BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO

CT312 – KHAI KHOÁNG DỮ LIỆU

ĐỀ TÀI:

PHÂN TÍCH TẬP DỮ LIỆU CAR INSURANCE

Giảng viên hướng dẫn:
Ths GVC Lưu Tiến Đạo

Sinh viên thực hiện:
Lê Nguyễn Thái Tuấn-B2113346

Cần thơ 30, 10, 2024

LỜI CÁM ƠN

Em xin gửi lời cảm ơn sâu sắc đến thầy Lưu Tiến Đạo vì đã tận tình hướng dẫn em trong suốt quá trình thực hiện đồ án. Sự hỗ trợ và chỉ dẫn của thầy đã giúp em vượt qua nhiều khó khăn, hoàn thiện kiến thức và kỹ năng để hoàn thành báo cáo này.

Em cũng xin chân thành cảm ơn các thầy cô trong khoa Công Nghệ Thông Tin và Truyền Thông của trường Đại Học Cần Thơ đã tạo điều kiện và truyền đạt cho em những kiến thức bổ ích trong suốt thời gian học tập. Nhờ sự dạy bảo tận tình của các thầy cô, em đã có được nền tảng vững chắc để tiến hành nghiên cứu và thực hiện đề tài này.

Bên cạnh đó, em xin gửi lời cảm ơn đến gia đình và bạn bè, những người đã luôn bên cạnh, động viên và khích lệ em vượt qua những thử thách trong quá trình học tập và làm đồ án.

Cần Thơ, ngày 5 tháng 11 năm 2024

Lê Nguyễn Thái Tuấn

MỤC LỤC

TÓM TẮT **6**
PHẦN GIỚI THIỆU **7**
1.Giới thiệu chung về đề tài 7
2.Mục tiêu nghiên cứu 7
3.Phạm vi nghiên cứu 7
4.Phương pháp thực hiện 8
PHẦN NỘI DUNG **9**
CHƯƠNG 1: TỔNG QUAN **9**
1.Mô tả chi tiết bài toán 9
2. Vấn đề và giải pháp liên quan đến bài toán 9
CHƯƠNG 2: TRỰC QUAN HÓA, TIỀN XỬ LÍ DỮ LIỆU **10**
1.Trực quan hóa tập dữ liệu 10
1.1 Tổng quan về tập dữ liệu 10
1.2 Chi tiết tập dữ liệu 10
1.3 Trực quan hóa dữ liệu 12
2.Tiền xử lí dữ liệu 20
2.1 Xử lí dữ liệu bị thiếu 21
2.2 Chuẩn hóa dữ liệu 22
CHƯƠNG 3: HUẤN LUYỆN MÔ HÌNH **30**
1. Tiêu chí đánh giá 30
2 Xây dựng mô hình phân lớp 30
2.1 Xây dựng mô hình phân lớp với KNN 31
2.2 Xây dựng mô hình phân lớp với Decision Tree 33
2.3 Xây dựng mô hình phân lớp với Random Forest 36
2.4 Xây dựng mô hình phân lớp với AdaBoosting 39
2.4 Xây dựng mô hình phân lớp với Gradient Boosting 43
2.5 So sánh giữa các mô hình máy học 45
Kết Luận Chung 46
TÀI LIỆU THAM KHẢO **47**

DANH MỤC HÌNH

Hình 1: Chi tiết tập dữ liệu 10
Hình 2: Chi tiết tập dữ liệu 11
Hình 3: Các giá trị bị thiếu trong tập dữ liệu 12
Hình 4:Biểu đồ cột biểu diễn Sex, Race, Driving_experience 13
Hình 5: Biểu đồ cột biểu diễn Education, Vehicle_ownership, Income 13
Hình 6: Biểu đồ cột biểu diễn Married, Children, Vehicle_type 14
Hình 7: Biểu đồ Histogram biểu diễn Credit_score 14
Hình 8: Biểu đồ Histogram biểu diễn Annuala_mileage 15
Hình 9: Biểu đồ cột biểu diễn Outcome 16
Hình 10: Biểu đồ Heatmap biểu diễn mối quan hệ giữa các đặc trưng 17
Hình 11: Biểu đồ cột biểu diễn mối quan hệ giữa Age và Outcome 18
Hình 12: Biểu đồ cột biểu diễn mối quan hệ giữa Driving_experience và Outcome 19
Hình 13: Chi tiết tập dữ liệu 20
Hình 14: Xử lí dữ liệu bị thiếu 21
Hình 15: Kiểm tra dữ liệu sau khi xử lí dữ liệu bị thiếu 21
Hình 16: Mã hóa đặc trưng Age 22
Hình 17: Chi tiết đặc trưng Sex 22
Hình 18: Mã hóa đặc trựng Sex 23
Hình 19: Mã hóa đặc trưng Race 23
Hình 20: Mã hóa đặc trưng Driving_experience 24
Hình 21: Mã hóa đặc trưng Education 25
Hình 22: Mã hóa đặc trưng Income 26
Hình 23: Mã hóa đặc trưng Vehicle_year 26
Hình 24: Xử lí đặc trưng Married 27
Hình 25: Xử lí đặc trưng Children 27
Hình 26: Mã hóa đặc trưng Postal_Code 28
Hình 27: Mã hóa đặc trưng Vehicle_type 28
Hình 28: Chi tiết tập dữ liệu sau xử lí 29
Hình 29: Chia tập dữ liệu 30
Hình 30: Chuẩn hóa Minmax 31
Hình 31: Xây dưng mô hình phân lớp với KNN 31
Hình 32: Kết quả huấn luyện với KNN 32
Hình 33: Ma trận nhầm lẫn KNN 33
Hình 34: Ví dụ về giải thuật Decision Tree 34
Hình 35: Xây dựng mô hình phân lớp với Decision Tree 35
Hình 36: Kết quả huấn luyện Decision Tree 35
Hình 37: Ma trận nhầm lẫn Decision Tree 36
Hình 38: Ví dụ về giải thuật Random Forest 37

