấp, hệ số z lớn → các hệ số ổn định.
- Các chỉ số đánh giá mô hình
  - AIC = -4464.593
  - BIC = -4439.338
  - HQIC = -4455.061 → Đây là các chỉ số tốt, cho thấy mô hình phù hợp với dữ liệu.
- Kiểm định Ljung-Box (Q = 0.03, p = 0.87) cho thấy phần dư không có tự tương quan đáng kể.

- Jarque-Bera (JB) có p-value = 0.00, tức phân dữ liệu không hoàn toàn tuân theo phân phối chuẩn — điều phổ biến với dữ liệu tài chính, nhưng không ảnh hưởng lớn đến hiệu quả dự báo.



Hình 3.195: Biểu đồ chẩn đoán mô hình ARIMA – Kiểm tra phân dữ

Nhận xét:

- Standardized Residuals (trên cùng bên trái): Phân dữ dao động xung quanh 0 và không có xu hướng rõ rệt → cho thấy mô hình không bỏ sót thông tin mang tính hệ thống.
- Histogram + KDE (trên cùng bên phải): Phân phối phân dữ gần giống chuẩn, tuy có hơi lệch nhẹ (khi so với đường  $N(0,1)$ ).
- Normal Q-Q plot (dưới trái): Phân lõn điểm nằm gần đường thẳng → phân dữ gần tuân theo phân phối chuẩn. Một vài điểm lệch nhẹ ở đuôi thể hiện hiện tượng outlier nhẹ, thường thấy trong dữ liệu tài chính.
- Correlogram (dưới phải): Các giá trị ACF của phân dữ nằm trong khoảng tin cậy (vùng xanh) → phân dữ không có tương quan tuần tự, mô hình đã loại bỏ được tính chuỗi từ dữ liệu gốc.

Kết luận:

Các biểu đồ chẩn đoán cho thấy:

- Mô hình ARIMA (2,1,2) phù hợp với dữ liệu.
- Phần dư gần như trắng (white noise).
- Không còn tự tương quan đáng kể.
- Phân phối phần dư tương đối chuẩn.

```
model = ARIMA(train_data, order=(3,1,2), trend='t')
fitted=model.fit()
print(fitted.summary())
✓ 0.7s
SARIMAX Results
=====
Dep. Variable:                  Close   No. Observations:             1155
Model:                 ARIMA(3, 1, 2)   Log Likelihood:          -2236.921
Date:         Tue, 08 Apr 2025   AIC:                         -4459.842
Time:             09:31:56   BIC:                         -4424.485
Sample:                      0   HQIC:                        -4446.498
                           - 1155
Covariance Type:                opg
=====
            coef    std err      z   P>|z|      [0.025]     [0.975]
-----
x1      0.0003    0.001    0.284    0.776    -0.002     0.003
ar.L1    0.8576    0.334    2.567    0.010     0.203     1.512
ar.L2   -0.3429    0.212   -1.616    0.106    -0.759     0.073
ar.L3   -0.0656    0.042   -1.564    0.118    -0.148     0.017
ma.L1   -0.8655    0.330   -2.622    0.009    -1.512    -0.219
ma.L2    0.4424    0.199    2.223    0.026     0.052     0.832
sigma2   0.0012   2.5e-05  48.460    0.000     0.001     0.001
=====
Ljung-Box (L1) (Q):            0.00   Jarque-Bera (JB):        4607.62
Prob(Q):                      0.99   Prob(JB):              0.00
Heteroskedasticity (H):        0.55   Skew:                  0.54
Prob(H) (two-sided):           0.00   Kurtosis:             12.73
=====
```

Hình 3.196: Kết quả huấn luyện mô hình ARIMA(3,1,2) có thành phần xu thế tuyến tính

