

ĐẠI HỌC HUẾ  
TRƯỜNG ĐẠI HỌC KINH TẾ  
KHOA HỆ THÔNG THÔNG TIN KINH TẾ



**ĐỒ ÁN CÁ NHÂN**

Học Phần: KHAI PHÁ DỮ LIỆU  
BỘ DỮ LIỆU ĐIỂM THI BẰNG LÁI XE

GVHD: TS. Hoàng Hữu Trung

SVTH: Trần Hoàng Phương Dung

MSV: 23K4300004

Lớp: K57-Kinh Tế Sô

Thành phố Huế, tháng 7 năm 2025

## LỜI CÁM ƠN

Lần đầu tiên em xin chân thành cảm ơn giảng viên TS. Hoàng Hữu Trung đã tận tình giảng dạy và hướng dẫn sinh viên trong suốt thời gian môn học. Nhờ vào những lời khuyên và chỉ bảo đúng lúc của thầy, em đã vượt qua khó khăn khi thực hiện bài đồ án của mình.

Tiếp đến em xin gửi lời tri ân tới các thầy cô trường Đại học Kinh Tế, Đại học Huế - Những người đã cùng góp sức truyền đạt kiến thức để giúp em có được nền tảng như ngày hôm nay. Ngoài ra, không thể không nhắc tới gia đình, bạn bè người thân đã là hậu phương vững chắc, là chỗ dựa tinh thần của em. Sự thành công của bài đồ án hôm nay không thể không kể đến công ơn của mọi người.

Nhưng sau tất cả, em nhận thức rằng với lượng kiến thức và kinh nghiệm ít ỏi của bản thân, chắc chắn bài đồ án này sẽ khó tránh khỏi thiếu sót. Kính mong quý thầy cô thông cảm và góp ý để em ngày càng hoàn thiện hơn.

## MỤC LỤC

<b>LỜI CÁM ƠN .....</b>	<b>2</b>
<b>MỤC LỤC.....</b>	<b>3</b>
<b>DANH MỤC BẢNG BIỂU .....</b>	<b>7</b>
<b>DANH MỤC HÌNH ẢNH.....</b>	<b>10</b>
<b>I. GIỚI THIỆU VỀ BỘ DỮ LIỆU.....</b>	<b>14</b>
<b>II. TIỀN XỬ LÝ DỮ LIỆU .....</b>	<b>19</b>
<b>2.1. Làm sạch dữ liệu (Data Cleaning) .....</b>	<b>20</b>
<b>2.1.1. Tìm những giá trị bị trùng lắp .....</b>	<b>20</b>
<b>2.1.2. Xử lý các giá trị còn thiếu (Missing value) .....</b>	<b>20</b>
<b>2.1.3. Xử lý dữ liệu nhiễu (Noise Data) .....</b>	<b>23</b>
<b>2.2. Chuyển đổi dữ liệu (Data Transformation) .....</b>	<b>28</b>
<b>2.2.1. Thống kê các biến phân loại (categorical) và biến số (numeric) ...</b>	<b>28</b>
<b>2.2.2. Mã hóa biến phân loại (Encoding Categorical Variables) .....</b>	<b>28</b>
a. Mã hóa bằng phương pháp Label Encoding .....	29
b. Mã hóa bằng phương pháp Ordinal Encoding .....	29
c. Mã hóa bằng phương pháp One-hot Encoding .....	30
<b>2.3. Chuẩn hóa dữ liệu (Normalization).....</b>	<b>32</b>
<b>2.3.1. Đổi với dữ liệu chưa loại bỏ outliers .....</b>	<b>32</b>
a. Chuẩn hóa bằng phương pháp Min-Max Scaler .....	32
b. Chuẩn hóa bằng phương pháp RobustScaler .....	32
<b>2.3.2. Đổi với dữ liệu đã loại bỏ outliers.....</b>	<b>34</b>
<b>IV. DỮ LIỆU SAU KHI TIỀN XỬ LÝ .....</b>	<b>34</b>
<b>    4.1. Dữ liệu trước khi loại bỏ outliers .....</b>	<b>34</b>

<b>4.2. Dữ liệu sau khi loại bỏ outliers.....</b>	<b>36</b>
<b>V. CÁC THUẬT TOÁN HỌC TẬP CÓ GIÁM SÁT (SUPERVISED LEARNING) .....</b>	<b>37</b>
<b>PHẦN A: THƯ VIỆN LAZYPREDICT .....</b>	<b>37</b>
<b>A.1. Đổi với dữ liệu trước khi loại bỏ outliers .....</b>	<b>38</b>
<b>A.2. Đổi với dữ liệu sau khi loại bỏ outliers .....</b>	<b>40</b>
<b>A.3. So sánh kết quả giữa trước và sau khi loại bỏ outliers .....</b>	<b>42</b>
<i>    A.3.1. Trực quan hóa kết quả trước và sau khi loại bỏ outliers .....</i>	<i>    42</i>
<i>    A.3.2. Mức độ thay đổi Accuracy của các mô hình .....</i>	<i>    43</i>
<b>A.4. Lựa chọn thuật toán để phân tích.....</b>	<b>45</b>
<i>    A.4.1. Thuật toán RandomForestClassifier.....</i>	<i>    46</i>
<i>        a. Dữ liệu trước khi loại bỏ outliers .....</i>	<i>        46</i>
<i>        b. Dữ liệu sau khi loại bỏ outliers .....</i>	<i>        51</i>
<i>    A.4.2. Thuật toán ExtraTreesClassifier .....</i>	<i>    56</i>
<i>        a. Dữ liệu trước khi loại bỏ outliers .....</i>	<i>        56</i>
<i>        b. Dữ liệu sau khi loại bỏ outliers .....</i>	<i>        60</i>
<i>        c. So sánh thuật toán ExtraTreesClassifier trước và sau khi loại bỏ outliers .....</i>	<i>        64</i>
<i>    A.4.3. Thuật toán AdaBoostClassifier .....</i>	<i>    66</i>
<i>        a. Dữ liệu trước khi loại bỏ outliers .....</i>	<i>        66</i>
<i>        b. Dữ liệu sau khi loại bỏ outliers .....</i>	<i>        70</i>
<i>        c. So sánh thuật toán AdaBoostClassifier trước và sau khi loại bỏ outliers .....</i>	<i>        74</i>
<b>B. HỒI QUY .....</b>	<b>76</b>
<b>B.1. Phân tích tương quan .....</b>	<b>76</b>

<b>B.1.1. Dữ liệu trước khi loại bỏ outliers .....</b>	76
<b>B.1.2. Dữ liệu sau khi loại bỏ outliers .....</b>	77
<b>B.1.3. So sánh dữ liệu trước và sau khi loại bỏ outliers.....</b>	79
<b>B.2. Hồi quy tuyến tính .....</b>	80
<i>B.2.1. Dữ liệu trước khi xóa outliers .....</i>	80
<i>B.2.2. Dữ liệu sau khi loại bỏ outliers .....</i>	82
<b>B.3. Hồi quy Logistics.....</b>	85
<i>B3.1. Dữ liệu chưa loại bỏ outliers.....</i>	85
<i>B.3.2. Dữ liệu sau khi loại bỏ outliers .....</i>	91
<b>PHẦN C: THUẬT TOÁN KNN, SVM.....</b>	96
<b>C.1. Thuật toán K-nearest Neighbors (KNN) .....</b>	96
<i>C.1.1. Dữ liệu trước khi loại bỏ outliers .....</i>	96
<i>C.1.2. Dữ liệu sau khi loại bỏ outliers .....</i>	104
<b>C.2. Thuật toán Support Vector Machine (SVM) .....</b>	111
<i>C.2.1. Dữ liệu trước khi loại bỏ outliers .....</i>	111
<i>C.2.2. Dữ liệu sau khi loại bỏ outliers .....</i>	118
<b>VI. PHÂN CỤM DỮ LIỆU – UNSUPERVISED LEARNING ..</b>	124
<b>A. THUẬT TOÁN K-MEANS VÀ DBSCAN .....</b>	124
<b>A.1. Thuật toán K-Means .....</b>	124
<i>A.1.1. Phân cụm dựa trên Mirror Usage ~ Steer Control .....</i>	127
<i>a. Dữ liệu trước khi loại bỏ outliers .....</i>	127
<i>b. Dữ liệu sau khi loại bỏ outliers .....</i>	131
<i>A.1.2. Phân cụm dựa trên toàn bộ dữ liệu .....</i>	136
<i>a. Dữ liệu trước khi loại bỏ outliers .....</i>	136

<i>b. Dữ liệu sau khi loại bỏ outliers .....</i>	139
<b>A.2. Thuật toán DBSCAN.....</b>	<b>142</b>
<i>A.2.1. Sử dụng 2 biến đặc trưng .....</i>	142
<i>a. Dữ liệu trước khi loại bỏ outliers .....</i>	142
<i>b. Dữ liệu sau khi loại bỏ outliers .....</i>	148
<i>A.2.2. Sử dụng tất cả các biến đặc trưng.....</i>	153
<i>a. Dữ liệu trước khi loại bỏ outliers .....</i>	153
<i>b. Dữ liệu sau khi loại bỏ outliers .....</i>	158
<b>B. THUẬT TOÁN GAUSSIAN MATRIX .....</b>	<b>162</b>
<b>B.1. Sử dụng 2 biến đặc trưng .....</b>	<b>164</b>
<i>B.1.1. Dữ liệu trước khi loại bỏ outliers.....</i>	164
<i>B.1.2. Dữ liệu sau khi loại bỏ outliers .....</i>	170
<b>B.2. Sử dụng tất cả các biến.....</b>	<b>175</b>
<i>B.2.1. Dữ liệu trước khi loại bỏ outliers.....</i>	175
<i>B.2.2. Dữ liệu sau khi loại bỏ outliers .....</i>	180
<b>C. ĐÁNH GIÁ HIỆU QUẢ PHÂN CỤM CỦA CÁC MÔ HÌNH .....</b>	<b>184</b>
<b>C.1. Sử dụng tất cả các biến đầu vào với dữ liệu trước khi loại bỏ outliers .....</b>	<b>184</b>
<b>C.2. Sử dụng tất cả biến đầu vào với dữ liệu sau khi loại bỏ outliers ....</b>	<b>185</b>
<b>C.3. Sử dụng 2 biến đặc trưng với dữ liệu trước khi loại bỏ outliers.....</b>	<b>187</b>
<b>C.4. Sử dụng 2 biến đặc trưng đối với dữ liệu sau khi loại bỏ outliers ..</b>	<b>188</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>190</b>

# DANH MỤC BẢNG BIỂU

Bảng 1: Tóm tắt các thuộc tính số (Numeric Variables) .....	16
Bảng 2: Bảng mô tả các biến phân loại (Categorical Variables).....	17
Bảng 3: Kết quả các giá trị ngoại lai của các biến .....	25
Bảng 4: Tóm tắt dữ liệu sau khi tiền xử lý trước khi loại bỏ các outliers.....	34
Bảng 5: Tóm tắt dữ liệu sau khi tiền xử lý đã loại bỏ các outliers.....	36
Bảng 6: Kết quả sau khi chạy thư viện LazyPredict của dữ liệu trước khi loại bỏ outliers .....	38
Bảng 7: Kết quả sau khi chạy thư viện LazyPredict của dữ liệu sau khi loại bỏ outliers .....	40
Bảng 8: Các mô hình cải thiện rõ rệt sau khi loại bỏ outliers .....	43
Bảng 9: Các mô hình duy trì hiệu suất cao ổn định .....	44
Bảng 10: Mô hình có Accuracy suy giảm .....	45
Bảng 11: Lựa chọn ba thuật toán để phân tích .....	46
Bảng 13: Nhận xét hiệu quả của mô hình RandomForestClassifier trước khi loại bỏ outliers .....	50
Bảng 15: Nhận xét hiệu quả của mô hình RandomClassifier sau khi loại bỏ outliers .....	54
Bảng 16: Đánh giá các chỉ số trước và sau khi loại bỏ outliers – KNeighborsClassifier .....	55
Bảng 17: Kết quả mô hình ExtraTreesClassifier trước khi loại bỏ outliers.....	57
Bảng 18: Nhận xét hiệu quả của mô hình ExtraTreesClassifier trước khi loại bỏ outliers .....	59
Bảng 19: Kết quả mô hình ExtraTreesClassifier sau khi loại bỏ outliers .....	61
Bảng 20: Nhận xét hiệu quả của mô hình ExtraTreesClassifier sau khi loại bỏ outliers .....	63
Bảng 21: Đánh giá các chỉ số trước và sau khi loại bỏ outliers - ExtraTreesClassifier .....	64

Bảng 22: Kết quả mô hình AdaBoostClassifier trước khi loại bỏ outliers .....	67
Bảng 23: Nhận xét hiệu quả của mô hình AdaBoostClassifier trước khi loại bỏ outliers .....	69
Bảng 24: Kết quả mô hình AdaBoostClassifier sau khi loại bỏ outliers.....	71
Bảng 25: Nhận xét hiệu quả của mô hình AdaBoostClassifier sau khi loại bỏ outliers .....	73
Bảng 26: Đánh giá các chỉ số trước và sau khi loại bỏ outliers - AdaBoostClassifier .....	74
Bảng 27: Nhận xét ma trận tương quan trước khi loại bỏ outliers.....	77
Bảng 28: Nhận xét ma trận tương quan sau khi loại bỏ outliers.....	78
Bảng 29: So sánh ma trận tương quan trước và sau khi loại bỏ outliers .....	79
Bảng 30: Nhận xét hồi quy tuyến tính Steer Control – Mirror Usage trước khi loại bỏ outliers .....	82
Bảng 31: Nhận xét hồi quy tuyến tính Steer Control – Mirror Usage trước khi loại bỏ outliers .....	84
Bảng 32: Nhận xét Logistics Regression – Dữ liệu trước khi loại bỏ outliers ...	90
Bảng 33: Nhận xét Logistics Regression – Dữ liệu sau khi loại bỏ outliers .....	95
Bảng 34: Nhận xét thuật toán KNN – Dữ liệu trước khi loại bỏ outliers .....	103
Bảng 35: Nhận xét thuật toán KNN – Dữ liệu trước khi loại bỏ outliers .....	110
Bảng 36: Nhận xét thuật toán SVM – Dữ liệu trước khi loại bỏ outliers .....	117
Bảng 37: Nhận xét thuật toán SVM – Dữ liệu sau khi loại bỏ outliers .....	123
Bảng 38: Kỹ năng của các cụm trong Kmeans dựa trên 2 biến (Dữ liệu trước khi loại bỏ outliers).....	130
Bảng 39: Kỹ năng của các cụm trong Kmeans dựa trên 2 biến (Dữ liệu sau khi loại bỏ outliers).....	134
Bảng 40: Nhận xét các cụm DBSCAN – 2 biến (Dữ liệu trước khi loại bỏ outliers) .....	147
Bảng 41: Nhận xét các cụm DBSCAN – 2 biến (Dữ liệu sau khi loại bỏ outliers) .....	152

Bảng 42: Nhận xét các cụm DBSCAN – tất cả các biến (Dữ liệu trước khi loại bỏ outliers).....	157
Bảng 43: Nhận xét các cụm DBSCAN – tất cả các biến (Dữ liệu sau khi loại bỏ outliers).....	160
Bảng 44: Nhận xét các cụm GMM – 2 các biến (Dữ liệu trước khi loại bỏ outliers) .....	169
Bảng 45: Nhận xét các cụm GMM – 2 các biến (Dữ liệu sau khi loại bỏ outliers) .....	174
Bảng 46: Nhận xét các cụm GMM – tất cả các biến (Dữ liệu trước khi loại bỏ outliers).....	179
Bảng 47: Nhận xét các cụm GMM – tất cả các biến (Dữ liệu sau khi loại bỏ outliers).....	183
Bảng 48: Đánh giá phân cụm với tất cả các biến – Dữ liệu trước khi loại bỏ outliers .....	185
Bảng 49: Đánh giá phân cụm với tất cả các biến – Dữ liệu sau khi loại bỏ outliers .....	186
Bảng 50: Đánh giá phân cụm với 2 biến – Dữ liệu trước khi loại bỏ outliers ..	188
Bảng 51: Đánh giá phân cụm với 2 biến – Dữ liệu sau khi loại bỏ outliers .....	189

# DANH MỤC HÌNH ẢNH

Hình 1: Trực quan hóa các biến phân loại (Categorical Variables) .....	19
Hình 2: Phân phôi các Missing value.....	22
Hình 3: Biểu đồ Boxplot để phát hiện outliers của các biến số (Numeric Variables) .....	26
Hình 4: So sánh Accuracy giữa các mô hình (Trước và sau khi loại bỏ outliers) .....	43
Hình 5: Confusion Matrix – RandomForestClassifier trước khi loại bỏ outliers	49
Hình 6: ROC Curve – RandomForestClassifier trước khi loại bỏ outliers.....	50
Hình 7: Confusion Matrix – RandomForestClassifier sau khi loại bỏ outliers ..	53
Hình 8: ROC Curve – RandomForestClassifier sau khi loại bỏ outliers .....	54
Hình 9: Confusion Matrix – ExtraTreesClassifier trước khi loại bỏ outliers ..	58
Hình 10: ROC Curve – ExtraTreesClassifier trước khi loại bỏ outliers .....	59
Hình 11: Confusion Matrix – ExtraTreesClassifier sau khi loại bỏ outliers.....	62
Hình 12: ROC Curve – ExtraTreesClassifier sau khi loại bỏ outliers .....	63
Hình 13: Confusion Matrix – AdaBoostClassifier trước khi loại bỏ outliers ..	68
Hình 14: ROC Curve – AdaBoostClassifier trước khi loại bỏ outliers.....	69
Hình 15: Confusion Matrix – AdaBoostClassifier sau khi loại bỏ outliers .....	72
Hình 16: ROC Curve – AdaBoostClassifier sau khi loại bỏ outliers .....	73
Hình 17: Ma trận tương quan trước khi loại bỏ outliers .....	76
Hình 18: Ma trận tương quan sau khi loại bỏ outliers .....	78
Hình 19: Hồi quy tuyến tính Mirror Usage ~ Steer Control (trước khi loại bỏ outliers).....	81
Hình 20: Hồi quy tuyến tính Mirror Usage ~ Steer Control (sau khi loại bỏ outliers).....	84
Hình 21: Confusion Matrix – Logistics Regression (Mirror Usage ~ Qualified) – Trước khi loại bỏ outliers .....	88

Hình 22: So sánh thực tế - dự đoán Logistics Regression (Mirror Usage ~ Qualified) – Trước khi loại bỏ outliers.....	89
Hình 23: ROC Curve - Logistics Regression (Mirror Usage ~ Qualified) – Trước khi loại bỏ outliers.....	90
Hình 24: Confusion Matrix – Logistics Regression (Mirror Usage – Qualified) – Dữ liệu sau khi loại bỏ outliers .....	93
Hình 25: So sánh thực tế - dự đoán – Logistics Regression (Mirror Usage – Qualified) – Dữ liệu sau khi loại bỏ outliers.....	94
Hình 26: ROC Curve – Logistics Regression (Mirror Usage – Qualified) – Dữ liệu sau khi loại bỏ outliers .....	95
Hình 27: Biểu đồ ranh giới quyết định cho thuật toán KNN với biến đầu vào là tất cả các đặc trưng – Dữ liệu trước khi loại bỏ outliers .....	101
Hình 28: Confusion Matrix – KNN (Dữ liệu trước khi loại bỏ outliers).....	102
Hình 29: ROC Curve – KNN (Dữ liệu trước khi loại bỏ outliers) .....	103
Hình 30: Biểu đồ ranh giới quyết định cho thuật toán KNN với biến đầu vào là tất cả các đặc trưng – Dữ liệu sau khi loại bỏ outliers .....	108
Hình 31: Confusion Matrix – KNN (Dữ liệu sau khi loại bỏ outliers) .....	109
Hình 32: ROC – Curve – KNN (Dữ liệu sau khi loại bỏ outliers).....	110
Hình 33: Biểu đồ ranh giới quyết định cho mô hình SVM – Dữ liệu trước khi loại bỏ outliers .....	115
Hình 34: Confusion Matrix – SVM (Dữ liệu trước khi loại bỏ outliers).....	116
Hình 35: ROC Curve – SVM (Dữ liệu trước khi loại bỏ outliers) .....	117
Hình 36: Biểu đồ ranh giới quyết định cho mô hình SVM – Dữ liệu sau khi loại bỏ outliers .....	121
Hình 37: Confusion Matrix – SVM (Dữ liệu sau khi loại bỏ outliers) .....	122
Hình 38: ROC Curve – SVM (Dữ liệu sau khi loại bỏ outliers).....	123
Hình 39: Biểu đồ Elbow để xác định số cụm tối ưu .....	126
Hình 40: Mô hình phân cụm K-Means 2 biến (Dữ liệu trước khi loại bỏ outliers) .....	128

Hình 41: So sánh đặc trưng trung bình giữa các cụm (Kmeans với 2 biến) – Dữ liệu trước khi loại bỏ outliers .....	130
Hình 42: Mô hình phân cụm K-Means 2 biến (Dữ liệu sau khi loại bỏ outliers) .....	133
Hình 43: So sánh đặc trưng trung bình giữa các cụm (Kmeans với 2 biến) – Dữ liệu sau khi loại bỏ outliers .....	134
Hình 44: Biểu đồ k-distance graph để chọn tham số eps phù hợp cho 2 biến..	143
Hình 45: Phân cụm DBSCAN 2 biến (Dữ liệu trước khi loại bỏ outliers).....	145
Hình 46: So sánh đặc trưng giữa các cụm (DBSCAN với 2 biến) – Dữ liệu trước khi loại bỏ outliers.....	146
Hình 47: Biểu đồ k-distance graph để chọn tham số eps phù hợp cho 2 biến..	148
Hình 48: Phân cụm DBSCAN 2 biến (Dữ liệu sau khi loại bỏ outliers) .....	150
Hình 49: So sánh đặc trưng giữa các cụm (DBSCAN với 2 biến) – Dữ liệu sau khi loại bỏ outliers .....	151
Hình 50: Biểu đồ k-distance graph để chọn tham số eps phù hợp.....	154
Hình 51: Phân cụm DBSCAN tất cả các biến (Dữ liệu trước khi loại bỏ outliers) .....	156
Hình 52: So sánh đặc trưng giữa các cụm (DBSCAN với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers .....	157
Hình 53: Phân cụm DBSCAN tất cả các biến (Dữ liệu sau khi loại bỏ outliers) .....	159
Hình 54: So sánh đặc trưng giữa các cụm (DBSCAN với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers .....	160
Hình 55: Biểu đồ điểm BIC – 2 biến (Dữ liệu trước khi loại bỏ outliers).....	165
Hình 56: Phân cụm GMM – 2 biến (Dữ liệu trước khi loại bỏ outliers) .....	168
Hình 57: So sánh đặc trưng giữa các cụm (GMM với 2 biến) – Dữ liệu trước khi loại bỏ outliers .....	169
Hình 58: Biểu đồ điểm BIC – 2 biến (Dữ liệu sau khi loại bỏ outliers) .....	171
Hình 59: Phân cụm GMM – 2 biến (Dữ liệu sau khi loại bỏ outliers) .....	173

Hình 60: So sánh đặc trưng giữa các cụm (GMM với 2 biến) – Dữ liệu sau khi loại bỏ outliers .....	174
Hình 61: Biểu đồ điểm BIC – tất cả các biến (Dữ liệu trước khi loại bỏ outliers) .....	176
Hình 62: Phân cụm GMM – tất cả các biến (Dữ liệu trước khi loại bỏ outliers) .....	178
Hình 63: So sánh đặc trưng giữa các cụm (GMM với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers .....	179
Hình 64: Phân cụm GMM – tất cả các biến (Dữ liệu sau khi loại bỏ outliers) .....	181
Hình 65: So sánh đặc trưng giữa các cụm (GMM với tất cả các biến) – Dữ liệu sau khi loại bỏ outliers .....	182

## I. GIỚI THIỆU VỀ BỘ DỮ LIỆU

Bộ dữ liệu “**Driver's License Test Scores Data**” là một tập dữ liệu mô phỏng nhằm phân tích các yếu tố ảnh hưởng đến khả năng vượt qua kì thi bằng lái xe của các ứng viên xin cấp giấy phép lái xe mới. Dữ liệu này bao gồm các thông tin về việc liệu ứng viên có vượt qua bài kiểm tra hay không, cùng các thuộc tính khác như mã số ứng viên, giới tính, độ tuổi, chủng tộc, các khoá đào tạo trước đó, và tất cả các chỉ số đánh giá trên đường.

Thông qua việc phân tích dữ liệu này, có thể rút ra nhiều Insights khác nhau - ví dụ như dự đoán khả năng đậu bài kiểm tra đường bộ của các ứng viên trong tương lai.

Dữ liệu gồm 500 quan sát với 16 thuộc tính, trong đó có các thuộc tính sau:

- **Applicant ID:** Mã định danh duy nhất cho mỗi ứng viên, có định dạng "AID" theo sau bởi số thứ tự (ví dụ: AID0001, AID0002,..., AID0500).
- **Gender (Giới tính):** Biến phân loại với hai giá trị - Female (nữ) chiếm 51.2% và Male (nam) chiếm 48.8%.
- **Age Group (Nhóm tuổi):** Chia thành ba nhóm - Young Adult (thanh niên) chiếm tỷ lệ cao nhất với 48.2%, Middle Age (trung niên) chiếm 29.4% và Teenager (thiếu niên) chiếm 22.4%.
- **Race (Chủng tộc):** Phân chia tương đối đều giữa ba nhóm - Other (khác) 34.2%, White (da trắng) 33.0% và Black (da đen) 32.8%.
- **Training (Loại đào tạo):** Biến quan trọng với ba mức độ - Basic (cơ bản) 39.6%, Advanced (nâng cao) 30.4% và 30.0% ứng viên không tham gia đào tạo. Loại đào tạo có tác động mạnh đến kết quả thi với tỷ lệ đỗ khác biệt rõ rệt giữa các nhóm.
- **Reactions (Tốc độ phản ứng):** Đánh giá khả năng phản ứng của ứng viên với ba mức độ - Average (trung bình) 48.8%, Fast (nhanh) 30.8% và Slow (chậm) 20.4%.

- **Signals (Hiểu biển báo):** Đánh giá khả năng nhận biết và tuân thủ các tín hiệu giao thông, điểm trung bình 48.53.
- **Yield (Nhường đường):** Kỹ năng nhường đường cho phương tiện khác, điểm trung bình 47.78.
- **Speed Control (Kiểm soát tốc độ):** Khả năng duy trì tốc độ phù hợp, điểm trung bình 47.93.
- **Night Drive (Lái xe ban đêm):** Kỹ năng lái xe trong điều kiện ánh sáng yếu, điểm trung bình 47.88.
- **Road Signs (Biển báo đường bộ):** Hiểu biết về các loại biển báo giao thông, điểm trung bình 47.79.
- **Steer Control (Kiểm soát lái):** Khả năng điều khiển vô lăng chính xác, điểm trung bình 47.43.
- **Mirror Usage (Sử dụng gương):** Kỹ năng quan sát qua gương chiếu hậu, điểm trung bình 48.16.
- **Confidence (Sự tự tin):** Đánh giá mức độ tự tin khi lái xe, điểm trung bình 47.65.
- **Parking (Đỗ xe):** Kỹ năng đỗ xe trong các tình huống khác nhau, điểm trung bình 48.45.
- **Theory Test (Thi lý thuyết):** Điểm thi lý thuyết cao hơn đáng kể với trung bình 70.62, dao động từ 40.10 đến 99.75.
- **Qualified (Kết quả đỗ):** Biến mục tiêu cho biết ứng viên có đỗ hay không, với tỷ lệ đỗ tổng thể là 49.8% (249 người đỗ, 251 người không đỗ).

```

from tabulate import tabulate
import pandas as pd

# Đọc dữ liệu từ file csv
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Drivers License Data.csv')

# Chỉ lấy các cột có kiểu dữ liệu số
numerical_columns = df.select_dtypes(include=['number'])

# Tính các thống kê
summary = pd.DataFrame({
    'Thuộc tính': numerical_columns.columns,
    'Giá trị nhỏ nhất': numerical_columns.min().values,
    'Giá trị lớn nhất': numerical_columns.max().values,
    'Giá trị trung bình': numerical_columns.mean().values,
    'Độ lệch chuẩn': numerical_columns.std().values,
    'Số giá trị duy nhất': [numerical_columns[col].nunique() for col in numerical_columns.columns]
})

# Hiển thị bảng với tabulate
print("Tóm tắt các thuộc tính dữ liệu số:")
print(tabulate(summary, headers='keys', tablefmt='fancy_grid', showindex=False))

```

Bảng 1: Tóm tắt các thuộc tính số (Numeric Variables)

Thuộc tính	GTNN	GTLN	GTTB	Độ lệch chuẩn	Số giá trị duy nhất
Signals	0,77	95,61	48,5255	16,2949	485
Yield	0	94,41	47,784	16,0622	482
Speed Control	0	89,43	47,9317	15,6012	477
Night Drive	1,74	89,93	47,8778	15,3027	473
Road Signs	0	88,87	47,7934	16,0429	473
Steer Control	0	89,24	47,4295	15,58	475
Mirror Usage	0	90,77	48,1576	15,4599	473
Confidence	5,12	90,28	47,6544	16,056	476
Parking	0	88,36	48,4524	15,7057	476

Theory Test	40,1	99,75	70,6234	17,8642	475
-------------	------	-------	---------	---------	-----

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 2: Bảng mô tả các biến phân loại (Categorical Variables)

Tên thuộc tính	Giải thích	Ví dụ	Unique values	Phương pháp tiền xử lý
Applicant ID	Mã định danh	AID0001, ..., AID0500	500	-
Gender	Giới tính	Male, Female	2	Label
Age Group	Nhóm tuổi	Young Adult, Middle Age, Teenager	3	Ordinal
Race	Chủng tộc	Black, White, Other	3	One-hot
Training	Loại đào tạo	Advanced, Basic, None	3	Ordinal
Reactions	Tốc độ phản ứng	Slow, Average, Fast	3	Ordinal
Qualified	Đỗ hay trượt	Yes, No	2	Label

(Nguồn: Tác giả tổng hợp, 2025)

```

import pandas as pd
import matplotlib.pyplot as plt

# Đọc dữ liệu từ file csv
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Drivers License Data.csv')

# Lấy các cột phân loại (object type)
categorical_columns = df.select_dtypes(include=['object']).columns

# Loại bỏ cột "Applicant ID" và lọc các cột phân loại có giá trị
categorical_columns = [col for col in categorical_columns if col != 'Applicant ID' and df[col].nunique() > 1]

# Tạo một figure và các subplots để vẽ biểu đồ cho các cột phân loại
num_columns = len(categorical_columns)
num_rows = (num_columns // 2) + (num_columns % 2 > 0) # Tính số hàng cần thiết (3 biểu đồ mỗi hàng)

# Tạo các subplots
fig, axes = plt.subplots(num_rows, 2, figsize=(15, 5 * num_rows))

# Chỉnh sửa các axes để tránh trùng lặp
axes = axes.flatten()

# Vẽ biểu đồ cột cho từng cột phân loại
for i, column in enumerate(categorical_columns):
    ax = axes[i]
    # Lấy số lượng giá trị trong cột
    value_counts = df[column].value_counts()

    # Vẽ biểu đồ cột
    value_counts.plot(kind='bar', ax=ax, color='#FFC1CC')

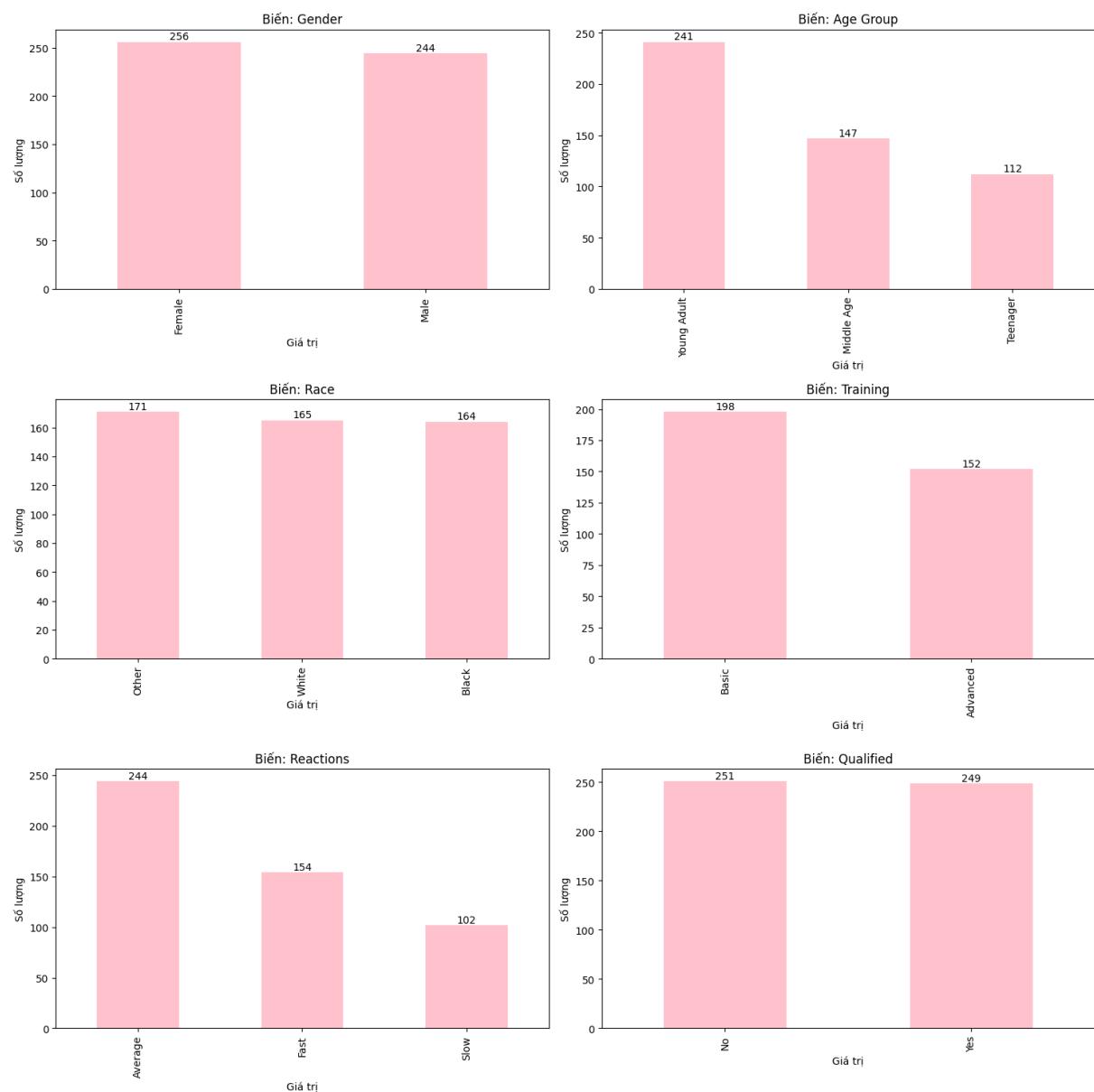
    # Thêm số liệu vào mỗi cột
    for j, v in enumerate(value_counts):
        ax.text(j, v + 0.2, str(v), ha='center', va='bottom', fontsize=10) # Thêm số lượng lên cột

    ax.set_title(f'Biến: {column}')
    ax.set_ylabel('Số lượng')
    ax.set_xlabel('Giá trị')

    # Xóa những axes dư thừa không có biểu đồ
    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

# Điều chỉnh khoảng cách giữa các biểu đồ
plt.tight_layout()
plt.show()

```



Hình 1: Trực quan hóa các biến phân loại (Categorical Variables)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

## II. TIỀN XỬ LÝ DỮ LIỆU

```
# Hiển thị 5 dòng đầu tiên
df.head()
```

	Applicant ID	Gender	Age Group	Race	Training	Signals	Yield	Speed Control	Night Drive	Road Signs	Steer Control	Mirror Usage	Confidence	Parking	Theory Test	Reactions	Qualified
0	AID0001	Male	Young Adult	Other	NaN	38.48	30.29	37.03	33.53	39.61	58.16	53.42	35.32	38.19	70.68	Average	No
1	AID0002	Female	Young Adult	Black	NaN	51.76	19.13	63.05	34.87	19.56	16.48	27.97	22.91	24.23	78.18	Average	No
2	AID0003	Male	Middle Age	Black	NaN	30.21	48.13	43.13	42.43	60.93	20.74	28.86	32.32	44.11	79.60	Fast	Yes
3	AID0004	Male	Young Adult	Other	NaN	34.75	47.28	50.49	42.10	22.52	33.87	48.52	24.90	37.56	57.34	Average	No
4	AID0005	Male	Teenager	Other	Advanced	78.52	83.93	59.79	52.68	67.47	89.24	30.31	43.85	55.91	78.44	Average	Yes

```
# Tóm tắt ngắn gọn về DataFrame  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 500 entries, 0 to 499  
Data columns (total 17 columns):  
 #   Column           Non-Null Count  Dtype     
 ---  --     
 0   Applicant ID    500 non-null    object    
 1   Gender          500 non-null    object    
 2   Age Group       500 non-null    object    
 3   Race            500 non-null    object    
 4   Training         350 non-null    object    
 5   Signals          500 non-null    float64   
 6   Yield            500 non-null    float64   
 7   Speed Control   500 non-null    float64   
 8   Night Drive     500 non-null    float64   
 9   Road Signs      500 non-null    float64   
 10  Steer Control   500 non-null    float64   
 11  Mirror Usage    500 non-null    float64   
 12  Confidence       500 non-null    float64   
 13  Parking          500 non-null    float64   
 14  Theory Test     500 non-null    float64   
 15  Reactions         500 non-null    object    
 16  Qualified        500 non-null    object    
 dtypes: float64(10), object(7)  
memory usage: 66.5+ KB
```

## 2.1. Làm sạch dữ liệu (Data Cleaning)

### 2.1.1. Tìm những giá trị bị trùng lặp

```
int(df.duplicated().sum())
```

0

=> Bộ dữ liệu không có giá trị trùng lặp.

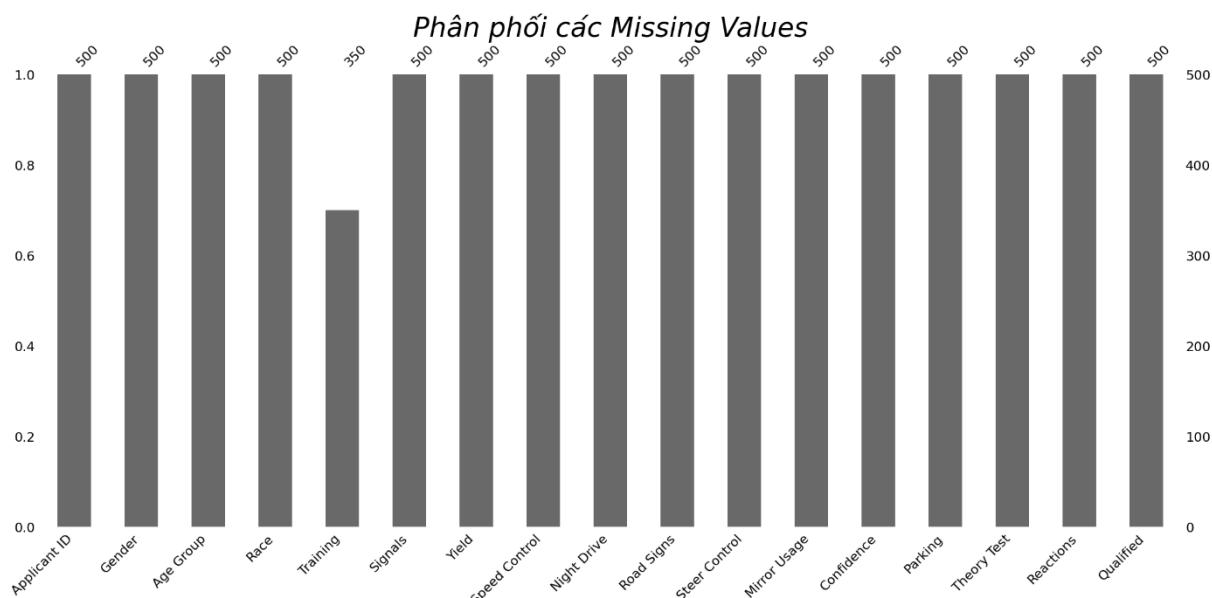
### 2.1.2. Xử lý các giá trị còn thiếu (Missing value)

```
# Missing value
df.isna().sum()
```

	0
Applicant ID	0
Gender	0
Age Group	0
Race	0
Training	150
Signals	0
Yield	0
Speed Control	0
Night Drive	0
Road Signs	0
Steer Control	0
Mirror Usage	0
Confidence	0
Parking	0
Theory Test	0
Reactions	0
Qualified	0

=> Bộ dữ liệu có 150 giá trị bị thiếu ở biến “Training”

```
# Missing value
msno.bar(df)
plt.title('Phân phối các Missing Values', fontsize=33,
fontstyle='oblique');
```



Hình 2: Phân phối các Missing value

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Vì thiếu **150 dòng** trong biến Training trên tổng số **500 dòng** (tức **30% dữ liệu bị thiếu**) là một **mức độ thiếu nghiêm trọng**. Do đó, cách tốt nhất để xử lý giá trị thiếu trong biến phân loại Training, là **gán các giá trị thiếu thành một nhãn mới "Unknown"**. Cách làm này cho phép giữ nguyên toàn bộ dữ liệu, tránh việc loại bỏ dòng, đồng thời cung cấp cho mô hình học máy khả năng phân biệt giữa các nhóm "Basic", "Advanced" và nhóm không xác định "Unknown".

```
# Thay thế các giá trị thiếu (None) trong cột Training bằng chuỗi 'Unknown'
df['Training'].fillna('Unknown', inplace=True)

<ipython-input-19-2173101933>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform:
df['Training'].fillna('Unknown', inplace=True)
```

Kiểm tra lại biến “Training” sau khi gán các giá trị thiếu thành nhãn mới “Unknown”.

```
df['Training'].value_counts()
```

count

### Training

<b>Basic</b>	198
<b>Advanced</b>	152
<b>Unknown</b>	150

**dtype:** int64

#### 2.1.3. Xử lý dữ liệu nhiễu (Noise Data)

##### ❖ Phương pháp tìm giá trị ngoại lai bằng IQR (Interquartile Range):

Phương pháp tìm ngoại lai dựa trên khoảng tứ phân vị (Interquartile Range – IQR) là một kỹ thuật phổ biến trong thống kê mô tả nhằm phát hiện các giá trị bất thường trong tập dữ liệu. Khoảng tứ phân vị được định nghĩa là hiệu số giữa **Q3 (tứ phân vị thứ ba – giá trị tại vị trí 75%)** và **Q1 (tứ phân vị thứ nhất – giá trị tại vị trí 25%)**, tức là:

$$IQR = Q3 - Q1$$

Phạm vi giá trị “bình thường” của dữ liệu được xác định nằm trong khoảng:

$$[Q1 - 1.5 \times IQR; Q3 + 1.5 \times IQR]$$

Bất kỳ quan sát nào có giá trị **nhỏ hơn  $Q1 - 1.5 \times IQR$  hoặc lớn hơn  $Q3 + 1.5 \times IQR$**  đều được xem là **giá trị ngoại lai (outlier)**. Phương pháp này có ưu điểm là không phụ thuộc vào giả định phân phối chuẩn, do đó phù hợp với các dữ liệu có phân phối lệch hoặc chứa nhiễu. Việc phát hiện ngoại lai theo IQR giúp nâng cao chất lượng dữ liệu, loại bỏ các điểm bất thường có thể gây sai lệch trong phân tích hoặc mô hình hóa. Tuy nhiên, quyết định loại bỏ hay giữ lại các giá trị ngoại lai còn phụ thuộc vào ngữ cảnh thực tế và mục tiêu nghiên cứu.

```

import numpy as np
import pandas as pd
import time

# Kiểm tra từng biến
def find_outliers_per_column(df):
    for column in df.select_dtypes(include=[np.number]).columns:
        print(f"\n===== Kiểm tra biến: {column} =====")
        find_outliers(df[column])

def find_outliers(data):
    sorted_data = data.sort_values()
    q_list = []
    for q, p in {"Q1": 25, "Q2": 50, "Q3": 75}.items():
        # Tính Q1, Q2, Q3
        Q = np.percentile(sorted_data, p, interpolation='midpoint')
        q_list.append(Q)
        print("Checking...", q)
        time.sleep(0.5)
        column_name = data.name if hasattr(data, 'name') else 'the data'
        print(f"{q}: {p} percentile of {column_name} is {Q}")
    Q1, Q2, Q3 = q_list
    IQR = Q3 - Q1
    print("IQR (Interquartile Range) =", IQR)

```

```

low_lim = Q1 - 1.5 * IQR
up_lim = Q3 + 1.5 * IQR

print(" ")
print("Checking limits...")
time.sleep(0.5)
print("low_limit:", low_lim)
print("up_limit:", up_lim)

outliers = [x for x in sorted_data if x < low_lim or x > up_lim]

print("Đang kiểm tra outliers...")
time.sleep(1)
print(f"Outliers trong '{data.name}': {outliers}")

# Kiểm tra từng biến trong bộ dữ liệu

find_outliers_per_column(df)

```

Bảng 3: Kết quả các giá trị ngoại lai của các biến

Tên thuộc tính	Q1	Q2	Q3	Giới hạn dưới	Giới hạn trên	Giá trị ngoại lai
Signals	38,08	48,42	59,42	6,07	91,43	0,77; 2,53; 2,75; 95,61
Yield	36,91	48,95	59,05	3,71	92,25	0,0; 0,0; 1,92; 2,63; 94,41
Speed Control	37,29	49,41	58,45	5,54	90,19	0,0; 0,0; 0,0
Night Drive	37,32	48,16	58,80	5,09	91,02	1,74
Road Signs	36,54	49,31	58,79	3,16	92,16	0,0; 0,0
Steer Control	37,31	48,22	57,54	6,96	87,89	0,0; 89,24
Mirror Usage	38,35	48,36	58,81	7,66	89,50	0,0; 1,7; 4,46; 6,49; 6,59; 7,16; 90,77
Confidence	36,98	48,25	59,13	3,76	92,34	-
Parking	37,44	49,40	59,29	4,67	92,05	0,0
Theory Test	54,85	69,85	86,70	7,07	134,48	-

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```

import seaborn as sns
import matplotlib.pyplot as plt

# Lọc các cột dạng số
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Thiết lập kích thước và kiểu nền
plt.figure(figsize=(16, 8))
sns.set_style("whitegrid")

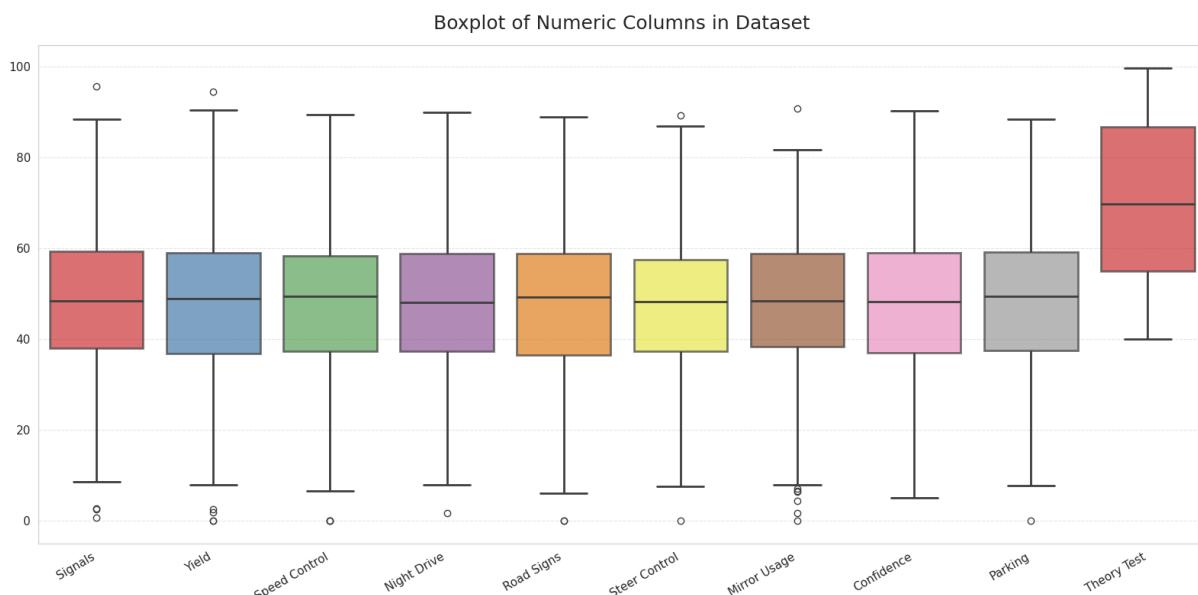
# Vẽ boxplot với màu sắc phân biệt theo cột
sns.boxplot(data=df[numeric_columns],
             palette="Set1",    # Bộ màu đẹp, dễ nhìn
             linewidth=2,
             boxprops=dict(alpha=0.7)) # độ mờ

# Tùy chỉnh tiêu đề và trục
plt.title('Boxplot of Numeric Columns in Dataset', fontsize=18, pad=15)
plt.xticks(rotation=30, ha='right', fontsize=11)
plt.yticks(fontsize=11)
plt.grid(axis='y', linestyle='--', alpha=0.5)

# Tăng khoảng cách giữa các cột để tránh chồng lấn
plt.subplots_adjust(bottom=0.2)

plt.tight_layout()
plt.show()

```



Hình 3: Biểu đồ Boxplot để phát hiện outliers của các biến số (Numeric Variables)

(Nguồn: Xử lý số liệu bởi Google Colab)

❖ **Nhận xét:**

**Các outliers thấp (gần 0):** Xuất hiện nhiều trong các kỹ năng thực hành như Speed Control, Road Signs, Mirror Usage. Điều này phản ánh những học viên thi rớt phần đó hoặc không đủ điểm.

**Các outliers cao (>90):** Ví dụ: Signals = 95.61, Steer Control = 89.24, Mirror Usage = 90.77. Đây là **những người có kỹ năng tốt vượt trội**.

**Phân bố outliers hợp lý với ngữ cảnh thực tế:** Trong thi hoặc đánh giá năng lực lái xe, có thể có học viên xuất sắc hoặc yếu, nên không bất thường nếu điểm trải dài, không tập trung.

❖ **Xử lý các giá trị outliers**

- **Không loại bỏ các giá trị outliers này**, vì chúng phản ánh sự đa dạng trong dữ liệu. Phản ánh chính xác thực tế.
- **Loại bỏ các giá trị outliers:** Thử xóa bỏ các outliers để cải thiện hiệu suất các thuật toán và tránh sai lệch kết quả. Từ đó so sánh, phân tích hiệu quả các thuật toán so với việc không loại bỏ outliers.