Hình 39: Xây dựng mô hình phân lớp với Random Forest.....37
Hình 40: Kết quả huấn luyện Random Forest 38
Hình 41: Biểu đồ thể hiện tầm quan trọng giữa các đặc trưng Random Forest.....38
Hình 42: Ma trận nhầm lẫn Ranadom Forest.....39
Hình 43: Xây dựng mô hình phân lớp với AdaBoosting.....40
Hình 44: Kết quả huấn luyện AdaBoosting 40
Hình 45: Ma trận nhầm lẫn AdaBoosting.....41
Hình 46: Biểu đồ thể hiện tầm quan trọng các đặc trưng AdaBoosting.....42
Hình 48: Xây dựng mô hình phân lớp với GBM 43
Hình 49: Kết quả huấn luyện GBM.....43
Hình 50 Ma trận nhầm lẫn GBM 44
Hình 51: Biểu đồ thể hiện tầm quan trọng giữa các đặc trưng GBM 45
Hình 52: So sánh giữa các mô hình.....45

TÓM TẮT

On the Road, một công ty bảo hiểm ô tô, đang đặt mục tiêu cải thiện quy trình dự đoán khả năng khách hàng yêu cầu bồi thường bảo hiểm trong thời hạn hợp đồng. Trong bối cảnh ngành bảo hiểm ngày càng cạnh tranh và yêu cầu phải tối ưu hóa chi phí, công ty nhận thấy tầm quan trọng của việc sử dụng dữ liệu để dự đoán chính xác hành vi của khách hàng. Do đó, họ đã quyết định áp dụng mô hình học máy để phân tích dữ liệu khách hàng và xác định khả năng xảy ra khiếu nại.

Mục tiêu chính của dự án là phát triển một mô hình máy học, với trọng tâm là tìm kiếm một mô hình có độ chính xác tương đối cao nhất trong việc dự đoán xem khách hàng có yêu cầu khoảng bồi thường hay không. Bằng cách này, On the Road mong muốn xây dựng một giải pháp đơn giản, dễ triển khai và quản lý, giúp họ có thể đưa ra các quyết định nhanh chóng và hiệu quả.

Với độ chính xác dự đoán cao, công ty có thể tối ưu hóa giá cả bảo hiểm, giảm thiểu rủi ro tài chính và cải thiện trải nghiệm khách hàng. Hơn nữa, việc dự đoán chính xác này sẽ giúp công ty đáp ứng các yêu cầu pháp lý hiện hành về bảo hiểm ô tô, nơi mà bảo hiểm là điều kiện bắt buộc khi tham gia giao thông trên đường công cộng. Qua đó, On the Road không chỉ có thể tăng cường vị thế cạnh tranh mà còn góp phần nâng cao hiệu quả hoạt động của mình trong thị trường bảo hiểm rộng lớn này.

PHẦN GIỚI THIỆU

1.Giới thiệu chung về đề tài

Trong bối cảnh ngành bảo hiểm ô tô ngày càng phát triển và cạnh tranh, việc dự đoán khả năng khách hàng yêu cầu bồi thường đã trở thành một yếu tố quan trọng để tối ưu hóa quy trình kinh doanh. Ngành bảo hiểm không chỉ phải đối mặt với áp lực từ các đối thủ cạnh tranh mà còn cần tuân thủ các quy định pháp lý nghiêm ngặt về bảo hiểm xe ô tô. Để cải thiện khả năng dự đoán này, việc áp dụng các mô hình học máy để phân tích dữ liệu khách hàng đang trở thành một giải pháp khả thi và cần thiết.