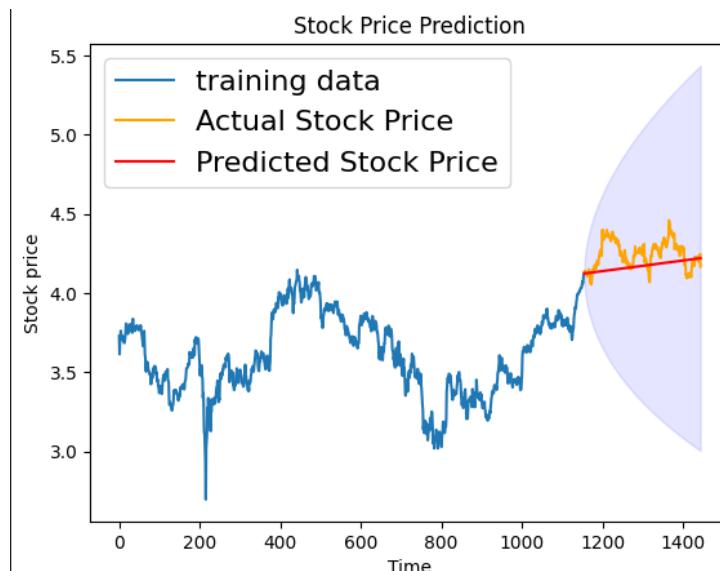
Nhận xét:

- Mô hình huấn luyện là ARIMA (3,1,2) với thành phần trend='t', tức có xu hướng tuyến tính theo thời gian.
- Trong các hệ số:
  - Chỉ có ar.L1, ma.L2, sigma2 là có p-value < 0.05 → có ý nghĩa thống kê.
  - Các hệ số còn lại (x1 – xu thế, ar.L2, ar.L3, ma.L1) đều có p-value > 0.05, cho thấy không đáng kể về mặt thống kê.
- So với mô hình ARIMA (2,1,2) trước đó:
  - AIC cao hơn (-4459.842 vs -4464.593)

- BIC cao hơn ( $-4424.485$  vs  $-4439.338$ ) → Mô hình ARIMA(3,1,2) tuy thêm thành phần xu thế nhưng không cải thiện chất lượng dự báo và ít hiệu quả hơn mô hình trước.
- Kiểm định phần dư:
  - Ljung-Box (Q) = 0.00, p = 0.87 → phần dư không có tương quan chuỗi mạnh.
  - Jarque-Bera (JB) = 4607.62, p = 0.00 → phần dư không hoàn toàn tuân chuẩn. Heteroskedasticity: 0.55, Skew: 0.54, Kurtosis: 12.73 → phần dư có đuôi dài, đúng tính chất dữ liệu tài chính.

Kết luận:

Mặc dù mô hình ARIMA (3,1,2) bổ sung xu thế tuyến tính, nhưng hiệu quả không vượt trội so với ARIMA (2,1,2). Do đó, mô hình ARIMA (2,1,2) vẫn là lựa chọn phù hợp hơn để sử dụng trong bước dự báo tiếp theo.



Hình 3.197: Biểu đồ dự báo giá cổ phiếu bằng mô hình ARIMA(2,1,2)

Nhận xét:

- Mô hình ARIMA (2,1,2) dự báo tốt chuỗi log giá đóng cửa trên tập kiểm tra.
- Đường dự báo (màu đỏ) nằm sát với đường giá thực tế (màu cam) và nằm trong vùng tin cậy 95%.

- Điều này chứng minh mô hình hoạt động ổn định và có khả năng dự báo đáng tin cậy

```

fc = fitted.get_forecast(len(test_data))
fc_value = fc.predicted_mean
fc_value.index = test_data.index
conf = fc.conf_int(alpha=0.05)
lower_series = conf['lower Close']
lower_series.index = test_data.index
upper_series = conf['upper Close']
upper_series.index = test_data.index
mse= mean_squared_error(test_data,fc_value)
print('test MSE: %.3f' % mse)
rmse=np.sqrt(mse)
print('test RMSE: {:.3f}'.format(rmse))
✓ 0.2s
test MSE: 0.013
test RMSE: 0.115

```

Hình 3.198: Dự báo và đánh giá sai số của mô hình ARIMA(2,1,2)