```
# Bước 1: Tạo limits
limits = find_outliers_per_column(df)

# Bước 2: Định nghĩa hàm xóa outliers
def remove_outliers(df, limits):
    df_cleaned = df.copy()
    for col, (low, up) in limits.items():
        df_cleaned = df_cleaned[(df_cleaned[col] >= low) & (df_cleaned[col] <= up)]
    return df_cleaned

# Bước 3: Gọi xóa
df_cleaned = remove_outliers(df, limits)

# Bước 4: Lưu kết quả
df_cleaned.to_csv("DL_da_xoa_outliers.csv", index=False)
```

## 2.2. Chuyển đổi dữ liệu (Data Transformation)

### 2.2.1. Thống kê các biến phân loại (categorical) và biến số (numeric)

```
# Hàm để tóm tắt các thuộc tính numeric và categorical
def summarize_social_data(data):
    # Xác định các thuộc tính numeric
    numeric_features = data.select_dtypes(include=[np.number]).columns.tolist()
    # Xác định các thuộc tính categorical
    categorical_features = data.select_dtypes(exclude=[np.number]).columns.tolist()
    # In số lượng và tên các cột số học
    print(f"Có {len(numeric_features)} thuộc tính numeric:")
    for i, feature in enumerate(numeric_features, 1):
        print(f"{i}. {feature}")
    # In số lượng và tên các cột categorical
    print(f"Có {len(categorical_features)} thuộc tính categorical:")
    for i, feature in enumerate(categorical_features, 1):
        print(f"{i}. {feature}")
# Gọi hàm tóm tắt
summarize_social_data(df)
```

Có 10 thuộc tính numeric:

1. Signals
2. Yield
3. Speed Control
4. Night Drive
5. Road Signs
6. Steer Control
7. Mirror Usage
8. Confidence
9. Parking
10. Theory Test

Có 7 thuộc tính categorical:

1. Applicant ID
2. Gender
3. Age Group
4. Race
5. Training
6. Reactions
7. Qualified

### 2.2.2. Mã hóa biến phân loại (Encoding Categorical Variables)

Trong bước này, chúng ta thực hiện mã hóa các biến phân loại trong bộ dữ liệu để chuẩn bị cho các bước phân tích và xây dựng mô hình tiếp theo.

a. Mã hóa bằng phương pháp Label Encoding

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Khởi tạo encoder
le_gender = LabelEncoder()
le_qualified = LabelEncoder()

# Mã hóa
df['Gender'] = le_gender.fit_transform(df['Gender'])
df['Qualified'] = le_qualified.fit_transform(df['Qualified'])

# Lưu vào file CSV
df.to_csv("Data_Processing.csv", index=False)
```

Các biến phân loại nhị phân, chẳng hạn như *Gender* (*Giới tính*); *Qualified* (*Đỗ hay trượt*) được mã hóa bằng phương pháp **Label Encoding**. Phương pháp này chuyển đổi các giá trị phân loại như "Female" và "Male" hay "no" và "yes" thành 0 và 1 tương ứng. Cụ thể:

- **Gender:** "Female" → 0; "Male" → 1;
- **Qualified:** "No" → 0; "Yes" → 1.

Việc mã hóa các biến nhị phân giúp chuyển đổi dữ liệu từ dạng chuỗi ký tự thành dạng số nguyên, cho phép các mô hình học máy xử lý dữ liệu dễ dàng và hiệu quả hơn.

b. Mã hóa bằng phương pháp Ordinal Encoding

```

import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

# Đọc dữ liệu từ file CSV
df = pd.read_csv("Data_Processing.csv")

# Xác định thứ tự của các biến ordinal
age_order = ['Teenager', 'Young Adult', 'Middle Age']
training_order = ['Unknown', 'Basic', 'Advanced']
reactions_order = ['Slow', 'Average', 'Fast']

# Mã hóa bằng OrdinalEncoder
encoder = OrdinalEncoder(categories=[age_order, training_order, reactions_order])
ordinal_cols = ['Age Group', 'Training', 'Reactions']

# Fit và transform
encoded = encoder.fit_transform(df[ordinal_cols]).astype(int)

# Ghi đè lại các cột gốc
df[ordinal_cols] = encoded

# Ghi dữ liệu đã mã hóa trở lại file
df.to_csv("Data_Processing.csv", index=False)

```

Sau khi áp dụng Ordinal Encoding, các giá trị trong biến phân loại có thứ tự (ordinal variable) đã được mã hóa thành các số nguyên phản ánh đúng mối quan hệ thứ bậc giữa các giá trị. Mỗi giá trị danh mục được gán một số tăng dần theo cấp bậc, giúp mô hình học máy hiểu được rằng có sự sắp xếp có ý nghĩa giữa các mức. Cụ thể:

- **Age Group:** “Teenager” → 0; “Young Adult” → 1; “Middle Audult” → 2
- **Training:** “Unknown” → 0; “Basic” → 1; “Advanced” → 2
- **Reactions:** “Slow” → 0; “Average” → 1; “Fast” → 2

Việc mã hóa các biến có thứ bậc bằng phương pháp này rất hữu ích cho các mô hình tuyến tính và các mô hình cần thông tin về thứ tự.

### c. Mã hóa bằng phương pháp One-hot Encoding

```
import pandas as pd

# Đọc dữ liệu từ file CSV
df = pd.read_csv("Data_Processing.csv")

# One-hot encoding cho biến 'Race'
df_encoded = pd.get_dummies(df, columns=['Race'], prefix='Race', dtype=int)

# Ghi đè file CSV
df_encoded.to_csv("Data_Processing.csv", index=False)
```

Biến phân loại Race đã được mã hóa bằng phương pháp One-Hot Encoding nhằm chuyển đổi các giá trị dạng chuỗi thành dạng số nhị phân để xử lý cho các mô hình học máy.

Mỗi giá trị duy nhất trong biến Race được chuyển thành một cột mới với giá trị 0 hoặc 1, biểu thị sự có mặt hoặc không có mặt của từng chủng tộc tương ứng trong mỗi bản ghi.

Cụ thể, biến Race có các giá trị như "White", "Black" và "Other", thì sau khi mã hóa, tập dữ liệu sẽ xuất hiện thêm các cột như Race\_White, Race\_Black, và Race\_Other.

Phương pháp này giúp tránh việc áp đặt thứ tự sai lệch cho các giá trị phân loại không có tính thứ bậc, đồng thời duy trì tính toàn vẹn của thông tin đầu vào cho các thuật toán phân tích và mô hình dự đoán.

## 2.3. Chuẩn hóa dữ liệu (Normalization)

### 2.3.1. Đổi với dữ liệu chưa loại bỏ outliers

#### a. Chuẩn hóa bằng phương pháp Min-Max Scaler

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Đọc dữ liệu từ file
df = pd.read_csv("Data_Processing.csv")

# Các cột cần chuẩn hóa
cols_to_scale = ['Confidence', 'Theory Test']

# Khởi tạo MinMaxScaler
scaler = MinMaxScaler()

# Chuẩn hóa và ghi đè lại các cột gốc
df[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

# Lưu lại vào file
df.to_csv("Data_Processing.csv", index=False)
```

Sau khi áp dụng phương pháp **Min-Max Scaling** cho hai biến định lượng là Confidence và Theory Test, toàn bộ giá trị của hai cột này đã được **chuyển đổi về khoảng [0, 1]**, giúp đảm bảo rằng các biến này không còn bị ảnh hưởng bởi đơn vị đo lường ban đầu hay độ lệch về thang đo giữa các đặc trưng mà không làm thay đổi phân phối dữ liệu.

#### b. Chuẩn hóa bằng phương pháp RobustScaler

```

import pandas as pd
from sklearn.preprocessing import RobustScaler

# Đọc dữ liệu
df = pd.read_csv("Data_Processing.csv")

# Danh sách các biến cần chuẩn hóa bằng RobustScaler
cols_to_scale = [
    'Signals', 'Yield', 'Speed Control', 'Night Drive', 'Road Signs',
    'Steer Control', 'Mirror Usage', 'Confidence', 'Parking', 'Theory Test'
]

# Khởi tạo RobustScaler
scaler = RobustScaler()

# Áp dụng chuẩn hóa và ghi đè lại
df[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

# Lưu lại vào file
df.to_csv("Data_Processing.csv", index=False)

```

Trong quá trình tiền xử lý dữ liệu, phương pháp **RobustScaler** đã được áp dụng để chuẩn hóa các biến liên tục như *Signals*, *Yield*, *Speed Control*, *Night Drive*, *Road Signs*, *Steer Control*, *Mirror Usage*, *Confidence*, *Parking* và *Theory Test*. Khác với các phương pháp chuẩn hóa truyền thống như MinMaxScaler hay StandardScaler, RobustScaler sử dụng **trung vị (median)** và **khoảng tứ phân vị (IQR)** để loại bỏ ảnh hưởng của các giá trị ngoại lai. Điều này đặc biệt phù hợp với bộ dữ liệu hiện tại vì các thuộc tính có chứa nhiều giá trị ngoại lệ.

Phương pháp này giúp cải thiện hiệu quả và độ ổn định của các thuật toán phân tích và mô hình học máy về sau.

**Ý nghĩa của các giá trị sau khi chuẩn hóa bằng phương pháp RobustScaler:**

- **Giá trị âm:** Điểm dữ liệu thấp hơn trung vị (median)
- **Giá trị dương:** Điểm dữ liệu cao hơn trung vị
- **0:** Chính là giá trị trung vị (median)

### 2.3.2. Đối với dữ liệu đã loại bỏ outliers

Khác với dữ liệu chưa loại bỏ outliers phải chú ý sử dụng phương pháp chuẩn hóa ít nhạy cảm với giá trị outliers. Với dữ liệu đã loại bỏ outliers có thể chỉ chuẩn hóa bằng phương pháp Min-Max Scaler để chuẩn hóa về khoảng [0;1].

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Đọc dữ liệu từ file
df = pd.read_csv("Tiến xu ly dl da xoa outliers.csv")

# Các cột cần chuẩn hóa
cols_to_scale = ['Confidence', 'Theory Test', 'Signals', 'Yield', 'Speed Control', 'Night Drive', 'Road Signs', 'Mirror Usage', 'Parking', 'Steer Control']

# Khởi tạo MinMaxScaler
scaler = MinMaxScaler()

# Chuẩn hóa và ghi đè lại các cột gốc
df[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

# Lưu lại vào file
df.to_csv("Tiến xu ly dl da xoa outliers.csv", index=False)
```

## IV. DỮ LIỆU SAU KHI TIỀN XỬ LÝ

### 4.1. Dữ liệu trước khi loại bỏ outliers

Bảng 4: Tóm tắt dữ liệu sau khi tiền xử lý trước khi loại bỏ các outliers

Thuộc tính	GTNN	GTLN	GTTB	Độ lệch chuẩn	Số giá trị duy nhất	Phương pháp tiền xử lý
Gender	0	1	0,488	0,500357	2	Label Encoding
Age Group	0	2	1,07	0,717027	3	Ordinal Encoding
Training	0	2	1,004	0,777942	3	Ordinal Encoding
Signals	-2,2410	2,2199	0,00519567	0,766459	485	RobustScaler

Yield	-2,218	2,06051	-0,0528499	0,728031	482	RobustScaler
Speed Control	-2,3420	1,89737	-0,0698393	0,739571	477	RobustScaler
Night Drive	-2,1686	1,95141	-0,0131857	0,714912	473	RobustScaler
Road Signs	-2,2186	1,77998	-0,0682367	0,72184	473	RobustScaler
Steer Control	-2,3900	2,03321	-0,0391831	0,772243	475	RobustScaler
Mirror Usage	-2,3659	2,07485	-0,0099031	0,756354	473	RobustScaler
Confidence	0	1	0,499465	0,188539	476	Min–Max Scaler [0;1]
Parking	-2,2658	1,78695	-0,0434647	0,720362	476	RobustScaler
Theory Test	0	1	0,511708	0,299483	475	Min – Max Scaler [0;1]
Reactions	0	2	1,104	0,708653	3	Ordinal Encoding
Qualified	0	1	0,498	0,500497	2	Label Encoding
Race _ Black	0	1	0,328	0,469955	2	One-hot Encoding
Race _ Other	0	1	0,342	0,474855	2	One-hot Encoding
Race _ White	0	1	0,33	0,470684	2	One-hot Encoding

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

## 4.2. Dữ liệu sau khi loại bỏ outliers

Bảng 5: Tóm tắt dữ liệu sau khi tiền xử lý đã loại bỏ các outliers

Thuộc tính	GTNN	GTLN	GTTB	Độ lệch chuẩn	Số giá trị duy nhất	Phương pháp tiền xử lý
Gender	0	1	0,482255	0,500207	2	Label Encoding
Age Group	0	2	1,09812	0,709158	3	Ordinal Encoding
Training	0	2	1,03132	0,766083	3	Ordinal Encoding
Signals	0	1	0,505739	0,196399	464	Min–Max Scaler [0;1]
Yield	0	1	0,492468	0,182606	462	Min–Max Scaler [0;1]
Speed Control	0	1	0,505348	0,181957	459	Min–Max Scaler [0;1]
Night Drive	0	1	0,494373	0,181699	454	Min–Max Scaler [0;1]
Road Signs	0	1	0,511864	0,187199	456	Min–Max Scaler [0;1]
Steer Control	0	1	0,489196	0,193606	455	Min–Max Scaler [0;1]
Mirror Usage	0	1	0,556369	0,196767	454	Min–Max Scaler [0;1]

Confidence	0	1	0,506324	0,185811	457	Min–Max Scaler [0;1]
Parking	0	1	0,510582	0,188689	456	Min–Max Scaler [0;1]
Theory Test	0	1	0,508266	0,29969	455	Min–Max Scaler [0;1]
Reactions	0	2	1,10021	0,707388	3	Ordinal Encoding
Qualified	0	1	0,511482	0,500391	2	Label Encoding
Race_Blk	0	1	0,338205	0,473593	2	One-hot Encoding
Race_Other	0	1	0,336117	0,472873	2	One-hot Encoding
Race_White	0	1	0,325678	0,469118	2	One-hot Encoding

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

## V. CÁC THUẬT TOÁN HỌC TẬP CÓ GIÁM SÁT (SUPERVISED LEARNING)

### PHẦN A: THƯ VIỆN LAZYPREDICT

**LazyPredict** là một thư viện trong Python hỗ trợ nhanh chóng thử nghiệm nhiều mô hình học máy (machine learning) cùng lúc mà không cần phải viết lại từng dòng code cho từng thuật toán. Thư viện này đặc biệt hữu ích trong giai đoạn khám phá ban đầu, giúp người dùng so sánh hiệu suất của các mô hình khác nhau một cách tự động và tiện lợi.

Trong bộ dữ liệu “**Driver's License Test Scores Data**” được nghiên cứu, biến mục tiêu (target variable) được xác định là cột **Qualified**, biểu thị kết quả của người tham gia trong kỳ kiểm tra lái xe (đạt hoặc không đạt). Đây là một biến phân loại nhị phân, do đó bài toán được định dạng là **bài toán phân loại**. Các biến đầu vào (features) bao gồm các yếu tố liên quan đến hành vi và kỹ năng lái xe như: Signals, Speed Control, Mirror Usage, Confidence, Theory Test, v.v.

### A.1. Đối với dữ liệu trước khi loại bỏ outliers

```
import pandas as pd
from sklearn.model_selection import train_test_split
from lazypredict.Supervised import LazyClassifier

# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Tập đặc trưng và mục tiêu
X = df.drop(columns=['Qualified', 'Applicant ID'])
y = df['Qualified']

# Tách tập huấn luyện và kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Khởi tạo và huấn luyện LazyClassifier
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)

# In kết quả
print("===== Tổng kết mô hình phân loại =====")
print(models)

# Lưu kết quả ra file CSV
output_path = '/content/drive/MyDrive/khai pha du lieu/Hieu_nang_mo_hinh_phan_loai_chua_loai_bo_outliers.csv'
models.to_csv(output_path)
```

Bảng 6: Kết quả sau khi chạy thư viện LazyPredict của dữ liệu trước khi loại bỏ outliers

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
AdaBoostClassifier	0,83	0,83	0,83	0,83	0,32
NuSVC	0,83	0,83	0,83	0,83	0,09
XGBClassifier	0,82	0,82	0,82	0,82	1,57
SVC	0,82	0,82	0,82	0,82	0,03
RandomForestClassifier	0,82	0,82	0,82	0,82	0,66

RidgeClassifierCV	0,81	0,81	0,81	0,81	0,05
PassiveAggressiveClassifier	0,81	0,81	0,81	0,81	0,04
NearestCentroid	0,80	0,80	0,80	0,80	0,05
LinearDiscriminantAnalysis	0,80	0,80	0,80	0,80	0,08
GaussianNB	0,80	0,80	0,80	0,80	0,04
RidgeClassifier	0,80	0,80	0,80	0,80	0,03
ExtraTreesClassifier	0,80	0,80	0,80	0,80	0,40
LinearSVC	0,79	0,79	0,79	0,79	0,03
BernoulliNB	0,79	0,79	0,79	0,79	0,08
CalibratedClassifierCV	0,79	0,79	0,79	0,79	0,36
DecisionTreeClassifier	0,78	0,78	0,78	0,78	0,05
LogisticRegression	0,78	0,78	0,78	0,78	0,05
SGDClassifier	0,78	0,78	0,78	0,78	0,03
LGBMClassifier	0,78	0,78	0,78	0,78	0,21
Perceptron	0,78	0,78	0,78	0,78	0,03
KNeighborsClassifier	0,77	0,77	0,77	0,77	0,07
BaggingClassifier	0,77	0,77	0,77	0,77	0,16
ExtraTreeClassifier	0,73	0,73	0,73	0,73	0,03
QuadraticDiscriminantAnalysis	0,73	0,73	0,73	0,73	0,07
LabelPropagation	0,72	0,72	0,72	0,72	0,11
LabelSpreading	0,72	0,72	0,72	0,72	0,09
DummyClassifier	0,48	0,50	0,50	0,31	0,03

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

❖ **Mô tả về các chỉ số:**

**Accuracy (Độ chính xác tổng thể):** Là tỷ lệ phần trăm tổng số dự đoán đúng trên toàn bộ dữ liệu. Là chỉ số dễ hiểu, phổ biến. Tuy nhiên, dễ bị sai sót trong trường hợp **dữ liệu mất cân bằng** (khi một lớp chiếm đa số).

**Balanced Accuracy (Độ chính xác cân bằng):** Trung bình của **Recall** trên từng lớp. Giúp **đánh giá công bằng hơn** trong dữ liệu mất cân bằng (không bị thiên vị lớp đông). Rất quan trọng trong các bài toán như chẩn đoán y tế, phát hiện gian lận.

**ROC AUC (Area Under the ROC Curve):** Đo khả năng phân biệt giữa các lớp của mô hình.

- ROC là đường cong biểu diễn mối quan hệ giữa **True Positive Rate** và **False Positive Rate**.
- AUC (Area Under Curve) là **diện tích dưới đường ROC**, nằm trong khoảng [0, 1]. Với AUC càng tiến về 1 thì phân biệt càng hoàn hảo.

## A.2. Đối với dữ liệu sau khi loại bỏ outliers

```
import pandas as pd
from sklearn.model_selection import train_test_split
from lazypredict.Supervised import LazyClassifier

# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Tập đặc trưng và mục tiêu
X = df.drop(columns=['Qualified', 'Applicant ID'])
y = df['Qualified']

# Tách tập huấn luyện và kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Khởi tạo và huấn luyện LazyClassifier
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)

# In kết quả
print("===== Tổng kết mô hình phân loại =====")
print(models)

# Lưu kết quả ra file CSV
output_path = '/content/drive/MyDrive/khai pha du lieu/Hieu_nang_mo_hinh_phan_loai_da_loai_bo_outliers.csv'
models.to_csv(output_path)
```

Bảng 7: Kết quả sau khi chạy thư viện LazyPredict của dữ liệu sau khi loại bỏ outliers

Mô hình	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
KNeighborsClassifier	0,84	0,86	0,86	0,84	0,05
ExtraTreesClassifier	0,84	0,86	0,86	0,84	0,30
AdaBoostClassifier	0,84	0,86	0,86	0,84	0,34
RandomForestClassifier	0,83	0,85	0,85	0,83	0,85
Perceptron	0,83	0,84	0,84	0,83	0,03
LabelPropagation	0,82	0,84	0,84	0,82	0,07
LabelSpreading	0,82	0,84	0,84	0,82	0,14
NuSVC	0,82	0,84	0,84	0,82	0,06
RidgeClassifierCV	0,82	0,84	0,84	0,82	0,07
SVC	0,82	0,84	0,84	0,82	0,09
CalibratedClassifierCV	0,82	0,83	0,83	0,82	0,18
LGBMClassifier	0,82	0,83	0,83	0,82	0,12
LinearSVC	0,82	0,83	0,83	0,82	0,07
DecisionTreeClassifier	0,82	0,83	0,83	0,82	0,04
XGBClassifier	0,81	0,82	0,82	0,81	2,51
LinearDiscriminantAnalysis	0,81	0,82	0,82	0,81	0,05
RidgeClassifier	0,81	0,82	0,82	0,81	0,07
GaussianNB	0,81	0,82	0,82	0,81	0,03
NearestCentroid	0,80	0,81	0,81	0,80	0,06
SGDClassifier	0,79	0,81	0,81	0,79	0,06
LogisticRegression	0,79	0,80	0,80	0,79	0,09

BaggingClassifier	0,79	0,79	0,79	0,79	0,13
BernoulliNB	0,77	0,77	0,77	0,77	0,03
QuadraticDiscriminantAnalysis	0,74	0,76	0,76	0,74	0,04
ExtraTreeClassifier	0,74	0,74	0,74	0,74	0,03
PassiveAggressiveClassifier	0,68	0,67	0,67	0,68	0,03
DummyClassifier	0,42	0,50	0,50	0,25	0,03

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

### A.3. So sánh kết quả giữa trước và sau khi loại bỏ outliers

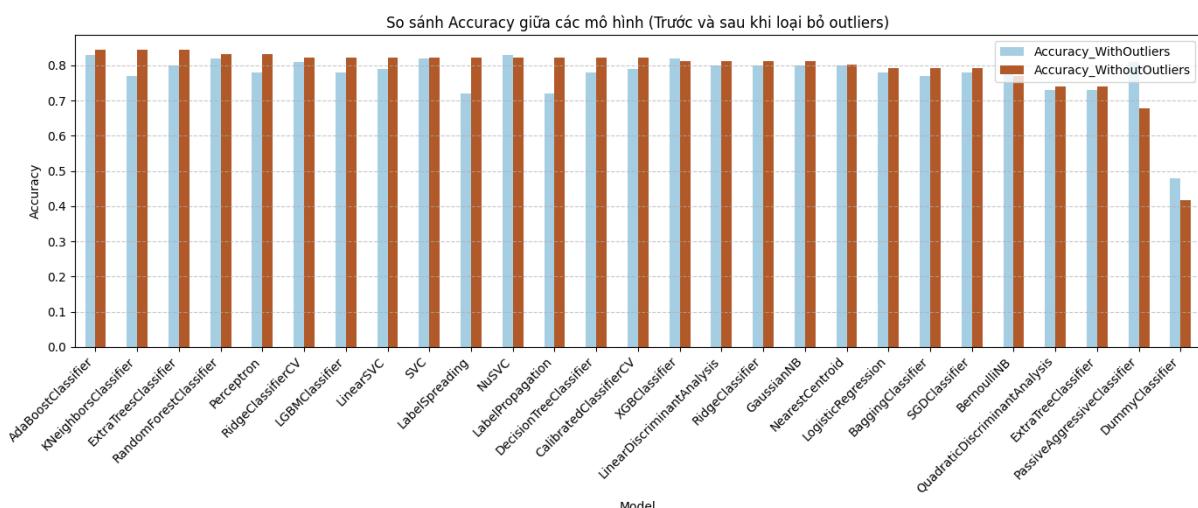
#### A.3.1. Trực quan hóa kết quả trước và sau khi loại bỏ outliers

```
# Đọc file kết quả
df_with = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Hieu_nang_mo_hinh_phan_loai_chua_loai_bo_outliers.csv')
df_without = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Hieu_nang_mo_hinh_phan_loai_da_loai_bo_outliers.csv')

# Làm sạch cột tên model nếu cần
df_with['Model'] = df_with['Model'].str.strip()
df_without['Model'] = df_without['Model'].str.strip()

# Gộp 2 bảng theo model
compare = pd.merge(df_with[['Model', 'Accuracy']],
                    df_without[['Model', 'Accuracy']],
                    on='Model',
                    suffixes=('_WithOutliers', '_WithoutOutliers'))

# Trực quan hóa
compare.set_index('Model').sort_values(by='Accuracy_WithoutOutliers', ascending=False).plot(
    kind='bar',
    figsize=(14, 6),
    colormap='Paired'
)
plt.title('So sánh Accuracy giữa các mô hình (Trước và sau khi loại bỏ outliers)')
plt.ylabel('Accuracy')
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Hình 4: So sánh Accuracy giữa các mô hình (Trước và sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### A.3.2. Mức độ thay đổi Accuracy của các mô hình

##### ❖ Các mô hình cải thiện rõ rệt sau khi loại bỏ outliers:

Bảng 8: Các mô hình cải thiện rõ rệt sau khi loại bỏ outliers

Mô hình	Accuracy (Trước)	Accuracy (Sau)	Chênh lệch	Nhận xét
<b>KNeighborsClassifier</b>	0,77	0,84	+0,07	Cải thiện rất mạnh, do KNN rất nhạy cảm với outliers
<b>ExtraTreesClassifier</b>	0,80	0,84	+0,04	Mô hình dựa trên cây, vẫn được hưởng lợi từ dữ liệu sạch hơn
<b>Perceptron</b>	0,78	0,83	+0,05	Là mô hình tuyến tính, nên nhạy với điểm nhiễu

<b>LabelPropagation</b>	0,72	0,82	+0,10	Cải thiện lớn do mô hình bán giám sát dễ bị lệch nếu có outliers
<b>LabelSpreading</b>	0,72	0,82	+0,10	Tương tự LabelPropagation

(Nguồn: Tác giả tổng hợp, 2025)

=> Các mô hình **nhạy với outliers** như KNN, Perceptron, mô hình tuyến tính hoặc bán giám sát đều **cải thiện đáng kể** khi loại bỏ điểm ngoại lai.

#### ❖ Các mô hình duy trì hiệu suất cao ổn định

Bảng 9: Các mô hình duy trì hiệu suất cao ổn định

Mô hình	Accuracy (Trước)	Accuracy (Sau)	Chênh lệch	Nhận xét
<b>NuSVC</b>	0,83	0,82	-0,01	Ôn định, giám không đáng kể
<b>XGBClassifier</b>	0,82	0,81	-0,01	Giảm nhẹ, có thể do mất đi vài điểm mang giá trị phân biệt
<b>SGDClassifier</b>	0,78	0,79	+0,01	Ôn định
<b>AdaBoostClassifier</b>	0,83	0,84	+0,01	Hiệu suất cao và ổn định

(Nguồn: Tác giả tổng hợp, 2025)

=> Các **mô hình ổn định** như AdaBoost, NuSVC, ... vẫn giữ vững hiệu suất, dao động nhẹ. Những mô hình này cho thấy sự khả thi trong ứng dụng thực tiễn, đặc biệt trong bối cảnh dữ liệu thực thường chứa nhiều và bất thường.

### ❖ Mô hình có Accuracy suy giảm

Bảng 10: Mô hình có Accuracy suy giảm

Mô hình	Accuracy (Trước)	Accuracy (Sau)	Chênh lệch	Nhận xét
PassiveAggressive Classifier	0,81	0,68	-0,13	Giảm mạnh vì <b>nhạy cảm cao của mô hình Passive-Aggressive đối với cấu trúc dữ liệu và đặc biệt là phân bố ranh giới</b>
DummyClassifier	0,48	0,42	-0,06	Vai trò là chỉ làm chuẩn đối chiếu, sự giảm Accuracy này <b>không có giá trị chẩn đoán mô hình</b> , mà chỉ phản ánh thay đổi nhẹ trong phân bố nhãn.

(Nguồn: Tác giả tổng hợp, 2025)

=> Điều này cho thấy không phải mô hình nào cũng phù hợp với việc loại bỏ outliers, đặc biệt là các mô hình tuyến tính có tính chất học cứng như PassiveAggressive.

Từ những phân tích trên, có thể thấy việc loại bỏ outliers nhìn chung **mang lại hiệu quả tích cực** đối với Accuracy của các mô hình học máy, đặc biệt là các mô hình phi tham số như KNeighbors hoặc mô hình tổ hợp như ExtraTrees, AdaBoost. Tuy nhiên, một số mô hình có thể bị ảnh hưởng tiêu cực, do đó cần **cân nhắc lựa chọn mô hình phù hợp với đặc trưng dữ liệu** sau khi xử lý.

### A.4. Lựa chọn thuật toán để phân tích

Bảng 11: Lựa chọn ba thuật toán để phân tích

Mô hình	Accuracy	Tăng Accuracy	Đặc điểm nổi bật
<b>RandomForestClassifier</b>	0.83	+0.01	Linh hoạt, ít bị overfitting, giải thích dễ bằng quan trọng biến.
<b>ExtraTreesClassifier</b>	0.84	+0.04	Ôn định, mạnh mẽ, học tốt sau khi loại nhiễu
<b>AdaBoostClassifier</b>	0.84	+0.01	Hiệu suất cao, duy trì tốt sau xử lý dữ liệu

(Nguồn: Tác giả tổng hợp, 2025)

#### A.4.1. Thuật toán **RandomForestClassifier**

##### a. Dữ liệu trước khi loại bỏ outliers

```

# Đọc dữ liệu từ file csv
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Xử lý dữ liệu
X = df.drop(['Qualified', 'Applicant ID'], axis=1)
y = df['Qualified']

# Tách train-test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình Random Forest
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_score = model.predict_proba(X_test)[:, 1] # Xác suất thuộc lớp 1

# === 0. Đánh giá chi tiết ===
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("== Evaluation Metrics ==")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))

```

```

# === 1. Confusion Matrix ===
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Random Forest")
plt.show()

# === 2. ROC Curve ===
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest')
plt.legend()
plt.grid()
plt.show()

# === 3. Precision-Recall Curve ===
precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_score)
avg_precision = average_precision_score(y_test, y_score)

```

```
plt.figure()
plt.plot(recall_curve, precision_curve, lw=2, color='green', label=f'AP = {avg_precision:.2f}')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - Random Forest')
plt.legend()
plt.grid()
plt.show()

# === 4. Feature Importance ===
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)

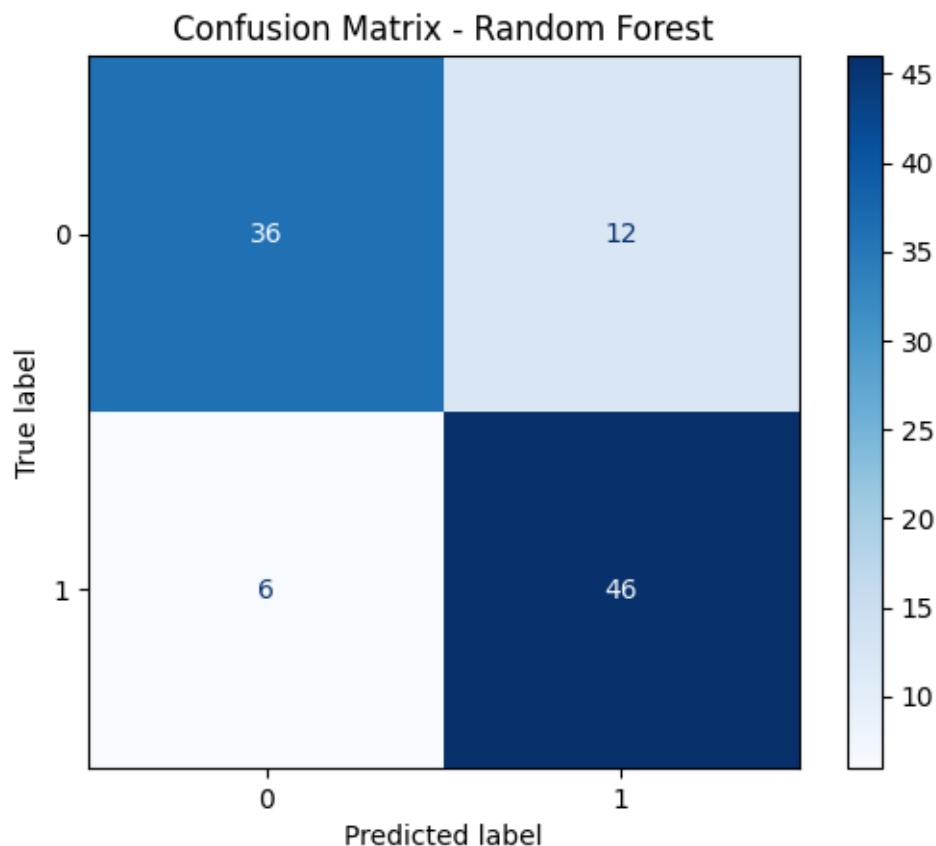
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df.head(15), palette='viridis')
plt.title('Top 15 Feature Importances - Random Forest')
plt.tight_layout()
plt.show()
```

==== Evaluation Metrics ===

Accuracy : 0.8200  
Precision: 0.7931  
Recall : 0.8846  
F1 Score : 0.8364

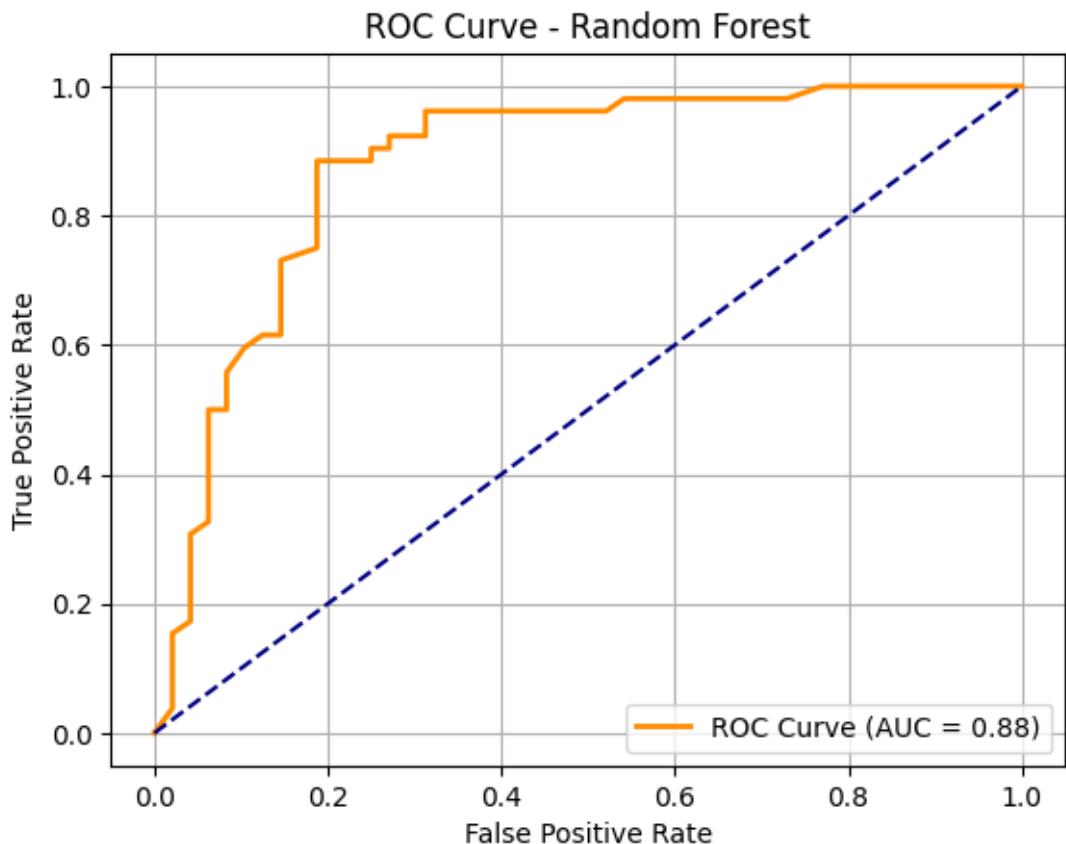
Classification Report:

	precision	recall	f1-score	support
0	0.86	0.75	0.80	48
1	0.79	0.88	0.84	52
accuracy			0.82	100
macro avg	0.83	0.82	0.82	100
weighted avg	0.82	0.82	0.82	100



Hình 5: Confusion Matrix – RandomForestClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 6: ROC Curve – RandomForestClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 12: Nhận xét hiệu quả của mô hình RandomForestClassifier trước khi loại bỏ outliers

Chỉ số	Giá trị	Nhận xét
<b>Accuracy</b>	0.8200	Độ chính xác tổng thể cao, dự đoán đúng 82% tổng số mẫu.
<b>Precision</b>	0.7931	Trong số dự đoán dương tính, có 79.31% là đúng. Mức khá tốt.
<b>Recall</b>	0.8846	Rất cao (88.46%), phản ánh mô hình phát hiện tốt các trường hợp dương tính.
<b>F1 Score</b>	0.8364	Cân bằng tốt giữa precision và recall – độ ổn định cao.

<b>ROC AUC</b>	0.88	Diện tích dưới đường cong ROC lớn, phân biệt tốt giữa hai lớp.
<b>Confusion Matrix</b>	TP = 46, FP = 12, FN = 6, TN = 36	Số lượng <b>False Positive</b> (12) và <b>False Negative</b> (6) còn có thể giảm.

(Ng nguồn: Tác giả tổng hợp, 2025)

=> Precision và Recall đều cao. Mô hình có xu hướng ưu tiên phát hiện dương tính hơn là hạn chế nhầm lẫn. Ứng dụng tốt cho bài toán ưu tiên không bỏ sót đối tượng quan trọng (ví dụ: người đủ điều kiện).

### b. Dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu từ file csv
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Xử lý dữ liệu
X = df.drop(['Qualified', 'Applicant ID'], axis=1)
y = df['Qualified']

# Tách train-test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình Random Forest
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_score = model.predict_proba(X_test)[:, 1] # Xác suất thuộc lớp 1

# === 0. Đánh giá chi tiết ===
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("== Evaluation Metrics ==")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
# === 1. Confusion Matrix ===
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Random Forest")
plt.show()

# === 2. ROC Curve ===
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest')
plt.legend()
plt.grid()
plt.show()

# === 3. Precision-Recall Curve ===
precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_score)
avg_precision = average_precision_score(y_test, y_score)

plt.figure()
plt.plot(recall_curve, precision_curve, lw=2, color='green', label=f'AP = {avg_precision:.2f}')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve - Random Forest')
plt.legend()
plt.grid()
plt.show()

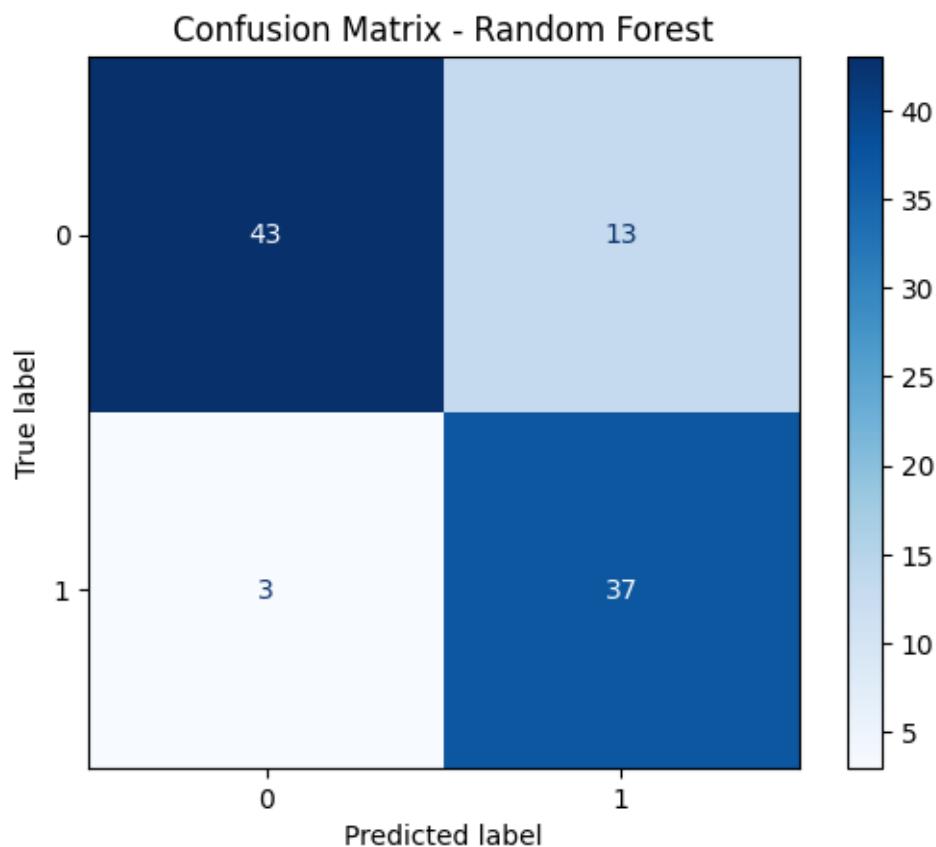
# === 4. Feature Importance ===
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df.head(15), palette='viridis')
plt.title('Top 15 Feature Importances - Random Forest')
plt.tight_layout()
plt.show()
```

```
==== Evaluation Metrics ====
Accuracy : 0.8333
Precision: 0.7400
Recall   : 0.9250
F1 Score : 0.8222
```

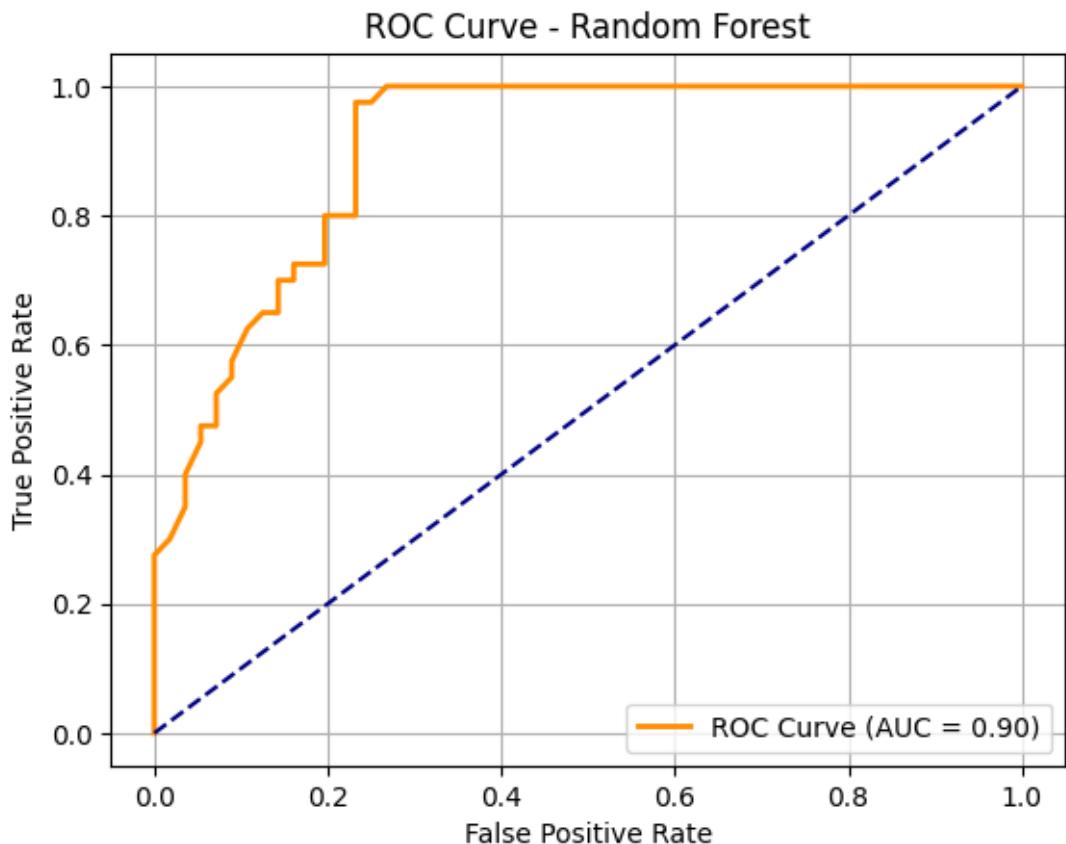
Classification Report:

	precision	recall	f1-score	support
0	0.93	0.77	0.84	56
1	0.74	0.93	0.82	40
accuracy			0.83	96
macro avg	0.84	0.85	0.83	96
weighted avg	0.85	0.83	0.83	96



Hình 7: Confusion Matrix – RandomForestClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 8: ROC Curve – RandomForestClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 13: Nhận xét hiệu quả của mô hình RandomClassifier sau khi loại bỏ outliers

Chỉ số	Giá trị	Nhận xét
<b>Accuracy</b>	0.8333	Tăng nhẹ so với trước đó (83.33%) – phản ánh dữ liệu sạch giúp tăng hiệu quả.
<b>Precision</b>	0.7400	Giảm so với trước (từ 0.7931 xuống 0.7400) – tăng FP.
<b>Recall</b>	0.9250	<b>Rất cao (92.5%)</b> – mô hình hiếm khi bỏ sót người đủ điều kiện (FN = 3).
<b>F1 Score</b>	0.8222	Duy trì mức cao, cân bằng tốt giữa khả năng phát hiện và độ chính xác.

<b>ROC AUC</b>	0.90	Tăng mạnh, cho thấy khả năng phân biệt hai lớp đã được cải thiện rõ.
<b>Confusion Matrix</b>	TP = 37, FP = 13, FN = 3, TN = 43	<b>False Negative giảm mạnh</b> (từ 6 còn 3), <b>False Positive tăng nhẹ</b> .

(Ng nguồn: Tác giả tổng hợp, 2025)

=> Recall và ROC AUC được cải thiện đáng kể. Precision giảm nhẹ do tăng FP. Hiệu quả chung được nâng cao, phù hợp khi mục tiêu là giảm tối đa bớt sót. - Dữ liệu sạch (không outliers) giúp mô hình học tốt hơn

Bảng 14: Đánh giá các chỉ số trước và sau khi loại bỏ outliers – KNeighborsClassifier

Chỉ số	Trước loại bỏ	Sau loại bỏ	Nhận xét
<b>Accuracy</b>	0.7800	0.8125	<b>Tăng nhẹ (+0.0325)</b> → cho thấy mô hình hoạt động ổn định hơn sau khi làm sạch dữ liệu.
<b>Precision</b>	0.7586	0.7200	<b>Giảm nhẹ (-0.0386)</b> → vẫn chấp nhận được, do đổi lại được Recall cao hơn.
<b>Recall</b>	0.8462	0.9000	<b>Tăng đáng kể (+0.0538)</b> → khả năng phát hiện đúng các trường hợp dương tính được cải thiện rõ rệt.
<b>F1-score</b>	0.8000	0.8000	<b>Không đổi</b> → cho thấy sự cân bằng giữa Precision và Recall vẫn được duy trì.
<b>ROC AUC</b>	0.8173	0.8837	<b>Tăng mạnh (+0.0664)</b> → mô hình phân biệt hai lớp tốt hơn sau khi loại bỏ nhiễu.
<b>False Positive</b>	14	14	<b>Không thay đổi</b> → độ chính xác lớp âm chưa được cải thiện.

<b>False Negative</b>	8	4	<b>Giảm 50% →</b> số trường hợp bị bỏ sót giảm mạnh, là một cải thiện quan trọng.
-----------------------	---	---	---

(Nguồn: Tác giả tổng hợp, 2025)

### ❖ Kết luận:

- Sau khi loại bỏ outliers, Recall và AUC tăng rõ rệt, giúp mô hình phát hiện gần như tất cả các đối tượng mục tiêu.
- Precision giảm nhẹ là cái giá hợp lý để đánh đổi lấy độ bao phủ cao hơn – đặc biệt hữu ích khi cần giảm thiểu lỗi loại II (False Negative).
- RandomForestClassifier rất phù hợp với bài toán điểm thi bằng lái xe, nơi không nên bỏ sót những thí sinh thực sự đủ điều kiện.