Đề tài này tập trung vào việc phát triển một mô hình dự đoán nhằm xác định khả năng khách hàng yêu cầu bồi thường bảo hiểm. Mục tiêu là tìm kiếm mô hình có độ chính xác tương đối cao, giúp tối ưu hóa quy trình định giá và quản lý rủi ro. Việc sử dụng mô hình học máy không chỉ giúp tiết kiệm thời gian trong quá trình phân tích mà còn nâng cao độ chính xác trong việc dự đoán hành vi của khách hàng.

2.Mục tiêu nghiên cứu

- **Tổng quan bộ dữ liệu**: Thể hiện góc nhìn về dữ liệu bằng cách trình bày các thông tin thống kê mô tả, bao gồm phân phối, mối quan hệ giữa các biến và các yếu tố quan trọng. Điều này sẽ bao gồm việc trực quan hóa dữ liệu thông qua biểu đồ và bảng để hiểu rõ hơn về các đặc trưng, cũng như xác định các biến có ảnh hưởng đến khả năng yêu cầu bồi thường.
- **Xác định mô hình có độ chính xác cao**: Tìm kiếm và phát triển mô hình dự đoán có độ chính xác tương đối cao nhất có thể, giúp công ty dự đoán khả năng khách hàng sẽ yêu cầu bồi thường bảo hiểm.
- **Đánh giá hiệu suất của các mô hình**: So sánh và đánh giá hiệu suất của các mô hình khác nhau trong việc dự đoán khả năng yêu cầu bồi thường, từ đó lựa chọn mô hình phù hợp nhất cho công ty.

3.Phạm vi nghiên cứu

- **Dữ liệu khách hàng**: Nghiên cứu sẽ sử dụng dữ liệu từ các khách hàng hiện có của công ty On the Road, bao gồm thông tin về hồ sơ khách hàng, lịch sử yêu cầu bồi thường và các yếu tố ảnh hưởng khác.

- **Thời gian nghiên cứu**: Nghiên cứu sẽ tập trung vào các dữ liệu trong một khoảng thời gian nhất định để đảm bảo tính chính xác và tính liên quan của mô hình.
- **Giới hạn về mô hình**: Nghiên cứu sẽ không giới hạn ở một mô hình cụ thể nào mà sẽ xem xét nhiều mô hình học máy khác nhau để tìm ra mô hình có độ chính xác cao nhất.

4.Phương pháp thực hiện

1. **Thu thập dữ liệu**: Thu thập dữ liệu khách hàng từ hệ thống của công ty On the Road, bao gồm các yếu tố như độ tuổi, giới tính, loại xe, lịch sử bồi thường và các thông tin liên quan khác.
2. **Tiền xử lý dữ liệu**: Thực hiện các bước tiền xử lý như xử lý giá trị thiếu, chuẩn hóa và mã hóa dữ liệu để đảm bảo dữ liệu sẵn sàng cho việc xây dựng mô hình.
3. **Phân tích dữ liệu**: Sử dụng các công cụ phân tích thống kê để hiểu rõ các yếu tố ảnh hưởng đến khả năng yêu cầu bồi thường.
4. **Xây dựng và đánh giá mô hình**: Phát triển nhiều mô hình học máy khác nhau và đánh giá độ chính xác của từng mô hình bằng các chỉ số như độ chính xác, độ nhạy và độ đặc hiệu.
5. **Chọn mô hình tốt nhất**: Dựa vào kết quả đánh giá, chọn mô hình có độ chính xác tương đối cao nhất và đề xuất cách áp dụng mô hình này trong thực tế.

PHẦN NỘI DUNG

CHƯƠNG 1: TỔNG QUAN

1.Mô tả chi tiết bài toán

Trong lĩnh vực bảo hiểm ô tô, việc dự đoán khả năng khách hàng sẽ yêu cầu bồi thường là một vấn đề quan trọng và cần thiết. Sự chính xác trong việc dự đoán này không chỉ ảnh hưởng đến quyết định định giá bảo hiểm mà còn liên quan đến chiến lược quản lý rủi ro của công ty bảo hiểm.

Chúng tôi sử dụng bộ dữ liệu bảo hiểm ô tô thu thập từ nghiên cứu của công ty On The Road, trong đó chứa các thông tin quan trọng của khách hàng như tuổi, giới tính, tình trạng hôn nhân, điểm tín dụng, và các đặc điểm liên quan đến xe cộ. Các biến này được kỳ vọng có mối quan hệ với khả năng yêu cầu bồi thường.

Mục tiêu chính của bài toán là xây dựng một mô hình phân loại nhị phân có khả năng dự đoán liệu một khách hàng có yêu cầu bồi thường hay không dựa trên các thông tin đầu vào. Mô hình này sẽ giúp công ty bảo hiểm tối ưu hóa quy trình quản lý yêu cầu bồi thường và định giá hợp lý hơn cho các hợp đồng bảo hiểm.