- Mô hình ARIMA (2,1,2) được sử dụng để dự báo chuỗi log giá đóng cửa trên tập kiểm tra.
- Sai số RMSE = 0.115, cho thấy mô hình có khả năng dự báo tốt, với độ lệch trung bình khá thấp so với dữ liệu thực tế.
- Dải tin cậy 95% (lower\_series, upper\_series) giúp đánh giá mức độ chắc chắn của dự đoán, phù hợp để trực quan hóa kết quả.

```

baseline_prediction = np.full_like(test_data, train_data.mean())
baseline_rmse=np.sqrt(mean_squared_error(test_data,baseline_prediction))

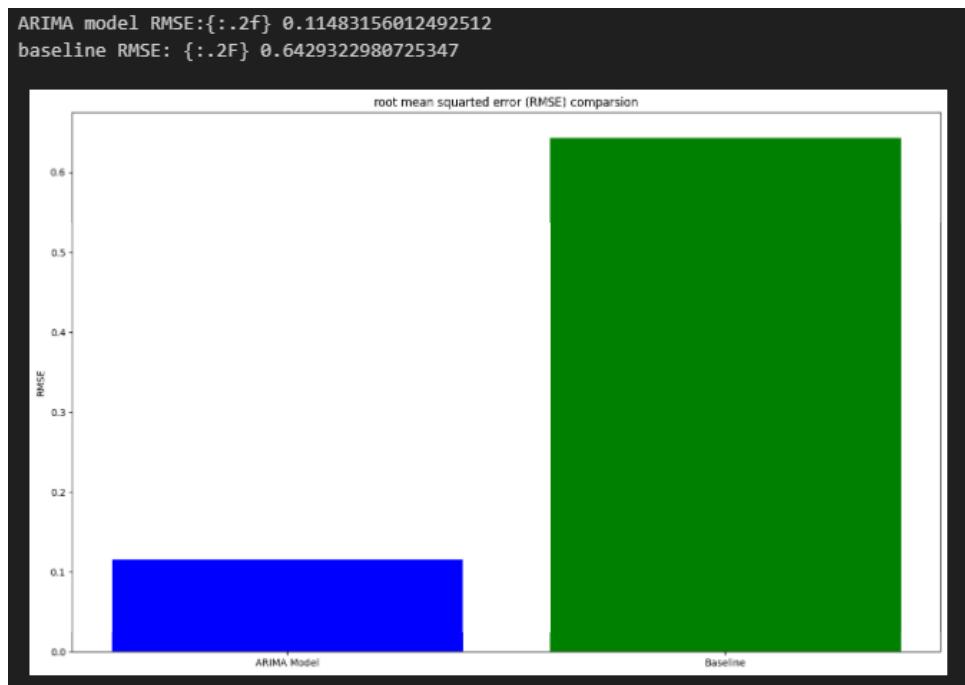
```

Hình 3.199: Dự báo baseline sử dụng trung bình tập huấn luyện

Giải thích:

- Dự đoán toàn bộ tập test bằng giá trị trung bình của tập huấn luyện.
- Tính RMSE để so sánh với mô hình ARIMA.

### 3.6.5. Đánh giá và so sánh kết quả



Hình 3.200: So sánh sai số RMSE giữa mô hình ARIMA và baseline

Nhận xét:

- Mô hình ARIMA (2,1,2) có RMSE = 0.115, nhỏ hơn rất nhiều so với baseline = 0.643.
- Điều này cho thấy mô hình ARIMA đã học được cấu trúc và xu hướng trong dữ liệu, thay vì chỉ dự đoán ngẫu nhiên theo giá trị trung bình.
- Khoảng cách giữa hai cột thể hiện mức độ cải thiện rõ rệt về độ chính xác.

Kết luận:

Mô hình ARIMA (2,1,2) vượt trội so với baseline về khả năng dự báo giá cổ phiếu Uber. Sai số RMSE giảm mạnh (chỉ còn khoảng 18% so với baseline), chứng tỏ mô hình đã nắm bắt được quy luật của chuỗi thời gian. Kết quả này khẳng định ARIMA là lựa chọn phù hợp và đáng tin cậy trong phân tích dự báo tài chính.