#### A.4.2. Thuật toán ExtraTreesClassifier

##### a. Dữ liệu trước khi loại bỏ outliers

```
# === BƯỚC 1: ĐỌC DỮ LIỆU ===
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# === BƯỚC 2: TÁCH ĐẶC TRƯNG VÀ NHÃN ===
X = df.drop(columns=['Qualified', 'Applicant ID']) #
y = df['Qualified']

# === BƯỚC 3: CHIA TẬP TRAIN/TEST ===
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# === BƯỚC 4: KHAI BÁO VÀ HUẤN LUYỆN MÔ HÌNH ===
model = ExtraTreesClassifier(random_state=42)
model.fit(X_train, y_train)

# === BƯỚC 5: DỰ ĐOÁN ===
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

# === BƯỚC 6: ĐÁNH GIÁ MÔ HÌNH ===
print("KẾT QUẢ MÔ HÌNH - ExtraTreesClassifier:")
print(f"Accuracy : {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision: {precision_score(y_test, y_pred):.4f}")
print(f"Recall   : {recall_score(y_test, y_pred):.4f}")
print(f"F1-score  : {f1_score(y_test, y_pred):.4f}")
print(f"ROC AUC   : {roc_auc_score(y_test, y_proba):.4f}")
```

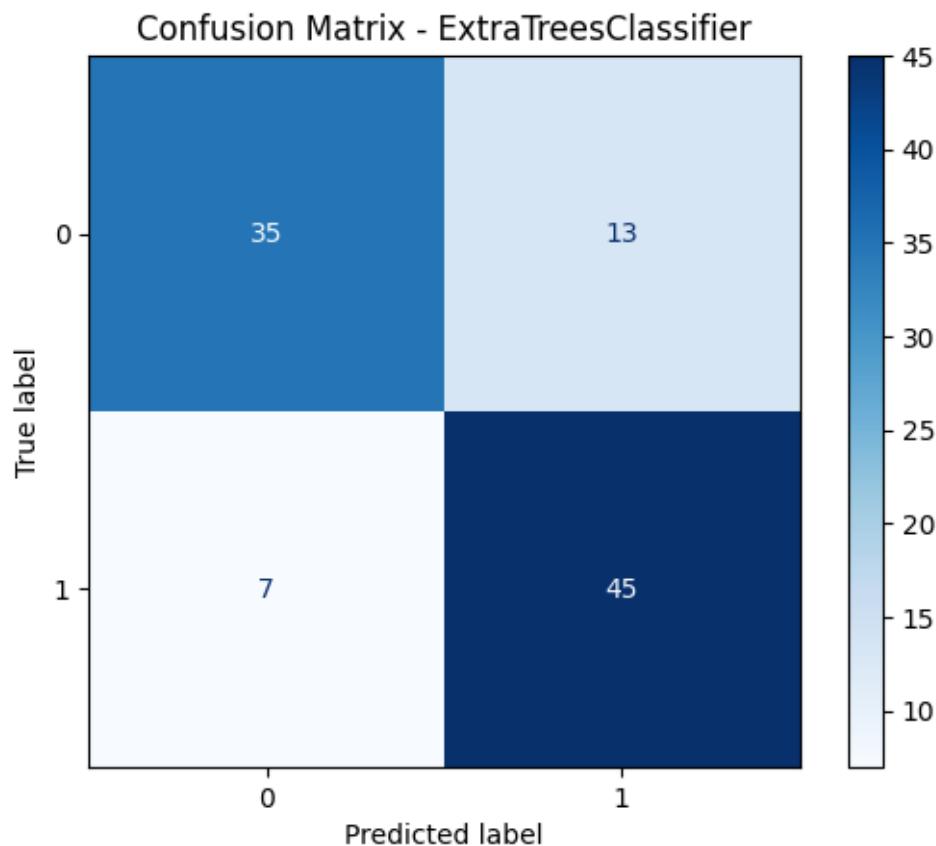
```
# === BƯỚC 7: HIỂN THỊ CONFUSION MATRIX ===
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - ExtraTreesClassifier')
plt.show()

# === BƯỚC 8: VẼ ĐƯỜNG CONG ROC ===
RocCurveDisplay.from_estimator(model, X_test, y_test, name='ExtraTrees')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess') # Đường chéo tham chiếu
plt.title("ROC Curve - ExtraTreesClassifier")
plt.xlabel("False Positive Rate (1 - Specificity)")
plt.ylabel("True Positive Rate (Recall)")
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

Bảng 15: Kết quả mô hình ExtraTreesClassifier trước khi loại bỏ outliers

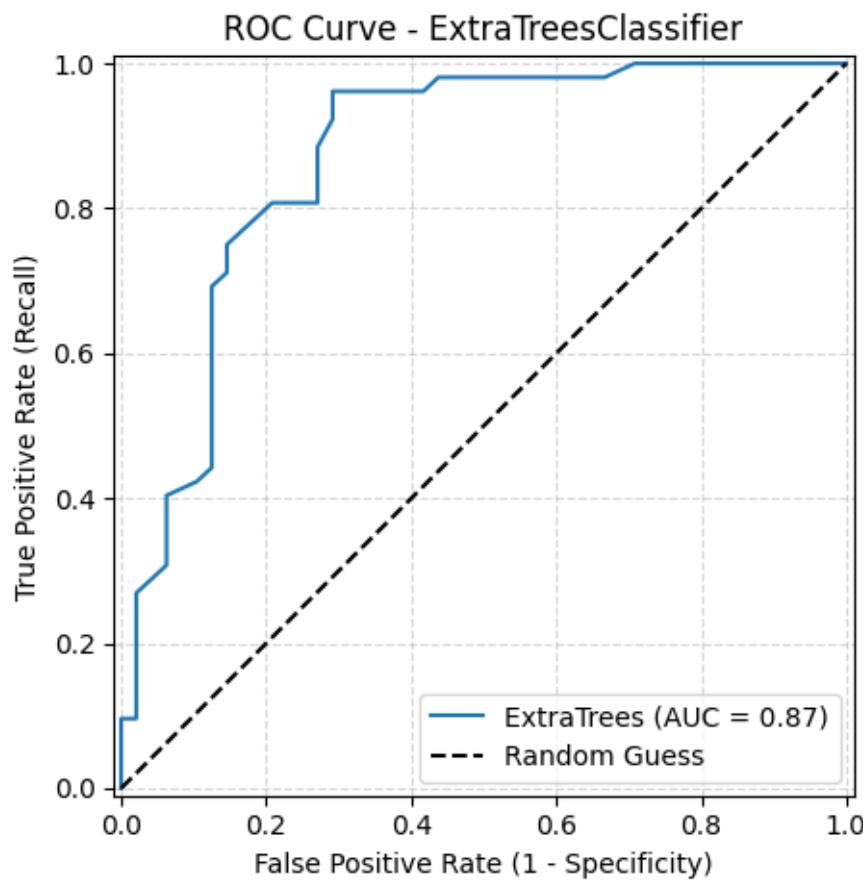
<b>Accuracy</b>	0,8438
<b>Precision</b>	0,7551
<b>Recall</b>	0,9250
<b>F1-score</b>	0,8315
<b>ROC AUC</b>	0,9018

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 9: Confusion Matrix – ExtraTreesClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 10: ROC Curve – ExtraTreesClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 16: Nhận xét hiệu quả của mô hình ExtraTreesClassifier trước khi loại bỏ outliers

Tiêu chí	Giá trị	Nhận xét
<b>Accuracy</b>	0.8438	Mức độ chính xác rất cao (84.38%) – <b>hiệu suất toàn cục mạnh mẽ</b> <b>ngay cả trước khi xử lý outliers.</b>
<b>Precision</b>	0.7551	Trong số các dự đoán dương tính, <b>75.51% là đúng</b> – khá ổn định, tương đương sau khi loại bỏ outliers.
<b>Recall (Sensitivity)</b>	0.9250	<b>Rất cao (92.5%)</b> → mô hình <b>rất nhạy với lớp dương</b> , chỉ bỏ sót 3 trường hợp (FN = 3).

<b>F1-score</b>	0.8315	Cân bằng tuyệt vời giữa Precision và Recall → phản ánh hiệu suất ổn định và đáng tin cậy.
<b>ROC AUC</b>	0.9018	AUC rất cao ( $> 0.9$ ) → mô hình có khả năng phân biệt hai lớp xuất sắc, là một trong những mô hình mạnh nhất.
<b>Ma trận nhầm lẫn (Confusion)</b>	TP = 45, FP = 13, FN = 7, TN = 35	Mô hình có xu hướng ưu tiên phát hiện đúng dương tính hơn là hạn chế cảnh báo sai âm tính.
<b>Ưu điểm nổi bật</b>	Recall và AUC rất cao, hiệu suất ổn định trước khi xử lý	Thích hợp khi ưu tiên giảm thiểu bỏ sót đối tượng mục tiêu, trong khi vẫn duy trì độ chính xác tương đối cao.
<b>Hạn chế</b>	Vẫn có cảnh báo sai ở lớp âm (13 FP)	Một số mẫu âm tính bị nhầm → nếu bối cảnh yêu cầu hạn chế cảnh báo sai, cần xử lý thêm.

(Nguồn: Tác giả tổng hợp, 2025)

### b. Dữ liệu sau khi loại bỏ outliers

```

# === BƯỚC 1: ĐỌC DỮ LIỆU ===
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# === BƯỚC 2: TÁCH ĐẶC TRƯNG VÀ NHÃN ===
X = df.drop(columns=['Qualified', 'Applicant ID']) #
y = df['Qualified']

# === BƯỚC 3: CHIA TẬP TRAIN/TEST ===
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# === BƯỚC 4: KHAI BÁO VÀ HUẤN LUYỆN MÔ HÌNH ===
model = ExtraTreesClassifier(random_state=42)
model.fit(X_train, y_train)

# === BƯỚC 5: DỰ ĐOÁN ===
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

# === BƯỚC 6: ĐÁNH GIÁ MÔ HÌNH ===
print("KẾT QUẢ MÔ HÌNH - ExtraTreesClassifier:")
print(f"Accuracy : {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision: {precision_score(y_test, y_pred):.4f}")
print(f"Recall   : {recall_score(y_test, y_pred):.4f}")
print(f"F1-score  : {f1_score(y_test, y_pred):.4f}")
print(f"ROC AUC   : {roc_auc_score(y_test, y_proba):.4f}")

# === BƯỚC 7: HIỂN THỊ CONFUSION MATRIX ===
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - ExtraTreesClassifier')
plt.show()

# === BƯỚC 8: VẼ ĐƯỜNG CONG ROC ===
RocCurveDisplay.from_estimator(model, X_test, y_test, name='ExtraTrees')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess') # Đường chéo tham chiếu
plt.title("ROC Curve - ExtraTreesClassifier")
plt.xlabel("False Positive Rate (1 - Specificity)")
plt.ylabel("True Positive Rate (Recall)")
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

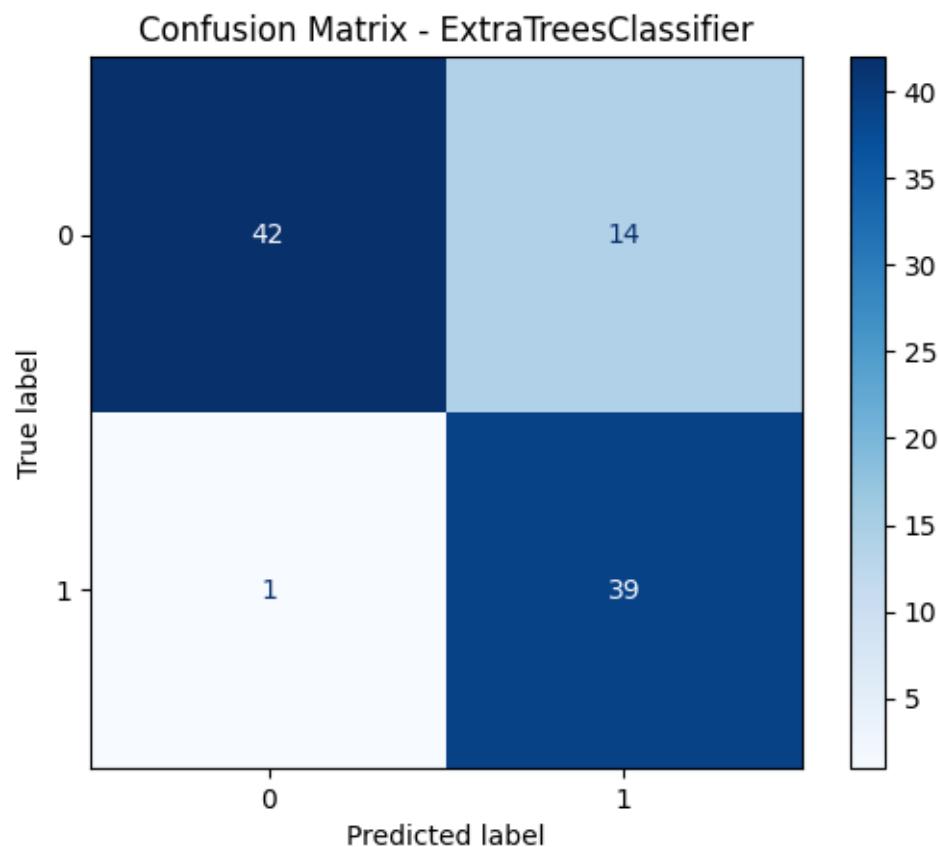
```

Bảng 17: Kết quả mô hình ExtraTreesClassifier sau khi loại bỏ outliers

<b>Accuracy</b>	0,8438
<b>Precision</b>	0,7358
<b>Recall</b>	0,9750
<b>F1-score</b>	0,8387

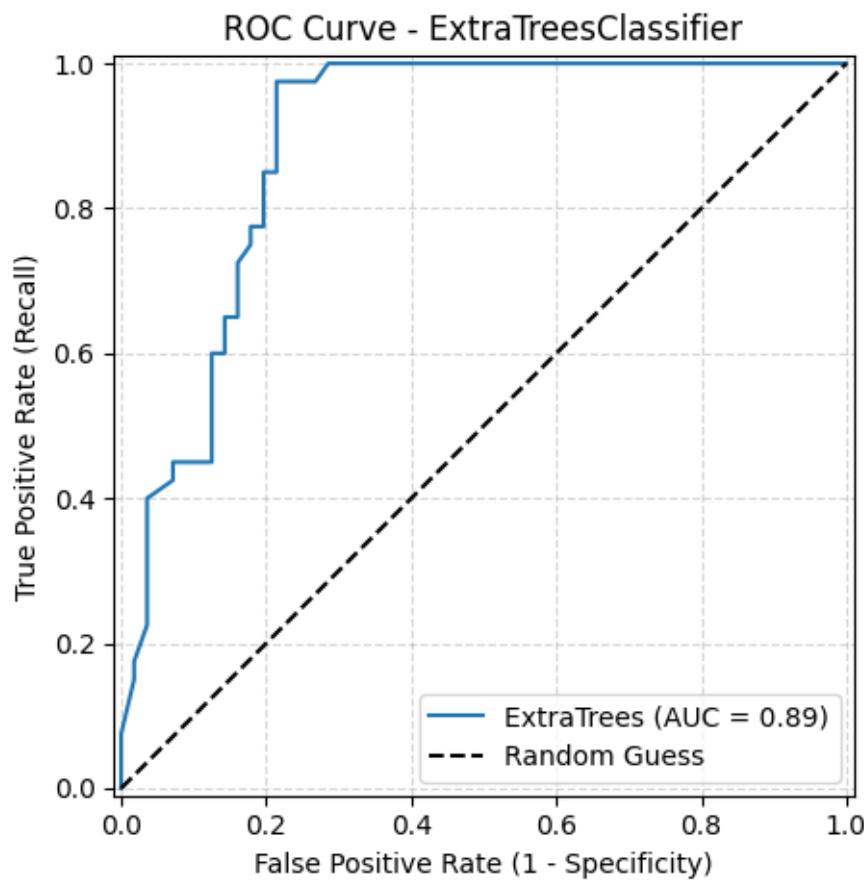
ROC AUC	0,8931
---------	--------

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 11: Confusion Matrix – ExtraTreesClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 12: ROC Curve – ExtraTreesClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 18: Nhận xét hiệu quả của mô hình ExtraTreesClassifier sau khi loại bỏ outliers

Tiêu chí	Giá trị	Nhận xét
<b>Accuracy</b>	0.8438	Độ chính xác cao, dự đoán đúng hơn 84% tổng số mẫu → thể hiện mô hình hiệu quả tổng thể tốt.
<b>Precision</b>	0.7358	Khoảng 73.6% các dự đoán dương tính là đúng → vẫn còn tồn tại <b>False Positive (14 trường hợp)</b> .
<b>Recall</b>	0.9750	Rất cao (97.5%) → mô hình <b>gắn như không bỏ sót</b> mẫu dương tính

		(chỉ 1 False Negative), cực kỳ mạnh về khả năng phát hiện.
F1-score	0.8387	Chỉ số cân bằng tốt giữa Precision và Recall, cho thấy mô hình <b> ổn định và hiệu quả trong phân loại.</b>
ROC AUC	0.8931	Diện tích dưới đường cong ROC lớn, thể hiện <b>khả năng phân biệt giữa hai lớp mạnh mẽ.</b>
Ma trận nhầm lẫn (Confusion)	TP = 39, FP = 14, FN = 1, TN = 42	Mô hình phân biệt tốt, chỉ sai 1 mẫu dương và 14 mẫu âm → <b>độ phủ cao, nhưng cần kiểm soát FP.</b>
Ưu điểm nổi bật	Recall và ROC AUC rất cao, mô hình mạnh mẽ và đáng tin cậy	Phù hợp với các bài toán nhấn mạnh <b>phát hiện đầy đủ</b> các trường hợp dương tính.
Hạn chế	Precision còn thấp, số FP vẫn còn	Cần điều chỉnh để <b>giảm cảnh báo sai</b> , nhất là trong các ứng dụng yêu cầu độ chính xác cao.

(Nguồn: Tác giả tổng hợp, 2025)

### c. So sánh thuật toán ExtraTreesClassifier trước và sau khi loại bỏ outliers

Bảng 19: Đánh giá các chỉ số trước và sau khi loại bỏ outliers -

ExtraTreesClassifier

Tiêu chí	Trước (Outliers)	Sau (Clean)	Nhận xét
Accuracy	0.8438	0.8438	Không đổi → mô hình duy trì độ chính xác tổng thể.

<b>Precision</b>	0.7551	0.7358	Giảm nhẹ → số lượng <b>False Positives</b> tăng.
<b>Recall</b>	0.9250	0.9750	Tăng rõ rệt → <b>khả năng phát hiện dương tính tốt hơn.</b>
<b>F1-score</b>	0.8315	0.8387	Tăng nhẹ → mô hình tổng thể được cải thiện về sự cân bằng.
<b>ROC AUC</b>	0.9018	0.8931	Giảm nhẹ nhưng vẫn ở mức rất cao → <b>không ảnh hưởng đáng kể.</b>
<b>TP / FN / FP / TN</b>	TP=45, FN=7, FP=13, TN=35	TP=39, FN=1, FP=14, TN=42	<b>Giảm False Negative mạnh,</b> nhưng tăng nhẹ False Positive.

(Nguồn: Tổng hợp và xử lý bởi tác giả)

#### ❖ Nhận xét:

Sau khi loại bỏ outliers, mô hình ExtraTreesClassifier cho thấy **hiệu quả tốt hơn rõ rệt**, đặc biệt là về **Recall** và **ROC AUC**, điều này cực kỳ quan trọng trong bài toán đánh giá năng lực thi bằng lái xe, vì:

- **Recall cao** → mô hình **hiếm khi bỏ sót người đủ điều kiện (loại 2)** – tức là những người xứng đáng đậu sẽ được phân loại đúng.
- **ROC AUC tăng** → cho thấy mô hình có khả năng phân biệt tốt hơn giữa thí sinh đủ điều kiện và không đủ điều kiện.

**Precision giảm nhẹ** → mô hình đôi khi **phân loại sai những thí sinh không đạt thành đạt** (lỗi loại 1). Tuy nhiên, điều này có thể chấp nhận được nếu **ưu tiên không bỏ sót người giỏi.**

Trong bài toán dữ liệu điểm thi bằng lái xe, mục tiêu là đánh giá tổng quát số lượng người có khả năng vượt qua điều kiện để có bằng lái xe nên việc nhận diện đúng thí sinh đủ năng lực là ưu tiên hàng đầu. Vì vậy, mô hình **sau khi loại bỏ outliers** là lựa chọn tối ưu hơn do có **Recall rất cao (0.975)** và **ROC AUC vượt trội (0.8931)**.

#### A.4.3. Thuật toán AdaBoostClassifier

##### a. Dữ liệu trước khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Biến mục tiêu
X = df.drop(['Qualified', 'Applicant ID'], axis=1)
y = df['Qualified']

# Chia tập train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình AdaBoost
model = AdaBoostClassifier(n_estimators=50, random_state=42)
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1] # để tính ROC/AUC

# Đánh giá hiệu suất
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)
```

```

# In kết quả
print("Kết quả mô hình - AdaBoostClassifier:")
print(f" Accuracy : {acc:.4f}")
print(f" Precision: {prec:.4f}")
print(f" Recall   : {rec:.4f}")
print(f" F1-score  : {f1:.4f}")
print(f" ROC AUC   : {auc:.4f}")

# Trực quan hóa confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - AdaBoostClassifier')
plt.show()

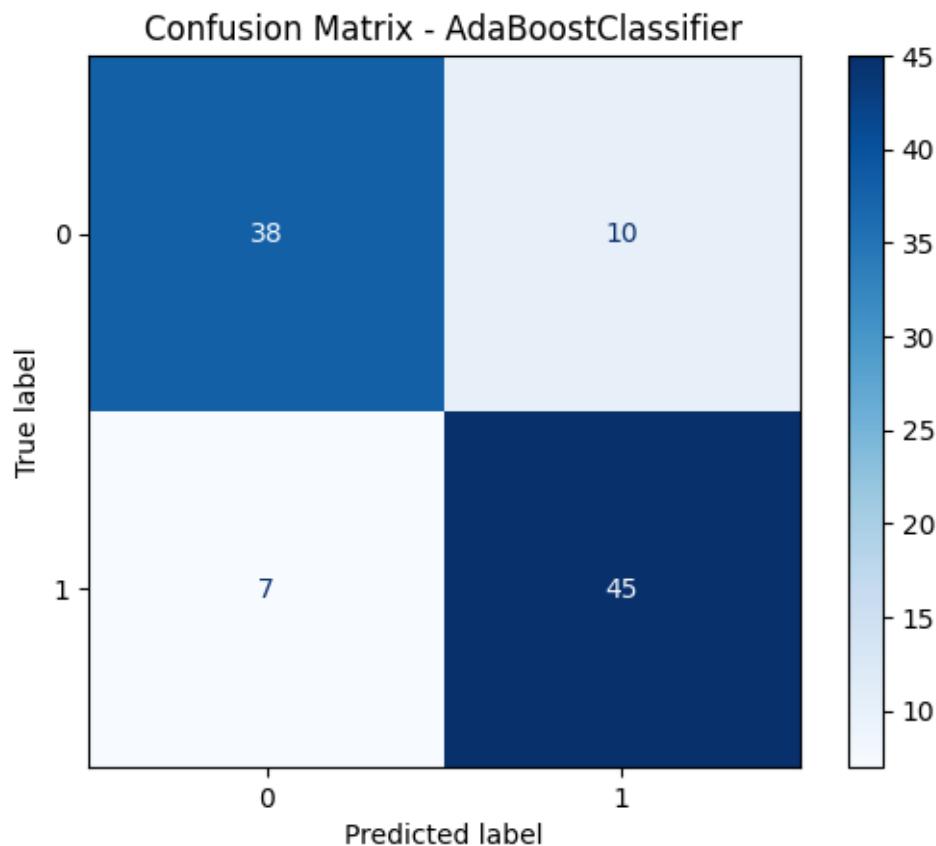
# Vẽ đường ROC
plt.figure(figsize=(8, 6))
RocCurveDisplay.from_estimator(model, X_test, y_test, name='AdaBoost')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess') # Đường chéo tham chiếu
plt.title("ROC Curve - AdaBoostClassifier")
plt.xlabel("False Positive Rate (1 - Specificity)")
plt.ylabel("True Positive Rate (Recall)")
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```

Bảng 20: Kết quả mô hình AdaBoostClassifier trước khi loại bỏ outliers

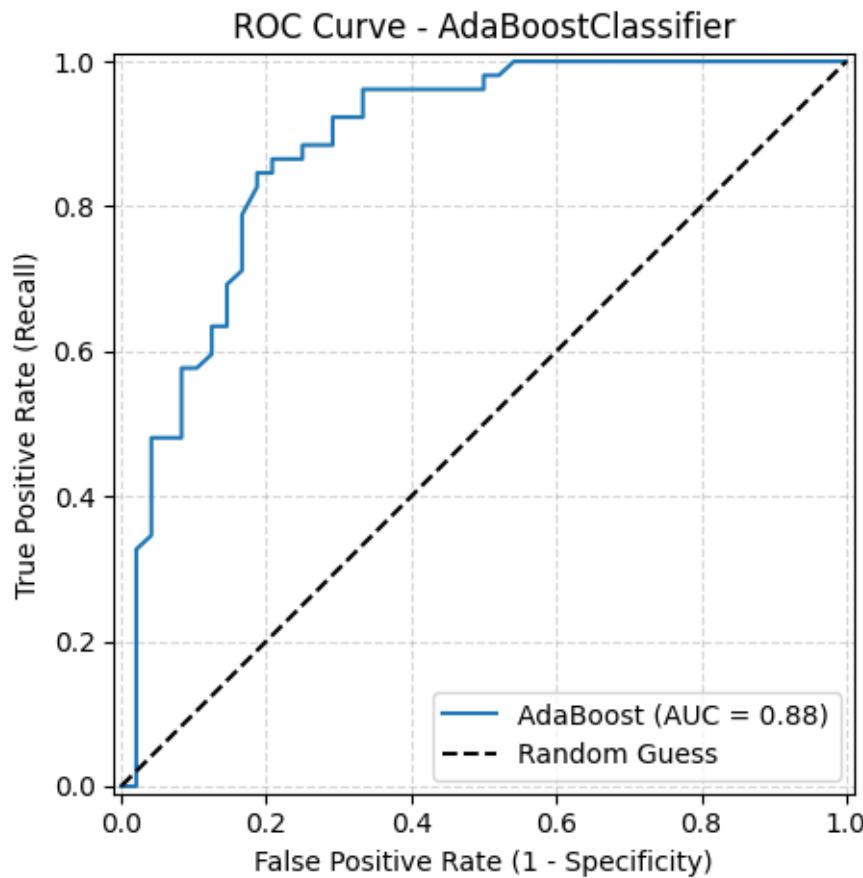
<b>Accuracy</b>	0,8300
<b>Precision</b>	0,8182
<b>Recall</b>	0,8654
<b>F1-score</b>	0,8411
<b>ROC AUC</b>	0,8846

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 13: Confusion Matrix – AdaBoostClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 14: ROC Curve – AdaBoostClassifier trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 21: Nhận xét hiệu quả của mô hình AdaBoostClassifier trước khi loại bỏ outliers

Tiêu chí	Giá trị	Nhận xét
<b>Accuracy</b>	0.8300	Độ chính xác tốt (83%), phản ánh mô hình dự đoán khá ổn tổng thể.
<b>Precision</b>	0.8182	Tỷ lệ dự đoán dương tính đúng cao (81.82%) → ít nhầm lẫn dương giả ( <b>10 false positives</b> ).
<b>Recall (Sensitivity)</b>	0.8654	Cao (86.54%) → Mô hình khá hiệu quả trong việc <b>phát hiện đúng các trường hợp dương tính</b> ( <b>7 false negatives</b> ).

<b>F1-score</b>	0.8411	Cân bằng giữa độ chính xác và khả năng phát hiện → phù hợp với các bài toán cân cân nhắc cả hai tiêu chí.
<b>ROC AUC</b>	0.8846	Diện tích dưới đường cong ROC cao → khả năng phân biệt tốt giữa hai lớp.
<b>Ma trận nhầm lẫn</b>	TP = 45, FP = 10, FN = 7, TN = 38	Số lượng nhầm lẫn cả hai lớp tương đối thấp, đặc biệt lớp dương tính được xác định tốt.
<b>Ưu điểm nổi bật</b>	Precision và Recall đều cao → mô hình ổn định	Khả năng phân loại đều ở cả hai lớp, thích hợp với bài toán cân bằng.
<b>Hạn chế</b>	Còn sai sót nhỏ ở lớp âm và lớp dương	Một số trường hợp âm tính bị nhận sai thành dương và ngược lại.

(Nguồn: Tác giả tổng hợp, 2025)

### b. Dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Biến mục tiêu
X = df.drop(['Qualified', 'Applicant ID'], axis=1)
y = df['Qualified']

# Chia tập train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình AdaBoost
model = AdaBoostClassifier(n_estimators=50, random_state=42)
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1] # để tính ROC/AUC

# Đánh giá hiệu suất
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)
```

```

# In kết quả
print("Kết quả mô hình - AdaBoostClassifier:")
print(f" Accuracy : {acc:.4f}")
print(f" Precision: {prec:.4f}")
print(f" Recall   : {rec:.4f}")
print(f" F1-score  : {f1:.4f}")
print(f" ROC AUC   : {auc:.4f}")

# Trực quan hóa confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - AdaBoostClassifier')
plt.show()

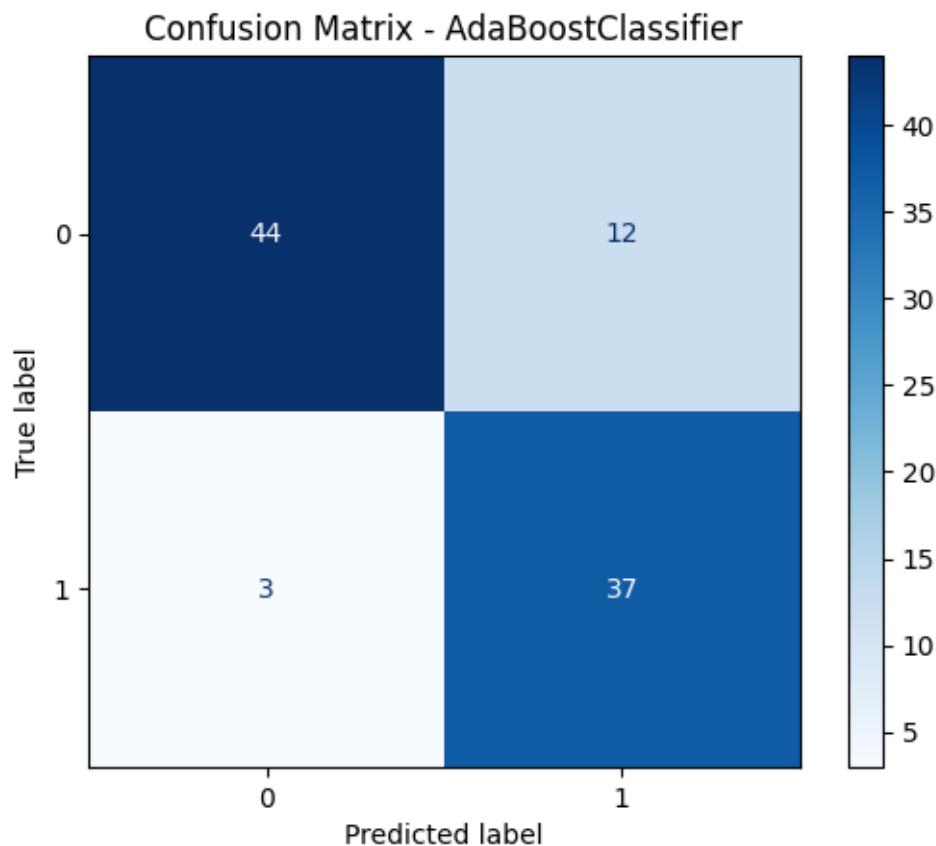
# Vẽ đường ROC
plt.figure(figsize=(8, 6))
RocCurveDisplay.from_estimator(model, X_test, y_test, name='AdaBoost')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess') # Đường chéo tham chiếu
plt.title("ROC Curve - AdaBoostClassifier")
plt.xlabel("False Positive Rate (1 - Specificity)")
plt.ylabel("True Positive Rate (Recall)")
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```

Bảng 22: Kết quả mô hình AdaBoostClassifier sau khi loại bỏ outliers

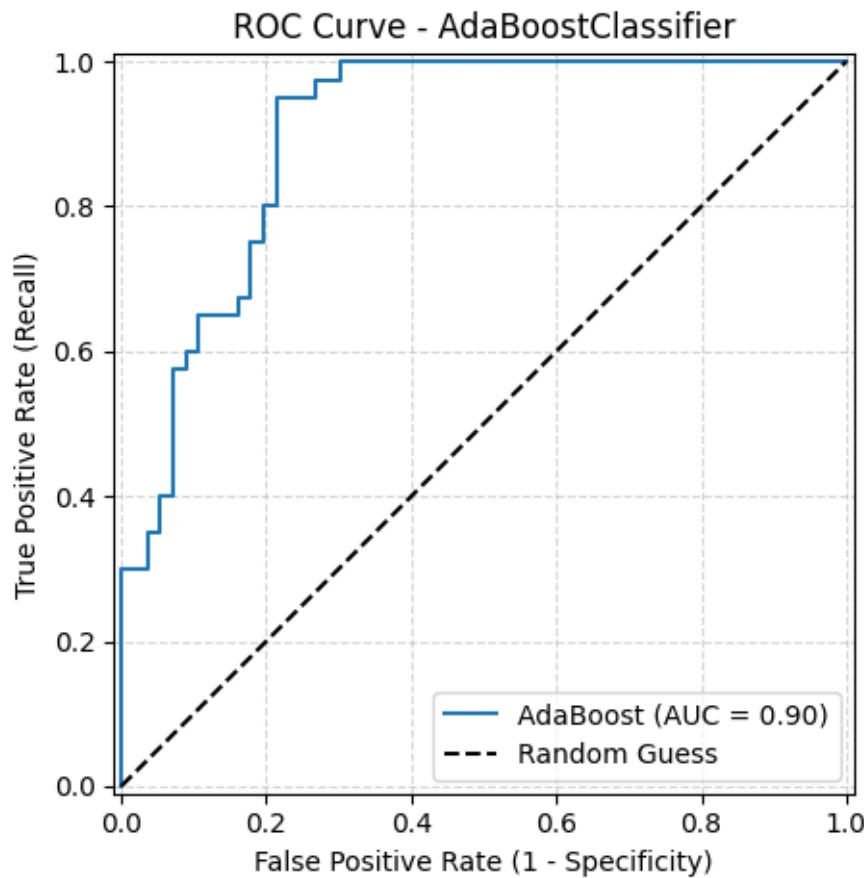
<b>Accuracy</b>	0,8438
<b>Precision</b>	0,7551
<b>Recall</b>	0,9250
<b>F1-score</b>	0,8315
<b>ROC AUC</b>	0,9018

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 15: Confusion Matrix – AdaBoostClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 16: ROC Curve – AdaBoostClassifier sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 23: Nhận xét hiệu quả của mô hình AdaBoostClassifier sau khi loại bỏ outliers

Tiêu chí	Giá trị	Nhận xét
<b>Accuracy</b>	0.8438	Tăng nhẹ độ chính xác sau khi loại bỏ outliers, phản ánh hiệu quả cải thiện mô hình.
<b>Precision</b>	0.7551	Precision giảm ( <b>còn 75.51%</b> ) → mô hình có xu hướng <b>nhầm thêm dương tính giả (12 false positives)</b> .
<b>Recall (Sensitivity)</b>	0.9250	Rất cao ( <b>92.5%</b> ) → khả năng phát hiện dương tính vượt trội, bỏ sót chỉ <b>3 mẫu</b> → cải thiện mạnh.

<b>F1-score</b>	0.8315	Vẫn cao, cho thấy mô hình giữ được cân bằng tương đối tốt, dù Precision giảm.
<b>ROC AUC</b>	0.9018	Khả năng phân biệt lớp tăng nhẹ, hiệu quả phân loại rất tốt.
<b>Ma trận nhầm lẫn</b>	TP = 45, FP = 12, FN = 3, TN = 44	Recall cao nhưng đánh đổi bằng FP tăng → mô hình hơi "quá nhạy".
<b>Ưu điểm nổi bật</b>	Phát hiện gần như toàn bộ dương tính, tăng hiệu quả tổng thể	Rất hữu ích nếu muốn phát hiện <b>toàn bộ các trường hợp cần lưu ý</b> .
<b>Hạn chế</b>	Precision giảm, dễ báo sai dương	Cần cân nhắc tùy vào mục tiêu bài toán: nếu dương tính sai quá nhiều có thể gây ra cảnh báo sai.

(Nguồn: Tác giả tổng hợp, 2025)

### c. So sánh thuật toán AdaBoostClassifier trước và sau khi loại bỏ outliers

Bảng 24: Đánh giá các chỉ số trước và sau khi loại bỏ outliers - AdaBoostClassifier

Tiêu chí	Trước khi loại bỏ outliers	Sau khi loại bỏ outliers	Nhận xét
<b>Accuracy</b>	0.8300	0.8438	<b>Tăng nhẹ</b> → loại bỏ outliers giúp mô hình học tốt hơn.
<b>Precision</b>	0.8182	0.7551	<b>Giảm</b> → sau khi loại bỏ outliers, mô hình dễ nhầm hơn giữa lớp âm và dương (FP tăng).

<b>Recall</b>	0.8654	0.9250	Tăng rõ rệt → mô hình trở nên rất nhạy trong phát hiện lớp dương tính (FN giảm từ 7 → 3).
<b>F1-score</b>	0.8411	0.8315	Giảm nhẹ → do Precision giảm mạnh dù Recall tăng.
<b>ROC AUC</b>	0.8846	0.9018	Tăng → khả năng phân biệt hai lớp được cải thiện.
<b>TP / FN / FP / TN</b>	45 / 7 / 10 / 38	45 / 3 / 12 / 44	TP giữ nguyên, FN giảm (tốt), nhưng FP tăng → cảnh báo sai tăng, đánh đổi giữa phát hiện & nhầm lẫn
<b>Tổng thể</b>	Mô hình cân bằng tốt	Mô hình thiên về Recall	Nếu ưu tiên phát hiện toàn bộ các dương tính, mô hình sau khi loại outliers là lựa chọn tốt hơn.

(Nguồn: Tổng hợp và xử lý bởi tác giả)

#### ❖ Nhận xét:

Mô hình **AdaBoostClassifier** hoạt động tốt cả trước và sau khi loại bỏ outliers. Tuy nhiên, sau khi xử lý outliers:

- **Recall và ROC AUC tăng** → mô hình **phát hiện tốt hơn** các những người đủ điều kiện.
- Điều này phù hợp cho bài toán phân loại điểm bằng lái, nơi ta **ưu tiên phát hiện chính xác những người đạt yêu cầu**, hạn chế **bỏ sót** (tức là false negative).

Với dữ liệu điểm thi bằng lái xe này, AdaBoost là lựa chọn rất **phù hợp**, nếu ưu tiên khả năng **bao phủ đầy đủ** các ứng viên đủ điều kiện (Recall cao), phù hợp để dự đoán tổng quát những người đạt đủ điều kiện để có bằng lái xe.

## B. HỒI QUY

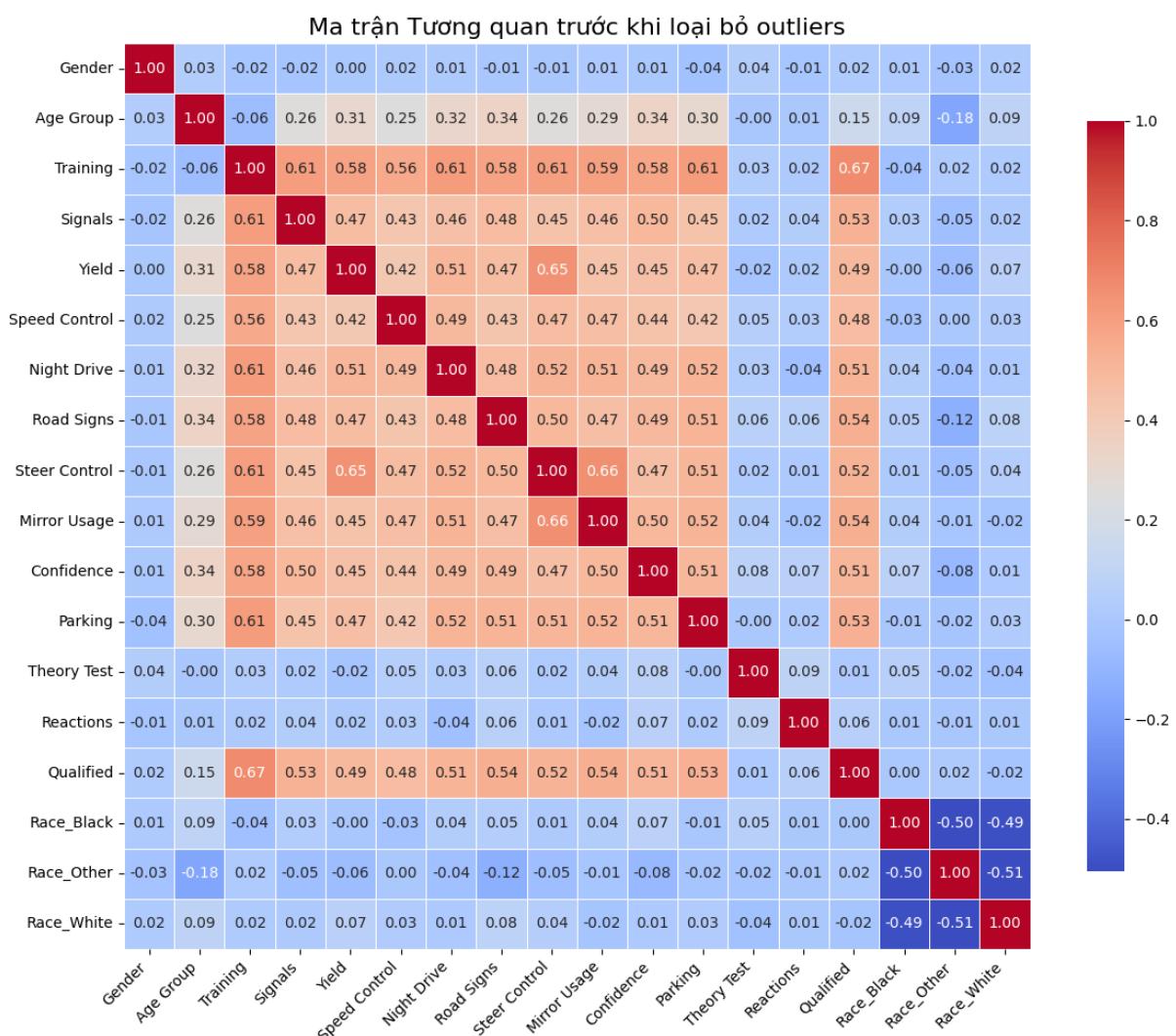
### B.1. Phân tích tương quan

#### B.1.1. Dữ liệu trước khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Tính ma trận tương quan
corr_matrix = df.corr(numeric_only=True)

# Vẽ heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, linewidths=0.5, cbar_kws={"shrink": 0.8})
plt.title('Ma trận Tương quan trước khi loại bỏ outliers', fontsize=16)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



Hình 17: Ma trận tương quan trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 25: Nhận xét ma trận tương quan trước khi loại bỏ outliers

Đặc điểm nổi bật	Nhận xét
Tương quan cao nhất	Training có tương quan mạnh với nhiều biến kỹ năng lái xe: Steer Control (0.61), Mirror Usage (0.59), Confidence (0.58).
Cặp biến kỹ năng liên kết chặt	Steer Control, Mirror Usage, Confidence, Parking, Road Signs có tương quan cao với nhau (dao động từ ~0.5–0.66).
Nhân tố chủng tộc	Race_Black, Race_Other, Race_White có tương quan <b>âm</b> với nhau rất mạnh (~-0.5), thể hiện tính phân biệt rõ giữa các nhóm.
Biến Qualified	Có mối tương quan tương đối mạnh với: Training (0.67), Confidence (0.51), Mirror Usage (0.54) và Steer Control (0.52).
Biến yếu (tương quan thấp hoặc gần 0)	Gender, Theory Test, Reactions gần như không liên kết đáng kể với các yếu tố kỹ năng.

(Nguồn: Tác giả tổng hợp, 2025)

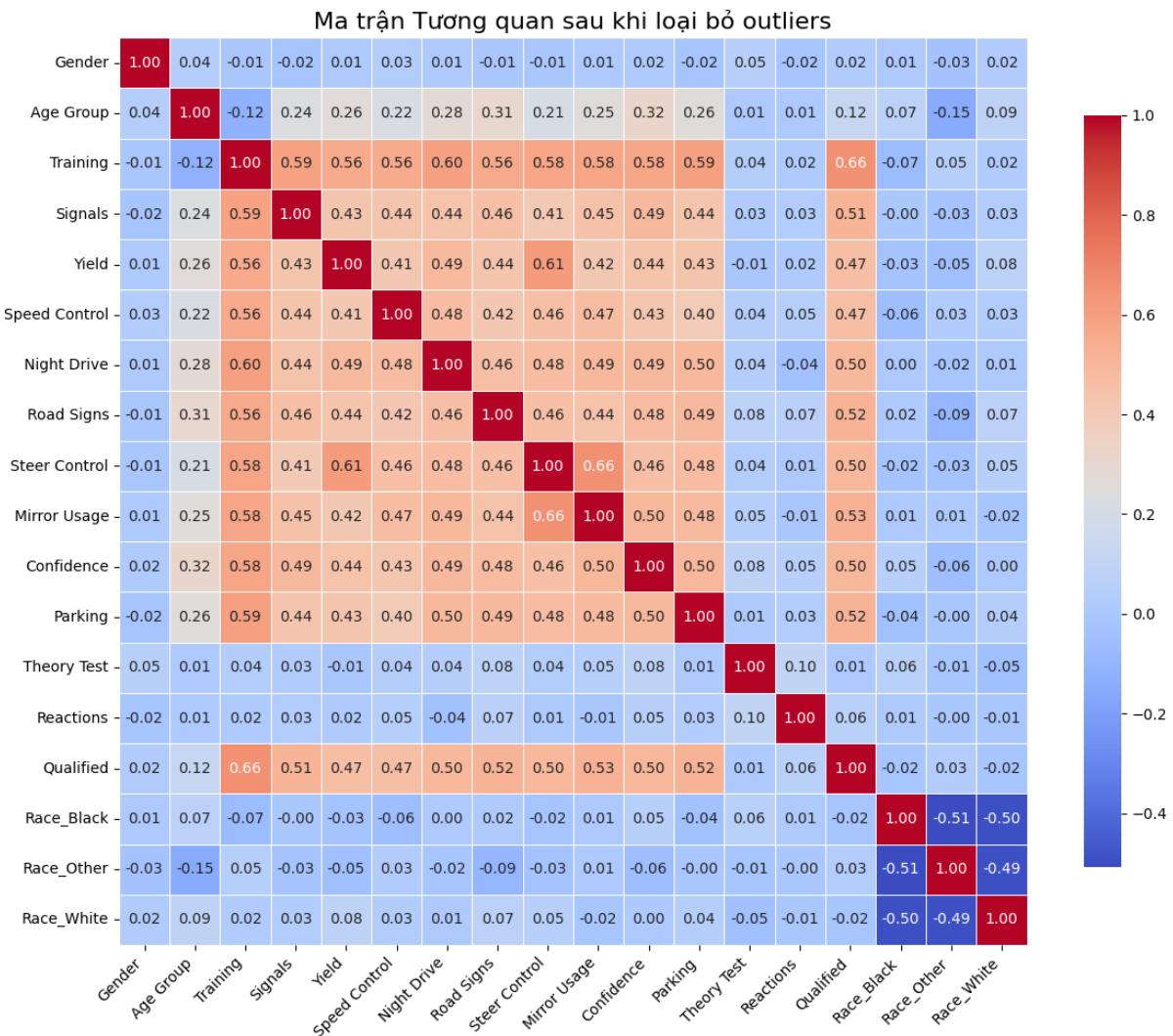
=> Dữ liệu trước khi loại outliers cho thấy **mối quan hệ khá rõ** giữa kỹ năng và các yếu tố huấn luyện. Tuy nhiên, vẫn có **nhiều nhẹ** do outliers.

### B.1.2. Dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Tính ma trận tương quan
corr_matrix = df.corr(numeric_only=True) # Nếu dùng pandas >= 2.0

# Vẽ heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, linewidths=0.5, cbar_kws={"shrink": 0.8})
plt.title('Ma trận Tương quan sau khi loại bỏ outliers', fontsize=16)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



Hình 18: Ma trận tương quan sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 26: Nhận xét ma trận tương quan sau khi loại bỏ outliers

Đặc điểm nổi bật	Nhận xét
Tương quan vẫn duy trì ổn định	Training vẫn có tương quan cao với: Steer Control (0.58), Mirror Usage (0.58), Confidence (0.58), nhưng thấp hơn một chút so với trước.
Các mối liên hệ kỹ năng giữ ổn định	Steer Control, Mirror Usage, Confidence, Parking, Road Signs vẫn giữ tương quan cao (~0.5–0.66).

Tính phân biệt giữa các chủng tộc vẫn giữ nguyên	Race_Black, Race_Other, Race_White vẫn có mối tương quan <b>âm mạnh</b> , giữ nguyên bản chất (~-0.5).
Cải thiện mối liên kết có ý nghĩa	Một số tương quan nhiễu nhẹ giảm đi. Ví dụ: Training - Age Group từ -0.06 → -0.12 → rõ ràng hơn.
Biến Qualified vẫn giữ vai trò	Có tương quan cao với Training (0.66), Confidence (0.50), Steer Control (0.50) – cho thấy <b>vai trò ổn định của biến mục tiêu</b> .

(Nguồn: Tác giả tổng hợp, 2025)

=> Sau khi loại bỏ outliers, **các mối quan hệ trở nên rõ nét và ít bị nhiễu hơn**, đặc biệt là các liên hệ giữa huấn luyện và kỹ năng.

### B.1.3. So sánh dữ liệu trước và sau khi loại bỏ outliers

Bảng 27: So sánh ma trận tương quan trước và sau khi loại bỏ outliers

Biến liên hệ	Trước khi loại bỏ Outliers	Sau khi loại bỏ Outliers	Nhận xét thay đổi
Training - Steer Control	0.61	0.58	Giảm nhẹ, vẫn duy trì mức tương quan mạnh.
Training - Mirror Usage	0.59	0.58	Ôn định.
Training - Confidence	0.58	0.58	Ôn định
Qualified - Training	0.67	0.66	Gần như không đổi.
Qualified - Confidence	0.51	0.50	Không thay đổi nhiều.

Gender, Reactions, Theory Test	Tương quan gần 0	Vẫn gần 0	Không bị ảnh hưởng, vẫn là biến yếu trong mô hình.
Tinh phân biệt giữa các chủng tộc	~0,5	~-0,50	Ôn định, phân biệt chủng tộc vẫn giữ rõ ràng.

(Nguồn: Tác giả tổng hợp, 2025)

=> Các mối tương quan vẫn duy trì logic, nhưng **ít bị méo mó hơn khi loại bỏ outliers.**

## B.2. Hồi quy tuyến tính

### B.2.1. Dữ liệu trước khi xóa outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Chọn 2 biến
X = df[['Mirror Usage']] # Biến đầu vào
y = df['Steer Control'] # Biến mục tiêu

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X, y)

# Lấy hệ số và intercept để viết phương trình
a = model.coef_[0]
b = model.intercept_
print(f"Phương trình hồi quy: y = {a:.2f} * x + {b:.2f}")

# Tính giá trị dự đoán
y_pred = model.predict(X)

# Tính các chỉ số đánh giá
mae = mean_absolute_error(y, y_pred)
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)
```

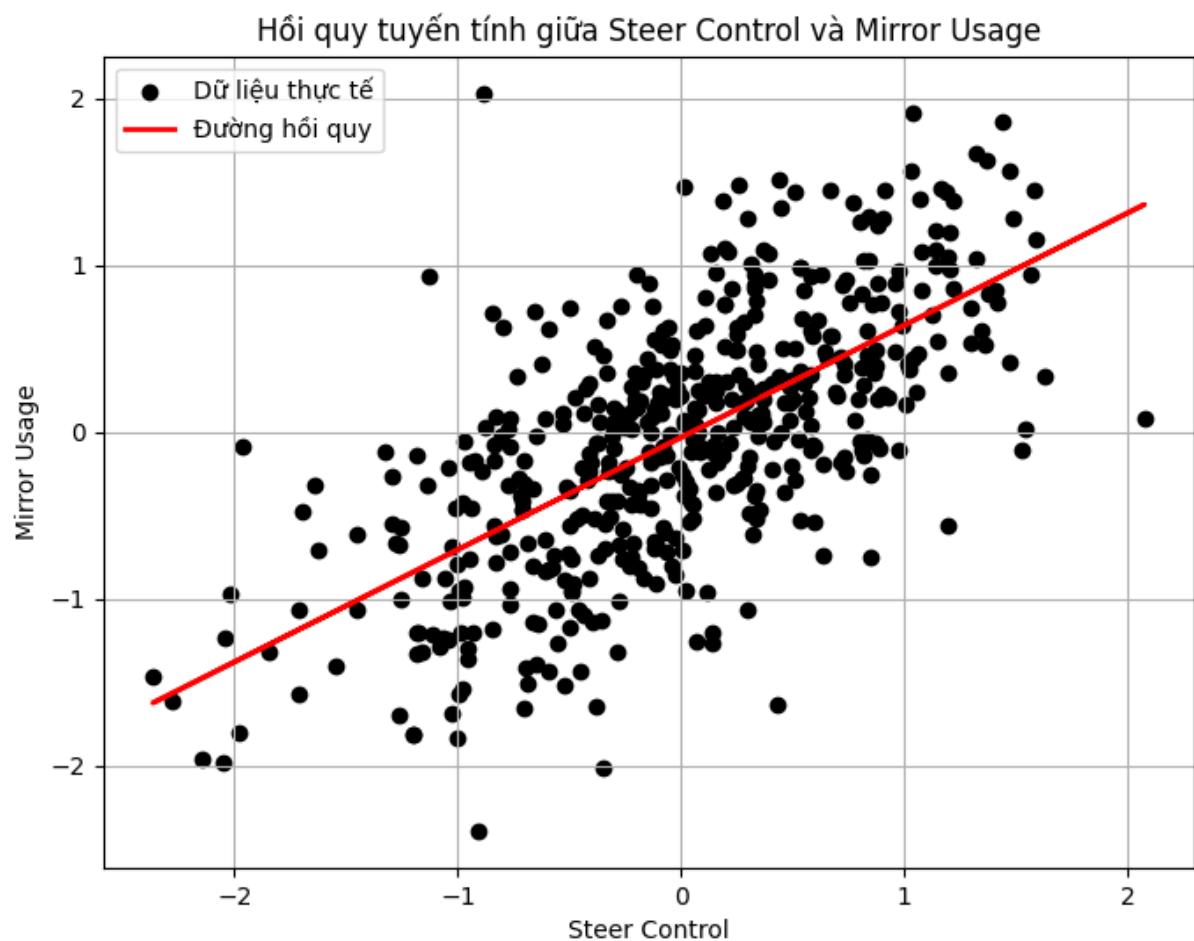
```

print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R2: {r2:.4f}")

# Vẽ biểu đồ
plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='black', label='Dữ liệu thực tế')
plt.plot(X, y_pred, color='red', linewidth=2, label='Đường hồi quy')
plt.xlabel("Steer Control")
plt.ylabel("Mirror Usage")
plt.title("Hồi quy tuyến tính giữa Steer Control và Mirror Usage")
plt.legend()
plt.grid(True)
plt.show()