### 3.6.6. Kết luận

Trong phần này, mô hình ARIMA đã được triển khai để phân tích và dự báo chuỗi thời gian log giá đóng cửa của cổ phiếu Uber. Quá trình xử lý bao gồm kiểm tra tính

dùng bằng kiểm định ADF và KPSS, phân tích tự tương quan (ACF, PACF), và lựa chọn tham số tối ưu bằng auto\_arima. Kết quả cho thấy mô hình ARIMA (2,1,2) phù hợp nhất với dữ liệu.

Việc huấn luyện và đánh giá trên tập kiểm tra mang lại sai số RMSE = 0.115, thấp hơn đáng kể so với phương pháp baseline. Đồng thời, kiểm định phần dư cho thấy mô hình đã loại bỏ được tính chuỗi và không còn tự tương quan đáng kể.

Như vậy, ARIMA (2,1,2) được xác nhận là mô hình hiệu quả và đáng tin cậy trong việc dự báo chuỗi giá cổ phiếu, đặt nền tảng cho các ứng dụng tài chính như dự đoán biến động giá và hỗ trợ ra quyết định đầu tư.

## **CHƯƠNG 4. KẾT LUẬN VÀ KIẾN NGHỊ**

### **4.1. Kết luận**

Báo cáo đã tiến hành nghiên cứu và áp dụng nhiều mô hình hồi quy và học máy thống kê vào các bài toán dự báo và phân tích dữ liệu kinh tế thực tế. Các mô hình bao gồm: ARIMA cho chuỗi thời gian, hồi quy logistic cho phân loại nhị phân, KNN và SVM cho phân loại điểm tín dụng và phát hiện tiền giả, mạng nơ-ron nhân tạo cho các bài toán phức tạp, và K-Means, DBSCAN cho phân cụm khách hàng. Qua thực nghiệm, mỗi mô hình đều cho thấy hiệu quả riêng tùy vào đặc điểm của dữ liệu và mục tiêu phân tích. Bên cạnh việc lựa chọn mô hình phù hợp, quá trình xử lý dữ liệu như chuẩn hóa, mã hóa, giảm chiều hay cân bằng lớp cũng đóng vai trò quyết định trong hiệu quả dự báo.

### **4.2. Kiến nghị**

Từ kết quả thu được, nhóm kiến nghị rằng cần linh hoạt lựa chọn mô hình theo từng loại dữ liệu và bài toán cụ thể. Với chuỗi thời gian ổn định, ARIMA là lựa chọn phù hợp; trong khi đó, các mô hình như SVM, Logistic Regression hay Neural Network lại phát huy hiệu quả với dữ liệu phân loại hoặc phức tạp. Việc xử lý dữ liệu đầu vào cần được thực hiện kỹ lưỡng, đồng thời nên thử nghiệm nhiều mô hình để so sánh hiệu suất thay vì chỉ dùng một phương pháp cố định. Trong tương lai, có thể mở rộng theo hướng kết hợp mô hình (ensemble) hoặc tích hợp với nền tảng dữ liệu lớn để phục vụ các bài toán kinh tế có quy mô lớn và tính chất thay đổi liên tục.