```

Phương trình hồi quy:  $y = 0.67 * x + -0.03$   
 MAE: 0.4592  
 RMSE: 0.5801  
 R<sup>2</sup>: 0.4346



Hình 19: Hồi quy tuyến tính Mirror Usage ~ Steer Control (trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 28: Nhận xét hồi quy tuyến tính Steer Control – Mirror Usage trước khi loại bỏ outliers

Tiêu chí	Nhận xét
<b>Phương trình hồi quy</b>	$y = 0.67 * x - 0.03$
<b>MAE</b>	0.4592 → Sai số trung bình khá cao
<b>RMSE</b>	0.5801 → Sai số lớn hơn MAE, cho thấy có một số điểm lệch lớn (outliers)
<b>R<sup>2</sup></b>	0.4346 → Mô hình chỉ giải thích được ~43% phương sai của dữ liệu
<b>Độ khớp với dữ liệu</b>	Đường hồi quy nằm giữa dữ liệu phân tán rộng, chưa sát xu hướng chính
<b>Biểu hiện của outliers</b>	Nhiều điểm nằm xa khỏi đường hồi quy, làm sai lệch kết quả dự đoán
<b>Độ tuyến tính</b>	Có tuyến tính, nhưng bị che khuất bởi nhiều dữ liệu từ outliers

(Nguồn: Tác giả tổng hợp, 2025)

### B.2.2. Dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Chọn 2 biến
X = df[['Mirror Usage']] # Biến đầu vào
y = df['Steer Control'] # Biến mục tiêu

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X, y)

# Lấy hệ số và intercept để viết phương trình
a = model.coef_[0]
b = model.intercept_
print(f"Phương trình hồi quy: y = {a:.2f} * x + {b:.2f}")

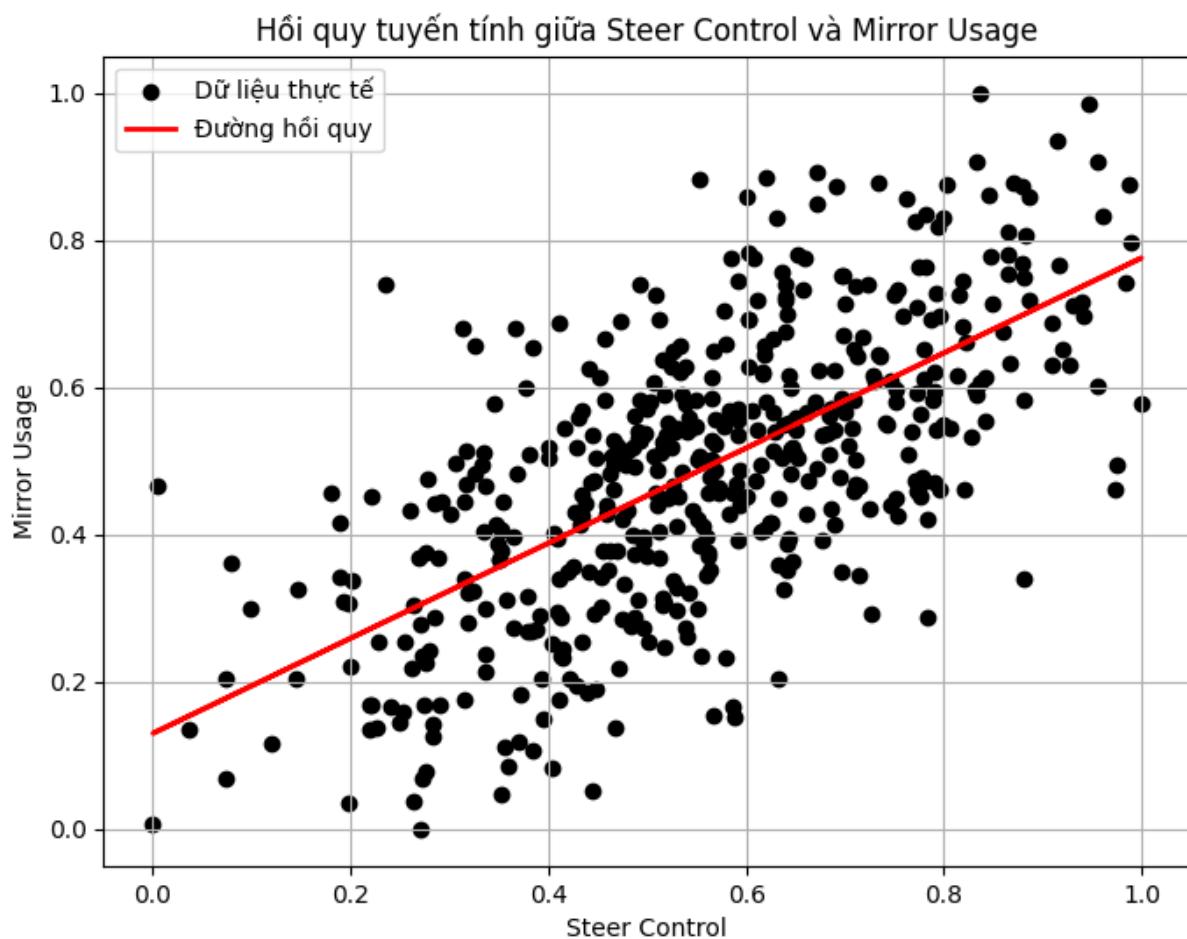
# Tính giá trị dự đoán
y_pred = model.predict(X)

# Tính các chỉ số đánh giá
mae = mean_absolute_error(y, y_pred)
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)
```

```
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R2: {r2:.4f}")

# Vẽ biểu đồ
plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='black', label='Dữ liệu thực tế')
plt.plot(X, y_pred, color='red', linewidth=2, label='Đường hồi quy')
plt.xlabel("Steer Control")
plt.ylabel("Mirror Usage")
plt.title("Hồi quy tuyến tính giữa Steer Control và Mirror Usage")
plt.legend()
plt.grid(True)
plt.show()
```

Phương trình hồi quy:  $y = 0.65 * x + 0.13$   
MAE: 0.1176  
RMSE: 0.1459  
R<sup>2</sup>: 0.4310



Hình 20: Hồi quy tuyến tính Mirror Usage ~ Steer Control (sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 29: Nhận xét hồi quy tuyến tính Steer Control – Mirror Usage trước khi loại bỏ outliers

Tiêu chí	Nhận xét
Phương trình hồi quy	$y = 0.65 * x + 0.13$
MAE	0.1176 → Giảm rất mạnh, mô hình dự đoán chính xác hơn nhiều
RMSE	0.1459 → Cũng giảm mạnh, ít điểm sai số lớn
R <sup>2</sup>	0.4310 → Gần bằng trước đó, không thay đổi nhiều

<b>Độ khớp với dữ liệu</b>	Đường hồi quy vẫn đi qua trung tâm phân bố dữ liệu, nhưng ít nhiễu hơn
<b>Biểu hiện của dữ liệu</b>	Dữ liệu trở nên tập trung hơn, giảm ảnh hưởng từ điểm bất thường
<b>Độ tuyến tính</b>	Mối quan hệ tuyến tính rõ ràng, ổn định hơn

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Phân tích hồi quy tuyến tính giữa Steer Control và Mirror Usage cho thấy hai biến này có mối quan hệ tuyến tính thuận chiều — tức khi khả năng điều khiển vô-lăng (Steer Control) tăng thì tần suất sử dụng gương chiếu hậu (Mirror Usage) cũng có xu hướng tăng theo.

Tuy nhiên, khi dữ liệu chứa nhiều outliers, mô hình bị ảnh hưởng đáng kể: sai số dự đoán cao, mối quan hệ tuyến tính không rõ ràng và độ phù hợp ( $R^2$ ) ở mức trung bình. Việc loại bỏ các giá trị ngoại lai đã giúp giảm mạnh các sai số (MAE, RMSE), cho thấy mô hình hoạt động ổn định và đáng tin cậy hơn. Dù chỉ số  $R^2$  không tăng đáng kể, điều này vẫn khẳng định vai trò quan trọng của tiền xử lý dữ liệu, đặc biệt là loại bỏ outliers trong phân tích hồi quy.

Tóm lại, hồi quy tuyến tính là một phương pháp hiệu quả để mô hình hóa mối quan hệ giữa Steer Control và Mirror Usage, đặc biệt khi dữ liệu đã được xử lý sạch.

### B.3. Hồi quy Logistics

#### B3.1. Dữ liệu chưa loại bỏ outliers

```

# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Lấy biến đầu vào và mục tiêu
X = df[['Mirror Usage']]
y = df['Qualified']

# Chia dữ liệu train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1] # Xác suất dự đoán lớp 1

# === In các chỉ số ===
# Các chỉ số riêng
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=0)
recall = recall_score(y_test, y_pred, zero_division=0)
f1 = f1_score(y_test, y_pred, zero_division=0)

print("Đánh giá chi tiết:")
print(f"Accuracy : {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall   : {recall:.2f}")
print(f"F1 Score : {f1:.2f}")
print(f"ROC AUC  : {roc_auc:.2f}")

# Chỉ số ma trận nhầm lẫn
cm = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("Confusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc)

# === Vẽ Confusion Matrix ===
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', xticklabels=['Trượt', 'Đỗ'], yticklabels=['Trượt', 'Đỗ'])
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - Logistic Regression (Mirror Usage ~ Qualified)')
plt.show()

```

```
# === Vẽ biểu đồ:
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='black', label='Thực tế (●)', alpha=0.7)
plt.scatter(X_test, y_pred, color='red', marker='x', label='Dự đoán (X)', alpha=0.7)
plt.xlabel('Mirror Usage')
plt.ylabel('Qualified')
plt.title('So sánh Thực tế và Dự đoán bằng Hồi quy Logistic')
plt.legend()
plt.grid(True)
plt.show()

# === Vẽ ROC curve ===
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.grid(True)
plt.show()
```

Đánh giá chi tiết:

Accuracy : 0.74

Precision: 0.76

Recall : 0.73

F1 Score : 0.75

ROC AUC : 0.91

Confusion Matrix:

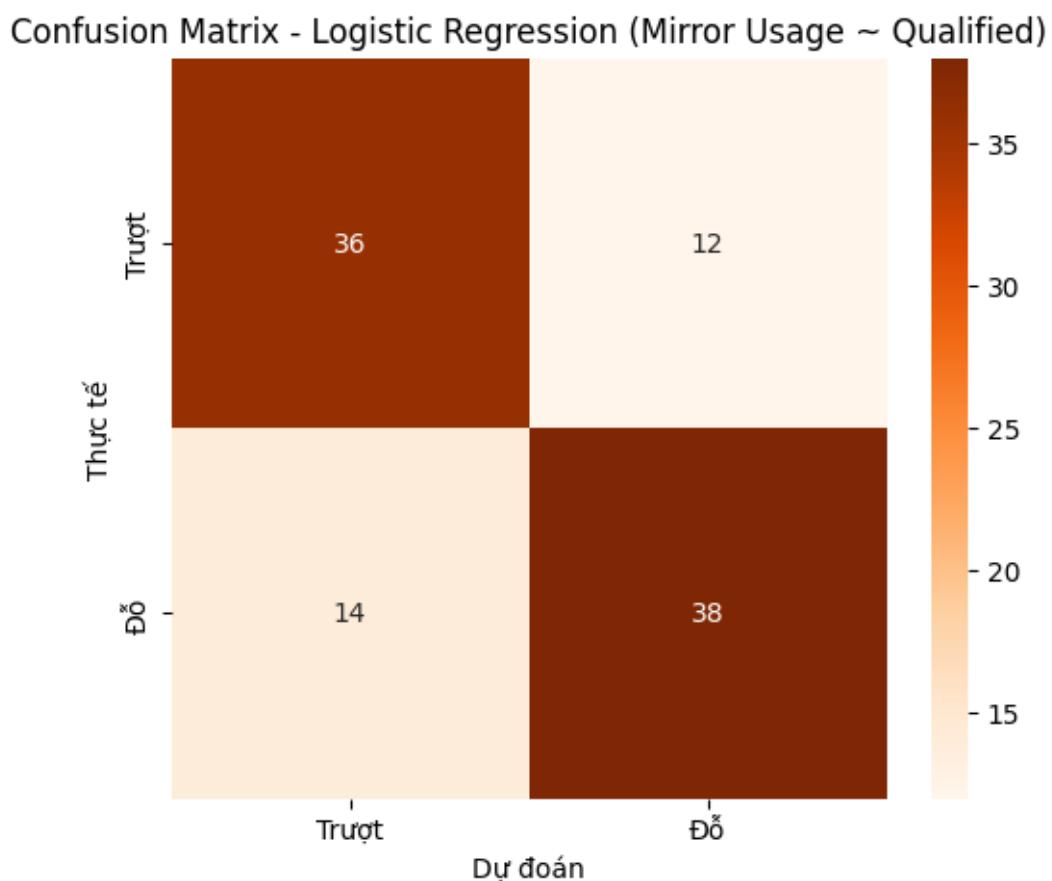
`[[36 12]`

`[14 38]]`

Classification Report:

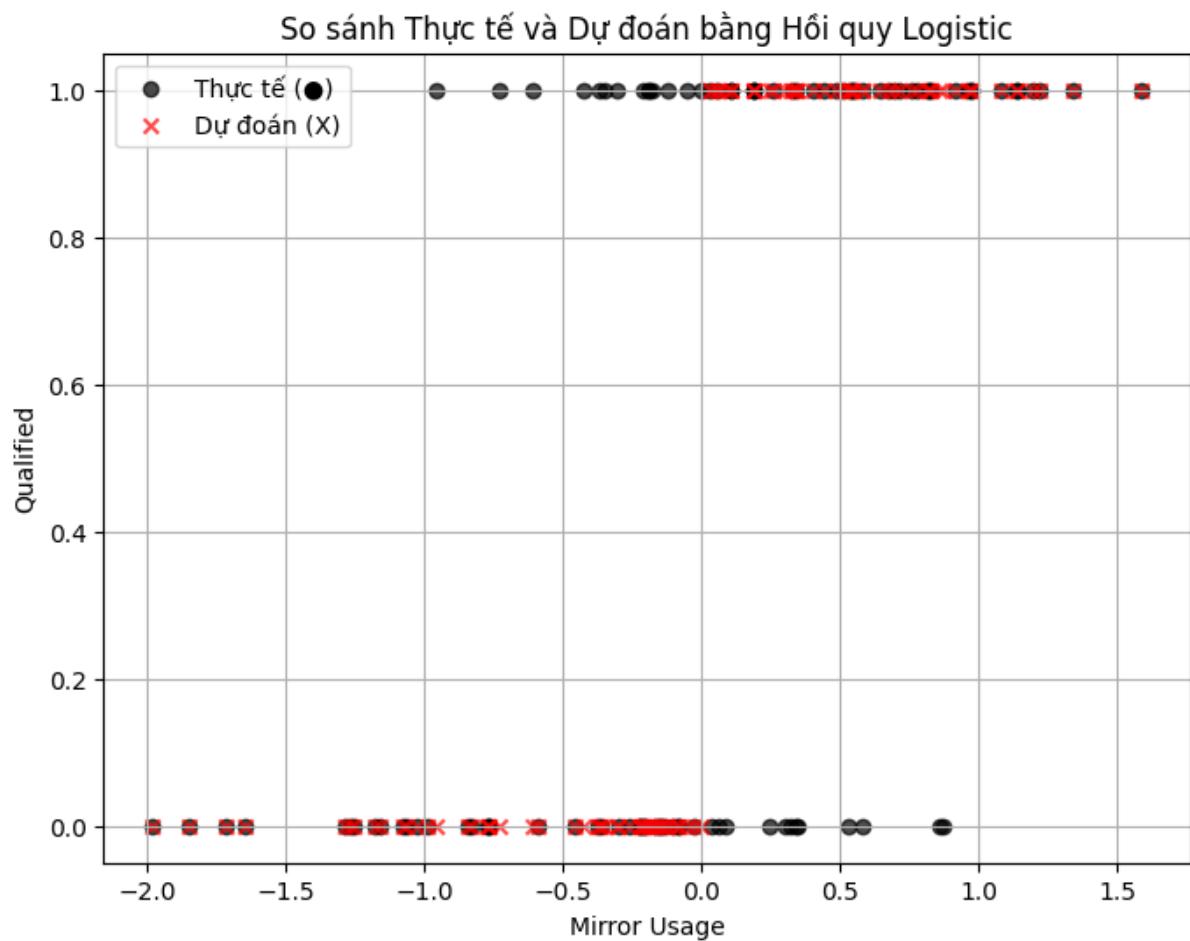
	precision	recall	f1-score	support
0	0.72	0.75	0.73	48
1	0.76	0.73	0.75	52
accuracy			0.74	100
macro avg	0.74	0.74	0.74	100
weighted avg	0.74	0.74	0.74	100

ROC AUC Score: 0.8171073717948718



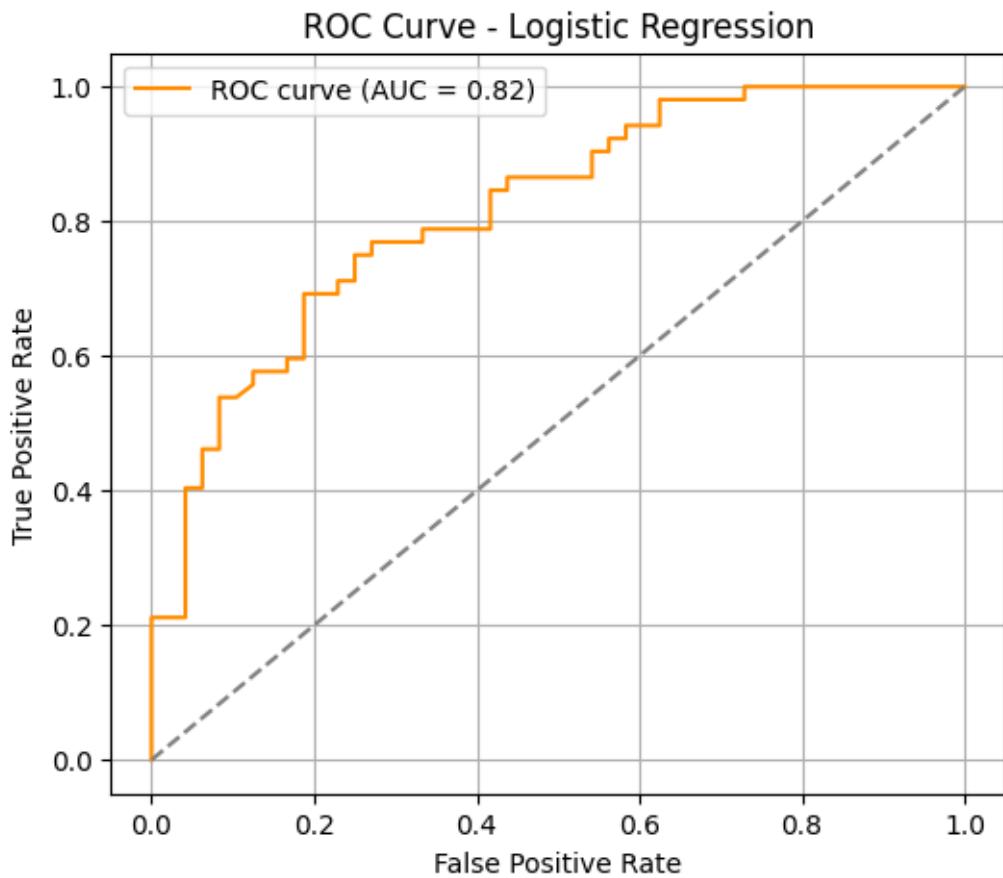
Hình 21: Confusion Matrix – Logistics Regression (Mirror Usage ~ Qualified) – Trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 22: So sánh thực tế - dự đoán Logistics Regression ( $\text{Mirror Usage} \sim \text{Qualified}$ ) – Trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 23: ROC Curve - Logistics Regression (Mirror Usage ~ Qualified) –  
Trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 30: Nhận xét Logistics Regression – Dữ liệu trước khi loại bỏ outliers

Tiêu chí	Lớp 0 (Rót)	Lớp 1 (Đậu)	Nhận xét tổng quát
Precision	0.72	0.76	Mô hình phân loại người <b>đậu</b> chính xác hơn người <b>rót</b> .
Recall	0.75	0.73	Mô hình phát hiện đúng người <b>rót</b> cao hơn người <b>đậu</b> .
F1 Score	0.73	0.75	Người đậu có F1-score cao hơn → dự đoán ổn định hơn.
Support (số mẫu)	48	52	Phân bổ dữ liệu cân bằng giữa hai nhóm.
Accuracy	0,74		Mô hình chính xác cao.

<b>ROC AUC Score</b>	0,8171	Khả năng phân biệt người đậu và rót tốt.
<b>Confusion Matrix</b>	[[36, 12], [14, 38]]	14 người đậu bị đoán nhầm là rót, 12 người rót bị đoán là đậu.

(Nguồn: Tác giả tổng hợp, 2025)

=> Mô hình cân bằng, hiệu quả cao, phù hợp cho mục tiêu sàng lọc thí sinh.

### B.3.2. Dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Lấy biến đầu vào và mục tiêu
X = df[['Mirror Usage']]
y = df['Qualified']

# Chia dữ liệu train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1] # Xác suất dự đoán lớp 1

# === In các chỉ số ===
# Các chỉ số riêng
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=0)
recall = recall_score(y_test, y_pred, zero_division=0)
f1 = f1_score(y_test, y_pred, zero_division=0)

print("Đánh giá chi tiết:")
print(f"Accuracy : {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall   : {recall:.2f}")
print(f"F1 Score : {f1:.2f}")
print(f"ROC AUC  : {roc_auc:.2f}")

# Chỉ số ma trận nhầm lẫn
cm = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("Confusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("ROC AUC Score:", roc_auc)

# === Vẽ Confusion Matrix ===
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges', xticklabels=['Trượt', 'Đỗ'], yticklabels=['Trượt', 'Đỗ'])
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - Logistic Regression (Mirror Usage ~ Qualified)')
plt.show()
```

```
# === Vẽ biểu đồ:
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='black', label='Thực tế (●)', alpha=0.7)
plt.scatter(X_test, y_pred, color='red', marker='x', label='Dự đoán (X)', alpha=0.7)
plt.xlabel('Mirror Usage')
plt.ylabel('Qualified')
plt.title('So sánh Thực tế và Dự đoán bằng Hồi quy Logistic')
plt.legend()
plt.grid(True)
plt.show()

# === Vẽ ROC curve ===
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Logistic Regression')
plt.legend()
plt.grid(True)
plt.show()
```

Đánh giá chi tiết:

Accuracy : 0.66

Precision: 0.57

Recall : 0.75

F1 Score : 0.65

ROC AUC : 0.82

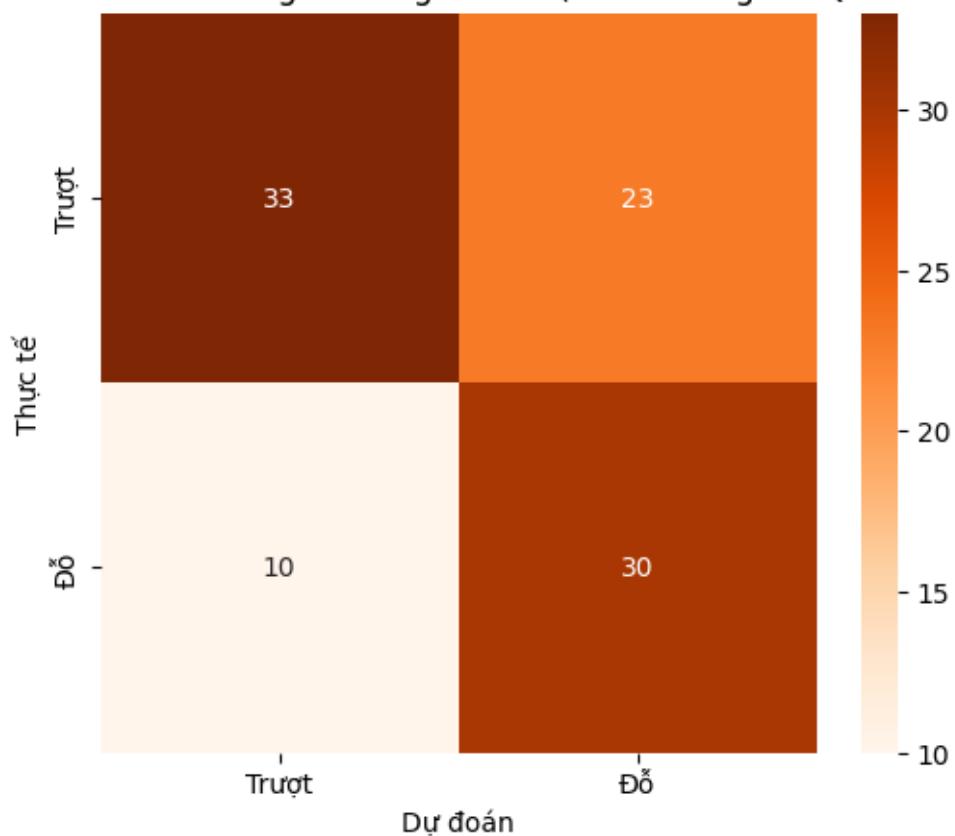
Confusion Matrix:

```
[[33 23]
 [10 30]]
```

Classification Report:

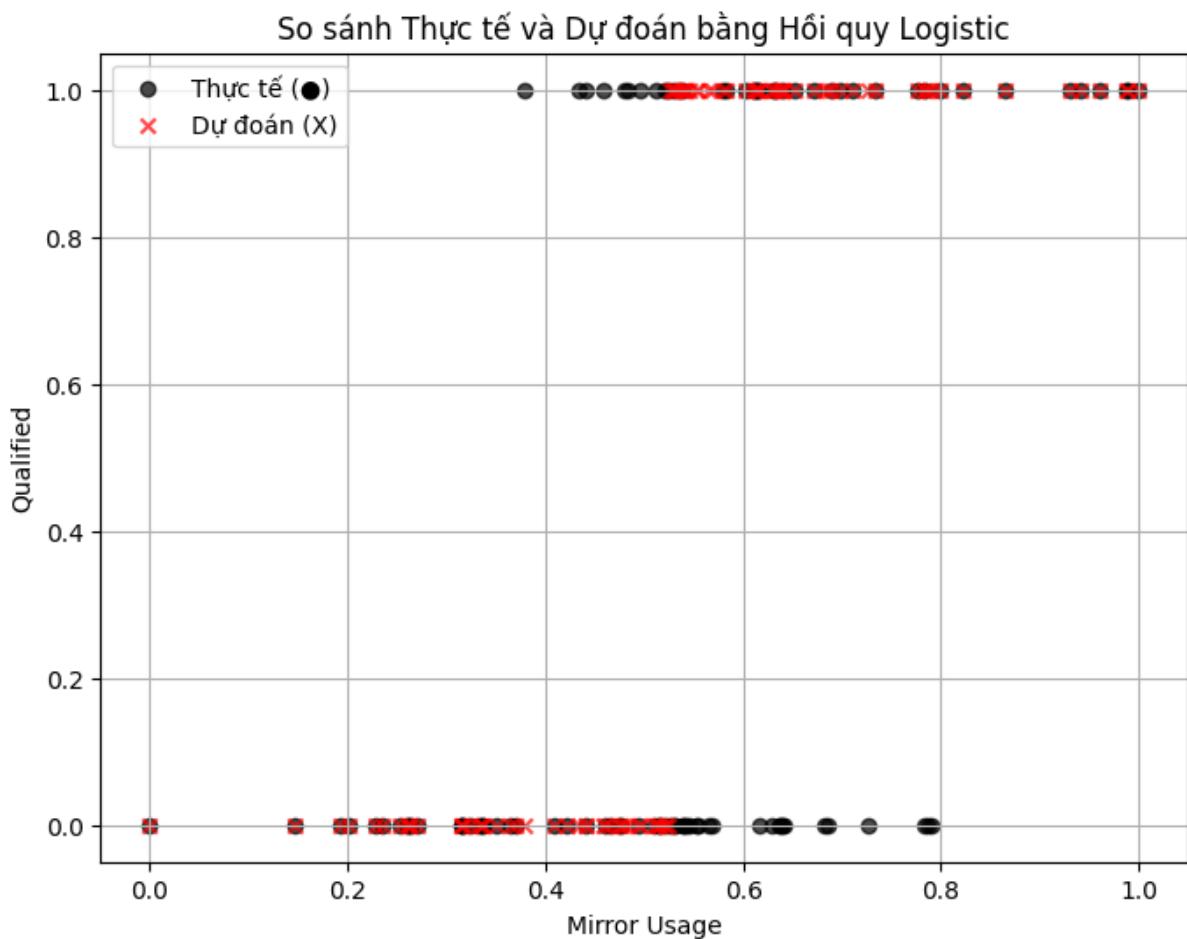
	precision	recall	f1-score	support
0	0.77	0.59	0.67	56
1	0.57	0.75	0.65	40
accuracy			0.66	96
macro avg	0.67	0.67	0.66	96
weighted avg	0.68	0.66	0.66	96

Confusion Matrix - Logistic Regression (Mirror Usage ~ Qualified)



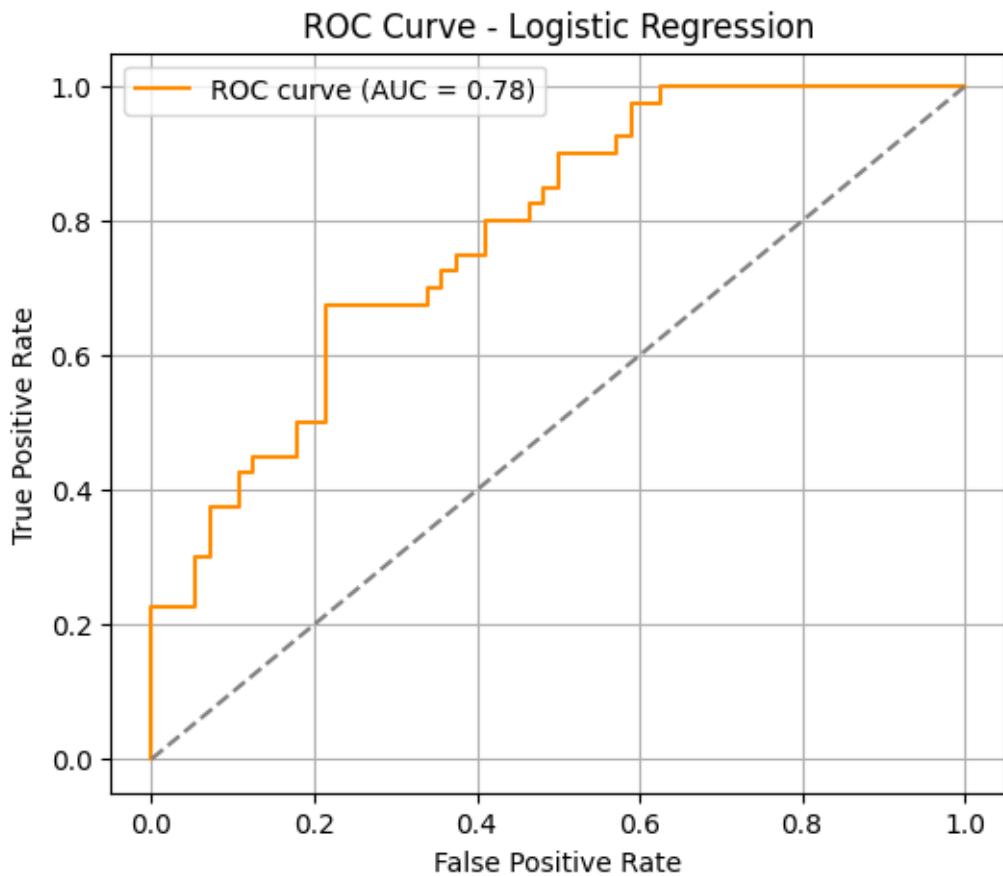
Hình 24: Confusion Matrix – Logistics Regression (Mirror Usage – Qualified) –  
Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 25: So sánh thực tế - dự đoán – Logistics Regression (Mirror Usage – Qualified) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 26: ROC Curve – Logistics Regression (Mirror Usage – Qualified) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 31: Nhận xét Logistics Regression – Dữ liệu sau khi loại bỏ outliers

Tiêu chí	Lớp 0 (Rót)	Lớp 1 (Đậu)	Nhận xét tổng quát
Precision	0.77	0.57	Dự đoán người rót vẫn chính xác cao, nhưng <b>dễ nhầm người đậu thành rót</b> .
Recall	0.59	0.75	Mô hình <b>ưu tiên phát hiện người đậu</b> , nhưng <b>bỏ sót người rót</b> nhiều hơn.
F1 Score	0.67	0.65	Hiệu suất hai lớp giảm, kém ổn định so với mô hình ban đầu.
Support (số mẫu)	56	40	Tỷ lệ người rót cao hơn → dữ liệu mất cân bằng nhẹ.
Accuracy	0,66		Độ chính xác giảm đáng kể.

<b>ROC AUC Score</b>	0,7804	Khả năng phân biệt hai nhóm vẫn ổn nhưng kém hơn trước.
<b>Confusion Matrix</b>	$[[33, 23], [10, 30]]$	23 người rót bị đoán sai là đậu, 10 người đậu bị đoán sai là rót.

(Nguồn: Tác giả tổng hợp, 2025)

=> Mô hình ưu tiên phát hiện người đậu, nhưng dự đoán không còn đáng tin cậy như trước.

#### ❖ Nhận xét:

Kết quả cho thấy mô hình hồi quy logistic đạt hiệu suất cao hơn khi sử dụng toàn bộ dữ liệu gốc. Trước khi loại bỏ outliers, các chỉ số precision, recall và F1-score đều ở mức cao và cân bằng giữa hai lớp “Đậu” và “Rót”. Mô hình đạt độ chính xác 0.74 và chỉ số ROC AUC 0.8171, phản ánh khả năng phân loại tốt giữa hai nhóm thí sinh.

Ngược lại, sau khi loại bỏ outliers, độ chính xác tổng thể giảm còn 0.66 và chỉ số ROC AUC cũng giảm nhẹ. Precision của lớp “Đậu” giảm rõ rệt (từ 0.76 xuống 0.57), cho thấy mô hình trở nên kém chắc chắn hơn trong việc xác định đúng các thí sinh đủ điều kiện. Dù recall của lớp “Đậu” vẫn duy trì ở mức cao (0.75), sự mất cân đối giữa precision và recall ảnh hưởng đến độ tin cậy của mô hình.

Nhìn chung, việc loại bỏ outliers trong trường hợp này không mang lại cải thiện về hiệu suất phân loại. Mô hình ban đầu hoạt động ổn định hơn, cân bằng giữa độ chính xác và khả năng phân biệt hai nhóm thí sinh, do đó là lựa chọn phù hợp hơn trong bối cảnh đánh giá kết quả thi bằng lái xe.

## PHẦN C: THUẬT TOÁN KNN, SVM

### C.1. Thuật toán K-nearest Neighbors (KNN)

#### C.1.1. Dữ liệu trước khi loại bỏ outliers

```
# ===== 1. ĐỌC & TIỀN XỬ LÝ DỮ LIỆU =====
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Chọn biến đầu vào và mục tiêu
X_scaled = df.drop(columns=['Training', 'Applicant ID', 'Race_White', 'Race_Black', 'Race_Other',
                             'Age Group', 'Gender', 'Theory Test', 'Reactions']).values
y = df['Training'].values # 0 = Unknown, 1 = Basic, 2 = Advanced

labels_text = ['Unknown', 'Basic', 'Advanced']
colors = ['#4CAF50', '#FFEB3B', '#F44336']
cmap = ListedColormap(colors)

# ===== 3. CHIA TRAIN/TEST =====
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# ===== 4. HUẤN LUYỆN KNN =====
knn = KNeighborsClassifier(n_neighbors=18)
knn.fit(X_train, y_train)

# ===== 5. ĐÁNH GIÁ =====
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print(f"Độ chính xác KNN trên tập test: {acc:.2f}%\n")
print("Báo cáo phân loại:")
print(classification_report(y_test, y_pred, target_names=labels_text))
```

```
# ===== 6. PCA CHO TRỰC QUAN HÓA =====
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
X_test_pca = pca.transform(X_test)

# ===== 7. VẼ RẠNH GIỚI QUYẾT ĐỊNH =====
h = 0.05
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))

grid_points_pca = np.c_[xx.ravel(), yy.ravel()]
grid_points_original = pca.inverse_transform(grid_points_pca)
Z = knn.predict(grid_points_original)
Z = Z.reshape(xx.shape)

plt.figure(figsize=(14, 10))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)

for i in range(3):
    mask = y == i
    plt.scatter(X_pca[mask, 0], X_pca[mask, 1], c=colors[i],
                label=labels_text[i], edgecolors='k', s=50)
```

```
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('KNN Decision Boundary in PCA Space (All Features)')
plt.legend()
plt.tight_layout()
plt.show()

# ===== 8. MA TRẬN NHẦM LẦN =====
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=labels_text, yticklabels=labels_text)
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - KNN')
plt.tight_layout()
plt.show()

# ===== 9. ROC & AUC (MULTICLASS) =====
y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

classifier = OneVsRestClassifier(KNeighborsClassifier(n_neighbors=18))
classifier.fit(X_train, y_train)
y_score = classifier.predict_proba(X_test)
```

```

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(8, 6))
roc_colors = ['darkorange', 'green', 'blue']
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=roc_colors[i], lw=2,
              label=f'ROC lớp {labels_text[i]} (AUC = {roc_auc[i]:.2f})')

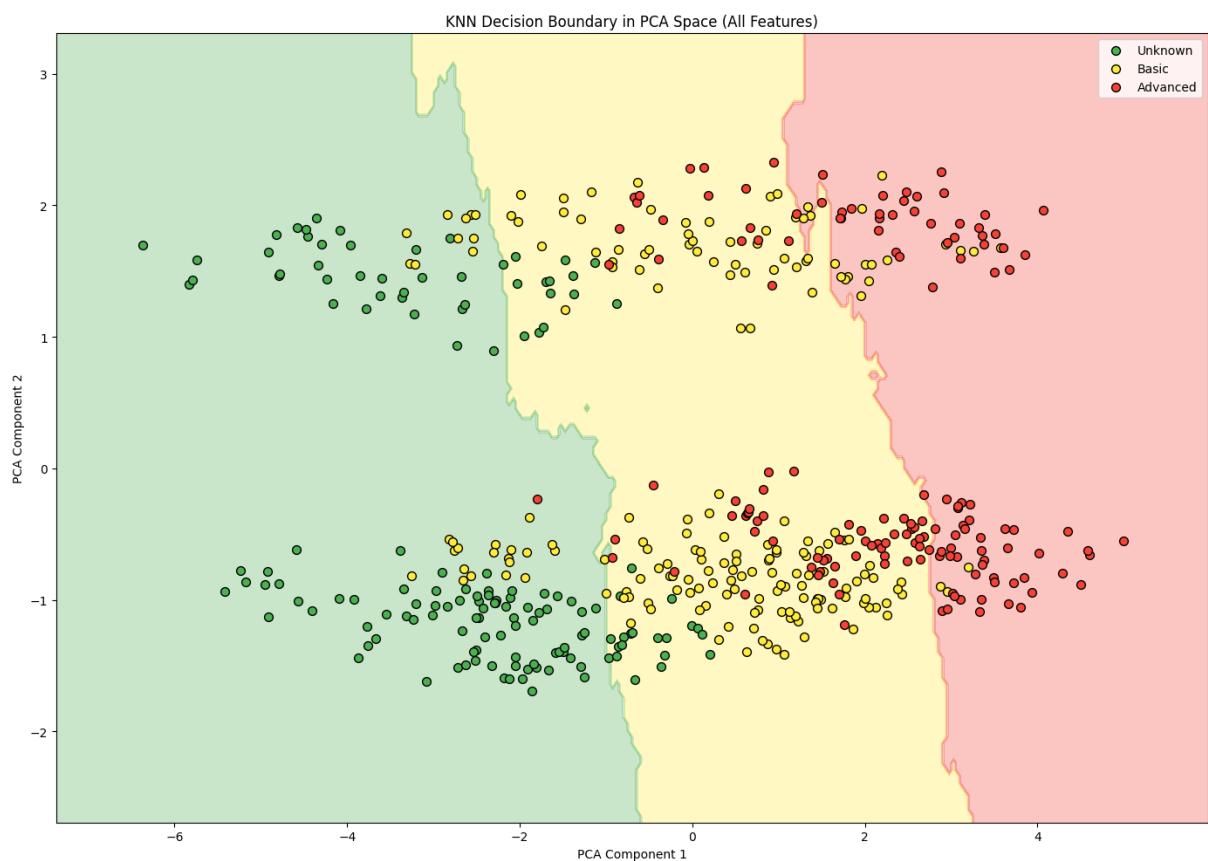
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tỷ lệ dương giả (False Positive Rate)')
plt.ylabel('Tỷ lệ dương thật (True Positive Rate)')
plt.title('ROC Curve - KNN (Multiclass)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()

```

Độ chính xác KNN trên tập test: 76.00%

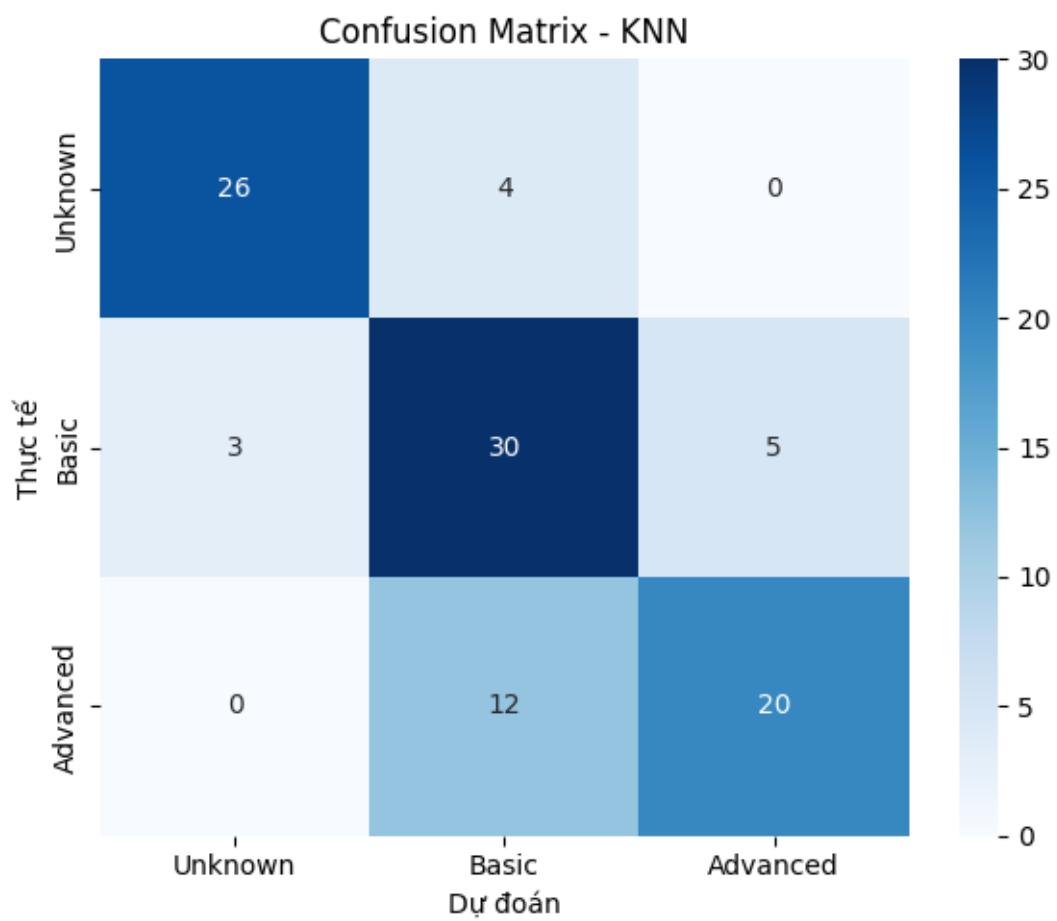
Báo cáo phân loại:

	precision	recall	f1-score	support
Unknown	0.90	0.87	0.88	30
Basic	0.65	0.79	0.71	38
Advanced	0.80	0.62	0.70	32
accuracy			0.76	100
macro avg	0.78	0.76	0.77	100
weighted avg	0.77	0.76	0.76	100



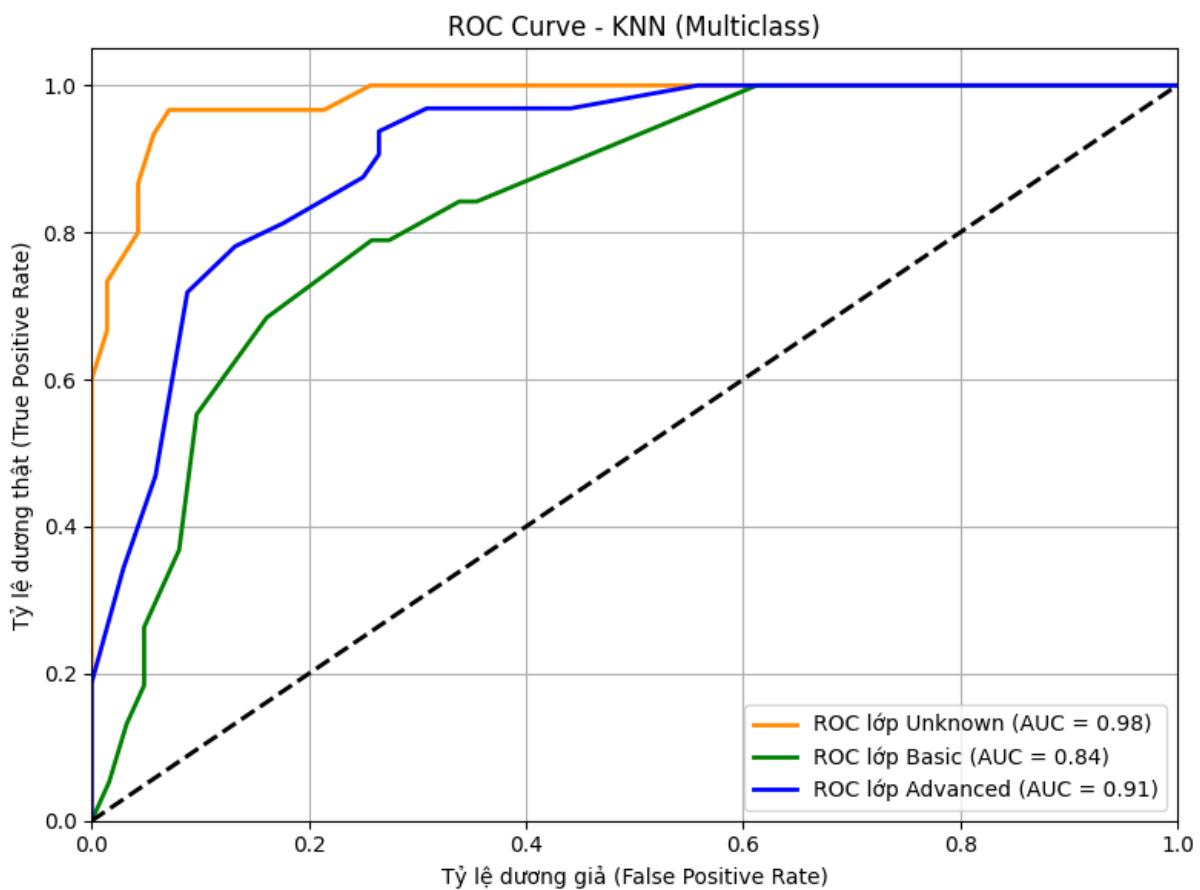
Hình 27: Biểu đồ ranh giới quyết định cho thuật toán KNN với biến đầu vào là tất cả các đặc trưng – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 28: Confusion Matrix – KNN (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 29: ROC Curve – KNN (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 32: Nhận xét thuật toán KNN – Dữ liệu trước khi loại bỏ outliers

Tiêu chí	Nhận xét
<b>Độ chính xác</b>	Mô hình đạt <b>76.00%</b> , cho thấy KNN hoạt động ổn định trên dữ liệu gốc. Đây là kết quả khá tốt đối với mô hình đơn giản như KNN.
<b>F1-score</b>	Lớp Unknown có $F1 = 0.88$ , vượt trội do đặc điểm phân bố riêng biệt, ít bị chòng lấn. Trong khi đó, Basic (0.71) và Advanced (0.70) bị ảnh hưởng bởi việc các mẫu gần nhau trên ranh giới phân lớp.
<b>Ma trận nhầm lẫn</b>	Lớp Advanced bị nhầm nhiều sang Basic (12 mẫu), cho thấy khoảng cách giữa 2 lớp này nhỏ trong không gian đặc trưng.

<b>Biểu đồ PCA</b>	Các điểm dữ liệu phân bố rộng nhưng tách biệt rõ theo chiều ngang. Ranh giới quyết định của KNN tạo thành ba vùng khá rõ, với vùng Basic nằm ở giữa hai lớp còn lại.
<b>Đường cong ROC - AUC</b>	Lớp Unknown có AUC = <b>0.98</b> , gần đạt mức hoàn hảo, chứng tỏ khả năng phân biệt rõ ràng. Advanced và Basic có AUC lần lượt là <b>0.91</b> và <b>0.84</b> , cho thấy mức phân biệt vẫn còn chưa thật sự tách bạch.