## TÀI LIỆU THAM KHẢO

- [1] “Hồi quy tuyến tính,” *Wikipedia tiếng Việt*. Nov. 29, 2024. Accessed: Mar. 28, 2025. [Online]. Available: [https://vi.wikipedia.org/w/index.php?title=H%e1%bb%93i\\_quy\\_tuy%e1%ba%bfn\\_t%C3%ADnh&oldid=71973970](https://vi.wikipedia.org/w/index.php?title=H%e1%bb%93i_quy_tuy%e1%ba%bfn_t%C3%ADnh&oldid=71973970)
- [2] “Hồi quy tuyến tính trong học máy - GeeksforGeeks.” Accessed: Mar. 28, 2025. [Online]. Available: <https://www.geeksforgeeks.org/ml-linear-regression/>
- [3] “[ALL] Hồi quy tuyến tính - mà bạn cần biết - Phân tích nghiệp vụ.” Accessed: Mar. 28, 2025. [Online]. Available: <https://phantichnghiepvu.com/all-hoi-quy-tuyen-tinh-ma-ban-can-biet/>
- [4] “Linear Regression in Machine learning,” GeeksforGeeks. Accessed: Mar. 29, 2025. [Online]. Available: <https://www.geeksforgeeks.org/ml-linear-regression/>
- [5] “Everything You Need to Know About Logistic Regression - Spiceworks,” Spiceworks Inc. Accessed: Mar. 29, 2025. [Online]. Available: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/>
- [6] “Logistic Regression in Machine Learning,” GeeksforGeeks. Accessed: Mar. 29, 2025. [Online]. Available: <https://www.geeksforgeeks.org/understanding-logistic-regression/>
- [7] “Hồi quy logistic: Tính xác suất bằng hàm sigmoid | Machine Learning,” Google for Developers. Accessed: Mar. 29, 2025. [Online]. Available: <https://developers.google.com/machine-learning/crash-course/logistic-regression/sigmoid-function?hl=vi>
- [8] Gupta M., “Logistic Regression in Machine Learning,” Applied AI Blog. Accessed: Mar. 30, 2025. [Online]. Available: <https://www.appliedaicourse.com/blog/logistic-regression-in-machine-learning/>

- [9] “Everything You Need to Know About Logistic Regression - Spiceworks,” Spiceworks Inc. Accessed: Mar. 30, 2025. [Online]. Available: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/>
- [10] “1.1. Linear Models,” scikit-learn. Accessed: Mar. 30, 2025. [Online]. Available: [https://scikit-learn/stable/modules/linear\\_model.html](https://scikit-learn/stable/modules/linear_model.html)
- [11] trituenhantao.io, “Bài 6: Logistic Regression (Hồi quy Logistic),” Trí tuệ nhân tạo. Accessed: Mar. 30, 2025. [Online]. Available: <https://trituenhantao.io/machine-learning-co-ban/bai-6-logistic-regression-hoi-quy-logistic/>
- [12] Singh A., “Decision Tree in Machine Learning,” Applied AI Blog. Accessed: Mar. 30, 2025. [Online]. Available: <https://www.appliedaicourse.com/blog/decision-tree-in-machine-learning/>
- [13] “What is a Decision Tree? | IBM.” Accessed: Mar. 30, 2025. [Online]. Available: <https://www.ibm.com/think/topics/decision-trees>
- [14] “Decision Tree,” GeeksforGeeks. Accessed: Mar. 30, 2025. [Online]. Available: <https://www.geeksforgeeks.org/decision-tree/>
- [15] “Decision Tree,” GeeksforGeeks. Accessed: Mar. 30, 2025. [Online]. Available: <https://www.geeksforgeeks.org/decision-tree/>
- [16] “Random Forest là gì trong Machine Learning? Giải thích các thuật ngữ, cách hoạt động, và ví dụ thực tế.” Accessed: Mar. 30, 2025. [Online]. Available: <https://statio.vn/blog/random-forest-la-gi-trong-machine-learning-giai-thichcac-thuat-ngu-cach-hoat-ong-va-vi-du-thuc-te>
- [17] “khai phá dữ liệu\_random forest - NGÂN HÀNG NHÀ NƯỚC VIỆT NAM BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC - Studocu.” Accessed: Mar. 30, 2025. [Online]. Available: <https://www.studocu.vn/vn/document/truong-dai-hoc-ngan-hang-thanh-pho-ho-chi-minh/data-mining/khai-pha-du-lieu-random-forest/79464521>