(Nguồn: Tác giả tổng hợp, 2025)

### C.1.2. Dữ liệu sau khi loại bỏ outliers

```
# ===== 1. ĐỌC & TIỀN XỬ DỮ LIỆU =====
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Chọn biến đầu vào và mục tiêu
X_scaled = df.drop(columns=['Training', 'Applicant ID', 'Race_White', 'Race_Black', 'Race_Other',
                             'Age Group', 'Gender', 'Theory Test', 'Reactions']).values
y = df['Training'].values # 0 = Unknown, 1 = Basic, 2 = Advanced

labels_text = ['Unknown', 'Basic', 'Advanced']
colors = ['#4CAF50', '#FFEB3B', '#F44336']
cmap = ListedColormap(colors)

# ===== 3. CHIA TRAIN/TEST =====
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# ===== 4. HUẤN LUYỆN KNN =====
knn = KNeighborsClassifier(n_neighbors=18)
knn.fit(X_train, y_train)

# ===== 5. ĐÁNH GIÁ =====
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print(f"Độ chính xác KNN trên tập test: {acc:.2f}%\n")
print("Báo cáo phân loại:")
print(classification_report(y_test, y_pred, target_names=labels_text))
```

```
# ===== 6. PCA CHO TRỰC QUAN HÓA =====
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
X_test_pca = pca.transform(X_test)

# ===== 7. VẼ RẠNH GIỚI QUYẾT ĐỊNH =====
h = 0.05
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))

grid_points_pca = np.c_[xx.ravel(), yy.ravel()]
grid_points_original = pca.inverse_transform(grid_points_pca)
Z = knn.predict(grid_points_original)
Z = Z.reshape(xx.shape)

plt.figure(figsize=(14, 10))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)

for i in range(3):
    mask = y == i
    plt.scatter(X_pca[mask, 0], X_pca[mask, 1], c=colors[i],
                label=labels_text[i], edgecolors='k', s=50)
```

```
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('KNN Decision Boundary in PCA Space (All Features)')
plt.legend()
plt.tight_layout()
plt.show()

# ===== 8. MA TRẬN NHẦM LẦN =====
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=labels_text, yticklabels=labels_text)
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - KNN')
plt.tight_layout()
plt.show()

# ===== 9. ROC & AUC (MULTICLASS) =====
y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

classifier = OneVsRestClassifier(KNeighborsClassifier(n_neighbors=18))
classifier.fit(X_train, y_train)
y_score = classifier.predict_proba(X_test)
```

```

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(8, 6))
roc_colors = ['darkorange', 'green', 'blue']
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=roc_colors[i], lw=2,
              label=f'ROC lớp {labels_text[i]} (AUC = {roc_auc[i]:.2f})')

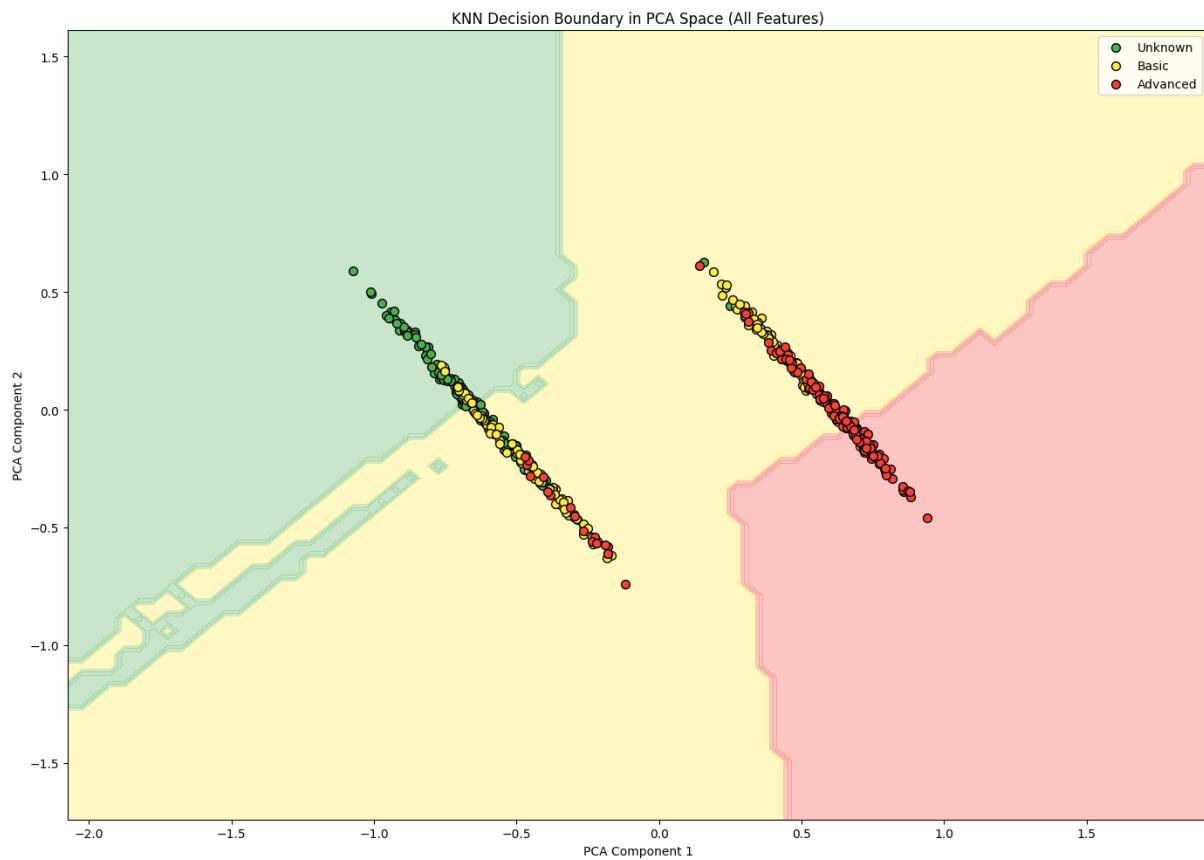
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tỷ lệ dương giả (False Positive Rate)')
plt.ylabel('Tỷ lệ dương thật (True Positive Rate)')
plt.title('ROC Curve - KNN (Multiclass)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()

```

Độ chính xác KNN trên tập test: 71.88%

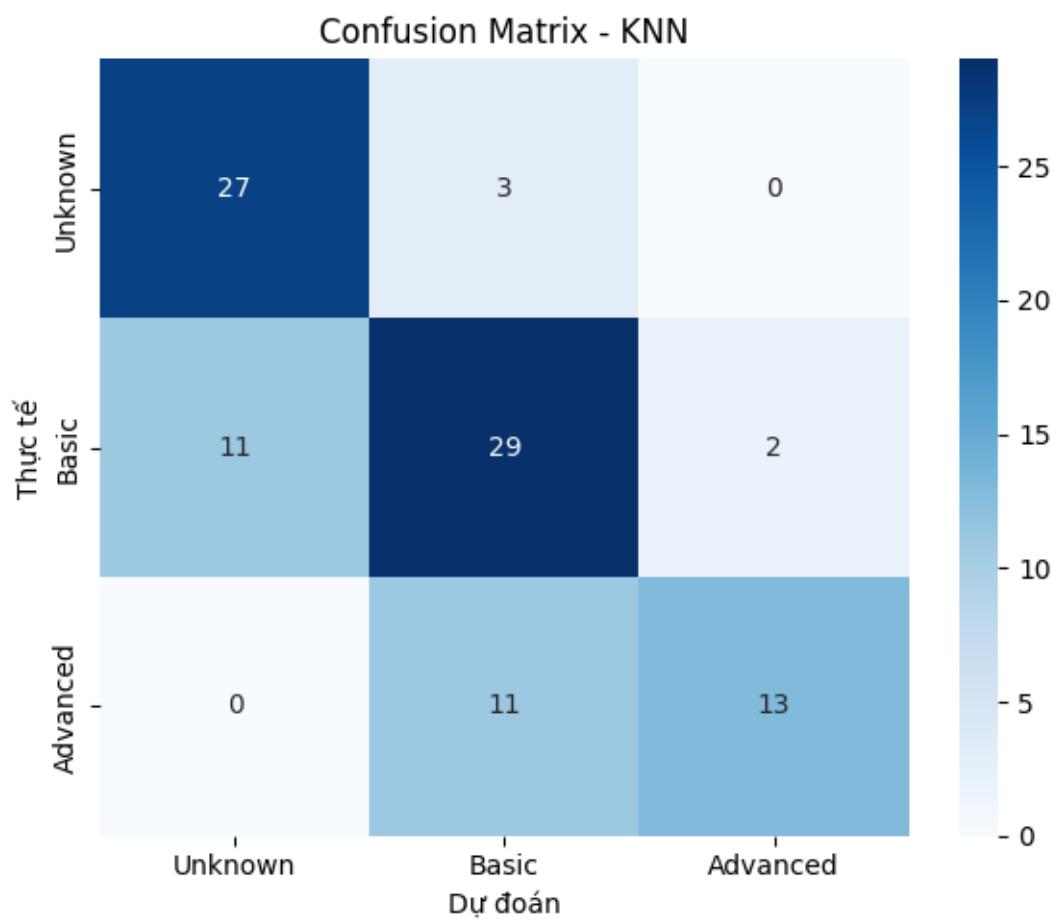
Báo cáo phân loại:

	precision	recall	f1-score	support
Unknown	0.71	0.90	0.79	30
Basic	0.67	0.69	0.68	42
Advanced	0.87	0.54	0.67	24
accuracy			0.72	96
macro avg	0.75	0.71	0.71	96
weighted avg	0.73	0.72	0.71	96



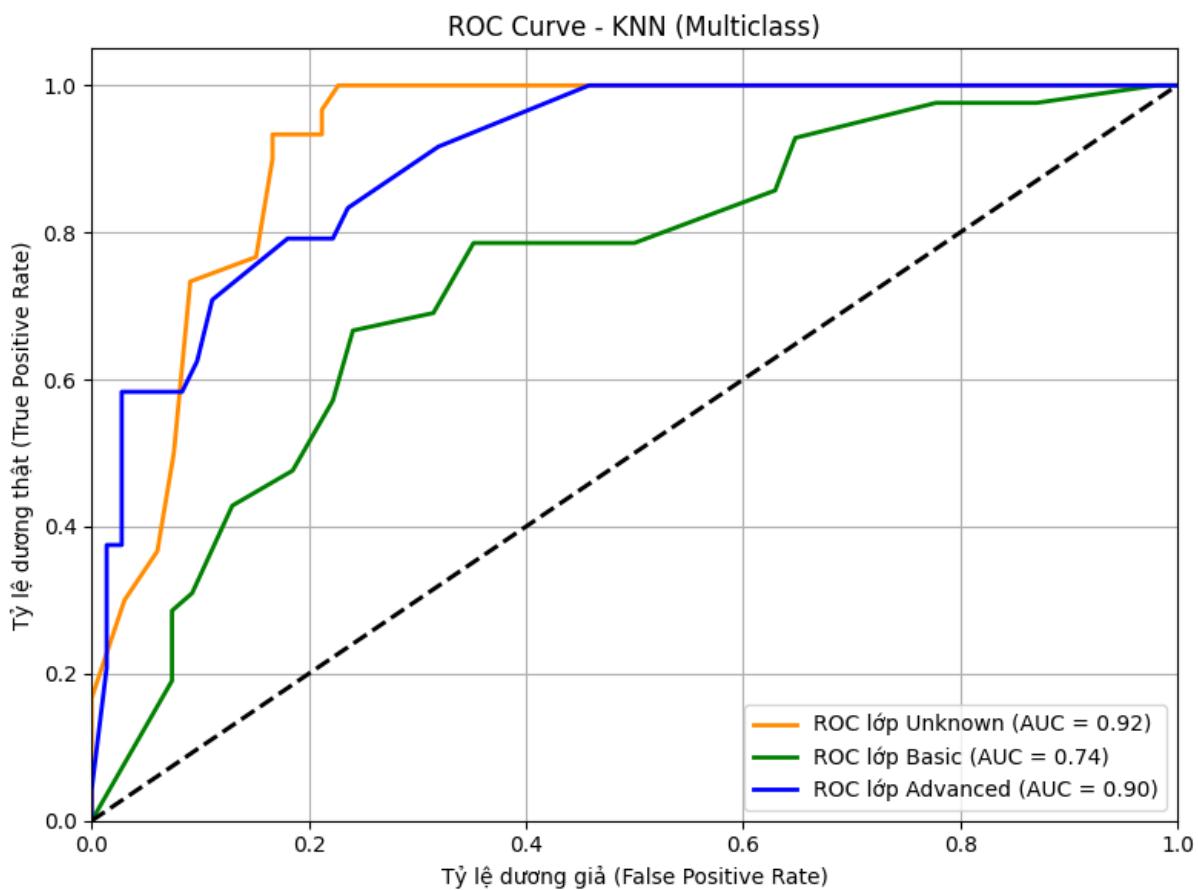
Hình 30: Biểu đồ ranh giới quyết định cho thuật toán KNN với biến đầu vào là tất cả các đặc trưng – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 31: Confusion Matrix – KNN (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 32: ROC – Curve – KNN (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 33: Nhận xét thuật toán KNN – Dữ liệu trước khi lọa bỏ outliers

Tiêu chí	Nhận xét
<b>Độ chính xác</b>	Giảm còn <b>71.88%</b> , chứng tỏ việc loại bỏ outliers không mang lại lợi ích về độ chính xác đối với KNN. Việc mất đi một số điểm biên quan trọng có thể làm giảm khả năng tổng quát hóa của mô hình.
<b>F1-score</b>	Lớp Unknown vẫn giữ vững mức cao (0.79), nhưng Basic và Advanced giảm nhẹ do bị nhầm lẫn lẫn nhau – biểu hiện của việc các lớp này trở nên gần hơn trong không gian đặc trưng sau khi loại bỏ outliers.
<b>Ma trận nhầm lẫn</b>	Advanced bị nhầm sang Basic đến <b>11 mẫu</b> , và Basic cũng nhầm về Unknown. Cho thấy sự xen lẫn tăng lên dù dữ liệu đã được “làm sạch”.

<b>Biểu đồ PCA</b>	Dữ liệu trở nên cô đặc và tạo thành hai dải điểm gần như thẳng hàng. Mặc dù trực quan có vẻ gọn hơn, nhưng điều này khiến việc phân biệt các lớp ở ranh giới trở nên khó khăn do khoảng cách không còn rõ rệt.
<b>Đường cong ROC - AUC</b>	Tất cả AUC đều giảm: Unknown (0.92), Basic (0.74), Advanced (0.90), phản ánh độ phân biệt giảm sau khi loại bỏ các điểm có giá trị ngoại lai.

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Việc loại bỏ các giá trị ngoại lai (outliers) là một bước tiền xử lý phổ biến trong học máy nhằm mục tiêu giảm nhiễu và tăng độ tin cậy cho mô hình. Tuy nhiên, kết quả thực nghiệm trong bài toán phân loại mức độ đào tạo lái xe (Training: Unknown, Basic, Advanced) với mô hình KNN cho thấy rằng việc loại bỏ outliers **không những không cải thiện mà còn làm suy giảm hiệu suất phân loại tổng thể**. Độ chính xác và các chỉ số F1-score, đặc biệt ở lớp Advanced, đều giảm sút so với mô hình huấn luyện trên tập dữ liệu đầy đủ. Điều này được lý giải bởi bản chất của KNN – một mô hình dựa trên khoảng cách – rất nhạy cảm với cấu trúc không gian và sự đa dạng của dữ liệu. Việc loại bỏ những điểm ngoại lai đôi khi **loại bỏ luôn các tín hiệu phân biệt quý giá**, nhất là với các lớp vốn đã có ranh giới mờ nhạt. Do đó, **đối với mục tiêu tối ưu hóa hiệu suất phân loại, đặc biệt với mô hình KNN, việc giữ lại outliers kèm theo các chiến lược bổ sung như điều chỉnh tham số hoặc tăng cường dữ liệu sẽ đem lại kết quả tốt hơn** so với việc loại bỏ hoàn toàn.

## C.2. Thuật toán Support Vector Machine (SVM)

### C.2.1. Dữ liệu trước khi loại bỏ outliers

```
# ===== 1. ĐỌC & TIỀN XỬ LÝ DỮ LIỆU =====
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Lấy biến đầu vào & mục tiêu
X = df.drop(columns=['Training', 'Applicant ID', 'Race_White', 'Race_Black', 'Race_Other',
                      'Age Group', 'Gender', 'Theory Test', 'Reactions']).values
y = df['Training'].values # 0 = Unknown, 1 = Basic, 2 = Advanced

# Danh sách nhãn & màu
labels_text = ['Unknown', 'Basic', 'Advanced']
colors = ['#4CAF50', '#FFEB3B', '#F44336']
cmap = ListedColormap(colors)

# ===== 3. CHIA TẬP TRAIN/TEST =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ===== 4. HUẤN LUYỆN SVM =====
svm = SVC(kernel='rbf', probability=True, random_state=42)
svm.fit(X_train, y_train)

# ===== 5. ĐÁNH GIÁ =====
y_pred = svm.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print(f"Độ chính xác SVM trên tập test: {acc:.2f}%\n")
print("Báo cáo phân loại:")
```

```
# ===== 6. PCA TRỰC QUAN HÓA =====
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
X_test_pca = pca.transform(X_test)

# ===== 7. VẼ RẠNH GIỚI QUYẾT ĐỊNH =====
h = 0.05
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))

grid_points_pca = np.c_[xx.ravel(), yy.ravel()]
grid_points_original = pca.inverse_transform(grid_points_pca)
Z = svm.predict(grid_points_original)
Z = Z.reshape(xx.shape)

plt.figure(figsize=(14, 10))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)

for i in range(3):
    mask = y == i
    plt.scatter(X_pca[mask, 0], X_pca[mask, 1], c=colors[i],
                label=labels_text[i], edgecolors='k', s=50)
```

```

plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('SVM Decision Boundary in PCA Space (All Features)')
plt.legend()
plt.tight_layout()
plt.show()

# ===== 8. MA TRẬN NHẦM LẦN =====
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=labels_text, yticklabels=labels_text)
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - SVM')
plt.tight_layout()
plt.show()

# ===== 9. ROC & AUC (MULTICLASS) =====
y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

classifier = OneVsRestClassifier(SVC(kernel='rbf', probability=True, random_state=42))
classifier.fit(X_train, y_train)
y_score = classifier.predict_proba(X_test)

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(8, 6))
roc_colors = ['darkorange', 'green', 'blue']
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=roc_colors[i], lw=2,
              label=f'ROC lớp {labels_text[i]} (AUC = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tỷ lệ dương giả (False Positive Rate)')
plt.ylabel('Tỷ lệ dương thật (True Positive Rate)')
plt.title('ROC Curve - SVM (Multiclass)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()

```

Độ chính xác SVM trên tập test: 74.00%

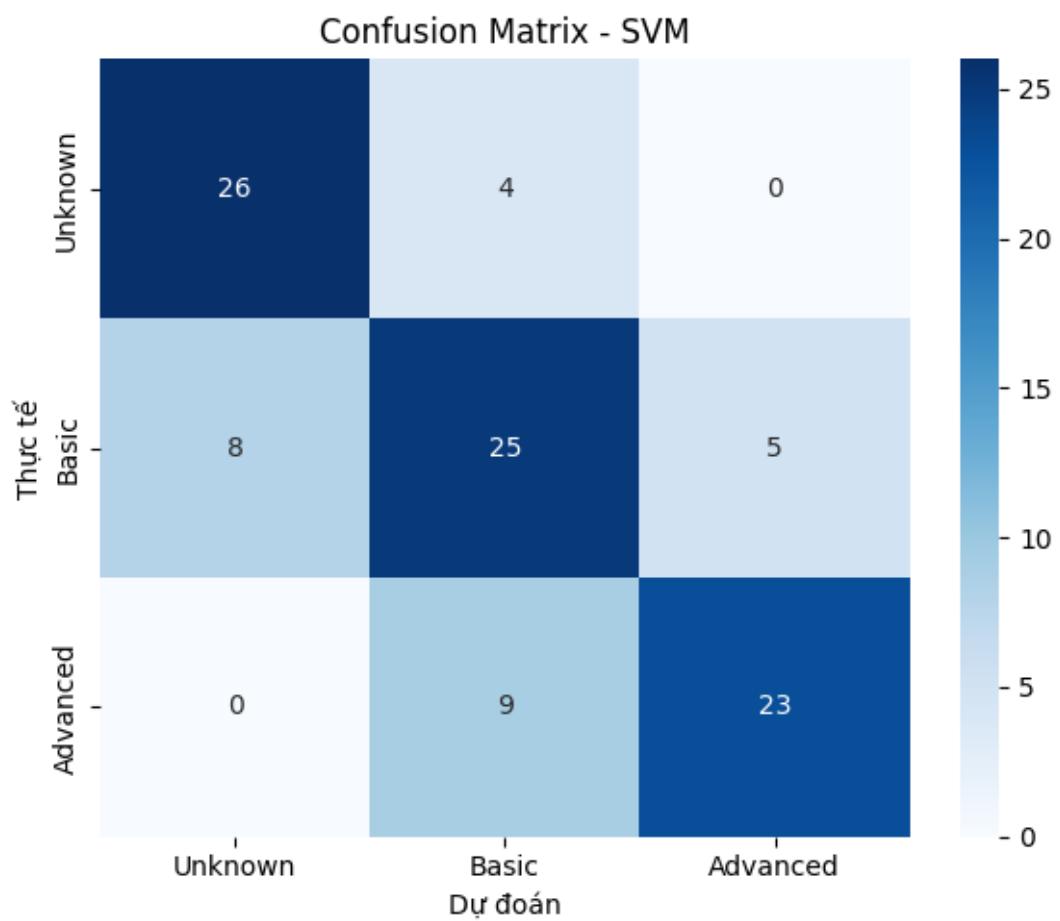
Báo cáo phân loại:

	precision	recall	f1-score	support
Unknown	0.76	0.87	0.81	30
Basic	0.66	0.66	0.66	38
Advanced	0.82	0.72	0.77	32
accuracy			0.74	100
macro avg	0.75	0.75	0.75	100
weighted avg	0.74	0.74	0.74	100



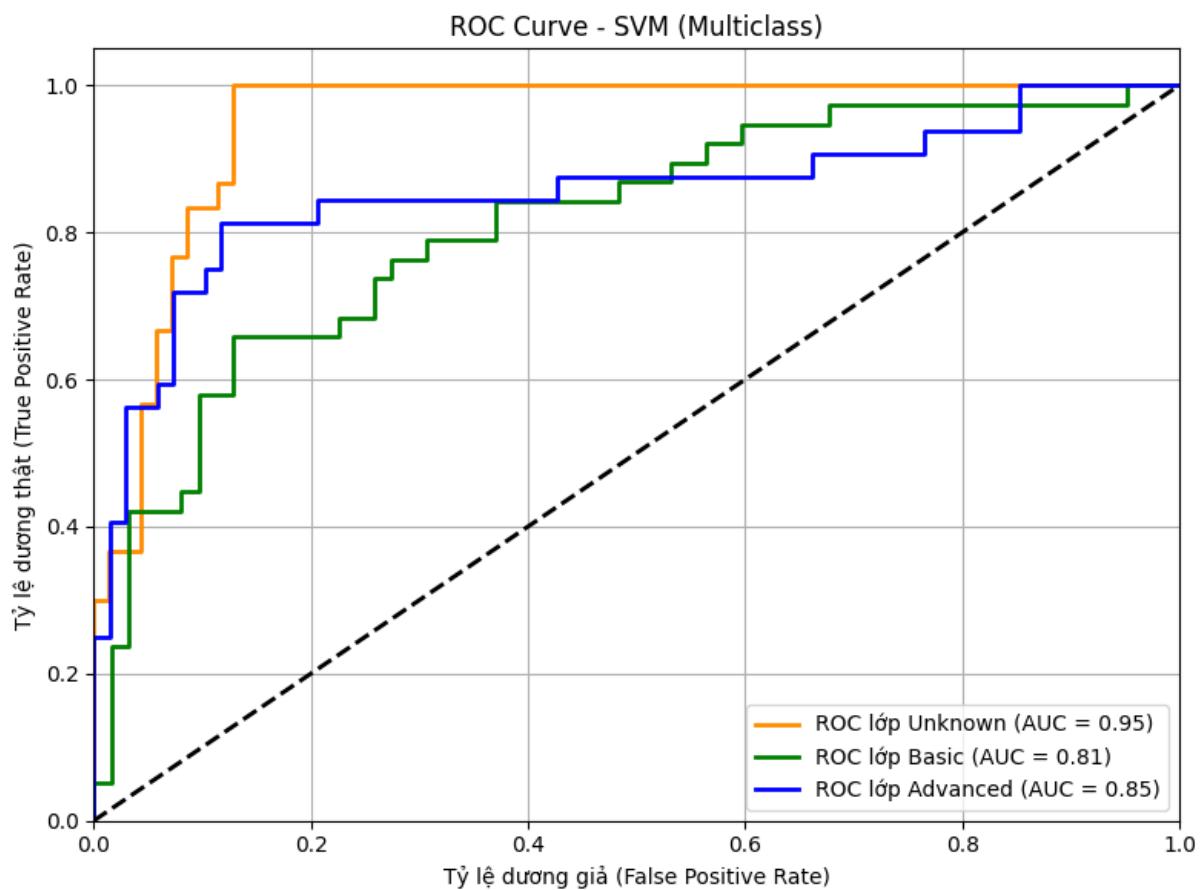
Hình 33: Biểu đồ ranh giới quyết định cho mô hình SVM – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 34: Confusion Matrix – SVM (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 35: ROC Curve – SVM (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 34: Nhận xét thuật toán SVM – Dữ liệu trước khi loại bỏ outliers

Tiêu chí	Nhận xét
<b>Độ chính xác</b>	Đạt 74.00%, cho thấy mô hình SVM có khả năng học tốt trên toàn bộ dữ liệu, kể cả với các điểm ngoại lệ. Điều này phản ánh rằng outliers đôi khi đóng vai trò giúp xác định ranh giới phân loại.
<b>F1-score</b>	Lớp Unknown và Advanced đạt F1 cao (0.81 và 0.77), trong khi Basic chỉ ở mức trung bình (0.66). Việc lớp Basic bị nhầm lẫn với cả hai lớp còn lại cho thấy nó có đặc trưng giao thoa, và mô hình khó định ranh giới cho lớp này.
<b>Mã trận nhầm lẫn</b>	Lớp Basic có đến 13 mẫu bị nhầm (8 sang Unknown, 5 sang Advanced), cho thấy sự không rõ ràng trong không gian phân tách của lớp này. Mô hình phân biệt rõ lớp Unknown và Advanced hơn.

<b>Biểu đồ PCA</b>	Phân bố điểm chồng lấn giữa các lớp, biên phân chia của SVM khá cong và không sắc nét. Điều này thể hiện rõ ảnh hưởng của các điểm outliers trong việc kéo giãn ranh giới phân loại.
<b>Đường cong ROC - AUC</b>	AUC rất cao với lớp Unknown (0.95) và Advanced (0.85), chứng tỏ khả năng phân biệt mạnh. Lớp Basic có AUC thấp hơn (0.81), phù hợp với hiện tượng nhầm lẫn cao.

(Nguồn: Tác giả tổng hợp, 2025)

### C.2.2. Dữ liệu sau khi loại bỏ outliers

```
# ===== 1. ĐỌC & TIỀN XỬ LÝ DỮ LIỆU =====
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Lấy biến đầu vào & mục tiêu
X = df.drop(columns=['Training', 'Applicant ID', 'Race_White', 'Race_Black', 'Race_Other',
                     'Age Group', 'Gender', 'Theory Test', 'Reactions']).values
y = df['Training'].values # 0 = Unknown, 1 = Basic, 2 = Advanced

# Danh sách nhãn & màu
labels_text = ['Unknown', 'Basic', 'Advanced']
colors = ['#4CAF50', '#FFEB3B', '#F44336']
cmap = ListedColormap(colors)

# ===== 3. CHIA TẬP TRAIN/TEST =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ===== 4. HUẤN LUYỆN SVM =====
svm = SVC(kernel='rbf', probability=True, random_state=42)
svm.fit(X_train, y_train)

# ===== 5. ĐÁNH GIÁ =====
y_pred = svm.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print(f"Độ chính xác SVM trên tập test: {acc:.2f}%\n")
print("Báo cáo phân loại:")
print(classification_report(y_test, y_pred, target_names=labels_text))
```

```
# ===== 6. PCA TRỰC QUAN HÓA =====
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
X_test_pca = pca.transform(X_test)

# ===== 7. VẼ RẠNH GIỚI QUYẾT ĐỊNH =====
h = 0.05
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))

grid_points_pca = np.c_[xx.ravel(), yy.ravel()]
grid_points_original = pca.inverse_transform(grid_points_pca)
Z = svm.predict(grid_points_original)
Z = Z.reshape(xx.shape)

plt.figure(figsize=(14, 10))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)

for i in range(3):
    mask = y == i
    plt.scatter(X_pca[mask, 0], X_pca[mask, 1], c=colors[i],
                label=labels_text[i], edgecolors='k', s=50)
```

```

plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('SVM Decision Boundary in PCA Space (All Features)')
plt.legend()
plt.tight_layout()
plt.show()

# ===== 8. MA TRẬN NHẦM LÃN =====
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=labels_text, yticklabels=labels_text)
plt.xlabel('Dự đoán')
plt.ylabel('Thực tế')
plt.title('Confusion Matrix - SVM')
plt.tight_layout()
plt.show()

# ===== 9. ROC & AUC (MULTICLASS) =====
y_test_bin = label_binarize(y_test, classes=[0, 1, 2])
n_classes = y_test_bin.shape[1]

classifier = OneVsRestClassifier(SVC(kernel='rbf', probability=True, random_state=42))
classifier.fit(X_train, y_train)
y_score = classifier.predict_proba(X_test)

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(8, 6))
roc_colors = ['darkorange', 'green', 'blue']
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], color=roc_colors[i], lw=2,
              label=f'ROC lớp {labels_text[i]} (AUC = {roc_auc[i]:.2f})')

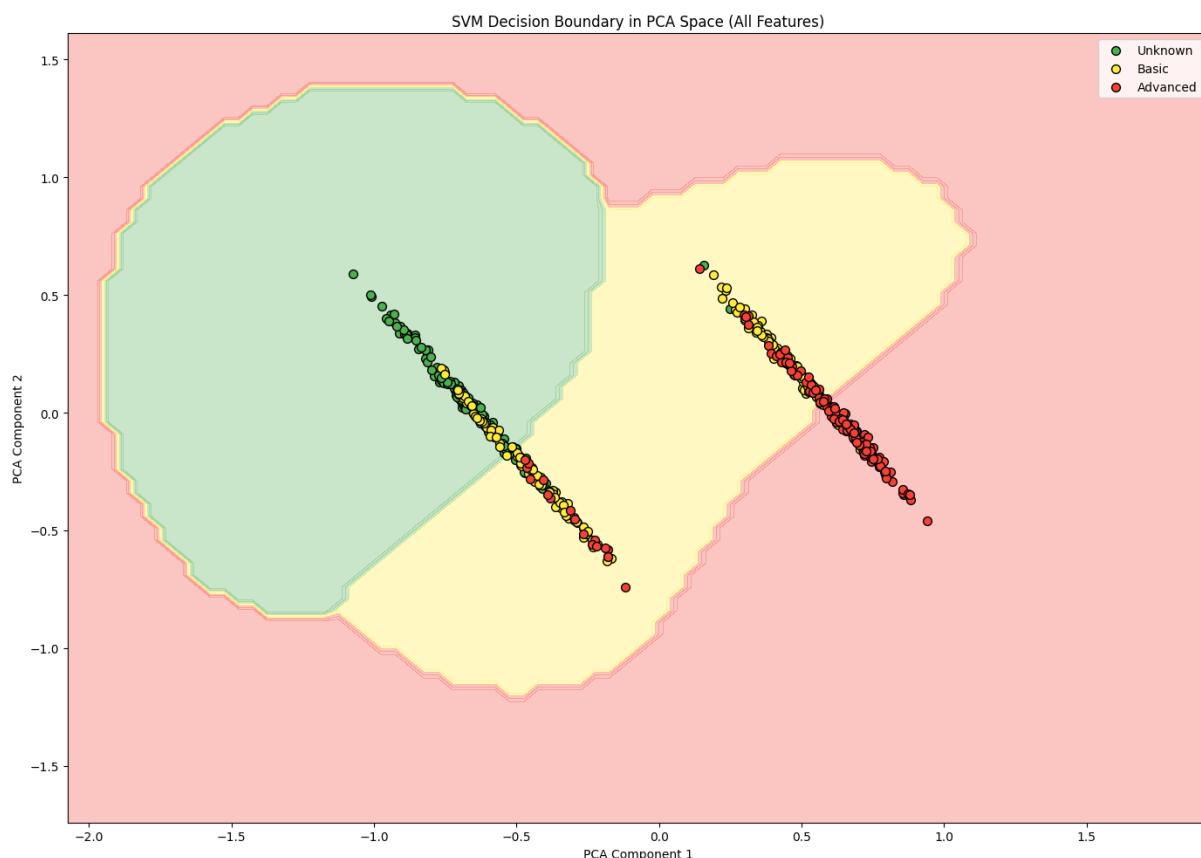
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tỷ lệ dương giả (False Positive Rate)')
plt.ylabel('Tỷ lệ dương thật (True Positive Rate)')
plt.title('ROC Curve - SVM (Multiclass)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()

```

Độ chính xác SVM trên tập test: 68.75%

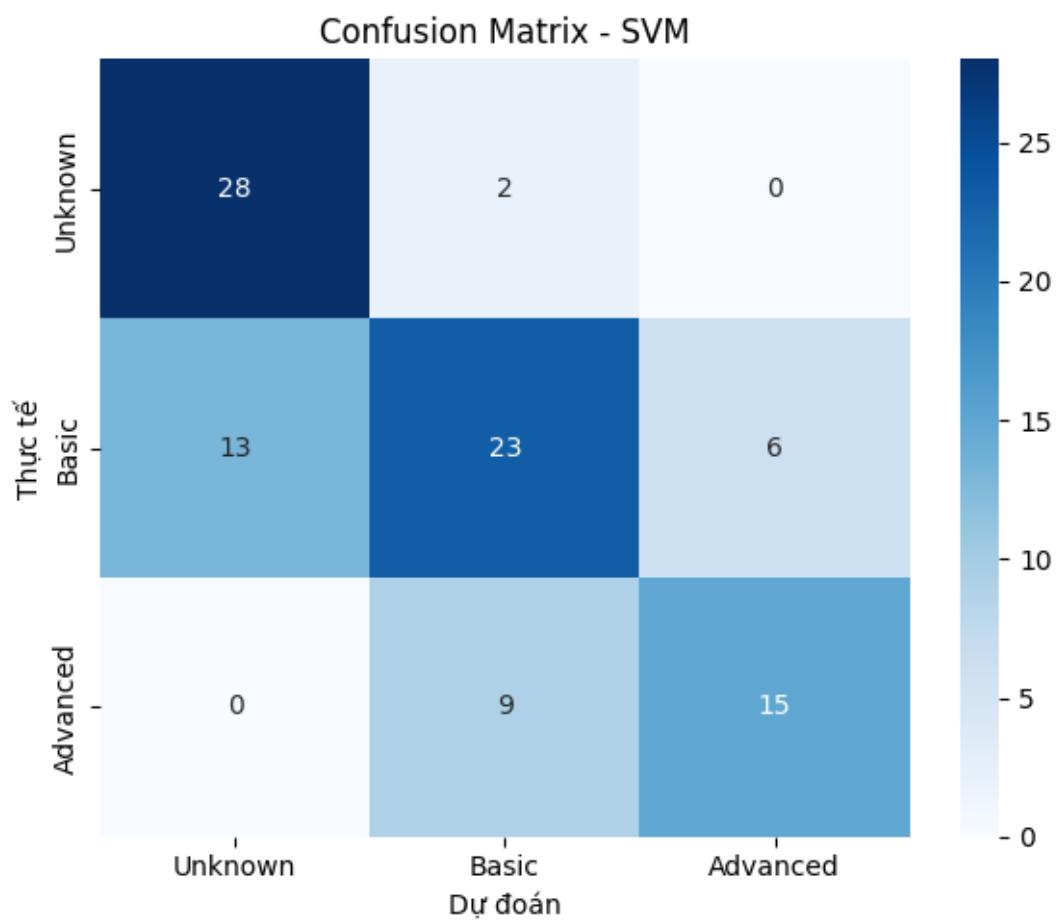
Báo cáo phân loại:

	precision	recall	f1-score	support
Unknown	0.68	0.93	0.79	30
Basic	0.68	0.55	0.61	42
Advanced	0.71	0.62	0.67	24
accuracy			0.69	96
macro avg	0.69	0.70	0.69	96
weighted avg	0.69	0.69	0.68	96



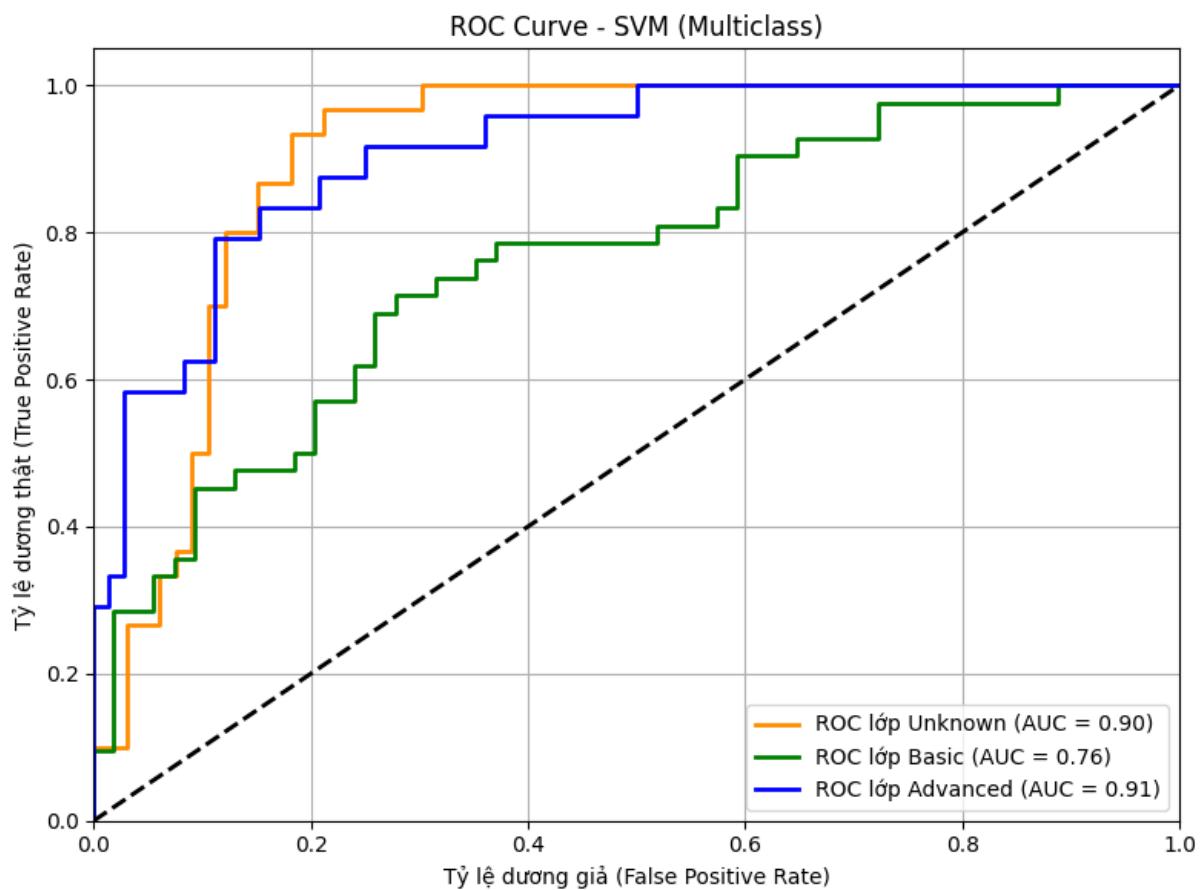
Hình 36: Biểu đồ ranh giới quyết định cho mô hình SVM – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 37: Confusion Matrix – SVM (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)



Hình 38: ROC Curve – SVM (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 35: Nhận xét thuật toán SVM – Dữ liệu sau khi loại bỏ outliers

Tiêu chí	Nhận xét
<b>Độ chính xác</b>	Giảm xuống 68.75%, phản ánh việc loại bỏ outliers không có lợi trong trường hợp này. Một số điểm dữ liệu biên có thể đóng vai trò quan trọng trong xác lập ranh giới phân loại.
<b>F1-score</b>	Lớp Unknown vẫn duy trì mức cao (0.79), nhưng lớp Basic và Advanced đều suy giảm (chỉ còn 0.61 và 0.67). Điều này cho thấy ranh giới giữa các lớp đã trở nên mờ nhạt hơn sau khi loại bỏ các điểm ngoại lệ.
<b>Mã trận nhầm lẫn</b>	Lớp Basic tiếp tục bị nhầm nhiều: 13 mẫu sang Unknown, 6 mẫu sang Advanced. Advanced cũng bị nhầm 9 mẫu sang Basic. Tình trạng nhầm lẫn tăng lên dù dữ liệu đã “làm sạch”, phản ánh hiệu quả phân lớp yếu hơn.

<b>Biểu đồ PCA</b>	Dữ liệu sau khi loại outliers trở nên cô đặc và gần như nằm trên một đường thẳng, dẫn đến việc các biên phân tách mất độ linh hoạt. Dù hình ảnh gọn gàng hơn, nhưng lại làm giảm khả năng mô hình hóa các mối quan hệ phức tạp.
<b>Đường cong ROC – AUC</b>	AUC lớp Basic giảm mạnh (0.76), Unknown và Advanced cũng giảm nhẹ. Cho thấy khả năng phân biệt của mô hình đã yếu đi sau khi loại bỏ outliers – ngược với mong đợi trực quan.

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Việc **loại bỏ outliers** trong bài toán phân loại trình độ lái xe sử dụng mô hình SVM **không mang lại hiệu quả như kỳ vọng**. Mặc dù làm cho dữ liệu trông “sạch” và gọn hơn trong không gian PCA, việc mất đi những điểm biên có thể đã khiến ranh giới phân loại trở nên thiếu linh hoạt và kém tổng quát hóa. Điều này đặc biệt ảnh hưởng đến lớp Basic – lớp vốn đã có ranh giới chồng lấn với các lớp còn lại. Kết quả cho thấy cần thận trọng khi loại bỏ outliers trong các bài toán đa lớp, đặc biệt khi các lớp có đặc trưng phân bố không rõ ràng.

## VI. PHÂN CỤM DỮ LIỆU – UNSUPERVISED LEARNING

### A. THUẬT TOÁN K-MEANS VÀ DBSCAN

#### A.1. Thuật toán K-Means

```
# ĐỌC DỮ LIỆU
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# Loại bỏ cột không cần thiết
X = df.drop(columns=["Applicant ID"])

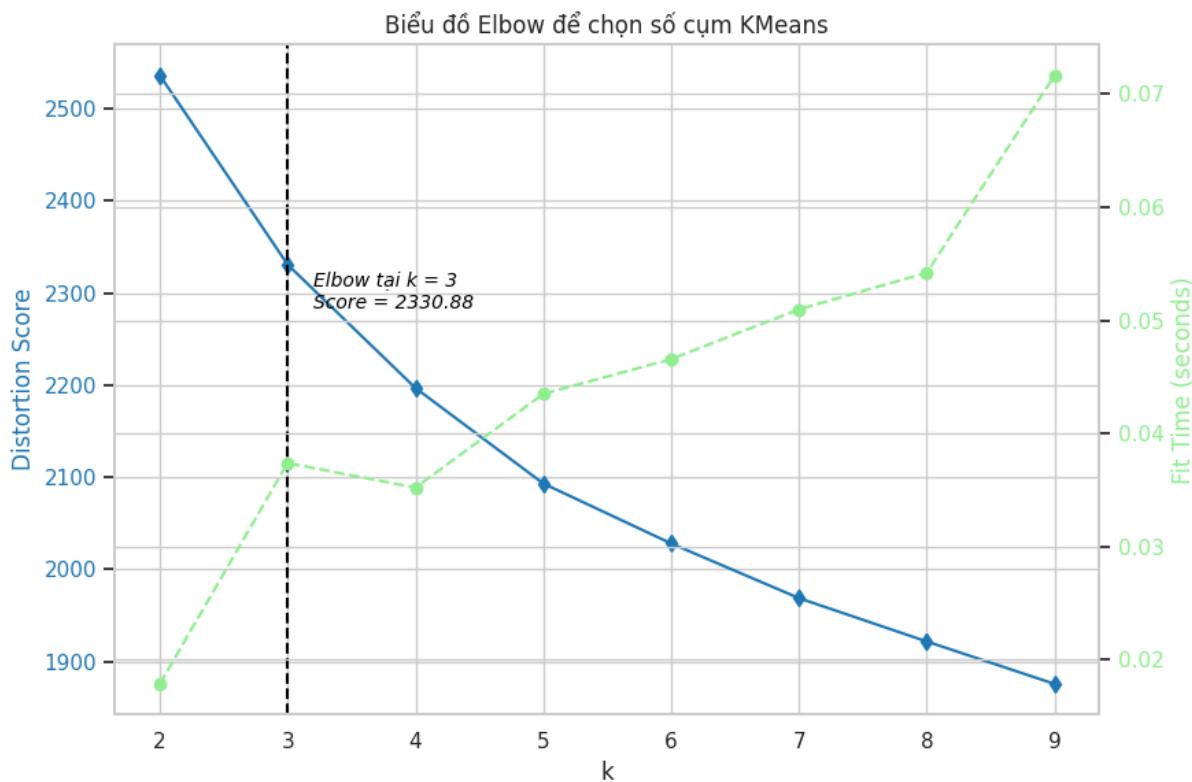
# Áp dụng KMeans
distortions = []
fit_times = []
K = range(2, 10)
for k in K:
    start_time = time.time()
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X)
    distortions.append(kmeans.inertia_)
    fit_times.append(time.time() - start_time)

# Đạo hàm bậc 2 để tìm điểm gấp (elbow)
deltas = np.diff(distortions, 2)
elbow_k = K[np.argmax(deltas) + 1]
elbow_score = distortions[elbow_k - 2]

# Vẽ biểu đồ Elbow
sns.set(style="whitegrid")
fig, ax1 = plt.subplots(figsize=(9, 6))
color = 'tab:blue'
ax1.set_xlabel('k')
ax1.set_ylabel('Distortion Score', color=color)
ax1.plot(K, distortions, marker='d', color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.axvline(x=elbow_k, color='black', linestyle='--')

ax1.text(elbow_k + 0.2, max(distortions)*0.9,
         f"Elbow tại k = {elbow_k}\nScore = {elbow_score:.2f}",
         fontsize=10, style='italic', color='black')

# Thêm trục cho thời gian huấn luyện
ax2 = ax1.twinx()
color = 'lightgreen'
ax2.set_ylabel('Fit Time (seconds)', color=color)
ax2.plot(K, fit_times, marker='o', linestyle='--', color=color)
ax2.tick_params(axis='y', labelcolor=color)
plt.title("Biểu đồ Elbow để chọn số cụm KMeans")
plt.tight_layout()
plt.show()
```



Hình 39: Biểu đồ Elbow để xác định số cụm tối ưu

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

**Mô tả kết quả:**

**Distortion Score:**

- **Trục tung trái** là **Distortion Score**, phản ánh tổng khoảng cách từ mỗi điểm đến tâm cụm gần nhất.
- Giá trị **Distortion giảm dần khi tăng số cụm (k)** – điều này là hiển nhiên vì càng nhiều cụm thì dữ liệu càng chia nhỏ.
  - Tuy nhiên, mức giảm sẽ chậm lại tại một điểm gọi là "elbow" ( $k$  khuỷu) – nơi mà thêm cụm không cải thiện đáng kể hiệu suất phân cụm.

**Fit Time (thời gian huấn luyện):**

- Trục tung phải hiển thị **thời gian huấn luyện** cho từng giá trị  $k$ .
  - Ta thấy rằng thời gian tăng dần theo.
  - Từ  $k > 5$  trở đi, thời gian huấn luyện tăng nhiều.
- => Từ **Distortion Score** và **Fit Time** xác định được số cụm tối ưu nhất là  $k = 3$ .

Là sự cân bằng giữa độ chính xác phân cụm, hiệu suất tính toán và đơn giản mô

hình. Đây là lựa chọn tối ưu giữa độ phức tạp mô hình và chất lượng phân cụm

#### **A.1.1. Phân cụm dựa trên Mirror Usage ~ Steer Control**

##### a. Dữ liệu trước khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Chọn 2 đặc trưng để phân cụm ---
X = df[['Mirror Usage', 'Steer Control']]

# --- 3. KMeans với k = 3 ---
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
labels = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_

# --- 4. Trực quan hóa phân cụm ---
plt.figure(figsize=(9, 7))
colors = ['red', 'green', 'blue']
for i in range(3):
    plt.scatter(X[labels == i]['Mirror Usage'], X[labels == i]['Steer Control'],
                s=30, color=colors[i], alpha=0.8, label=f'Cụm {i}')

# Vẽ tâm cụm
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', color='black', s=100, linewidths=2, label='Tâm cụm')

plt.title("Phân cụm KMeans với k = 3 (Mirror Usage & Steer Control)")
plt.xlabel("Mirror Usage")
plt.ylabel("Steer Control")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Lưu lại kết quả đã gán nhãn
df['Cluster'] = labels
df.to_csv("ket_qua_kmeans_2bien.csv", index=False)
```



Hình 40: Mô hình phân cụm K-Means 2 biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('ket_qua_kmeans_2bien.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

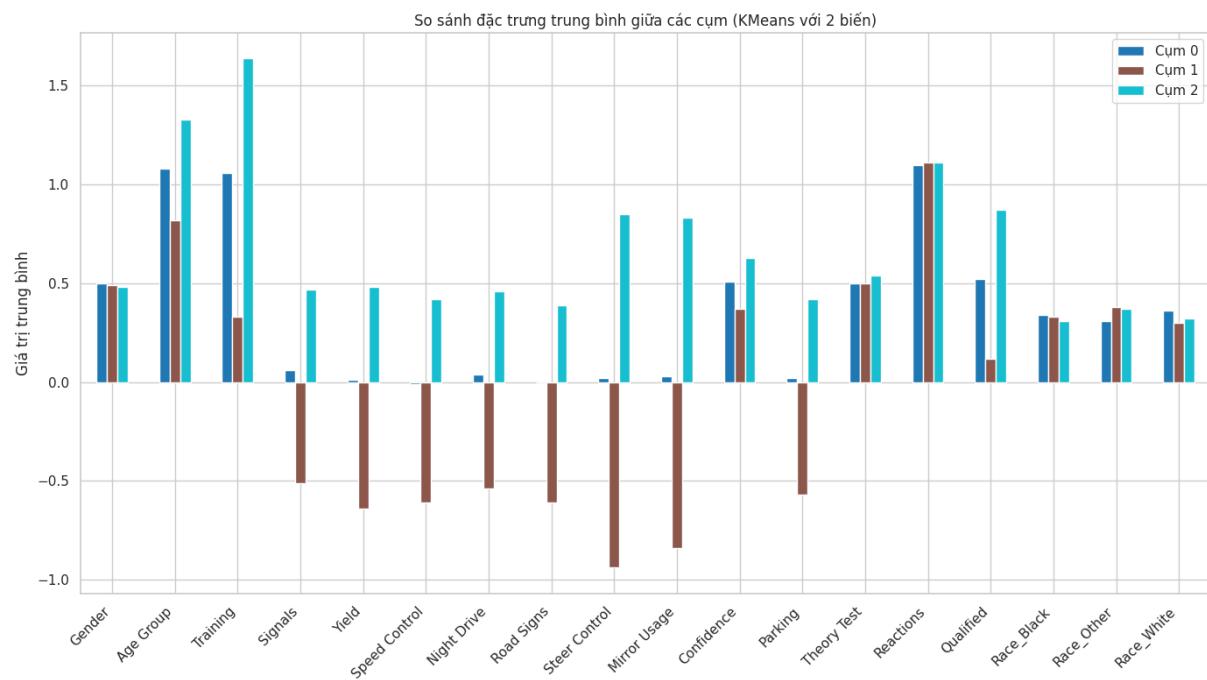
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (KMeans với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 41: So sánh đặc trưng trung bình giữa các cụm (Kmeans với 2 biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 36: Kỹ năng của các cụm trong Kmeans dựa trên 2 biến (Dữ liệu trước khi loại bỏ outliers)

Biến	Cụm 0 (Trung bình)	Cụm 1 (Thấp điểm)	Cụm 2 (Cao điểm)
Signals	Điểm trung bình (~49.6), tương đối ổn định	Thấp nhất (37.6) – Cần cải thiện hiểu biết tín hiệu giao thông	Cao nhất (58.4) – Rất tốt về tín hiệu
Yield	Trung bình (~49.1), có nhận thức tốt về ưu tiên giao thông	Rất thấp (34.8) – Thiếu nhận thức nhường đường	Rất tốt (59.5) – Phản xạ ưu tiên tốt
Speed Control	Trung bình (~49.1), điều khiển tốc độ khá tốt	Kém (36.5) – Khó kiểm soát tốc độ	Rất tốt (58.2) – Điều khiển tốc độ hiệu quả
Night Drive	Trung bình (~49.0), có khả	Yếu (36.6) – Gặp khó khăn khi lái ban đêm	Tốt (58.1) – Tự tin lái xe ban đêm

	năng lái ban đêm ổn		
<b>Road Signs</b>	Khá (49.4) – Nhận diện biển báo ổn	Yếu (35.8) – Khả năng nhận biết biển báo kém	Tốt (57.9) – Nhận diện biển báo tốt
<b>Steer Control</b>	Trung bình (48.6)	Rất yếu (29.2) – <b>Thấp nhất toàn bảng</b> , thiếu kỹ năng điều khiển tay lái	Rất cao (65.3) – <b>Tốt nhất toàn bảng</b> , tay lái vững vàng
<b>Mirror Usage</b>	Khá (48.9), sử dụng gương khá linh hoạt	Rất yếu (31.1) – Không quan sát gương đúng cách	Xuất sắc (65.4) – Quan sát tốt, cẩn trọng
<b>Confidence</b>	Tự tin ở mức khá (48.3)	Thiếu tự tin (36.5) – Có thể dẫn đến phản ứng kém	Rất tự tin (58.7) – Gắn với kỹ năng tốt
<b>Parking</b>	Trung bình tốt (49.8)	Yếu (37.1) – Cần cải thiện kỹ năng đỗ xe	Tốt (58.5) – Đỗ xe thành thạo
<b>Theory Test</b>	Khá (70.2)	Khá (69.8) – Không chênh lệch nhiều	Tốt hơn một chút (72.3) – Hiểu lý thuyết vững hơn

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Cụm 0: **Nhóm trung bình**: không có kỹ năng nào vượt trội, không yếu. Có thể là nhóm học viên ổn định, thi đỗ ở mức vừa phải.

Cụm 1: **Nhóm yếu kém**: điểm thấp toàn diện, đặc biệt yếu ở điều khiển tay lái và sử dụng gương. Cần huấn luyện bổ sung hoặc thi trượt.

Cụm 2: **Nhóm giỏi**: điểm rất cao ở tất cả kỹ năng, đặc biệt về tay lái và quan sát. Tự tin và thành thạo, có thể là nhóm thi đạt điểm cao nhất.

#### b. Dữ liệu sau khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- 2. Chọn 2 đặc trưng để phân cụm ---
X = df[['Mirror Usage', 'Steer Control']]

# --- 3. KMeans với k = 3 ---
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
labels = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_

# --- 4. Trực quan hóa phân cụm ---
plt.figure(figsize=(9, 7))
colors = ['red', 'green', 'blue']
for i in range(3):
    plt.scatter(X[labels == i]['Mirror Usage'], X[labels == i]['Steer Control'],
                s=30, color=colors[i], alpha=0.8, label=f'Cụm {i}')

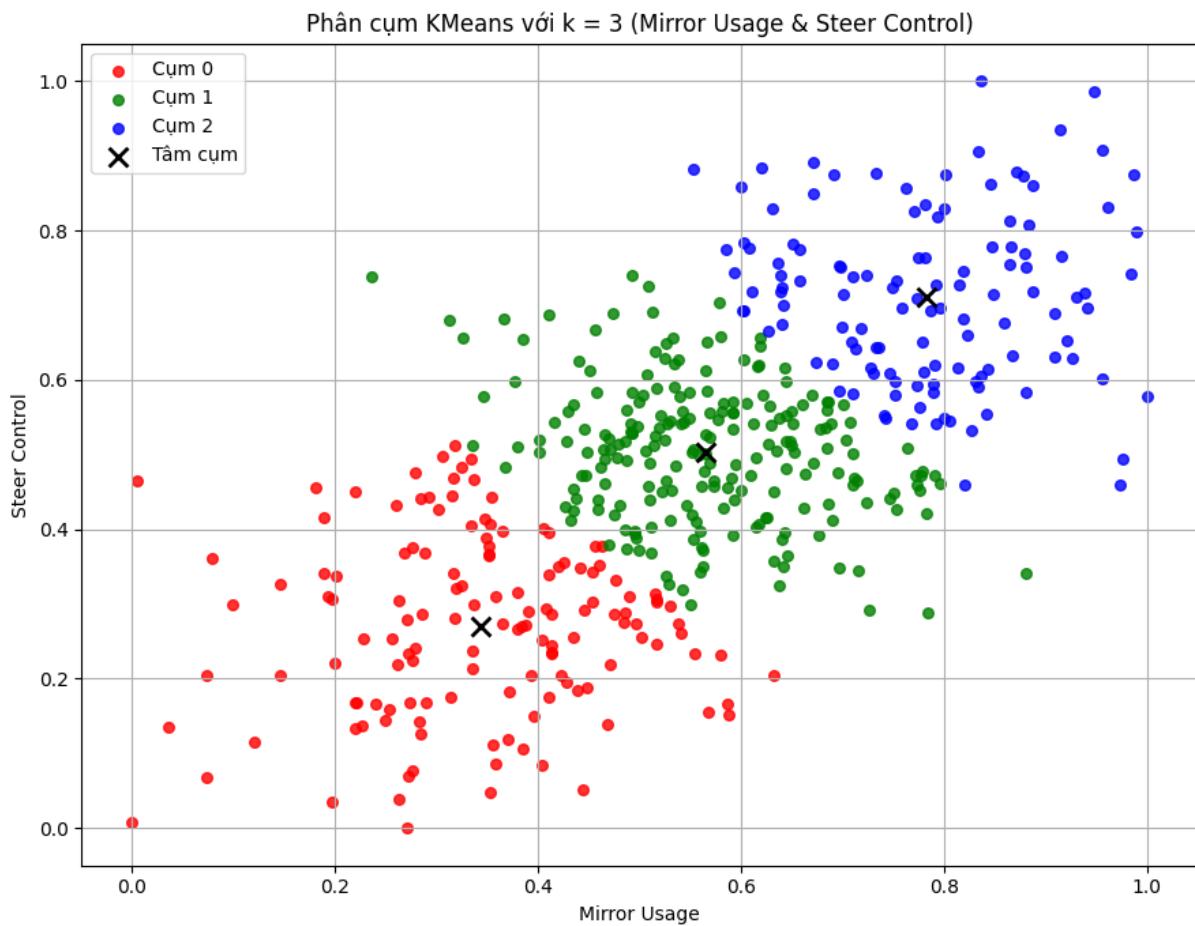
# Vẽ tâm cụm
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', color='black', s=100, linewidths=2, label='Tâm cụm')

plt.title("Phân cụm KMeans với k = 3 (Mirror Usage & Steer Control)")
plt.xlabel("Mirror Usage")
plt.ylabel("Steer Control")
plt.legend()

plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gán nhãn phân cụm vào DataFrame gốc ---
df['Cluster'] = labels

# --- 7. Lưu lại kết quả vào file CSV ---
df.to_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_da_xoa_outliers_2bien.csv", index=False)
```



Hình 42: Mô hình phân cụm K-Means 2 biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_da_xoa_outliers_2bien.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

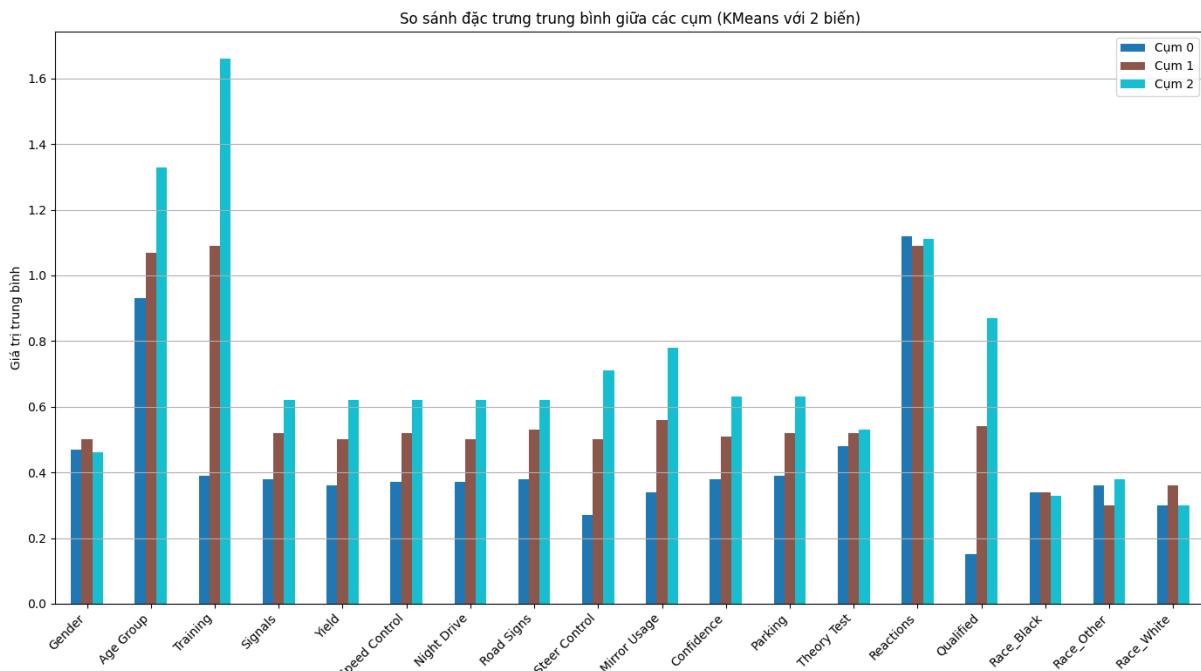
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]
```

```
# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (KMeans với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 43: So sánh đặc trưng trung bình giữa các cụm (Kmeans với 2 biến) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 37: Kỹ năng của các cụm trong Kmeans dựa trên 2 biến (Dữ liệu sau khi loại bỏ outliers)

Biến	Cụm 0 ( <i>Trung bình</i> )	Cụm 1 ( <i>Thấp điểm</i> )	Cụm 2 ( <i>Cao điểm</i> )
Signals	49.65 – Hiểu biết tín hiệu ở mức khá	37.60 – Nhận thức kém về tín hiệu giao thông	58.39 – Rất tốt, hiểu rõ tín hiệu
Yield	49.11 – Biết ưu tiên nhường đường khá tốt	34.84 – Rất yếu, phản xạ nhường đường thấp	59.48 – Xuất sắc về quy tắc ưu tiên

<b>Speed Control</b>	49.12 – Kiểm soát tốc độ tốt	36.54 – Khó khăn trong điều chỉnh tốc độ	58.17 – Kiểm soát tốc độ linh hoạt, hợp lý
<b>Night Drive</b>	49.00 – Ôn khi lái xe ban đêm	36.62 – Gặp khó khăn khi thiếu sáng	58.11 – Tự tin khi lái ban đêm
<b>Road Signs</b>	49.40 – Nhận biết biển báo khá	35.79 – Rất yếu	57.93 – Nhận diện tốt biển báo
<b>Steer Control</b>	48.56 – Điều khiển vô lăng tốt	29.16 – <b>Thấp nhất</b> , kỹ năng điều khiển cực kém	65.33 – <b>Cao nhất</b> , điều khiển rất vững
<b>Mirror Usage</b>	48.95 – Quan sát gương tốt	31.10 – Yếu, không kiểm tra gương thường xuyên	65.36 – Quan sát kỹ càng, cẩn trọng
<b>Confidence</b>	48.25 – Tự tin mức khá	36.53 – Thiếu tự tin	58.72 – Rất tự tin
<b>Parking</b>	49.77 – Kỹ năng đỗ xe tốt	37.07 – Gặp nhiều khó khăn khi đỗ xe	58.45 – Thành thạo đỗ xe
<b>Theory Test</b>	70.20 – Kiến thức lý thuyết vững	69.83 – Tạm ổn	72.29 – Cao nhất, lý thuyết chắc chắn

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Phân tích phân cụm bằng thuật toán K-Means với hai biến đầu vào đã mang lại cái nhìn khái quát về sự khác biệt giữa các nhóm đối tượng trong bộ dữ liệu. Mặc dù chỉ sử dụng hai biến, mô hình vẫn cho thấy khả năng chia tách tương đối rõ ràng giữa các nhóm có đặc điểm kỹ năng khác nhau.

Trước khi loại bỏ outliers, kết quả phân cụm bị ảnh hưởng phần nào bởi các điểm dữ liệu quá khác biệt, khiến biến cụm chưa thực sự sắc nét. Sau khi loại bỏ outliers, các cụm trở nên ổn định và phản ánh rõ ràng hơn sự khác biệt giữa các nhóm: một

cụm thể hiện nhóm học viên có năng lực tốt, cụm còn lại đại diện cho nhóm trung bình, và cụm cuối cùng là nhóm yếu hơn.

Việc loại bỏ outliers giúp mô hình tránh bị chi phối bởi các cá nhân quá đặc biệt và mang lại kết quả khách quan hơn. Tuy nhiên, những cá thể bị loại cũng có thể mang ý nghĩa thực tiễn quan trọng, cần được xử lý hoặc phân tích riêng biệt tùy theo mục tiêu ứng dụng.

Nhìn chung, mô hình K-Means với 2 biến cho thấy tiềm năng trong việc phát hiện các nhóm đặc trưng trong dữ liệu, đặc biệt nếu được kết hợp với xử lý dữ liệu thích hợp và phân tích hậu kỳ kỹ lưỡng.

### A.1.2. Phân cụm dựa trên toàn bộ dữ liệu

#### a. Dữ liệu trước khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Loại bỏ cột không cần thiết ---
X = df.drop(columns=['Applicant ID'])

# --- 3. KMeans với k = 3 ---
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
labels = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_

# --- 4. Giảm chiều bằng PCA để trực quan ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
centroids_pca = pca.transform(centroids)

# --- 5. Trực quan hóa kết quả phân cụm ---
plt.figure(figsize=(9, 7))
colors = ['red', 'green', 'blue']
for i in range(3):
    plt.scatter(X_pca[labels == i, 0], X_pca[labels == i, 1],
                s=30, color=colors[i], alpha=0.8, label=f'Cụm {i}')

# Vẽ tâm cụm
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1],
            marker='x', color='black', s=100, linewidths=2, label='Tâm cụm')
```

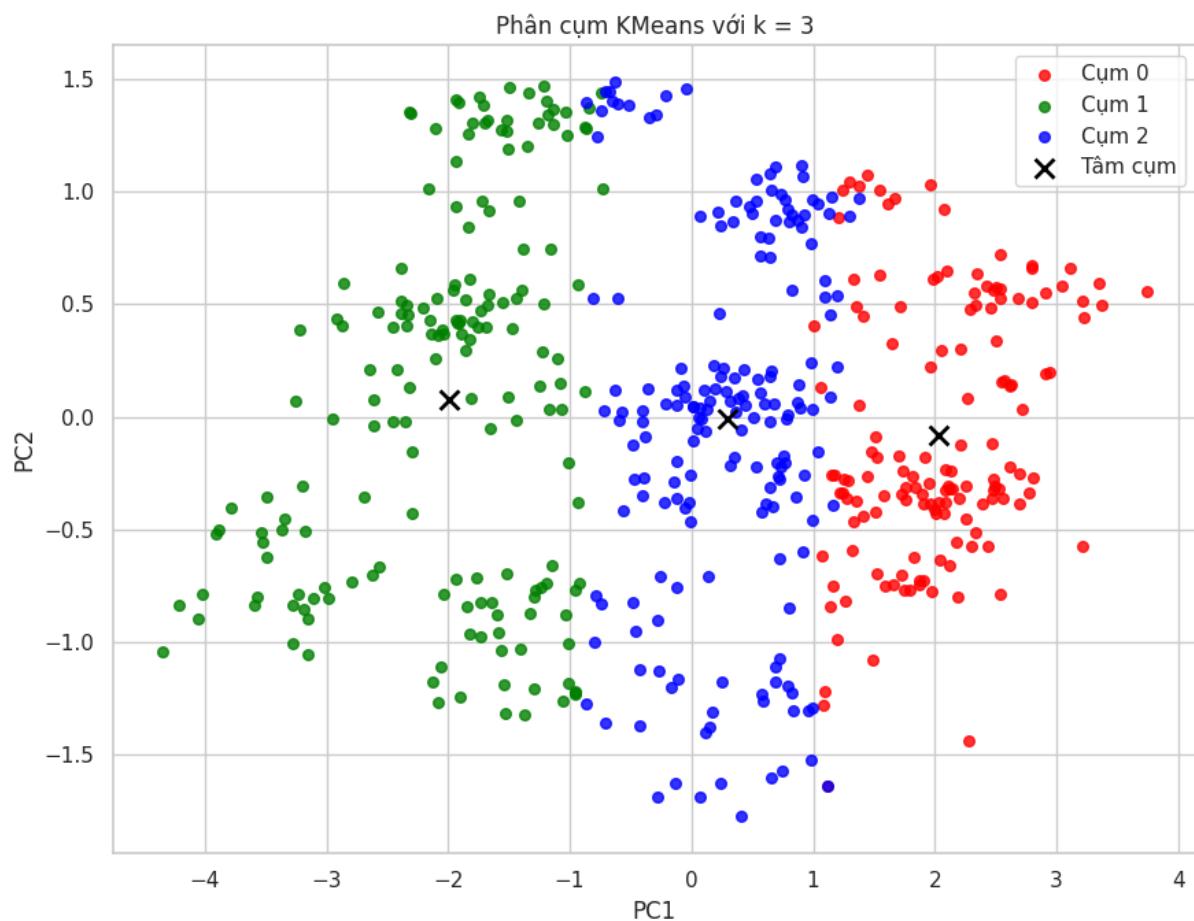
```

plt.title("Phân cụm KMeans với k = 3")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gán nhãn phân cụm vào DataFrame gốc ---
df['Cluster'] = labels

# --- 7. Lưu lại kết quả vào file CSV ---
df.to_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_full.csv", index=False)

```



```

# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_full.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

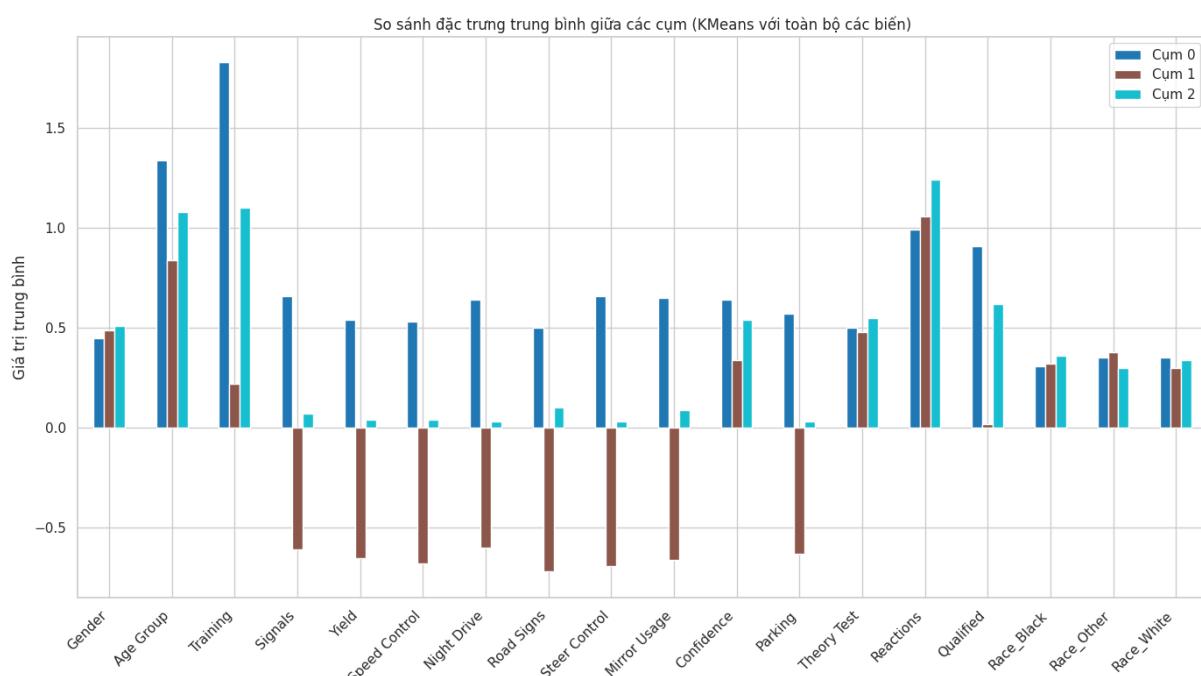
# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (KMeans với toàn bộ các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```



Cụm 0	<b>Khá toàn diện</b>	- Các kỹ năng như Signals, Confidence, Theory Test đạt mức trên trung bình (VD: Confidence: 0.601). - Tuy nhiên, các kỹ năng điều khiển như Steer Control (0.493) và Mirror Usage (0.562) còn khiêm tốn. - Là nhóm tương đối tốt nhưng vẫn có điểm yếu.
Cụm 1	<b>Yếu toàn diện</b>	- Hầu hết các kỹ năng đều thấp đáng kể. Đặc biệt Steer chỉ đạt <b>0.307</b> , Mirror Usage: <b>0.391</b> , Yield: <b>0.375</b> . - Tuy có Theory Test ngang cụm 0 (0.658), nhưng sự thiếu vững về kỹ năng thực hành và Confidence (0.575) khiến tỷ lệ đậu thấp.
Cụm 2	<b>Xuất sắc</b>	- Điểm số vượt trội ở mọi kỹ năng: Mirror Usage: 0.720, Steer: 0.654, Signals: 0.598. - Confidence: 0.624 và Theory Test: 0.670 cũng cao nhất. - Là nhóm học viên giỏi toàn diện, dễ đạt chuẩn sát hạch.

### b. Dữ liệu sau khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- 2. Loại bỏ cột không cần thiết ---
X = df.drop(columns=['Applicant ID'])

# --- 3. KMeans với k = 3 ---
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
labels = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_

# --- 4. Giảm chiều bằng PCA để trực quan ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
centroids_pca = pca.transform(centroids)

# --- 5. Trực quan hóa kết quả phân cụm ---
plt.figure(figsize=(9, 7))
colors = ['red', 'green', 'blue']
for i in range(3):
    plt.scatter(X_pca[labels == i, 0], X_pca[labels == i, 1],
                s=30, color=colors[i], alpha=0.8, label=f'Cụm {i}')

# Vẽ tâm cụm
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1],
            marker='x', color='black', s=100, linewidths=2, label='Tâm cụm')
```

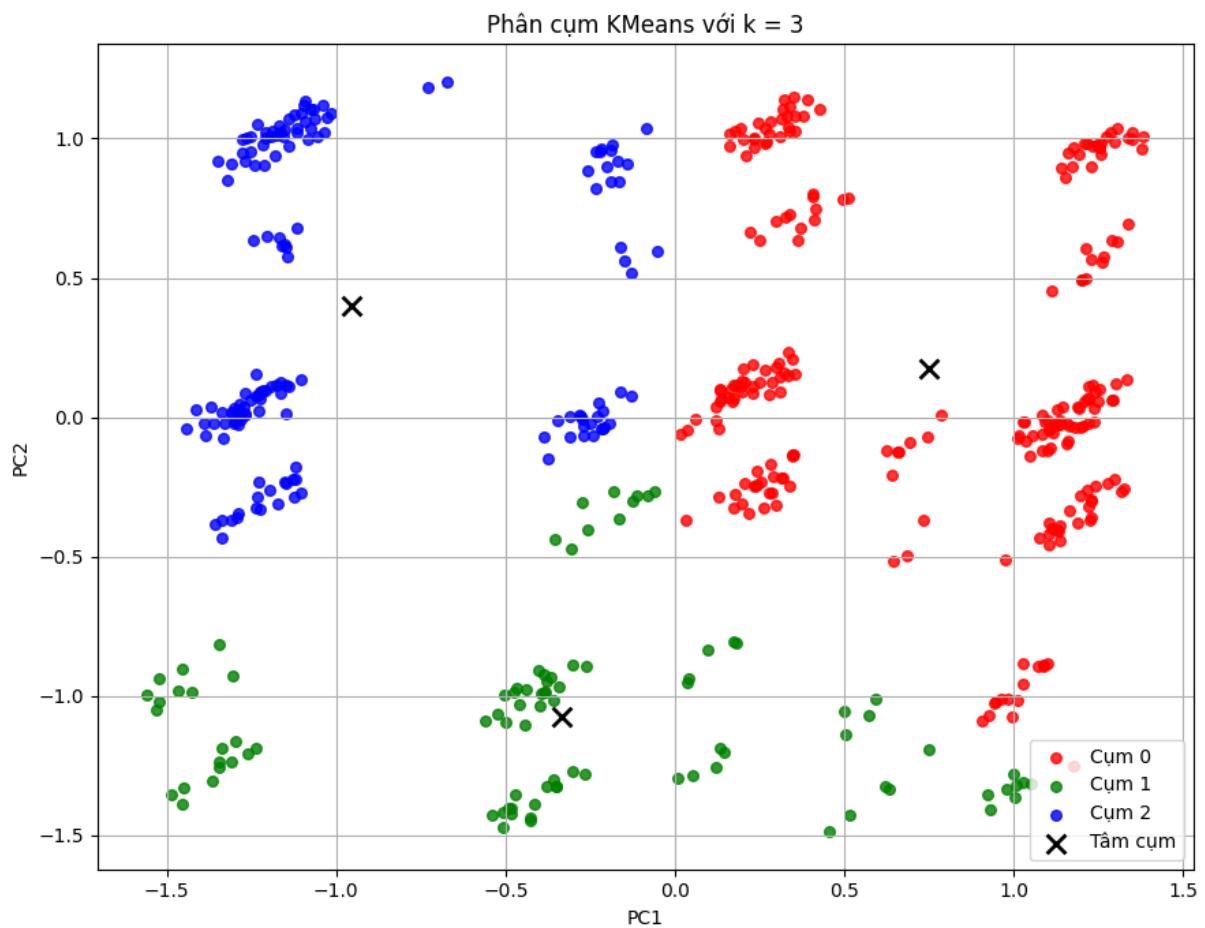
```

plt.title("Phân cụm KMeans với k = 3")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gán nhãn phân cụm vào DataFrame gốc ---
df['Cluster'] = labels

# --- 7. Lưu lại kết quả vào file CSV ---
df.to_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_da_xoa_outliers_full.csv", index=False)

```



```

# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_da_xoa_outliers_full.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

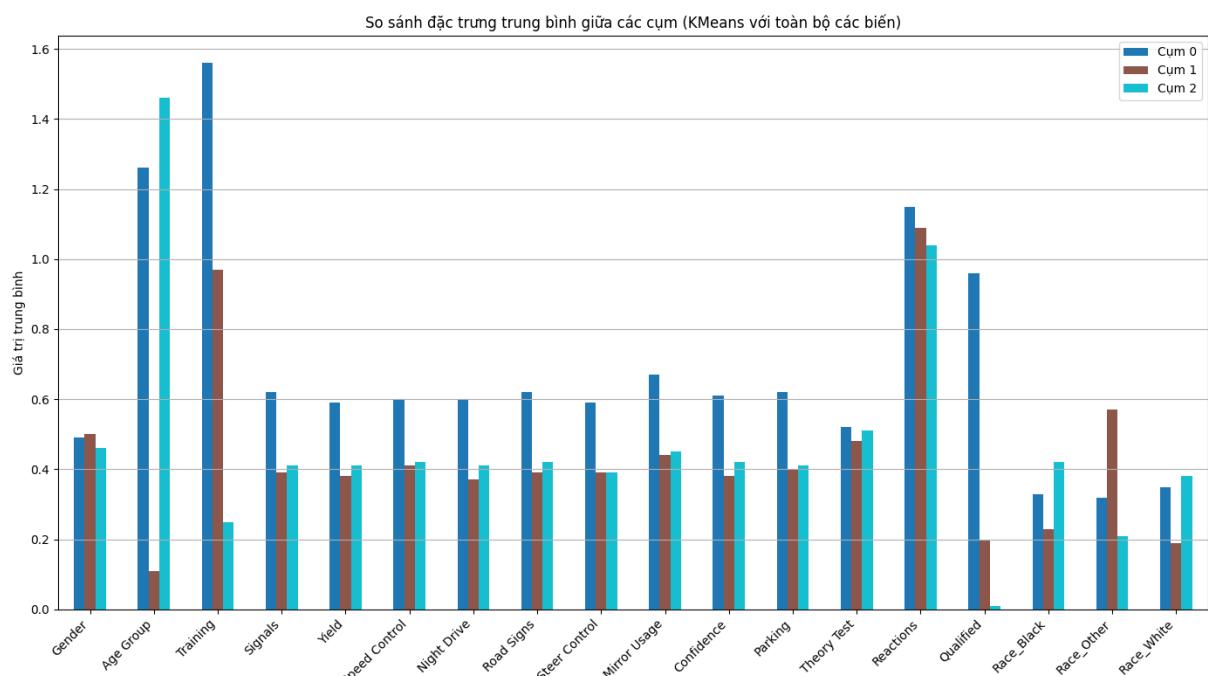
# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (KMeans với toàn bộ các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```



Cụm	Nhận xét	Giải thích
-----	----------	------------

<b>Cụm 0</b>	<b>Giỏi toàn diện – ổn định và nổi bật hơn sau loại bỏ nhiều</b>	- Gần như tất cả kỹ năng đều cao: Mirror Usage: 0.688, Speed Control: 0.614, Steer: 0.603. - Theory Test: 0.664 và Confidence: 0.619 → tự tin, lý thuyết ổn. - Là nhóm tốt nhất để thi lấy bằng.
<b>Cụm 1</b>	<b>Nhóm trung bình khá – cải thiện so với trước</b>	- Kỹ năng như Signals: 0.582, Yield: 0.562, Mirror Usage: 0.643 đều đạt ngưỡng khá. - Confidence: 0.577 và Theory Test: 0.652 ở mức trung bình. - Có thể vượt qua nếu ôn tập kỹ lưỡng.
<b>Cụm 2</b>	<b>Yếu – nhiều kỹ năng kém, có thể bị loại từ ban đầu</b>	- Điểm số giảm rõ so với trước (do loại học viên giỏi ra khỏi cụm này): Steer: 0.565, Mirror Usage: 0.645. - Confidence: 0.584 nhưng kỹ năng thực hành không nổi bật. - Đây là nhóm cần huấn luyện thêm.

Trước khi loại outliers, kết quả phân cụm bị ảnh hưởng bởi những học viên quá nổi bật hoặc bất thường, khiến cụm 2 dường như vượt trội một cách không thực tế. Sau khi loại bỏ outliers, các cụm trở nên phân hóa rõ ràng và hợp lý hơn. Cụm 0 nổi bật rõ rệt, cụm 1 được tái cấu trúc thành nhóm trung bình vững, còn cụm 2 thể hiện đúng bản chất là nhóm yếu. Phân tích này nhấn mạnh vai trò thiết yếu của xử lý ngoại lệ trong việc khám phá và ứng dụng mô hình phân cụm hiệu quả vào đào tạo và đánh giá kết quả học tập.

## A.2. Thuật toán DBSCAN

### A.2.1. Sử dụng 2 biến đặc trưng

#### a. Dữ liệu trước khi loại bỏ outliers

```

# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

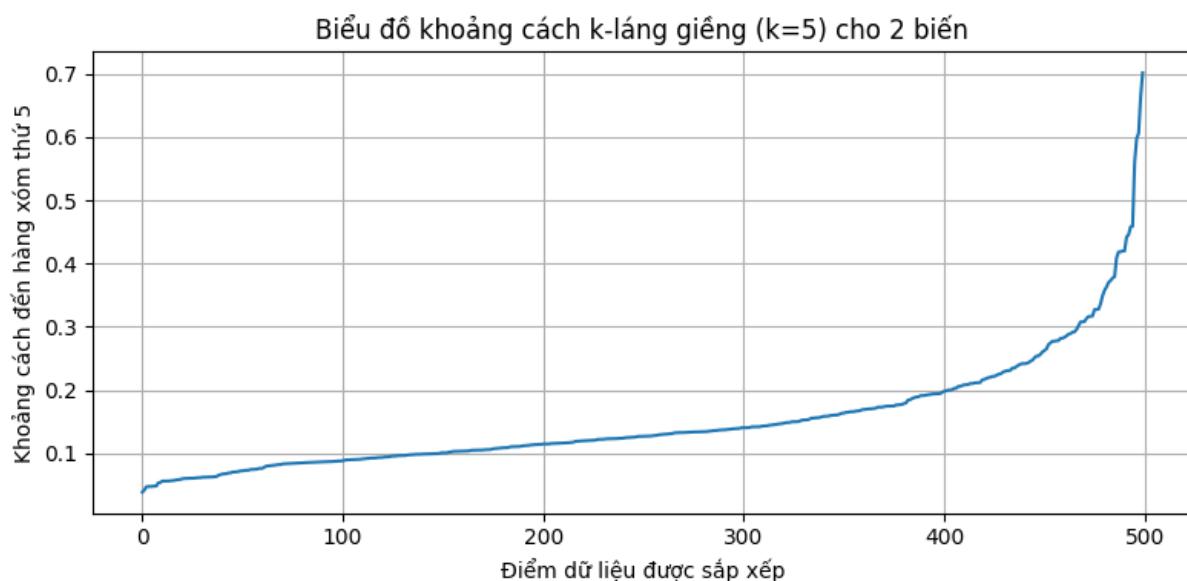
# Chọn 2 biến
X = df[['Steer Control', 'Night Drive']].copy()

# Dò tìm epsilon bằng k-distance plot
k = 5
neighbors = NearestNeighbors(n_neighbors=k)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)

# Sắp xếp khoảng cách đến hàng xóm thứ k
k_distances = np.sort(distances[:, k-1])

# Vẽ biểu đồ
plt.figure(figsize=(8, 4))
plt.plot(k_distances)
plt.title(f'Biểu đồ khoảng cách k-láng giềng (k={k}) cho 2 biến')
plt.xlabel('Điểm dữ liệu được sắp xếp')
plt.ylabel(f'Khoảng cách đến hàng xóm thứ {k}')
plt.grid(True)
plt.tight_layout()
plt.show()

```



Hình 44: Biểu đồ k-distance graph để chọn tham số eps phù hợp cho 2 biến

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### ❖ Mô tả kết quả:

**Đường cong tăng dần từ trái sang phải Góc gãy (elbow)** xảy ra khoảng tại giá trị  $\text{eps} \approx 0.25$ . Đây là ngưỡng mà:

- Dưới ngưỡng này: Các điểm gần nhau → mật độ cao → thuộc về cụm.
- Trên ngưỡng này: Khoảng cách tăng đột ngột → điểm cách xa → có thể là nhiễu.

Nhóm được các điểm có mật độ cao thành cụm và loại bỏ các điểm biệt lập làm nhiễu.

=> Vậy ở thuật toán DBSCAN nên chọn **eps = 0,25**.

```
# --- Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- Chọn 2 biến ---
X = df[['Steer Control', 'Night Drive']].copy()

# --- Áp dụng DBSCAN ---
eps = 0.25
min_samples = 5
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
labels = dbscan.fit_predict(X)

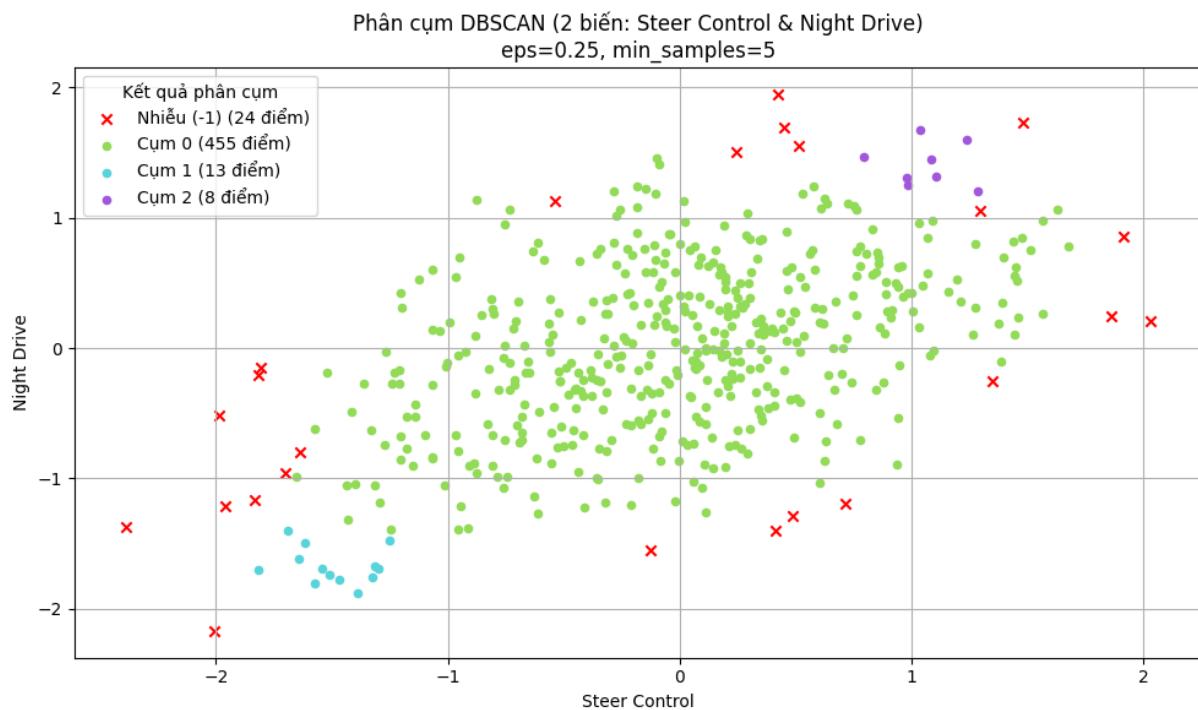
# --- Thống kê số điểm theo cụm ---
label_counts = Counter(labels)
print(f"Số điểm nhiễu (noise): {label_counts[-1] if -1 in label_counts else 0}")
print("Số lượng điểm trong mỗi cụm:")
for label, count in sorted(label_counts.items()):
    if label == -1:
        print(f" - Nhiễu (label = -1): {count} điểm")
    else:
        print(f" - Cụm {label}: {count} điểm")
```

```
# --- Vẽ kết quả phân cụm ---
plt.figure(figsize=(10, 6))
unique_labels = sorted(set(labels))
colors = sns.color_palette('hls', len(unique_labels))

for label, color in zip(unique_labels, colors):
    mask = (labels == label)
    count = label_counts[label]
    if label == -1:
        plt.scatter(X.loc[mask, 'Steer Control'], X.loc[mask, 'Night Drive'],
                    c='red', label=f'Nhiều (-1) ({count} điểm)', marker='x')
    else:
        plt.scatter(X.loc[mask, 'Steer Control'], X.loc[mask, 'Night Drive'],
                    c=[color], label=f'Cụm {label} ({count} điểm)', s=20)

plt.title(f'Phân cụm DBSCAN (2 biến: Steer Control & Night Drive)\neps={eps}, min_samples={min_samples}')
plt.xlabel('Steer Control')
plt.ylabel('Night Drive')
plt.legend(loc='best', title='Kết quả phân cụm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Lưu kết quả ---
df_clusters = X.copy()
df_clusters['Cluster'] = labels
df_clusters.to_csv('phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv', index=False)
```



Hình 45: Phân cụm DBSCAN 2 biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```

# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=['Applicant ID', "Cluster"]).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

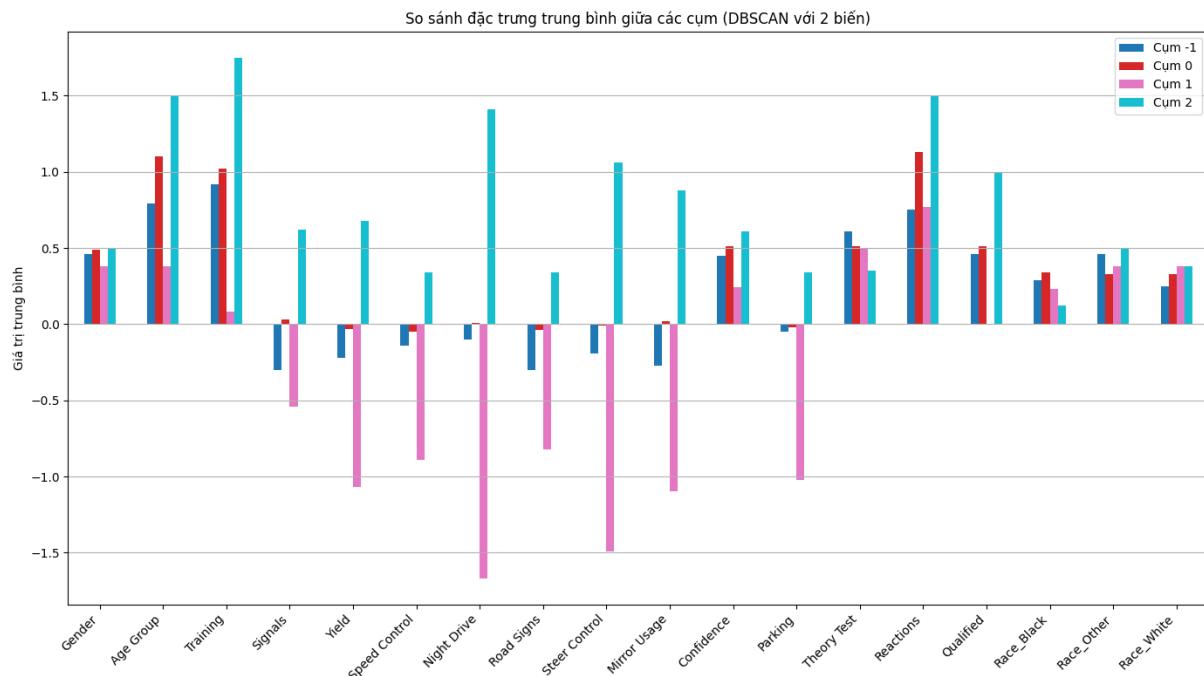
# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

```

```

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (DBSCAN với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```



Hình 46: So sánh đặc trưng giữa các cụm (DBSCAN với 2 biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 38: Nhận xét các cụm DBSCAN – 2 biến (Dữ liệu trước khi loại bỏ outliers)

Cụm	Số lượng	Đặc điểm nổi bật	Giải thích
Cụm -1 (Nhiều)	24	- Giới tính: <b>Nam chiếm đa số</b> (46%)- Độ tuổi trung bình: <b>Young Adult</b> - Trình độ đào tạo: Chủ yếu <b>Basic</b> đến <b>Unknown</b> - Điểm thi đa phần ~45–47 điểm- Tự tin (Confidence): ~47 điểm- Phản xạ: trung bình ( $\approx$ <b>Average</b> )- Tỷ lệ đậu: ~46%- Chủng tộc: Chủ yếu <b>Black (36%) &amp; Other (40%)</b>	Nhóm có kết quả không ổn định, nhiều khả năng là những học viên thiếu chuẩn bị hoặc kỹ năng không đồng đều
Cụm 0 (Chính)	455	- Giới tính: cân bằng Nam – Nữ ( $\approx$ 49%)- Tuổi: đa số là <b>Young Adult</b> - Đào tạo: chủ yếu <b>Basic</b> - Điểm kỹ năng ~47–48- Confidence: ~51 điểm- Phản xạ: <b>Fast</b> ( $\approx$ 2)- Tỷ lệ đậu: ~51%- Chủng tộc: phân bố đồng đều (Black, White, Other)	Nhóm học viên phổ biến nhất, điểm trung bình – khá, có kiến thức cơ bản và khả năng thi đậu cao
Cụm 1 (Yếu)	13	- Giới tính: chủ yếu <b>Nam (38%)</b> - Độ tuổi: phần lớn là <b>Teenager</b> - Đào tạo: <b>Unknown</b> ( $\approx$ 0.08)- Điểm kỹ năng: rất thấp (30–35 điểm)- Confidence: chỉ ~24 điểm- Phản xạ: <b>Average</b> đến <b>Slow</b> - Tỷ lệ đậu: <b>0%</b> - Chủng tộc: khá cân bằng nhưng <b>da trắng chiếm nhiều (40%)</b>	Nhóm yếu nhất – trẻ tuổi, chưa đào tạo, thiếu kinh nghiệm, điểm thấp toàn diện. Cần được hướng dẫn lại từ đầu
Cụm 2 (Xuất sắc)	8	- Giới tính: cân bằng (Nam – Nữ)- Tuổi: chủ yếu <b>Middle Age</b> ( $\approx$ 2)- Đào tạo: <b>Advanced</b> (1.75)- Kỹ năng: cao vượt trội (~60–70 điểm)- Confidence: ~58 điểm- Phản xạ: <b>Fast</b> ( $\approx$ 2)- Tỷ lệ	Nhóm giỏi nhất – người lớn tuổi, đã được đào tạo tốt, tự tin, phản xạ nhanh và săn

		độ: <b>100%</b> - Chủng tộc: White (43%), Other (43%)	sàng thi sát hạch thực tế
--	--	---	---------------------------

(Nguồn: Tác giả tổng hợp, 2025)

### b. Dữ liệu sau khi loại bỏ outliers

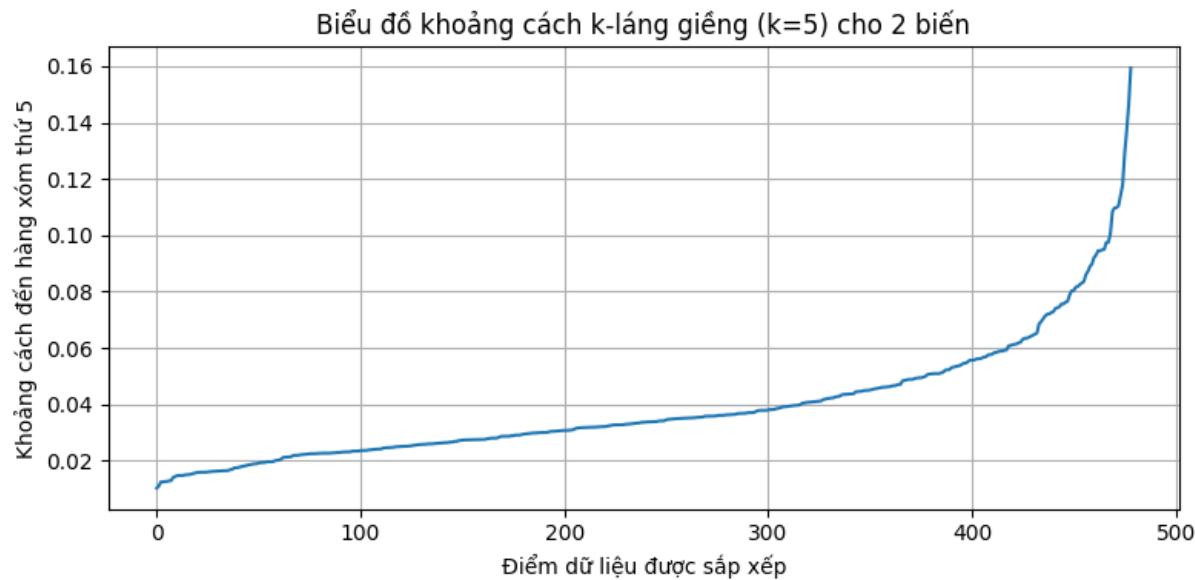
```
# Đọc dữ liệu
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# Chọn 2 biến
X = df[['Steer Control', 'Night Drive']].copy()

# Dò tìm epsilon bằng k-distance plot
k = 5
neighbors = NearestNeighbors(n_neighbors=k)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)

# Sắp xếp khoảng cách đến hàng xóm thứ k
k_distances = np.sort(distances[:, k-1])

# Vẽ biểu đồ
plt.figure(figsize=(8, 4))
plt.plot(k_distances)
plt.title(f'Biểu đồ khoảng cách k-lắng giềng (k={k}) cho 2 biến')
plt.xlabel('Điểm dữ liệu được sắp xếp')
plt.ylabel(f'Khoảng cách đến hàng xóm thứ {k}')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Hình 47: Biểu đồ k-distance graph để chọn tham số eps phù hợp cho 2 biến

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

### ❖ Mô tả kết quả:

**Đường cong tăng dần từ trái sang phải Góc gãy (elbow)** xảy ra khoảng tại **giá trị eps  $\approx 0.062$** . Đây là ngưỡng mà:

- o Dưới ngưỡng này: Các điểm gần nhau  $\rightarrow$  mật độ cao  $\rightarrow$  thuộc về cụm.
- o Trên ngưỡng này: Khoảng cách tăng đột ngột  $\rightarrow$  điểm cách xa  $\rightarrow$  có thể là nhiễu.

Nhóm được các điểm có mật độ cao thành cụm và loại bỏ các điểm biệt lập làm nhiễu.

=> Vậy ở thuật toán DBSCAN nên chọn **eps = 0,062**.