- [18] “Support Vector Machine là gì? Tâm quan trọng của SVM trong ML.” Accessed: Mar. 30, 2025. [Online]. Available: <https://interdata.vn/blog/support-vector-machine-la-gi/>
- [19] “What Is Support Vector Machine? | IBM.” Accessed: Mar. 30, 2025. [Online]. Available: <https://www.ibm.com/think/topics/support-vector-machine>
- [20] T. Vu, “Bài 21: Kernel Support Vector Machine,” Tiep Vu’s blog. Accessed: Apr. 03, 2025. [Online]. Available: <https://machinelearningcoban.com/2017/04/22/kernelsmv/>
- [21] “Support Vector Machine (SVM): xử lý dữ liệu phi tuyến - Alcandy.” Accessed: Apr. 04, 2025. [Online]. Available: <https://aicandy.vn/support-vector-machine-svm-xu-ly-du-lieu-phi-tuyen/>
- [22] M. Saeed, “Method of Lagrange Multipliers: The Theory Behind Support Vector Machines (Part 1: The Separable Case),” MachineLearningMastery.com. Accessed: Apr. 04, 2025. [Online]. Available: <https://www.machinelearningmastery.com/method-of-lagrange-multipliers-theory-behind-support-vector-machines-part-1-the-separable-case/>
- [23] Lang N., “Support Vector Machine (SVM) - giải thích dễ hiểu! | Data Basecamp.” Accessed: Apr. 04, 2025. [Online]. Available: <https://databacecamp.de/en/ml/svm-explained>
- [24] admin, “Support Vector Machine (SVM): xử lý dữ liệu phi tuyến,” Alcandy. Accessed: Apr. 04, 2025. [Online]. Available: <https://aicandy.vn/support-vector-machine-svm-xu-ly-du-lieu-phi-tuyen/>
- [25] techslang, “What is an SVM? — Techslang,” Techslang — Tech Explained in Simple Terms. Accessed: Apr. 04, 2025. [Online]. Available: <https://www.techslang.com/definition/what-is-an-svm/>
- [26] “What is the k-nearest neighbors algorithm? | IBM.” Accessed: Apr. 01, 2025. [Online]. Available: <https://www.ibm.com/think/topics/knn>

- [27] “What is k-Nearest Neighbor (kNN)? | A Comprehensive k-Nearest Neighbor Guide.” Accessed: Apr. 01, 2025. [Online]. Available: <https://www.elastic.co/what-is/knn>
- [28] “What is the k-nearest neighbors algorithm? | IBM.” Accessed: Apr. 01, 2025. [Online]. Available: <https://www.ibm.com/think/topics/knn>
- [29] “What is the K-Nearest Neighbors (KNN) Algorithm?,” DataStax. Accessed: Apr. 01, 2025. [Online]. Available: <https://www.datastax.com/guides/what-is-k-nearest-neighbors-knn-algorithm>
- [30] “K-Nearest Neighbors (KNN): Real-World Applications | Keylabs,” Keylabs: latest news and updates. Accessed: Apr. 01, 2025. [Online]. Available: <https://keylabs.ai/blog/k-nearest-neighbors-knn-real-world-applications/>
- [31] Sơn N. N., “K means là gì? Các công bố khoa học về K means,” Scholar Hub. Accessed: Apr. 03, 2025. [Online]. Available: <https://scholarhub.vn/topic/k%20means>
- [32] Sharma N., “K-Means Clustering Explained,” neptune.ai. Accessed: Apr. 03, 2025. [Online]. Available: <https://neptune.ai/blog/k-means-clustering>
- [33] “Determining The Optimal Number Of Clusters: 3 Must Know Methods,” Datanovia. Accessed: Apr. 03, 2025. [Online]. Available: <https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/>
- [34] admin, “K-Means Clustering: Ưu nhược điểm và hướng áp dụng,” Alcandy. Accessed: Apr. 03, 2025. [Online]. Available: <https://alcandy.vn/k-means-clusteringuu-nhuoc-diem-va-huong-ap-dung/>
- [35] Sharma N., “K-Means Clustering Explained,” neptune.ai. Accessed: Apr. 03, 2025. [Online]. Available: <https://neptune.ai/blog/k-means-clustering>
- [36] “DBSCAN,” *Wikipedia tiếng Việt*. Mar. 01, 2021. Accessed: Apr. 03, 2025. [Online]. Available: [https://vi.wikipedia.org/w/index.php?title=DBSCAN&oldid=64535681#cite\\_note-2](https://vi.wikipedia.org/w/index.php?title=DBSCAN&oldid=64535681#cite_note-2)