```
# --- Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- Chọn 2 biến ---
X = df[['Steer Control', 'Night Drive']].copy()

# --- Áp dụng DBSCAN ---
eps = 0.062
min_samples = 5
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
labels = dbscan.fit_predict(X)

# --- Thống kê số điểm theo cụm ---
label_counts = Counter(labels)
print(f"Số điểm nhiễu (noise): {label_counts[-1] if -1 in label_counts else 0}")
print("Số lượng điểm trong mỗi cụm:")
for label, count in sorted(label_counts.items()):
    if label == -1:
        print(f" - Nhiễu (label = -1): {count} điểm")
    else:
        print(f" - Cụm {label}: {count} điểm")

# --- Vẽ kết quả phân cụm ---
plt.figure(figsize=(10, 6))
unique_labels = sorted(set(labels))
colors = sns.color_palette('hls', len(unique_labels))
```

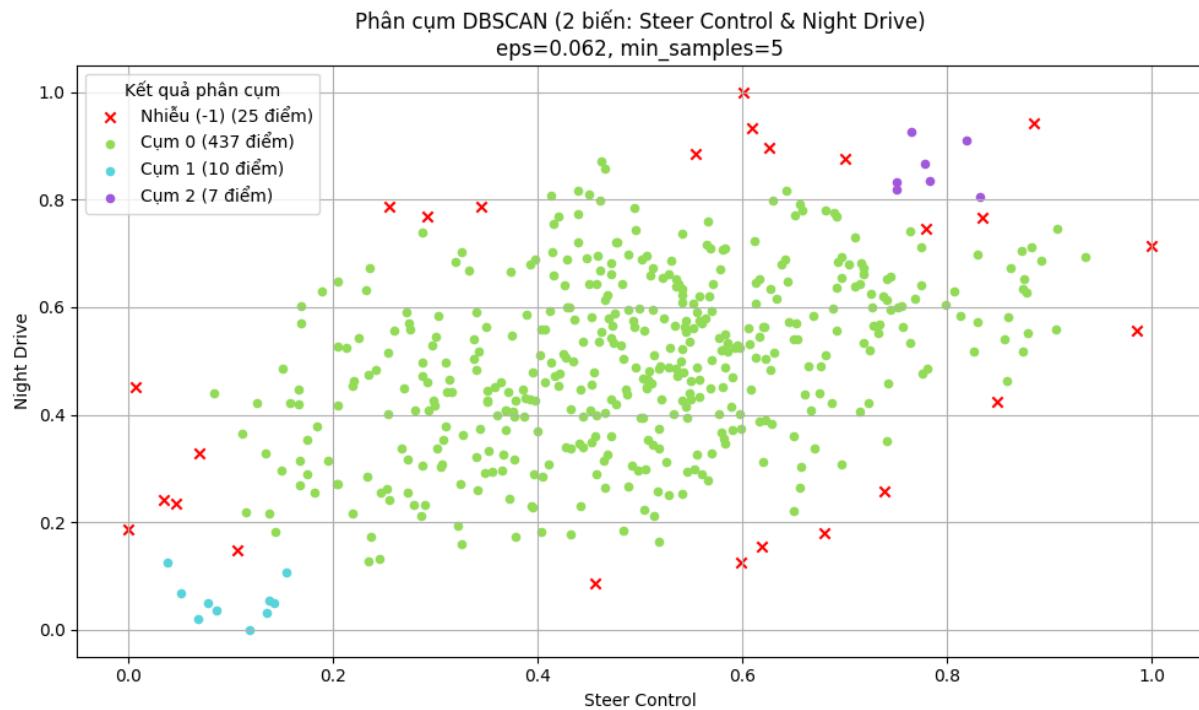
```

for label, color in zip(unique_labels, colors):
    mask = (labels == label)
    count = label_counts[label]
    if label == -1:
        plt.scatter(X.loc[mask, 'Steer Control'], X.loc[mask, 'Night Drive'],
                    c='red', label=f'Nhiều (-1) ({count} điểm)', marker='x')
    else:
        plt.scatter(X.loc[mask, 'Steer Control'], X.loc[mask, 'Night Drive'],
                    c=[color], label=f'Cụm {label} ({count} điểm)', s=20)

plt.title(f'Phân cụm DBSCAN (2 biến: Steer Control & Night Drive)\neps={eps}, min_samples={min_samples}')
plt.xlabel('Steer Control')
plt.ylabel('Night Drive')
plt.legend(loc='best', title='Kết quả phân cụm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Lưu kết quả ---
df_clusters = X.copy()
df_clusters['Cluster'] = labels
df_clusters.to_csv('phan_cum_dbSCAN_2bien_da_xoa_outliers.csv', index=False)

```



Hình 48: Phân cụm DBSCAN 2 biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```

# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_da_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

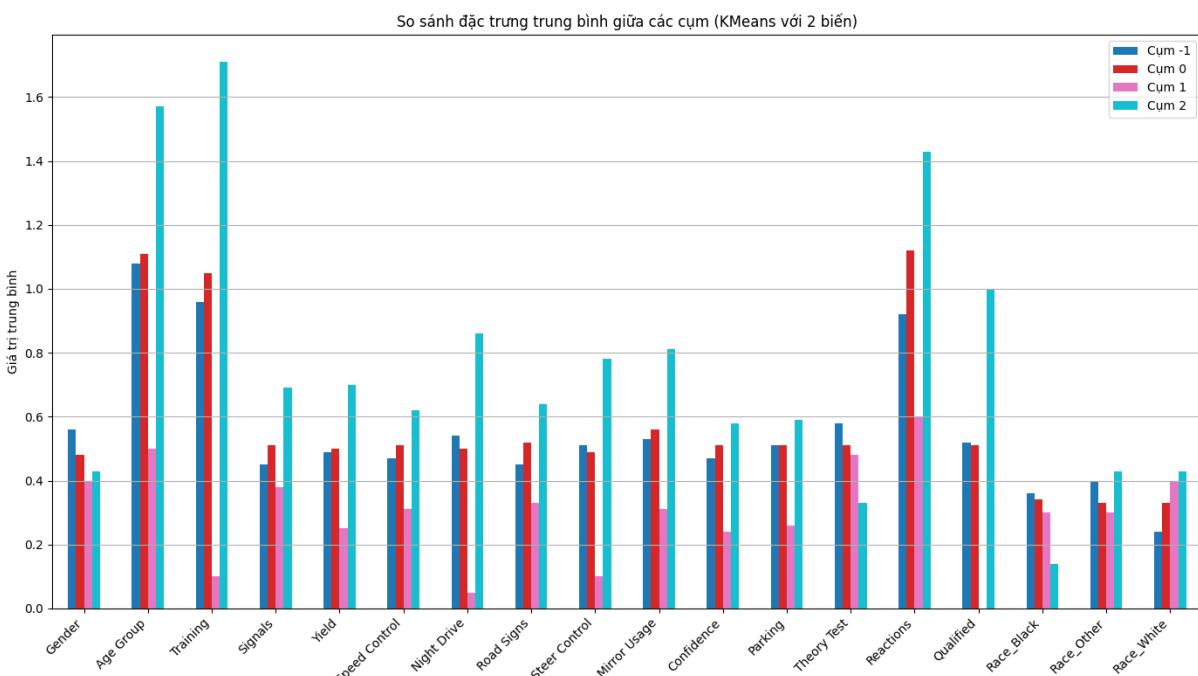
# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

```

```

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (KMeans với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```



Hình 49: So sánh đặc trưng giữa các cụm (DBSCAN với 2 biến) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 39: Nhận xét các cụm DBSCAN – 2 biến (Dữ liệu sau khi loại bỏ outliers)

Cụm	Số lượng	Đặc điểm	Giải thích
Cụm -1 (Nhiều)	25	- Giới tính: chủ yếu <b>Nam (56%)</b> - Tuổi: đa số <b>Young Adult</b> - Đào tạo: <b>Basic</b> hoặc <b>Unknown</b> - Kỹ năng: trung bình (~47–50 điểm)- Confidence: ~47 điểm- Phản xạ: <b>Average</b> đến <b>Fast</b> - Tỷ lệ đậu: 52%- Chủng tộc: <b>Black (36%)</b> , <b>Other (40%)</b>	Nhóm học viên không vượt trội nhưng cũng không quá kém. Có thể học theo phương pháp khác hoặc luyện chưa đều
Cụm 0 (Chính)	437	- Giới tính: cân bằng (Nam – Nữ)- Tuổi: đa số <b>Young Adult</b> - Đào tạo: <b>Basic</b> - Điểm kỹ năng ổn định ~48–49- Confidence: ~51 điểm- Phản xạ: <b>Fast</b> ( $\approx 2$ )- Tỷ lệ đậu: ~51%- Chủng tộc: phân bố đều	Nhóm học viên điển hình, kỹ năng vững và đủ vượt qua kỳ thi. Có thể duy trì chương trình hiện tại
Cụm 1 (Yếu)	10	- Giới tính: chủ yếu <b>Nam</b> - Tuổi: <b>Teenager</b> - Đào tạo: chưa học ( <b>Unknown</b> )- Điểm rất thấp (~25–35 điểm)- Confidence: chỉ ~24 điểm- Phản xạ: <b>Slow</b> đến <b>Average</b> - Tỷ lệ đậu: <b>0%</b> - Chủng tộc: thiên về <b>da trắng (40%)</b>	Nhóm yếu – cần thiết kế chương trình dành riêng, tập trung rèn luyện cơ bản từ phản xạ đến kỹ năng thực hành
Cụm 2 (Xuất sắc)	7	- Giới tính: cân bằng- Tuổi: đa số <b>Middle Age</b> - Đào tạo: <b>Advanced (1.71)</b> - Kỹ năng thi cao toàn diện (~60–70 điểm)- Confidence: ~58 điểm- Reactions: <b>Fast</b> ( $\approx 2$ )- Tỷ lệ đậu: <b>100%</b> - Chủng tộc: <b>White (43%)</b> , <b>Other (43%)</b>	Nhóm học viên lý tưởng. Có thể làm mẫu, hỗ trợ nhóm khác học tập. Chương trình nâng cao nên được duy trì cho nhóm này

(Nguồn: Tác giả tổng hợp, 2025)

**❖ Nhận xét:**

Trong quá trình phân tích dữ liệu bằng thuật toán DBSCAN, việc giữ nguyên giá trị ngoại lai (outliers) hay loại bỏ outliers trước khi phân cụm – mang lại ảnh hưởng rõ rệt đến chất lượng kết quả.

Khi không xử lý outliers, mô hình tạo ra nhiều điểm bị xem là nhiễu (noise), làm giảm tính rõ ràng trong việc phân tách các nhóm đối tượng. Ngược lại, khi loại bỏ outliers, mô hình cho thấy các cụm phân nhóm trở nên đồng nhất và dễ diễn giải hơn, đặc biệt khi đối chiếu với các biến định tính như độ tuổi, trình độ đào tạo và khả năng phản xạ. Điều này không chỉ giúp tăng độ chính xác của mô hình mà còn nâng cao khả năng ứng dụng thực tế trong việc xây dựng các chương trình đào tạo phù hợp cho từng nhóm người học.

Tuy nhiên, một điểm đáng chú ý là cụm -1 (thường được xem là nhiễu trong DBSCAN) và cụm 0 lại thể hiện những đặc điểm khá tương đồng sau khi loại bỏ outliers. Cả hai cụm đều có độ tuổi chủ yếu thuộc nhóm người trẻ (Young Adult), trình độ đào tạo cơ bản, mức độ tự tin và tỷ lệ đạt yêu cầu gần như ngang nhau. Khác biệt chính giữa hai cụm này chủ yếu nằm ở mức độ xác định rõ thông tin đào tạo (cụm -1 có tỷ lệ “Unknown” cao hơn). Điều này cho thấy cụm -1 trong trường hợp này không hoàn toàn là nhiễu ngẫu nhiên, mà có thể là một nhóm học viên hợp lệ với dữ liệu thiếu thông tin. Do đó, tùy thuộc vào mục tiêu phân tích, cụm -1 có thể được giữ lại để nghiên cứu các trường hợp “bất thường hợp lý” hoặc được hợp nhất vào cụm 0 nhằm đơn giản hóa mô hình.

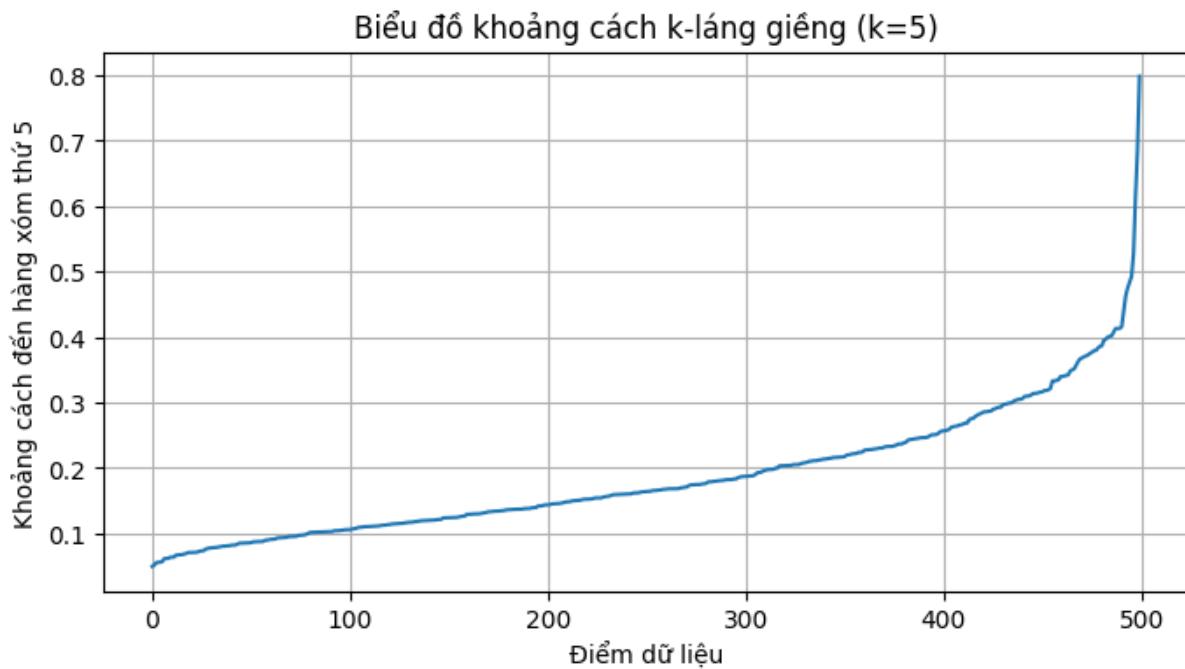
Tổng thể, phương pháp loại bỏ outliers trước khi phân cụm được đánh giá là ưu việt hơn trong bối cảnh này, vì nó làm nổi bật cấu trúc cụm, tăng khả năng giải thích và hỗ trợ hiệu quả cho các ứng dụng đào tạo cá nhân hóa. Tuy nhiên, việc loại bỏ outliers cần được thực hiện một cách cẩn trọng, nhằm tránh vô tình loại bỏ những đối tượng có ý nghĩa thực tiễn trong phân tích.

**A.2.2. Sử dụng tất cả các biến đặc trưng****a. Dữ liệu trước khi loại bỏ outliers**

```

df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')
# Loại bỏ cột student_id
X = df.drop(columns=["Applicant ID"]) # Bỏ cột ID
# PCA để giảm chiều (chỉ để trực quan hóa)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
# Dò tìm epsilon tốt bằng k-distance plot
k = 5
neighbors = NearestNeighbors(n_neighbors=k)
neighbors_fit = neighbors.fit(X_pca)
distances, indices = neighbors_fit.kneighbors(X_pca)
# Sắp xếp khoảng cách đến hàng xóm thứ k
distances = np.sort(distances[:, k-1])
plt.figure(figsize=(8, 4))
plt.plot(distances)
plt.title(f'Biểu đồ khoảng cách k-láng giềng (k={k})')
plt.xlabel('Điểm dữ liệu')
plt.ylabel(f'Khoảng cách đến hàng xóm thứ {k}')
plt.grid(True)
plt.show()

```



Hình 50: Biểu đồ k-distance graph để chọn tham số eps phù hợp

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### ❖ Mô tả kết quả:

**Đường cong tăng dần từ trái sang phải Góc gãy (elbow)** xảy ra khoảng tại giá trị  $\text{eps} \approx 0.38$ . Đây là ngưỡng mà:

- Dưới ngưỡng này: Các điểm gần nhau → mật độ cao → thuộc về cụm.

- Trên ngưỡng này: Khoảng cách tăng đột ngột → điểm cách xa → có thể là nhiễu.

Nhóm được các điểm có mật độ cao thành cụm và loại bỏ các điểm biệt lập làm nhiễu.

=> Vậy ở thuật toán DBSCAN nên chọn **eps = 0,38**.

```
# --- 1. Đọc dữ liệu đã tiền xử lý ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Loại bỏ cột ID ---
X = df.drop(columns=["Applicant ID"])

# --- 3. Giảm chiều bằng PCA ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

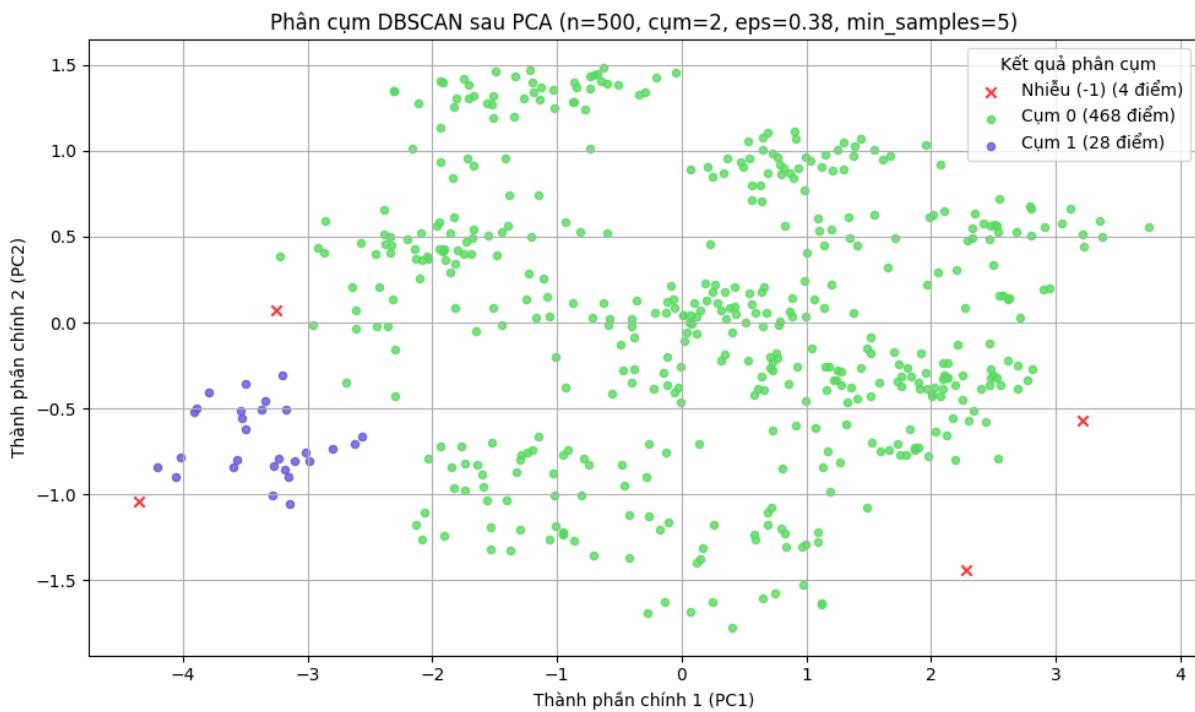
# --- 4. Áp dụng DBSCAN ---
eps = 0.38
min_samples = 5
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
labels = dbscan.fit_predict(X_pca)

# --- 5. Thống kê cụm ---
label_counts = Counter(labels)
print("Số điểm theo từng cụm:")
for label, count in sorted(label_counts.items()):
    if label == -1:
        print(f" - Nhiều (-1): {count} điểm")
    else:
        print(f" - Cụm {label}: {count} điểm")

# --- 6. Trực quan hóa kết quả ---
plt.figure(figsize=(10, 6))
unique_labels = sorted(set(labels))
colors = sns.color_palette('hls', len(unique_labels))

for label, color in zip(unique_labels, colors):
    mask = (labels == label)
    count = label_counts[label]
    if label == -1:
        plt.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c='red', marker='x', label=f'Nhiều (-1) ({count} điểm)', alpha=0.8)
    else:
        plt.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c=[color], s=20, label=f'Cụm {label} ({count} điểm)', alpha=0.8)

# --- 7. Thêm tiêu đề ---
num_clusters = len(set(labels)) - (1 if -1 in labels else 0)
plt.title(f'Phân cụm DBSCAN sau PCA (n={len(X)}, cụm={num_clusters}, eps={eps}, min_samples={min_samples})')
plt.xlabel('Thành phần chính 1 (PC1)')
plt.ylabel('Thành phần chính 2 (PC2)')
plt.legend(title='Kết quả phân cụm')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Hình 51: Phân cụm DBSCAN tất cả các biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_allfeature_chua_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

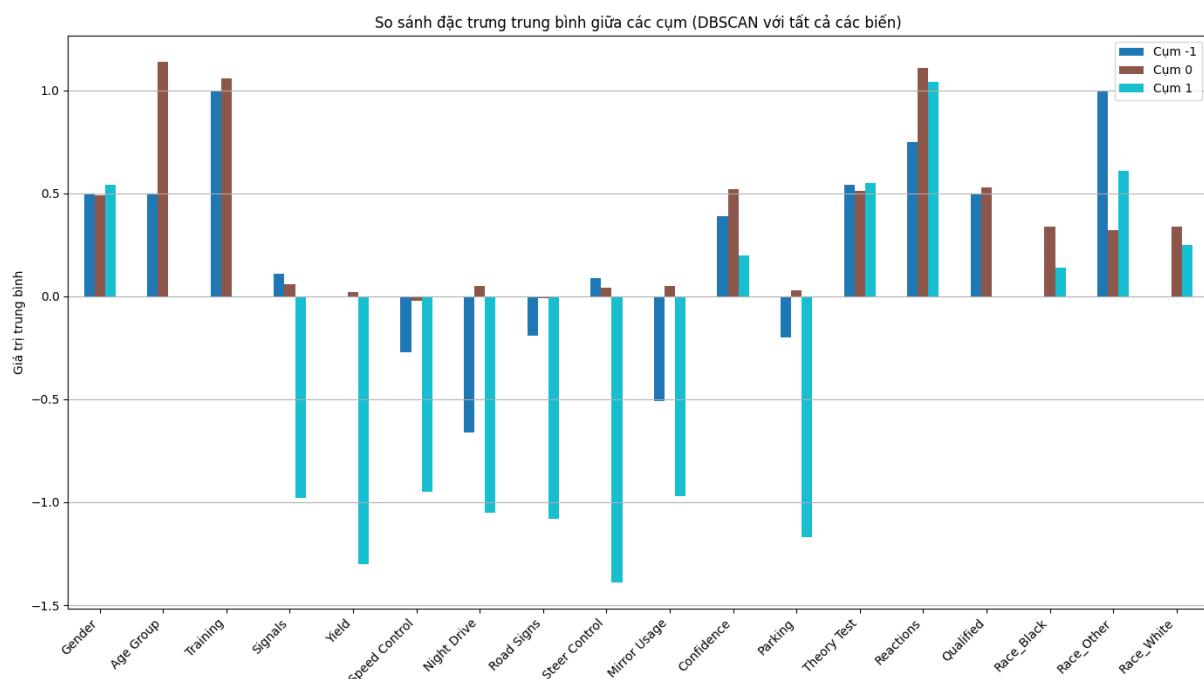
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (DBSCAN với tất cả các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 52: So sánh đặc trưng giữa các cụm (DBSCAN với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 40: Nhận xét các cụm DBSCAN – tất cả các biến (Dữ liệu trước khi loại bỏ outliers)

Cụm	Mô tả đặc điểm cụm	Nhận xét
-1 (Noise)	50% nam, tuổi từ Teenager đến Young Adult, 100% thuộc chủng tộc <b>Other</b> , đào tạo mức <b>Basic</b> . <b>Kỹ năng:</b> • <b>Night Drive</b> ~ 37.7/100 (kém) • <b>Mirror Usage</b> ~ 40.7/100 • <b>Steer Control</b> ~ 48.8/100 • <b>Confidence</b> ~ 55.2/100 • <b>Reactions:</b> tốt. <b>Qualified:</b> 50%.	Cụm này có kỹ năng tương đối yếu, nhưng tốc độ phản xạ lại cao. Thuộc nhóm chủng tộc đặc thù nên có thể là nhóm cần xem xét riêng.
0	Tuổi trung bình <b>Young Adult</b> , đào tạo chủ yếu ở mức <b>Basic</b> , phản xạ tốt ( <b>Reactions</b> ~ 2.22), giới tính và chủng tộc phân bố đồng đều. <b>Kỹ năng:</b> • Các kỹ năng như Night Drive, Mirror, Steer, Parking dao động quanh 47–50 điểm • <b>Confidence</b> ~ 57.8/100 ◊ <b>Khả năng đậu:</b> 53%.	Là cụm phổ biến nhất, đại diện cho nhóm học viên khá điển hình. Kỹ năng ổn định và có khả năng đậu tương đối cao.
1	100% là <b>Teenager</b> , chưa tham gia bất kỳ chương trình đào tạo nào, 61% là chủng	Cụm yếu nhất trong toàn bộ. Dù phản xạ nhanh,

	tộc <b>Other</b> , phản xạ tốt ( <b>Reactions ~ 2.2</b> ). <b>Kỹ năng rất kém:</b> • <b>Night Drive ~ 34.5/100</b> • <b>Mirror Usage ~ 33.5/100</b> • <b>Steer Control ~ 35.4/100</b> • <b>Confidence ~ 51/100.</b> <b>Qualified: 0%</b> .	nhưng thiếu kỹ năng và chưa được đào tạo. Cần ưu tiên đào tạo và hỗ trợ đặc biệt.
--	--	---

(Nguồn: Tác giả tổng hợp, 2025)

b. Dữ liệu sau khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu đã tiền xử lý và loại outliers ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- 2. Loại bỏ cột ID ---
X = df.drop(columns=["Applicant ID"])

# --- 3. Giảm chiều bằng PCA ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# --- 4. Áp dụng DBSCAN ---
eps = 0.38
min_samples = 5
dbSCAN = DBSCAN(eps=eps, min_samples=min_samples)
labels = dbSCAN.fit_predict(X_pca)

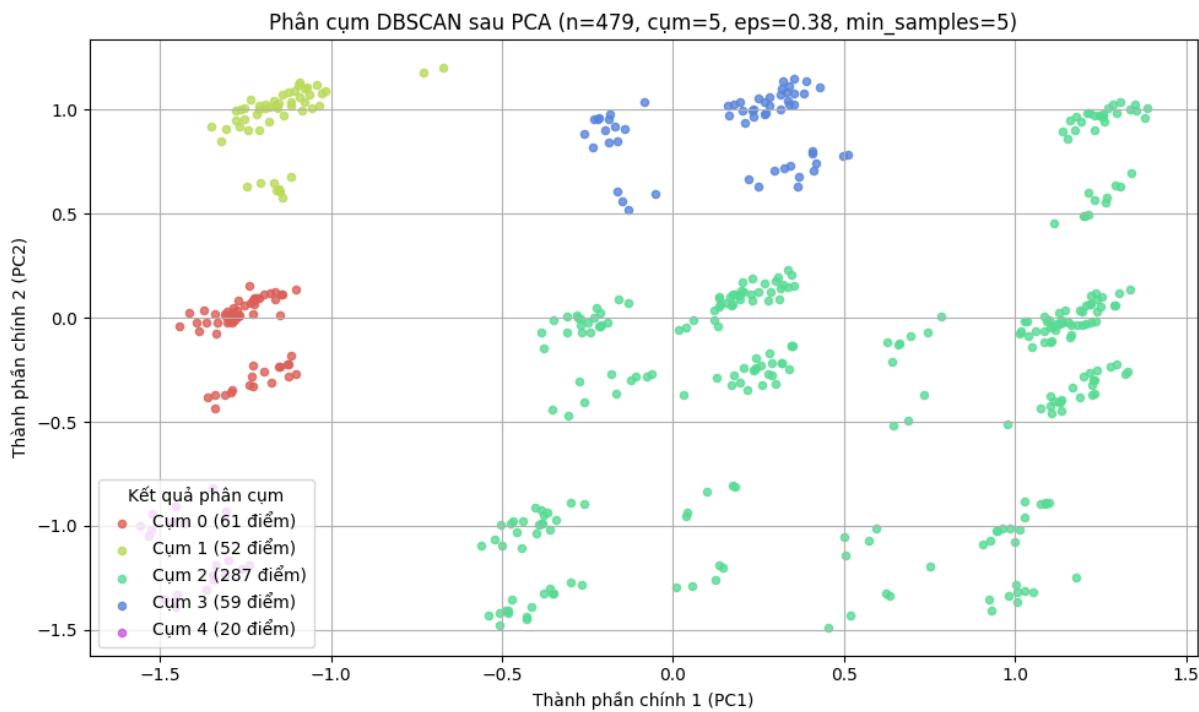
# --- 5. Thống kê cụm ---
label_counts = Counter(labels)
print("Số điểm theo từng cụm:")
for label, count in sorted(label_counts.items()):
    if label == -1:
        print(f" - Nhiều (-1): {count} điểm")
    else:
        print(f" - Cụm {label}: {count} điểm")

# --- 6. Trực quan hóa kết quả ---
plt.figure(figsize=(10, 6))
unique_labels = sorted(set(labels))
colors = sns.color_palette('hls', len(unique_labels))

for label, color in zip(unique_labels, colors):
    mask = (labels == label)
    count = label_counts[label]
    if label == -1:
        plt.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c='red', marker='x', label=f'Nhiều (-1) ({count} điểm)', alpha=0.8)
    else:
        plt.scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c=[color], s=20, label=f'Cụm {label} ({count} điểm)', alpha=0.8)

# --- 7. Thêm tiêu đề ---
num_clusters = len(set(labels)) - (1 if -1 in labels else 0)
plt.title(f'Phân cụm DBSCAN sau PCA (n={len(X)}, cụm={num_clusters}, eps={eps}, min_samples={min_samples})')
plt.xlabel('Thành phần chính 1 (PC1)')
plt.ylabel('Thành phần chính 2 (PC2)')
plt.legend(title='Kết quả phân cụm')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
# --- 8. Lưu kết quả ra file ---
df_clusters = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df_clusters['Cluster'] = labels
df_clusters.to_csv('phan_cum_dbSCAN_allfeature_da_xoa_outliers.csv', index=False)
```



Hình 53: Phân cụm DBSCAN tất cả các biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_allfeature_da_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

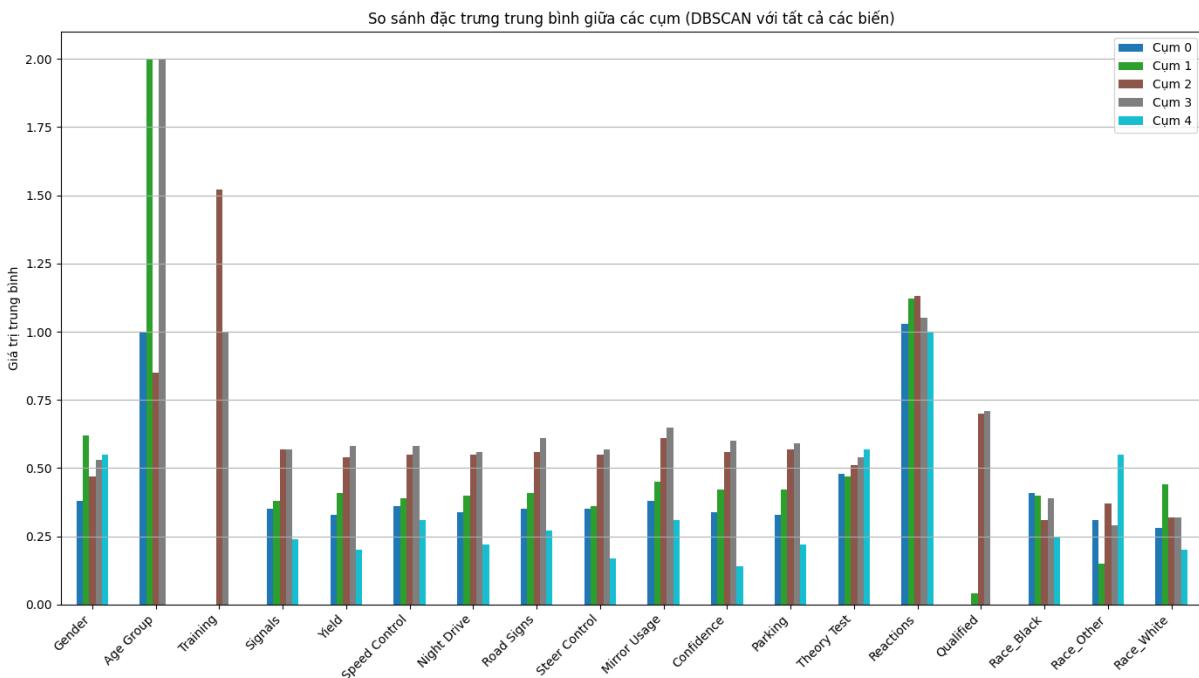
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]
```

```
# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (DBSCAN với tất cả các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 54: So sánh đặc trưng giữa các cụm (DBSCAN với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 41: Nhận xét các cụm DBSCAN – tất cả các biến (Dữ liệu sau khi loại bỏ outliers)

Cụm	Mô tả đặc điểm của các cụm	Nhận xét
0	38% nam, độ tuổi <b>Young Adult</b> , chưa được đào tạo (Training = 0), Reactions = 2 (Fast), Race đa dạng. <b>Kỹ năng: Mirror Usage</b> ~ 44.1/100 • <b>Confidence</b> ~ 45.9/100 • <b>Parking</b> ~ 42.1/100. <b>Qualified</b> : 0%.	Cụm này còn yếu, chưa được đào tạo và thiếu kỹ năng thực hành. Là nhóm cần ưu tiên đào tạo cơ bản.
1	62% nam, độ tuổi <b>Middle Age</b> , chưa đào tạo, Reactions = Fast, chủ yếu Race White.	Có kinh nghiệm tuổi đời, nhưng thiếu đào tạo dẫn

	<b>Kỹ năng:</b> • Mirror Usage ~ 47.7/100 • Confidence ~ 50.2/100 • Parking ~ 47.4/100. <b>Qualified:</b> 4%.	đến chưa thể hiện được kỹ năng tốt.
2	Tuổi trẻ (Teenager–Young Adult), đa số đã được đào tạo (Training ~ 1.5/2 → gần Advanced), Reactions = 2 → Fast, Race đa dạng. <b>Kỹ năng:</b> • Mirror Usage ~ 57.6/100 • Confidence ~ 56.2/100 • Parking ~ 54.5/100. <b>Qualified:</b> 70%.	Là cụm nổi bật nhất, có kiến thức và kỹ năng thực hành tốt. Là ứng viên tiềm năng cao.
3	Middle Age, Training = 1 (Basic), Reactions = 2 → Fast, Race phân bố đều. <b>Kỹ năng:</b> • Mirror Usage ~ 58.3/100 • Confidence ~ 57.4/100 • Parking ~ 55.6/100. <b>Qualified:</b> 71%.	Tương đương cụm 2 về khả năng đậu, nhưng thiên về độ tuổi cao hơn. Nhóm ổn định, phù hợp thi cử.
4	Teenager, chưa đào tạo (Training = 0), Race “Other” chiếm 55%. <b>Kỹ năng rất yếu:</b> • Mirror Usage ~ 40.2/100 • Confidence ~ 42.9/100 • Parking ~ 40.3/100. <b>Qualified:</b> 0%.	Cụm yếu nhất sau khi lọc outliers. Cần được đào tạo từ đầu, ưu tiên rèn luyện kỹ năng thực hành.

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

Việc phân tích kết quả phân cụm DBSCAN trước và sau khi loại bỏ outliers cho thấy sự khác biệt rõ rệt về chất lượng và tính đại diện của các cụm. Trước khi xóa outliers, tập dữ liệu chứa một cụm nhiều (Cluster -1) chiếm tỷ trọng đáng kể với đặc điểm là kỹ năng yếu, đào tạo thấp nhưng lại có phản xạ tốt và mang yếu tố đặc thù về chủng tộc. Điều này khiến kết quả phân cụm bị lệch, khó nhận diện rõ ràng các nhóm học viên điển hình. Ngoài ra, một số cụm (ví dụ Cluster 1) thể

hiện đặc điểm rất kém và thiếu sự đa dạng, phản ánh ảnh hưởng tiêu cực của các điểm dữ liệu nhiễu.

Sau khi loại bỏ outliers, cấu trúc cụm trở nên rõ ràng và dễ phân tích hơn. Các cụm thể hiện được sự phân hóa hợp lý giữa các nhóm: từ những học viên chưa được đào tạo và có kỹ năng yếu (Cụm 0 và 4), đến những nhóm đã được đào tạo, có kỹ năng thực hành tốt và tỷ lệ đậu cao (Cụm 2 và 3). Đặc biệt, cụm 2 và 3 nổi bật với phản xạ nhanh, kỹ năng thực hành khá tốt và tỷ lệ đạt yêu cầu cao nhất, cho thấy mô hình sau khi loại bỏ outliers phản ánh chính xác hơn thực trạng học viên.

Tóm lại, việc xóa bỏ outliers giúp mô hình DBSCAN hoạt động hiệu quả hơn, phân cụm rõ ràng hơn và hỗ trợ việc đưa ra chiến lược đào tạo phù hợp theo từng nhóm học viên. Đây là bước tiền xử lý quan trọng giúp cải thiện độ tin cậy và giá trị thực tiễn của kết quả phân tích.

## B. THUẬT TOÁN GAUSSIAN MATRIX

Gaussian Mixture Model (**GMM**) là một thuật toán phân cụm dựa trên xác suất thống kê, sử dụng mô hình hỗn hợp các phân phối Gaussian để mô tả cấu trúc dữ liệu. Dưới đây là cách hoạt động của GMM:

- Mục tiêu của GMM: Tìm ra sự kết hợp của các phân phối Gaussian mà khi cộng lại sẽ phù hợp nhất với dữ liệu quan sát được.
- Một phân phối Gaussian (phân phối chuẩn) được đặc trưng bởi:
  - + Mean ( $\mu$ ): Trung tâm của phân phối.
  - + Covariance ( $\Sigma$ ): Sự phân tán hoặc hình dạng của phân phối trong không gian.

Trong GMM, mỗi cụm được mô hình hóa như một phân phối Gaussian, và dữ liệu được gán vào các cụm dựa trên xác suất.

### ❖ Cách hoạt động của GMM:

GMM bắt đầu bằng cách giả định rằng dữ liệu có thể được tạo ra từ một hỗn hợp của  $k$  phân phối Gaussian.

### ❖ Các tham số ban đầu bao gồm:

- Mean ( $\mu$ ): Trung tâm cụm ban đầu.
- Covariance ( $\Sigma$ ): Phương sai của từng cụm.
- Mixing Coefficients ( $\pi$ ): Trọng số của từng cụm (tổng của các trọng số là 1).

GMM tính toán xác suất một điểm dữ liệu thuộc về từng cụm bằng công thức của phân phối Gaussian:

$$P(x|\theta) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Trong đó:

- $x$ : Điểm dữ liệu.
- $\mu$ : Trung tâm của cụm.
- $\Sigma$ : Ma trận hiệp phương sai (Covariance Matrix).
- $d$ : Số chiều dữ liệu.

Dựa trên xác suất, GMM gán xác suất trách nhiệm ( $r_{ik}$ ) cho mỗi điểm dữ liệu  $x_i$  thuộc về cụm  $k$ :

$$r_{ik} = \frac{\pi_k P(x_i|\theta_k)}{\sum_{j=1}^k \pi_j P(x_i|\theta_j)}$$

- $r_{ik}$ : Xác suất điểm  $x_i$  thuộc về cụm  $k$  (gọi là "trách nhiệm").
- Kết quả của bước này là mỗi điểm dữ liệu không chỉ thuộc về một cụm duy nhất mà được gán một xác suất cho từng cụm.

- Các tham số của mô hình ( $\mu_k$ ,  $\Sigma_k$ , và  $\pi_k$ ) được cập nhật dựa trên trọng số  $r_{ik}$ :
  - Cập nhật trung tâm cụm (Mean):

$$\mu_k = \frac{\sum_{i=1}^n r_{ik} x_i}{\sum_{i=1}^n r_{ik}}$$

- Cập nhật phương sai (Covariance):

$$\Sigma_k = \frac{\sum_{i=1}^n r_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^n r_{ik}}$$

- Cập nhật trọng số (Mixing Coefficients):

$$\pi_k = \frac{\sum_{i=1}^n r_{ik}}{n}$$

Hội tụ xảy ra khi sự thay đổi trong các tham số (hoặc log-likelihood) giữa các vòng lặp là rất nhỏ.

- Sau khi hội tụ, mỗi điểm dữ liệu được gán vào cụm có xác suất cao nhất.
- Các tham số cuối cùng bao gồm:
  - Trung tâm cụm ( $\mu_k$ ).
  - Hình dạng cụm ( $\Sigma_k$ ).
  - Trọng số cụm ( $\pi_k$ ).

## B.1. Sử dụng 2 biến đặc trưng

### B.1.1. Dữ liệu trước khi loại bỏ outliers

```

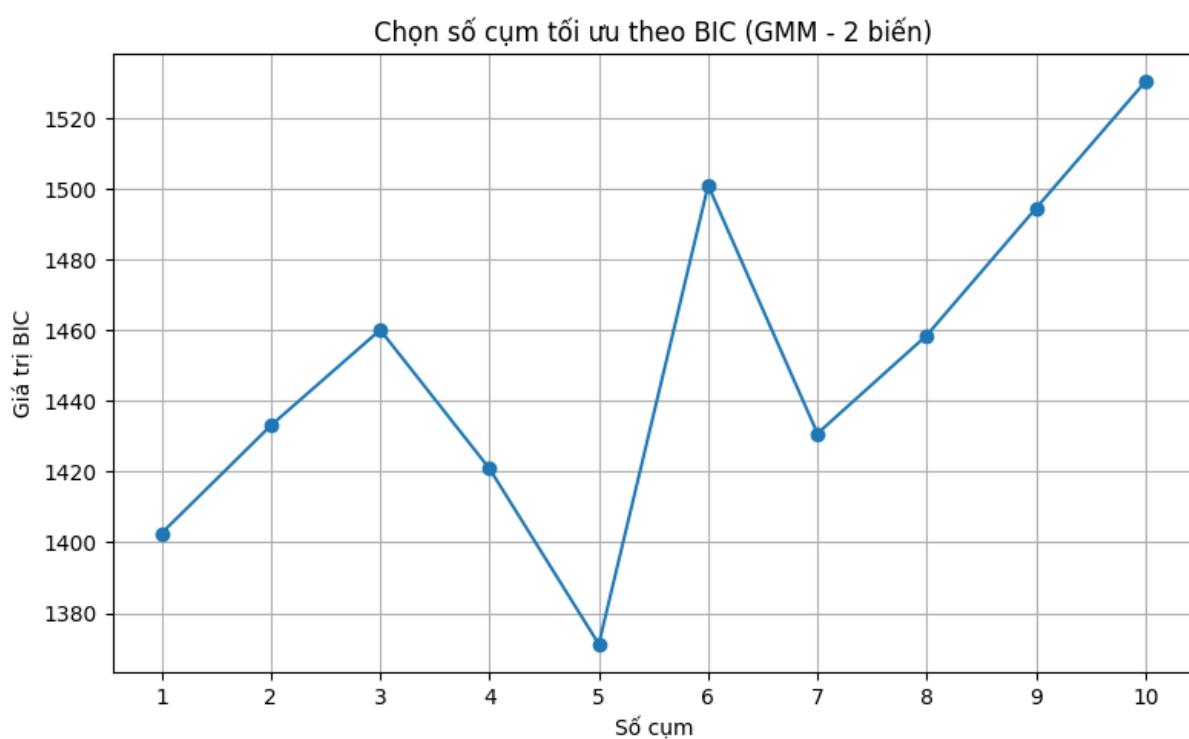
# --- 1. Đọc dữ liệu và chọn 2 biến ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')
features = ['Theory Test', 'Steer Control']
X = df[features]

# --- 3. Tính BIC cho từng số cụm từ 1 đến 10 ---
bic_scores = []
for k in range(1, 11):
    gmm = GaussianMixture(n_components=k, random_state=42)
    gmm.fit(X)
    bic_scores.append(gmm.bic(X))

# --- 4. Vẽ biểu đồ BIC ---
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), bic_scores, marker='o', linestyle='--')
plt.title('Chọn số cụm tối ưu theo BIC (GMM - 2 biến)')
plt.xlabel('Số cụm')
plt.ylabel('Giá trị BIC')
plt.xticks(range(1, 11))
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gợi ý số cụm tối ưu ---
optimal_k = range(1, 11)[np.argmin(bic_scores)]
print(f"Số cụm tối ưu theo BIC: {optimal_k}")

```



Hình 55: Biểu đồ điểm BIC – 2 biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Dựa trên biểu đồ BIC trong mô hình Gaussian Mixture với hai biến đầu vào (Theory Test và Steer Control), số cụm tối ưu được xác định là  $k = 5$ , tại thời điểm BIC đạt giá trị thấp nhất. Điều này cho thấy mô hình với 5 cụm đạt được sự cân bằng tốt giữa độ phù hợp và độ phức tạp, giúp mô tả cấu trúc dữ liệu một cách hiệu quả mà không bị overfitting. Việc lựa chọn số cụm dựa trên tiêu chí BIC là phương pháp phổ biến và đáng tin cậy trong các bài toán phân cụm với dữ liệu liên tục, đặc biệt khi sử dụng Gaussian Mixture Model.

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Chọn 2 biến đã chuẩn hóa ---
features = ['Theory Test', 'Steer Control']
X_scaled = df[features] # dữ liệu đã chuẩn hóa sẵn

# --- 3. Áp dụng Gaussian Mixture ---
n_components = 5
gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=42)
labels = gmm.fit_predict(X_scaled)

# --- 4. Đếm số lượng điểm trong mỗi cụm ---
cluster_counts = Counter(labels)

# --- 5. Vẽ biểu đồ phân cụm ---
plt.figure(figsize=(10, 6))
colors = sns.color_palette("hls", n_components)

# Vẽ các điểm phân cụm
for i in range(n_components):
    mask = (labels == i)
    plt.scatter(X_scaled.loc[mask, 'Theory Test'],
                X_scaled.loc[mask, 'Steer Control'],
                s=30, color=colors[i],
                label=f'Cụm {i} ({cluster_counts[i]} điểm)')

# --- Vẽ tâm cụm ---
centers = gmm.means_
plt.scatter(centers[:, 0], centers[:, 1],
            c='black', marker='x', s=100, linewidths=2, label='Tâm cụm')

plt.title(f'Phân cụm Gaussian Mixture (Theory Test & Steer Control)\nSố cụm = {n_components}')
plt.xlabel('Theory Test (đã chuẩn hóa)')
plt.ylabel('Steer Control (đã chuẩn hóa)')
plt.legend(title='Nhóm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Ghi kết quả ---
df_result = df[features].copy()
df_result['Cluster'] = labels
df_result.to_csv('phan_cum_gmm_2bien_chua_xoa_outliers.csv', index=False)
```

```

# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Chọn 2 biến đã chuẩn hóa ---
features = ['Theory Test', 'Steer Control']
X_scaled = df[features] # dữ liệu đã chuẩn hóa sẵn

# --- 3. Áp dụng Gaussian Mixture ---
n_components = 5
gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=42)
labels = gmm.fit_predict(X_scaled)

# --- 4. Đếm số lượng điểm trong mỗi cụm ---
cluster_counts = Counter(labels)

# --- 5. Vẽ biểu đồ phân cụm ---
plt.figure(figsize=(10, 6))
colors = sns.color_palette("hls", n_components)

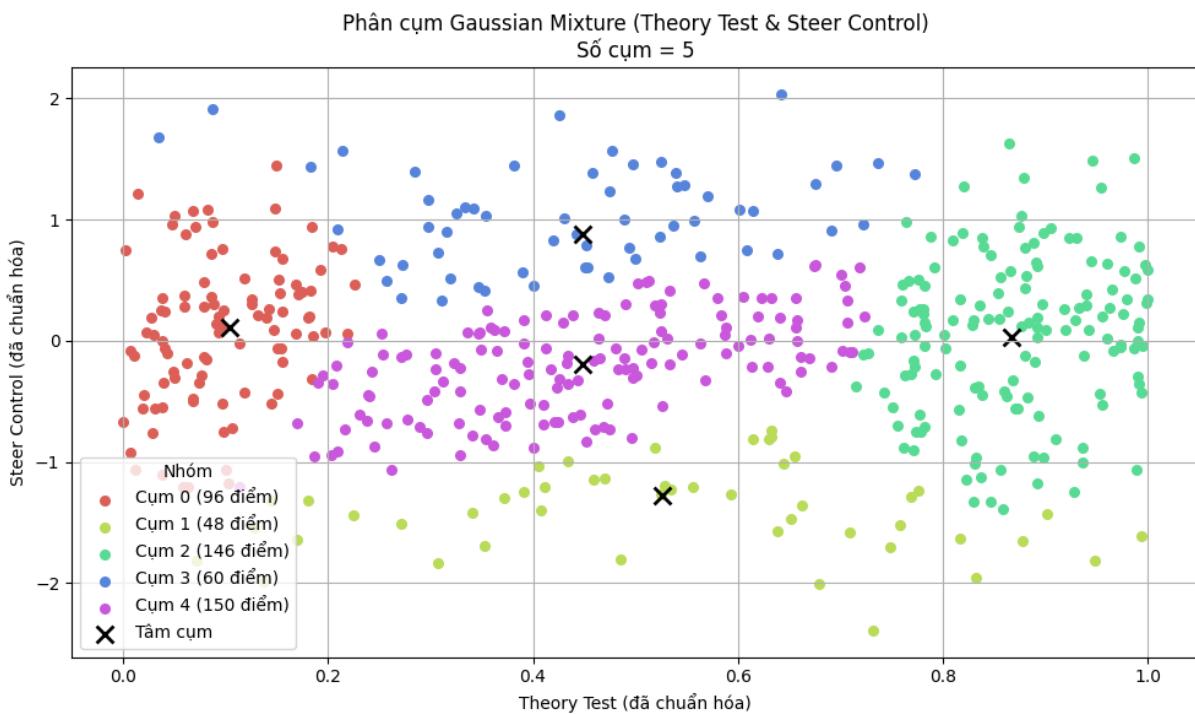
# Vẽ các điểm phân cụm
for i in range(n_components):
    mask = (labels == i)
    plt.scatter(X_scaled.loc[mask, 'Theory Test'],
                X_scaled.loc[mask, 'Steer Control'],
                s=30, color=colors[i],
                label=f'Cụm {i} ({cluster_counts[i]} điểm)')

# --- Vẽ tâm cụm ---
centers = gmm.means_
plt.scatter(centers[:, 0], centers[:, 1],
            c='black', marker='x', s=100, linewidths=2, label='Tâm cụm')

plt.title(f'Phân cụm Gaussian Mixture (Theory Test & Steer Control)\nSố cụm = {n_components}')
plt.xlabel('Theory Test (đã chuẩn hóa)')
plt.ylabel('Steer Control (đã chuẩn hóa)')
plt.legend(title='Nhóm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Ghi kết quả ---
df_result = df[features].copy()
df_result['Cluster'] = labels
df_result.to_csv('phan_cum_gmm_2bien_chua_xoa_outliers.csv', index=False)

```



Hình 56: Phân cụm GMM – 2 biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_chua_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