- [37] D. R. Yehoshua, “DBSCAN: Density-Based Clustering,” Medium. Accessed: Apr. 03, 2025. [Online]. Available: <https://ai.plainenglish.io/dbscan-density-based-clustering-aaebd76e2c8c>
- [38] “DBSCAN Clustering Algorithm Demystified,” Built In. Accessed: Apr. 03, 2025. [Online]. Available: <https://builtin.com/articles/dbscan>
- [39] “DBSCAN vs. K-Means: A Guide in Python,” New Horizons. Accessed: Apr. 03, 2025. [Online]. Available: <https://www.newhorizons.com/resources/blog/dbscan-vs-kmeans-a-guide-in-python>
- [40] Verma A., “Unveiling the Power of DBSCAN: A Comprehensive Guide,” Medium. Accessed: Apr. 03, 2025. [Online]. Available: <https://medium.com/@ajayverma23/unveiling-the-power-of-dbscan-a-comprehensive-guide-bc954d742611>
- [41] “Mạng nơ ron nhân tạo là gì? Ứng dụng mạng nơ ron.” Accessed: Apr. 04, 2025. [Online]. Available: <https://www.elcom.com.vn/mang-no-ron-nhan-tao-la-gi-tai-sao-chung-ta-can-mang-no-ron-nhan-tao-1669018888>
- [42] “Cơ bản và khái niệm về mạng nơ-ron – Học bằng Marketing.” Accessed: Apr. 04, 2025. [Online]. Available: <https://www.learnbymarketing.com/methods/neural-networks/>
- [43] “Mạng nơ-ron là gì? – Giải thích về mạng nơ-ron nhân tạo – AWS,” Amazon Web Services, Inc. Accessed: Apr. 04, 2025. [Online]. Available: <https://aws.amazon.com/vi/what-is/neural-network/>
- [44] Hưng N., “Neural Network là gì? Cách hoạt động, cấu trúc và ứng dụng.” Accessed: Apr. 04, 2025. [Online]. Available: <https://vietnix.vn/neural-network/>
- [45] “Hàm kích hoạt — Tài liệu ML Glossary.” Accessed: Apr. 04, 2025. [Online]. Available: [https://ml-glossary-vn.readthedocs.io/vi/latest/activation\\_functions.html?utm\\_source=chatgpt.com](https://ml-glossary-vn.readthedocs.io/vi/latest/activation_functions.html?utm_source=chatgpt.com)
- [46] “Neural Network Examples, Applications, and Use Cases,” Coursera. Accessed: Apr. 04, 2025. [Online]. Available: <https://www.coursera.org/articles/neural-network-example>

- [47] Sơn N. N., “Arima là gì? Các công bố khoa học về Arima,” Scholar Hub. Accessed: Apr. 08, 2025. [Online]. Available: <https://scholarhub.vn/topic/arima>
- [48] J. Brownlee, “How to Create an ARIMA Model for Time Series Forecasting in Python,” MachineLearningMastery.com. Accessed: Apr. 08, 2025. [Online]. Available: <https://www.machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>
- [49] Chetty R. J., “Introduction to the Autoregressive Integrated Moving Average (ARIMA) model,” Knowledge Tank. Accessed: Apr. 08, 2025. [Online]. Available: <https://www.projectguru.in/introduction-to-the-autoregressive-integrated-moving-average-arima-model/>
- [50] “Autoregressive integrated moving average,” Wikipedia. Mar. 09, 2025. Accessed: Apr. 08, 2025. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Autoregressive\\_integrated\\_moving\\_average&oldid=1279677557](https://en.wikipedia.org/w/index.php?title=Autoregressive_integrated_moving_average&oldid=1279677557)