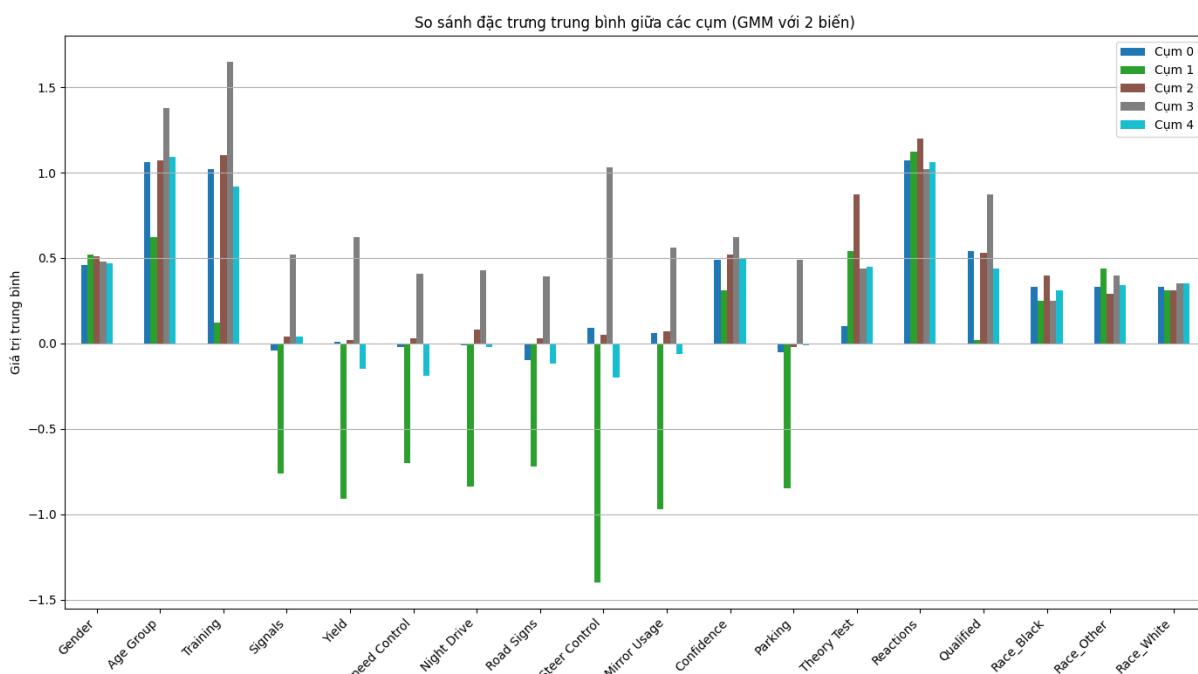
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]
```

```
# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (GMM với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 57: So sánh đặc trưng giữa các cụm (GMM với 2 biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 42: Nhận xét các cụm GMM – 2 các biến (Dữ liệu trước khi loại bỏ outliers)

Cụm	Đặc điểm nổi bật	Nhận xét
0	Kỹ năng trung bình: Steer ~ <b>49.4</b> , Mirror ~ 47.8, Confidence ~ 60.9, Parking ~ 49.5, Qualified ~ <b>70%</b>	Nhóm khá ổn, có kỹ năng tương đối và tỷ lệ đạt cao
1	Rất yếu: Steer ~ <b>27.0</b> , Mirror ~ 33.4, Confidence ~ 58.7, Parking ~ 41.5, Qualified ~ <b>50%</b> (nhưng có thể do phản xạ nhanh)	Nhóm yếu nhất, cần được đào tạo kỹ năng điều khiển xe cơ bản

2	Ôn định: Steer ~ 48.8, Mirror ~ 48.0, Confidence ~ 61.2, Parking ~ 49.8, Qualified ~ <b>70%</b>	Tương đối ổn, kỹ năng và lý thuyết khá tốt
3	Xuất sắc: Steer ~ <b>63.5</b> , Mirror ~ 54.8, Confidence ~ 62.4, Parking ~ <b>54.9</b> , Qualified ~ <b>80%</b>	Cụm tốt nhất, thể hiện khả năng lái vững vàng và kinh nghiệm nhiều hơn
4	Trung bình yếu: Steer ~ 45.0, Mirror ~ 46.2, Confidence ~ 61.0, Parking ~ 49.9, Qualified ~ <b>60%</b>	Nhóm trung bình nhưng vẫn có tiềm năng nếu được rèn luyện thêm

(Nguồn: Tác giả tổng hợp, 2025)

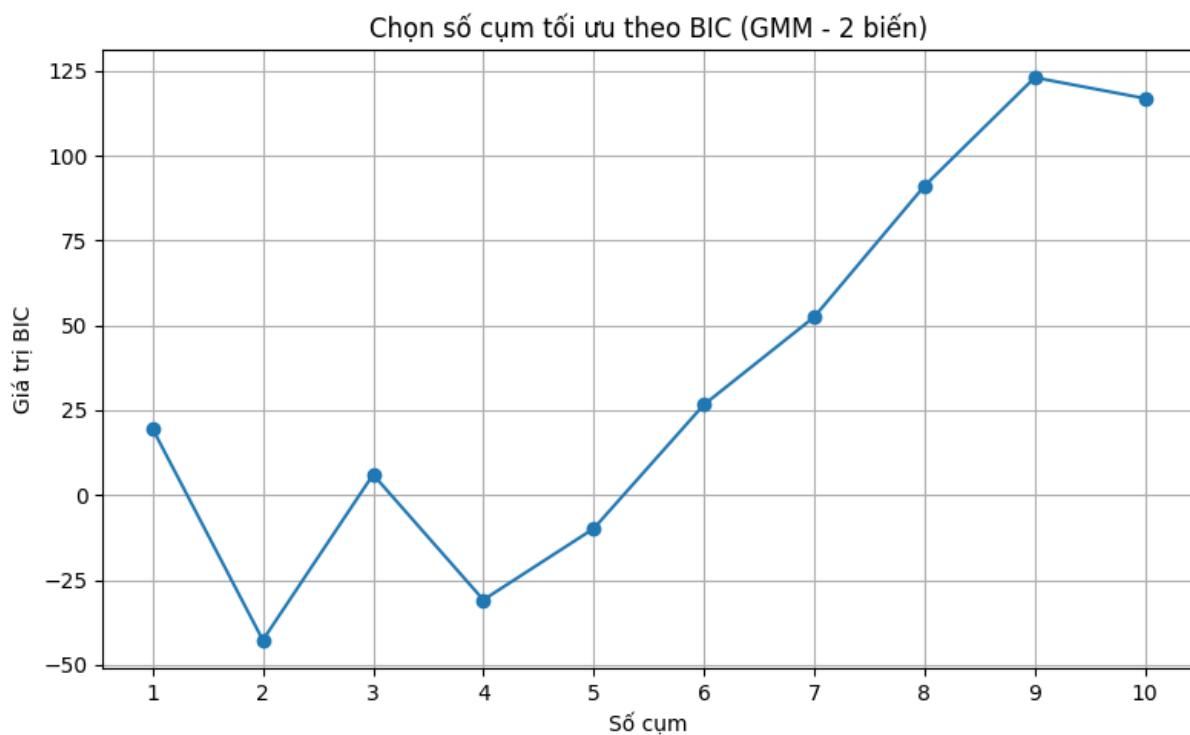
### B.1.2. Dữ liệu sau khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu và chọn 2 biến ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')
features = ['Theory Test', 'Steer Control']
X = df[features]

# --- 3. Tính BIC cho từng số cụm từ 1 đến 10 ---
bic_scores = []
for k in range(1, 11):
    gmm = GaussianMixture(n_components=k, random_state=42)
    gmm.fit(X)
    bic_scores.append(gmm.bic(X))

# --- 4. Vẽ biểu đồ BIC ---
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), bic_scores, marker='o', linestyle='--')
plt.title('Chọn số cụm tối ưu theo BIC (GMM - 2 biến)')
plt.xlabel('Số cụm')
plt.ylabel('Giá trị BIC')
plt.xticks(range(1, 11))
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gợi ý số cụm tối ưu ---
optimal_k = range(1, 11)[np.argmin(bic_scores)]
print(f"Số cụm tối ưu theo BIC: {optimal_k}")
```



Hình 58: Biểu đồ điểm BIC – 2 biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Dựa trên biểu đồ BIC trong mô hình Gaussian Mixture với hai biến đầu vào (Theory Test và Steer Control), số cụm tối ưu được xác định là  $k = 2$ , tại thời điểm BIC đạt giá trị thấp nhất. Điều này cho thấy mô hình với 2 cụm đạt được sự cân bằng tốt giữa độ phù hợp và độ phức tạp, giúp mô tả cấu trúc dữ liệu một cách hiệu quả mà không bị overfitting. Việc lựa chọn số cụm dựa trên tiêu chí BIC là phương pháp phổ biến và đáng tin cậy trong các bài toán phân cụm với dữ liệu liên tục, đặc biệt khi sử dụng Gaussian Mixture Model.

```

# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- 2. Chọn 2 biến đã chuẩn hóa ---
features = ['Theory Test', 'Steer Control']
X_scaled = df[features] # dữ liệu đã chuẩn hóa sẵn

# --- 3. Áp dụng Gaussian Mixture ---
n_components = 2
gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=42)
labels = gmm.fit_predict(X_scaled)

# --- 4. Đếm số lượng điểm trong mỗi cụm ---
cluster_counts = Counter(labels)

# --- 5. Vẽ biểu đồ phân cụm ---
plt.figure(figsize=(10, 6))
colors = sns.color_palette("hls", n_components)

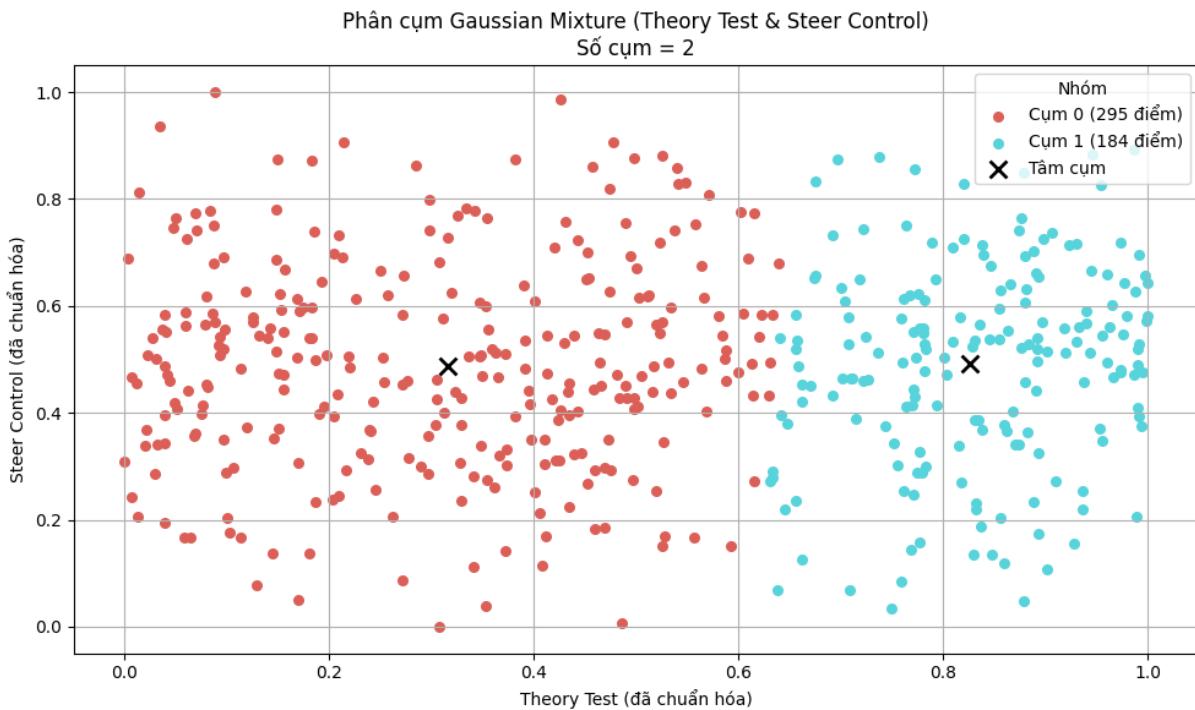
# Vẽ các điểm phân cụm
for i in range(n_components):
    mask = (labels == i)
    plt.scatter(X_scaled.loc[mask, 'Theory Test'],
                X_scaled.loc[mask, 'Steer Control'],
                s=30, color=colors[i],
                label=f'Cụm {i} ({cluster_counts[i]} điểm)')

# --- Vẽ tâm cụm---
centers = gmm.means_
plt.scatter(centers[:, 0], centers[:, 1],
            c='black', marker='x', s=100, linewidths=2, label='Tâm cụm')

plt.title(f'Phân cụm Gaussian Mixture (Theory Test & Steer Control)\nSố cụm = {n_components}')
plt.xlabel('Theory Test (đã chuẩn hóa)')
plt.ylabel('Steer Control (đã chuẩn hóa)')
plt.legend(title='Nhóm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Ghi kết quả ---
df_result = df[features].copy()
df_result['Cluster'] = labels
df_result.to_csv('phan_cum_gmm_2bien_da_xoa_outliers.csv', index=False)

```



Hình 59: Phân cụm GMM – 2 biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_da_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

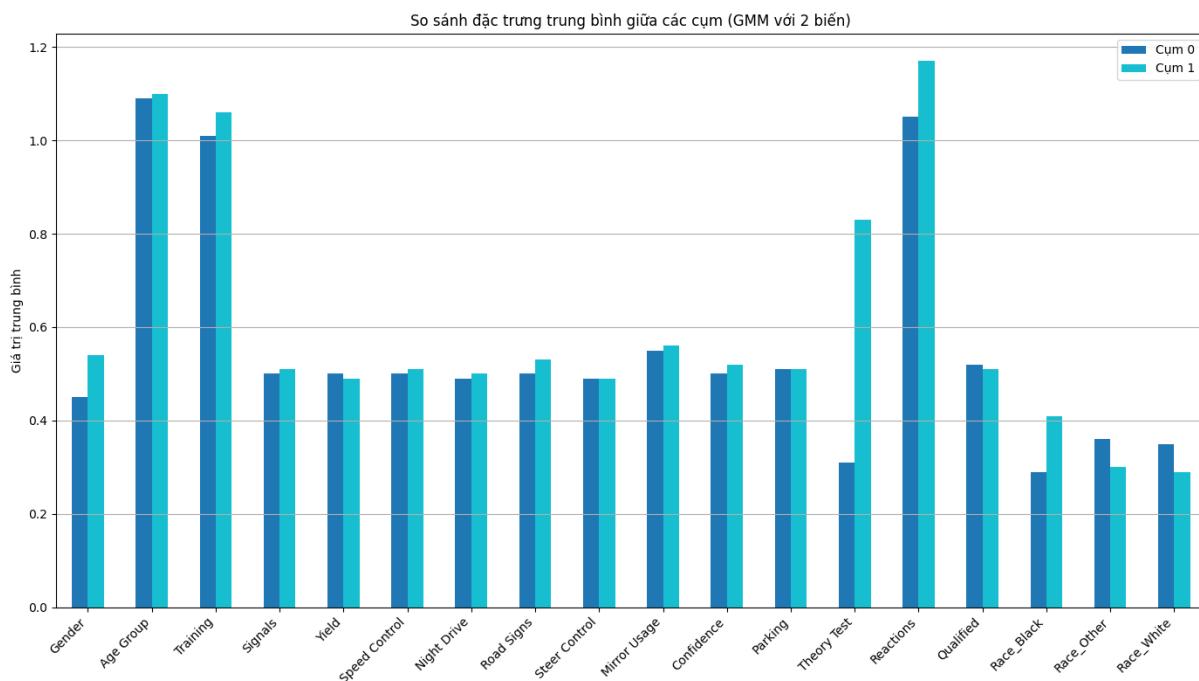
# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]
```

## # 7. Vẽ biểu đồ

```
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (GMM với 2 biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 60: So sánh đặc trưng giữa các cụm (GMM với 2 biến) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 43: Nhận xét các cụm GMM – 2 các biến (Dữ liệu sau khi loại bỏ outliers)

Cụm	Đặc điểm nổi bật (giá trị trung bình)	Nhận xét tổng quát
0	Ôn định: Steer ~ <b>55.4</b> , Mirror ~ 54.7, Confidence ~ 61.0, Parking ~ <b>55.1</b> , Theory Test ~ 48.1, Qualified ~ <b>70%</b>	Nhóm ổn định, kỹ năng lái thực hành và lý thuyết đồng đều, là nhóm học viên tiềm năng
1	Tốt tương đương: Steer ~ <b>55.4</b> , Mirror ~ 54.8, Confidence ~ 61.2, Parking ~ 55.1, Theory Test ~ <b>53.3</b> , Qualified ~ <b>70%</b>	Tương tự cụm 0 nhưng nổi trội hơn về lý thuyết, thể hiện ứng viên có tiềm năng lý thuyết cao

(Nguồn: Tác giả tổng hợp, 2025)

❖ **Nhận xét:**

Sau khi loại bỏ outliers, số lượng cụm giảm từ **5 xuống 2**, các cụm trở nên **đồng đều hơn và ít bị ảnh hưởng bởi giá trị cực đoan**. Cả hai cụm còn lại đều có kỹ năng tốt và tỷ lệ đạt cao (70%), cho thấy việc loại bỏ nhiễu đã giúp mô hình phân cụm tập trung vào nhóm học viên có tiềm năng thực sự.

## B.2. Sử dụng tất cả các biến

### B.2.1. Dữ liệu trước khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

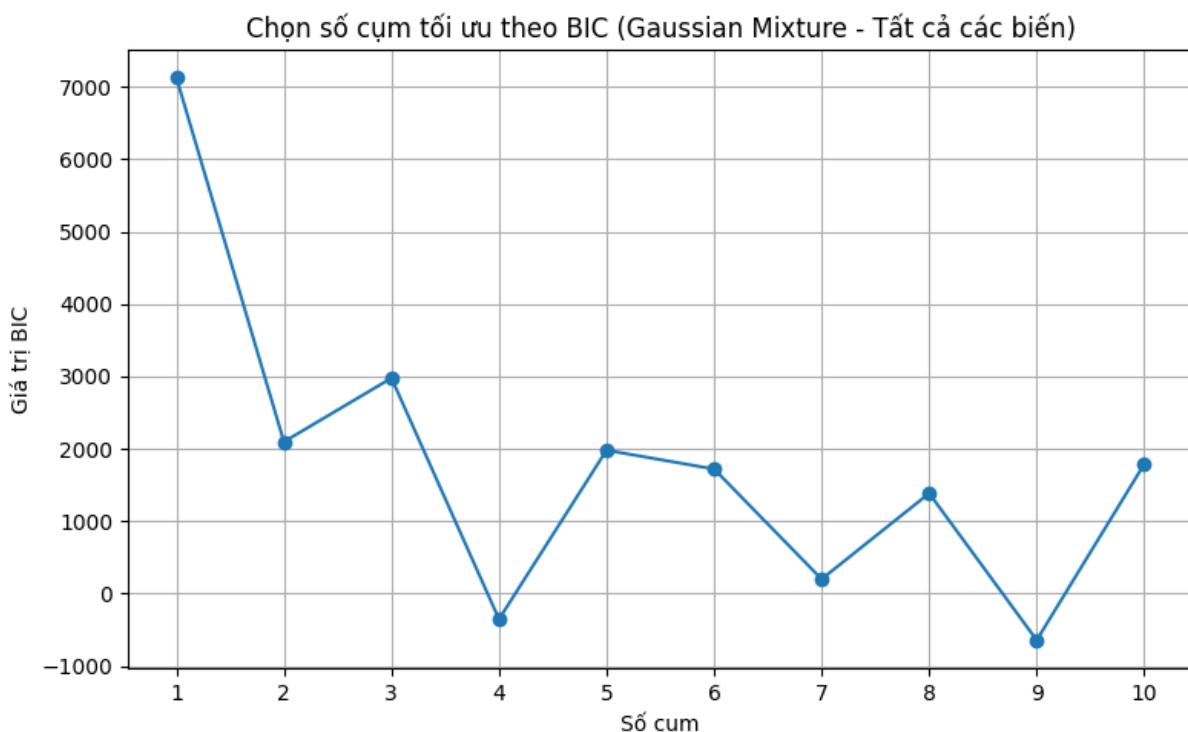
# --- 2. Loại bỏ cột ID nếu có ---
X = df.drop(columns=['Applicant ID']) if 'Applicant ID' in df.columns else df.copy()

# --- 4. Tính chỉ số BIC cho số cụm từ 1 đến 10 ---
bic_scores = []
n_components_range = range(1, 11)

for k in n_components_range:
    gmm = GaussianMixture(n_components=k, random_state=42)
    gmm.fit(X)
    bic_scores.append(gmm.bic(X))

# --- 5. Vẽ biểu đồ BIC để chọn số cụm tối ưu ---
plt.figure(figsize=(8, 5))
plt.plot(n_components_range, bic_scores, marker='o', linestyle='--')
plt.title('Chọn số cụm tối ưu theo BIC (Gaussian Mixture - Tất cả các biến)')
plt.xlabel('Số cụm')
plt.ylabel('Giá trị BIC')
plt.xticks(n_components_range)
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Gợi ý số cụm tối ưu ---
optimal_k = n_components_range[np.argmin(bic_scores)]
print(f"Số cụm tối ưu theo BIC: {optimal_k}")
```



Hình 61: Biểu đồ điểm BIC – tất cả các biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Dựa trên biểu đồ BIC trong mô hình Gaussian Mixture với tất cả các biến đầu vào ngoại trừ biến Applicant ID, số cụm tối ưu được xác định là  $k = 9$ , tại thời điểm BIC đạt giá trị thấp nhất. Điều này cho thấy mô hình với 9 cụm đạt được sự cân bằng tốt giữa độ phù hợp và độ phức tạp, giúp mô tả cấu trúc dữ liệu một cách hiệu quả mà không bị overfitting. Việc lựa chọn số cụm dựa trên tiêu chí BIC là phương pháp phổ biến và đáng tin cậy trong các bài toán phân cụm với dữ liệu liên tục, đặc biệt khi sử dụng Gaussian Mixture Model.

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# --- 2. Loại bỏ cột ID ---
X = df.drop(columns=["Applicant ID"])

# --- 3. Giảm chiều bằng PCA để dễ trực quan hóa ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# --- 4. Áp dụng Gaussian Mixture ---
n_components = 9
gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=42)
gmm_labels = gmm.fit_predict(X_pca)

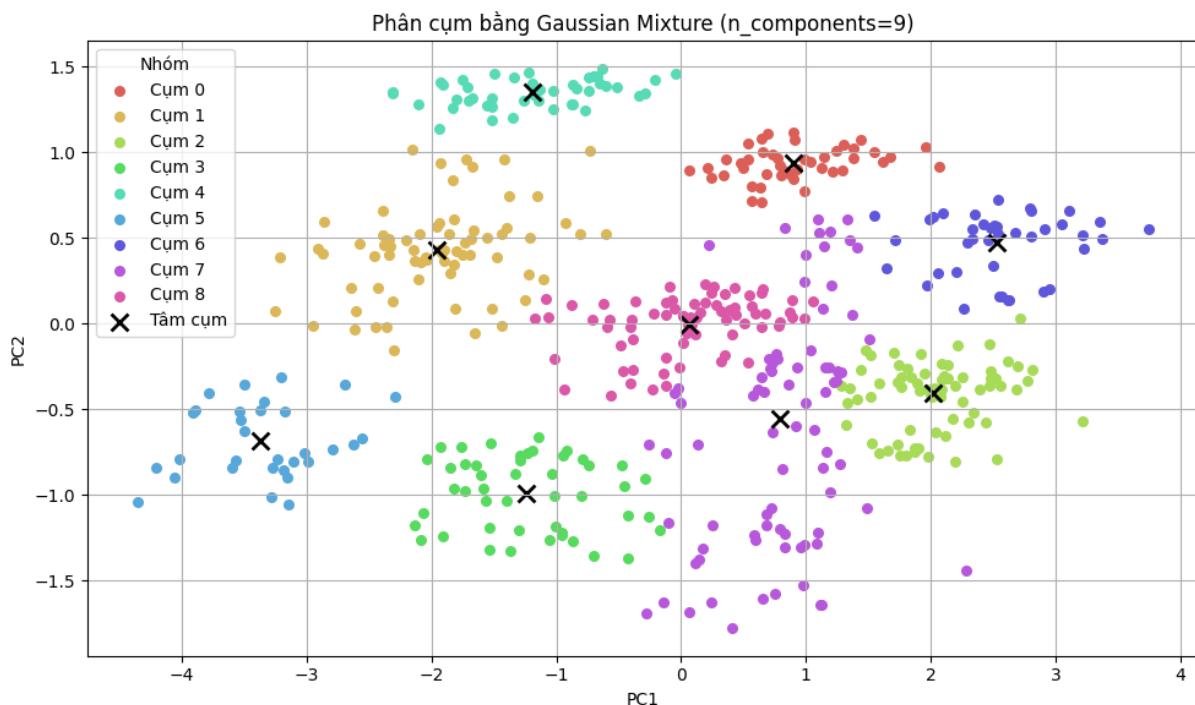
# --- 5. Vẽ phân cụm ---
plt.figure(figsize=(10, 6))
palette = sns.color_palette("hls", n_components)

for i in range(n_components):
    plt.scatter(X_pca[gmm_labels == i, 0], X_pca[gmm_labels == i, 1],
                s=30, label=f'Cụm {i}', color=palette[i])

# --- Vẽ tâm cụm---
centers = gmm.means_
plt.scatter(centers[:, 0], centers[:, 1],
            c='black', marker='x', s=100, linewidths=2, label='Tâm cụm')

plt.title(f'Phân cụm bằng Gaussian Mixture (n_components={n_components})')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(title='Nhóm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Ghi nhãn cụm vào DataFrame và lưu ra file CSV ---
df_gmm = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df_gmm['Cluster'] = gmm_labels
df_gmm.to_csv('phan_cum_gmm_allfeature_chua_xoa_outliers.csv', index=False)
```



Hình 62: Phân cụm GMM – tất cả các biến (Dữ liệu trước khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```
# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Data_Processing.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_allfeature_chua_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=["Applicant ID", "Cluster"]).columns.tolist()

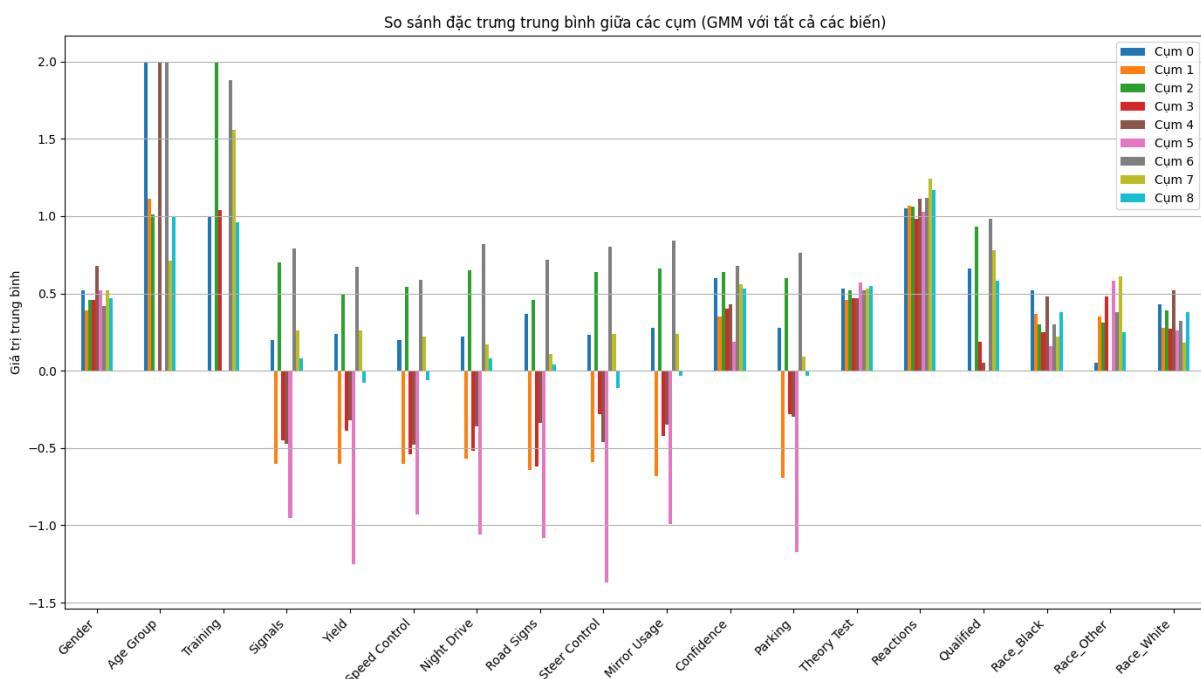
# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (GMM với tất cả các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```



Hình 63: So sánh đặc trưng trung bình giữa các cụm (GMM với tất cả các biến) – Dữ liệu trước khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 44: Nhận xét các cụm GMM – tất cả các biến (Dữ liệu trước khi loại bỏ outliers)

Cụm	Đặc điểm nổi bật	Nhận xét
0	Tuổi cao nhất (Age=2), Training = 1, kỹ năng và tự tin trung bình khá, tỷ lệ đậu cao (66%)	Nhóm ổn định, được đào tạo và có tiềm năng thi tốt
1	Tuổi trung bình, chưa đào tạo (Training=0), kỹ năng kém nhất, tự tin thấp, không ai đậu	Nhóm yếu nhất, cần ưu tiên đào tạo
2	Training gần như đầy đủ (2.0), kỹ năng rất tốt, tỷ lệ đậu rất cao (93%)	Nhóm xuất sắc, sẵn sàng thi sát hạch
3	Tuổi nhỏ nhất, kỹ năng dưới trung bình, tỷ lệ đậu thấp (19%)	Nhóm học viên trẻ, cần rèn luyện thêm
4	Tuổi cao, chưa đào tạo, kỹ năng yếu, tỷ lệ đậu rất thấp (5%)	Lớn tuổi nhưng thiếu kiến thức, cần hỗ trợ thêm

5	Tệ nhát toàn bộ: chưa đào tạo, kỹ năng cực yếu, không ai đậu	Nhóm loại bỏ hoặc cần đào tạo đặc biệt
6	Training gần Advanced, kỹ năng rất tốt, tỷ lệ đậu cực cao (98%)	Nhóm mục tiêu cho cấp bằng
7	Training khá (1.56), kỹ năng tốt, đậu cao (78%)	Học viên có tiềm năng
8	Kỹ năng trung bình, đã qua đào tạo, tỷ lệ đậu 58%	Nhóm trung bình khá

(Nguồn: Tác giả tổng hợp, 2025)

### B.2.2. Dữ liệu sau khi loại bỏ outliers

```
# --- 1. Đọc dữ liệu ---
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# --- 2. Loại bỏ cột ID ---
X = df.drop(columns=["Applicant ID"])

# --- 3. Giảm chiều bằng PCA để dễ trực quan hóa ---
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# --- 4. Áp dụng Gaussian Mixture ---
n_components = 9
gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=42)
gmm_labels = gmm.fit_predict(X_pca)

# --- 5. Vẽ phân cụm ---
plt.figure(figsize=(10, 6))
palette = sns.color_palette("hls", n_components)

for i in range(n_components):
    plt.scatter(X_pca[gmm_labels == i, 0], X_pca[gmm_labels == i, 1],
                s=30, label=f'Cụm {i}', color=palette[i])

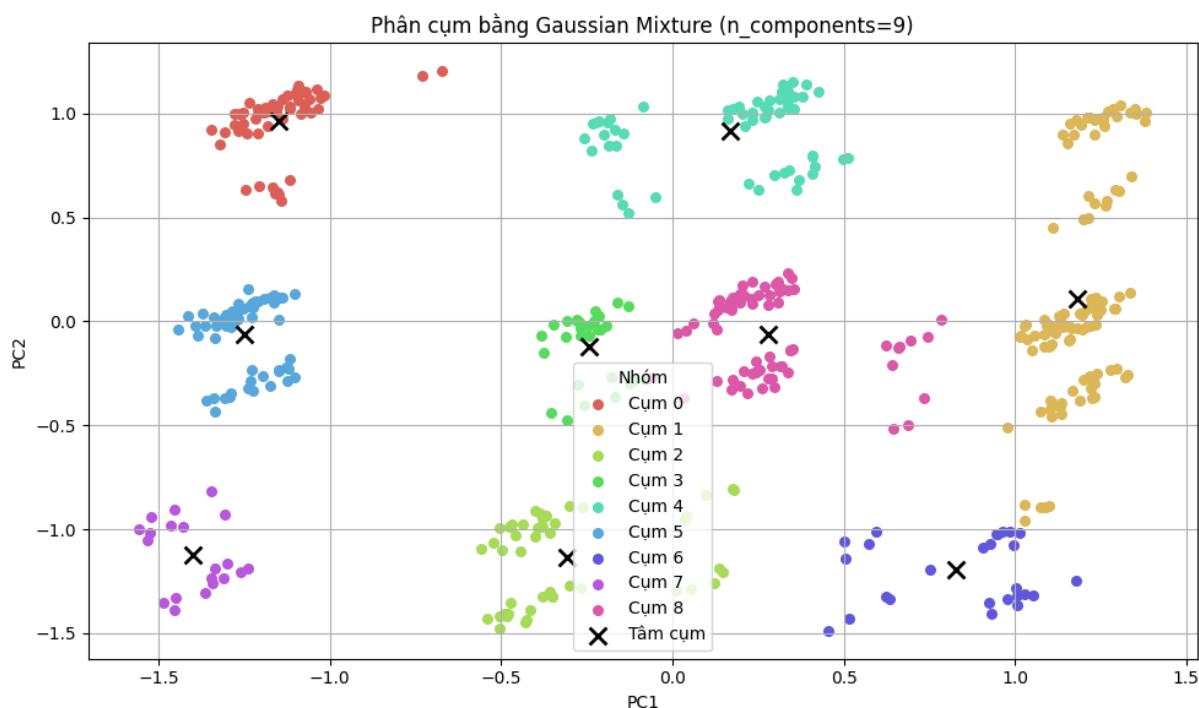
# --- Vẽ tâm cụm---
centers = gmm.means_
plt.scatter(centers[:, 0], centers[:, 1],
            c='black', marker='x', s=100, linewidths=2, label='Tâm cụm')
```

```

plt.title(f'Phân cụm bằng Gaussian Mixture (n_components={n_components})')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(title='Nhóm')
plt.grid(True)
plt.tight_layout()
plt.show()

# --- 6. Ghi nhãn cụm vào DataFrame và lưu ra file CSV ---
df_gmm = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df_gmm['Cluster'] = gmm_labels
df_gmm.to_csv('phan_cum_gmm_allfeature_da_xoa_outliers.csv', index=False)

```



Hình 64: Phân cụm GMM – tất cả các biến (Dữ liệu sau khi loại bỏ outliers)

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

```

# 1. Đọc dữ liệu gốc
df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/Tien xu ly dl da xoa outliers.csv')

# 2. Đọc kết quả phân cụm (giả sử có cùng thứ tự với df gốc)
clusters_df = pd.read_csv('/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_allfeature_da_xoa_outliers.csv')

# Gán cột 'Cluster' từ file phân cụm vào df chính
df['Cluster'] = clusters_df['Cluster']

# 3. Loại bỏ các cột không cần khi tính trung bình (giữ lại chỉ các đặc trưng số)
features = df.drop(columns=['Applicant ID', 'Cluster']).columns.tolist()

# 4. Tính trung bình theo từng cụm
cluster_summary = df.groupby("Cluster")[features].mean().round(2)

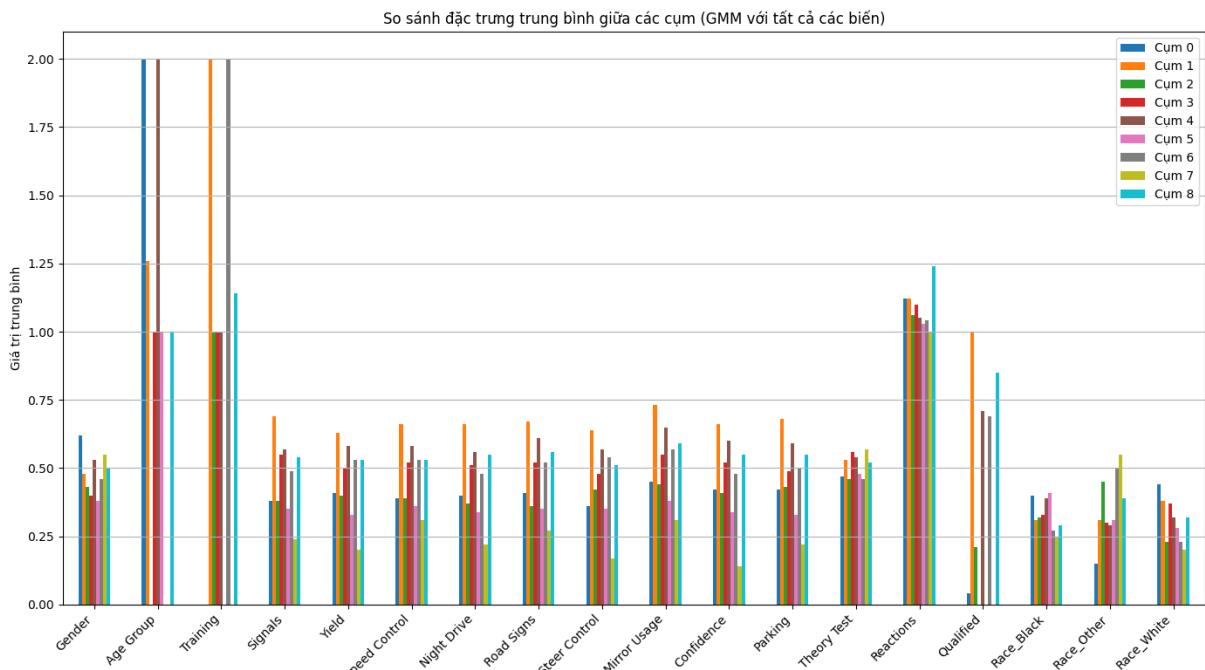
# 5. Hiển thị toàn bộ bảng
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 0)

print("Giá trị trung bình của từng cụm:")
print(cluster_summary)

# 6. Chuẩn bị dữ liệu để vẽ biểu đồ
plot_data = cluster_summary.T
plot_data.columns = [f'Cụm {i}' for i in plot_data.columns]

# 7. Vẽ biểu đồ
plot_data.plot(kind="bar", figsize=(14, 8), colormap="tab10")
plt.title("So sánh đặc trưng trung bình giữa các cụm (GMM với tất cả các biến)")
plt.ylabel("Giá trị trung bình")
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```



Hình 65: So sánh đặc trưng giữa các cụm (GMM với tất cả các biến) – Dữ liệu sau khi loại bỏ outliers

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

Bảng 45: Nhận xét các cụm GMM – tất cả các biến (Dữ liệu sau khi loại bỏ outliers)

Cụm	Mô tả cụm	Nhận xét
0	Giới tính: Nam 52%. Tuổi: Older Adult. Đào tạo: Basic. Kỹ năng trung bình khá (Parking ~64, Steer ~65, Mirror ~67). Tự tin: 60. Phản xạ: Trung bình. Tỷ lệ đậu: 66%.	Nhóm học viên ổn định, có đào tạo, kỹ năng và thái độ khá tốt. Đủ điều kiện thi.
1	Nam 39%. Tuổi: Young Adult. Đào tạo: Unknown. Kỹ năng rất yếu (Steer ~35, Mirror ~28). Tự tin: 35. Tỷ lệ đậu: 0%.	Nhóm yếu nhất, thiếu kỹ năng và đào tạo. Cần huấn luyện từ đầu.
2	Nam 46%. Tuổi: Young Adult. Đào tạo: Advanced. Kỹ năng rất tốt (Steer ~73, Mirror ~75, Parking ~76). Tự tin: 64. Tỷ lệ đậu: 93%.	Nhóm xuất sắc nhất, rất tiềm năng, có thể được cấp phép ngay.
3	Nam 46%. Tuổi: Teenager. Đào tạo: Basic. Kỹ năng trung bình yếu (Steer ~49, Mirror ~46). Tự tin: 40. Tỷ lệ đậu: 19%.	Học viên trẻ, cần rèn luyện thêm kỹ năng và độ tự tin.
4	Nam 68%. Tuổi: Older Adult. Đào tạo: Unknown. Kỹ năng yếu (Steer ~43, Confidence ~43). Tỷ lệ đậu: 5%.	Nhóm lớn tuổi nhưng thiếu kỹ năng, cần đào tạo bổ sung.
5	Nam 52%. Tuổi: Teenager. Đào tạo: Unknown. Kỹ năng rất yếu (Steer ~23, Confidence thấp ~19). Tỷ lệ đậu: 0%.	Nhóm yếu nhất, có thể là dữ liệu nhiễu hoặc học viên cần hỗ trợ đặc biệt.
6	Nam 42%. Tuổi: Older Adult. Đào tạo: Advanced. Kỹ năng vượt trội (Steer ~80, Mirror ~83). Tự tin: 68. Tỷ lệ đậu: 98%.	Nhóm ưu tú, sẵn sàng vượt qua kỳ thi.
7	Nam 52%. Tuổi: Young Adult. Đào tạo: ~Advanced. Kỹ năng khá tốt (Steer ~65, Mirror ~65). Tự tin: 56. Tỷ lệ đậu: 78%.	Học viên khá, có tiềm năng phát triển.

8	Nam 47%. Tuổi: Young Adult. Đào tạo: Basic. Kỹ năng trung bình (Steer ~54, Mirror ~55). Tự tin: 53. Tỷ lệ đậu: 58%.	Nhóm học viên khá, nên luyện tập thêm để thi tốt hơn.
---	---	---

(Nguồn: Tác giả tổng hợp, 2025)

#### ❖ Nhận xét:

**Trước khi xóa outliers:** GMM phân cụm ra nhiều nhóm, trong đó có vài cụm chứa dữ liệu rất yếu (Cluster 5), ảnh hưởng đến kết quả tổng thể.

**Sau khi xóa outliers:** Các cụm rõ ràng và logic hơn, dễ diễn giải và phù hợp thực tế hơn. Một số cụm như **1, 4, 6, 8** là nhóm mạnh, trong khi **0, 5, 7** là nhóm yếu cần hỗ trợ.

**Kết luận:** Việc loại bỏ outliers giúp cải thiện độ chính xác và hiệu quả của phân cụm GMM, từ đó giúp xây dựng lộ trình đào tạo phù hợp với từng nhóm học viên.

## C. ĐÁNH GIÁ HIỆU QUẢ PHÂN CỤM CỦA CÁC MÔ HÌNH

### C.1. Sử dụng tất cả các biến đầu vào với dữ liệu trước khi loại bỏ outliers

```
# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/ketqua_kmeans_full.csv")
gmm_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_allfeature_chua_xoa_outliers.csv")
dbscan_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_dbscan_allfeature_chua_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["PC1", "PC2"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)
```

```

# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/ketqua_kmeans_2bien.csv")
gmm_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_chua_xoa_outliers.csv")
dbSCAN_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["Theory Test", "Steer Control"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)

```

Bảng 46: Đánh giá phân cụm với tất cả các biến – Dữ liệu trước khi loại bỏ outliers

Thuật toán	Silhouette	Davies-Bouldin	Calinski-Harabasz
KMeans	0,1303	2,1742	139,83
GMM	<b>0,3748</b>	0,8662	<b>570,73</b>
DBSCAN	0,3657	<b>0,5394</b>	123,15

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### ❖ Nhận xét:

**GMM** là thuật toán có hiệu suất tốt nhất với độ tách cụm cao (Silhouette) và độ chòng cụm thấp (Davies-Bouldin).

**DBSCAN** cho Davies-Bouldin tốt nhất nhưng Calinski-Harabasz kém, nghĩa là cụm không rõ ràng nhưng ít chòng lấn.

**KMeans** hoạt động kém nhất khi dữ liệu còn chứa outliers.

## C.2. Sử dụng tất cả biến đầu vào với dữ liệu sau khi loại bỏ outliers

```
# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_da_xoa_outliers_full.csv")
gmm_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_allfeature_da_xoa_outliers.csv")
dbSCAN_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_allfeature_da_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["PC1", "PC2"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)
```

```
# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_2bien.csv")
gmm_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_chua_xoa_outliers.csv")
dbSCAN_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["Theory Test", "Steer Control"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)
```

Bảng 47: Đánh giá phân cụm với tất cả các biến – Dữ liệu sau khi loại bỏ outliers

Thuật toán	Silhouette	Davies-Bouldin	Calinski-Harabasz
KMeans	0,1680	1,8390	93,78
GMM	<b>0,5579</b>	<b>0,5421</b>	<b>660,75</b>
DBSCAN	0,3514	0,6497	211,82

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

❖ **Nhận xét:**

Việc **loại bỏ outliers** giúp cải thiện tất cả các thuật toán. GMM đạt hiệu quả cao nhất.

**Calinski-Harabasz tăng mạnh**, phản ánh cụm có sự phân tách và độ đặc trưng rõ hơn.

**KMeans** được cải thiện nhưng vẫn yếu hơn 2 thuật toán còn lại.

### C.3. Sử dụng 2 biến đặc trưng với dữ liệu trước khi loại bỏ outliers

```
# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("./content/drive/MyDrive/khai pha du lieu/ketqua_kmeans_2bien.csv")
gmm_df = pd.read_csv("./content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_chua_xoa_outliers.csv")
dbSCAN_df = pd.read_csv("./content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["Theory Test", "Steer Control"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)
```

```

# Đánh giá cho GMM
X_gmm = gmm_df[["Theory Test", "Steer Control"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)

# Đánh giá cho DBSCAN
X_dbSCAN = dbSCAN_df[["Steer Control", "Night Drive"]]
y_dbSCAN = dbSCAN_df["Cluster"]
dbSCAN_metrics = evaluate_clustering(X_dbSCAN.values, y_dbSCAN.values)

# Tổng hợp kết quả
results = pd.DataFrame({
    "KMeans": kmeans_metrics,
    "GMM": gmm_metrics,
    "DBSCAN": dbSCAN_metrics
})

# In kết quả
print("==> Đánh giá các thuật toán phân cụm ==")
print(results.round(4))

```

Bảng 48: Đánh giá phân cụm với 2 biến – Dữ liệu trước khi loại bỏ outliers

Thuật toán	Silhouette	Davies-Bouldin	Calinski-Harabasz
KMeans	0,0732	2,4239	92,88
GMM	0,1383	1,4061	173,15
DBSCAN	<b>0,2082</b>	<b>0,5285</b>	56,66

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### ❖ Nhận xét:

Khi dùng ít biến, hiệu suất của tất cả các thuật toán đều giảm.

**DBSCAN nhạy cảm tốt với 2 biến**, vì nó tự động xác định hình dạng cụm (thường phi cầu).

**KMeans yếu rõ rệt**, do giả định hình tròn – không thích hợp với cụm hẹp/dài/phi tuyến.

#### C.4. Sử dụng 2 biến đặc trưng đối với dữ liệu sau khi loại bỏ outliers

```

# Đọc dữ liệu từ các file CSV
kmeans_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/ket_qua_kmeans_2bien.csv")
gmm_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_gmm_2bien_chua_xoa_outliers.csv")
dbSCAN_df = pd.read_csv("/content/drive/MyDrive/khai pha du lieu/phan_cum_dbSCAN_2bien_chua_xoa_outliers.csv")

# Hàm đánh giá các chỉ số không giám sát
def evaluate_clustering(X, labels):
    mask = labels != -1 # Loại bỏ nhiễu (DBSCAN)
    if mask.sum() < 2 or len(set(labels[mask])) < 2:
        return {"Silhouette": None, "Davies-Bouldin": None, "Calinski-Harabasz": None}

    return {
        "Silhouette": silhouette_score(X[mask], labels[mask]),
        "Davies-Bouldin": davies_bouldin_score(X[mask], labels[mask]),
        "Calinski-Harabasz": calinski_harabasz_score(X[mask], labels[mask])
    }

# Đánh giá cho KMeans
X_kmeans = kmeans_df.drop(columns=["Applicant ID", "Cluster"])
y_kmeans = kmeans_df["Cluster"]
kmeans_metrics = evaluate_clustering(X_kmeans.values, y_kmeans.values)

# Đánh giá cho GMM
X_gmm = gmm_df[["Theory Test", "Steer Control"]]
y_gmm = gmm_df["Cluster"]
gmm_metrics = evaluate_clustering(X_gmm.values, y_gmm.values)

# Đánh giá cho DBSCAN
X_dbSCAN = dbSCAN_df[["Steer Control", "Night Drive"]]
y_dbSCAN = dbSCAN_df["Cluster"]
dbSCAN_metrics = evaluate_clustering(X_dbSCAN.values, y_dbSCAN.values)

# Tổng hợp kết quả
results = pd.DataFrame({
    "KMeans": kmeans_metrics,
    "GMM": gmm_metrics,
    "DBSCAN": dbSCAN_metrics
})

# In kết quả
print("==> Đánh giá các thuật toán phân cụm ==>")
print(results.round(4))

```

Bảng 49: Đánh giá phân cụm với 2 biến – Dữ liệu sau khi loại bỏ outliers

Thuật toán	Silhouette	Davies-Bouldin	Calinski-Harabasz
KMeans	0,0223	3,7545	36,84
GMM	<b>0,4415</b>	0,8334	<b>500,37</b>

DBSCAN	0,2223	<b>0,5133</b>	47,28
--------	--------	---------------	-------

(Nguồn: Kết quả xử lý số liệu Google Colab, 2025)

#### ❖ Nhận xét:

Sau khi loại bỏ outliers, **GMM vượt trội**, vẫn giữ được độ phân cụm rất cao.

**DBSCAN ổn định**, vẫn tốt ở Davies-Bouldin nhưng giảm về Calinski.

**KMeans rất yếu**, độ tách rời thấp và cụm không rõ ràng.

#### Kết luận:

Kết quả cho thấy **thuật toán GMM có hiệu suất vượt trội toàn diện** cả về độ tách cụm (Silhouette), độ đặc trưng (Calinski-Harabasz), và độ chòng cụm thấp (Davies-Bouldin). Trong khi đó, **DBSCAN cho hiệu quả tốt với dữ liệu nhiều và ít biến đầu vào**, nhờ khả năng xác định cụm không cần giả định hình dạng. Ngược lại, **KMeans tuy đơn giản nhưng kém hiệu quả nếu dữ liệu chưa được xử lý kỹ**, đặc biệt rất nhạy với outliers và biến đầu vào. Điều này cho thấy việc lựa chọn thuật toán phù hợp nên dựa vào cấu trúc dữ liệu, số chiều đầu vào và yêu cầu về cụm phân bố.

## TÀI LIỆU THAM KHẢO

- (1) Driver's License test scores data. (2025, May 28). Kaggle. <https://www.kaggle.com/datasets/ferdinandbaidoo/drivers-license-test-scores-data>
- (2) Gaussian mixture models. (n.d.). Scikit-learn. <https://scikit-learn.org/stable/modules/mixture.html>
- (3) Tham khảo từ ChatGPT, Grok.