2. Vấn đề và giải pháp liên quan đến bài toán

Để dự đoán khả năng khách hàng sẽ yêu cầu bồi thường bảo hiểm ô tô, cần có cái nhìn trực quan về bộ dữ liệu và mối quan hệ giữa các đặc trưng với kết quả. Mục tiêu chính của nghiên cứu bao gồm việc tìm hiểu cấu trúc và nội dung của bộ dữ liệu để có cái nhìn tổng quan, xác định mối quan hệ giữa các đặc trưng và khả năng yêu cầu bồi thường nhằm hiểu rõ yếu tố nào quan trọng nhất, và lựa chọn mô hình học máy thích hợp để dự đoán khả năng yêu cầu bồi thường một cách chính xác.

Để giải quyết những vấn đề này, chúng tôi sẽ thực hiện một số bước cụ thể. Đầu tiên, chúng tôi sẽ tiến hành phân tích dữ liệu thông qua các phương pháp mô tả và trực quan hóa, giúp có cái nhìn rõ ràng về bộ dữ liệu. Sau đó, chúng tôi sẽ thực hiện tiền xử lý dữ liệu, bao gồm việc làm sạch dữ liệu, xử lý các giá trị thiếu và chuẩn hóa các biến để đảm bảo tính nhất quán. Cuối cùng, chúng tôi sẽ áp dụng các thuật toán học máy như KNN, Decision Tree, Random Forest … để xây dựng các mô hình phân loại, và so sánh hiệu suất của các mô hình này nhằm tìm ra mô hình có độ chính xác tốt nhất cho việc dự đoán khả năng yêu cầu bồi thường.

CHƯƠNG 2: TRỰC QUAN HÓA, TIỀN XỬ LÍ DỮ LIỆU

1.Trực quan hóa tập dữ liệu

1.1 Tổng quan về tập dữ liệu

Tập dữ liệu được sử dụng cung cấp bởi công ty bảo hiểm ô tô On the Road. Dữ liệu này chứa thông tin về khách hàng và các yếu tố liên quan đến yêu cầu bồi thường bảo hiểm ô tô. Tập dữ liệu gốc bao gồm tổng cộng 19 cột, trong đó có 18 cột thể hiện các tính năng (đặc trưng) mà công ty thu thập được từ hồ sơ khách hàng.

Các cột trong tập dữ liệu đại diện cho các đặc điểm thực tế của khách hàng, bao gồm thông tin như tuổi, giới tính, tình trang hôn nhân, và nhiều đặc điểm khác có liên quan đến khách hàng. Cột cuối cùng, được gọi là “Outcome" cho biết liệu khách hàng đã yêu cầu khoản bồi thường bảo hiểm hay không. Nếu khách hàng đã yêu cầu bồi thường, giá trị sẽ là 1, nếu không, giá trị sẽ là 0.

Dữ liệu được đọc trực tiếp từ:

https://raw.githubusercontent.com/Tuan3198263/CT312/main/Car_Insurance_datasets.xlsx

1.2 Chi tiết tập dữ liệu

Tập dữ liệu này bao gồm 22 cột với tổng cộng 10.000 bản ghi. Mỗi cột thể hiện một đặc trưng cụ thể liên quan đến thông tin của khách hàng và yêu cầu bảo hiểm. Mốt số đặc trưng được thêm vào để gây nhiễu sẽ được loại bỏ(ID, Work, Name, Salary)

	ID	Name	Work	AGE	SEX	RACE	DRIVING_EXPERIENCE	EDUCATION	INCOME	CREDIT_SCORE	...	VEHICLE_YEAR
0	569520	N1	CTU	65+	female	majority	0-9y	high school	upper class	0.629027	...	after 2015
1	750365	N2	CTU	16-25	male	majority	0-9y	none	poverty	0.357757	...	before 2015
2	199901	N3	CTU	16-25	female	majority	0-9y	high school	working class	0.493146	...	before 2015
3	478866	N4	CTU	16-25	male	majority	0-9y	university	working class	0.206013	...	before 2015
4	731664	N5	CTU	26-39	male	majority	10-19y	none	working class	0.388366	...	before 2015

Hình 1: Chi tiết tập dữ liệu

MARRIED	CHILDREN	POSTAL_CODE	ANNUAL_MILEAGE	VEHICLE_TYPE	SPEEDING_VIOLATIONS	DUIS	PAST_ACCIDENTS	OUTCOME
0.0	1.0	10238.0	12000.0	sedan	0.0	0	0	0
0.0	0.0	10238.0	16000.0	sedan	0.0	0	0	1
0.0	0.0	10238.0	11000.0	sedan	0.0	0	0	0
0.0	1.0	32765.0	11000.0	sedan	0.0	0	0	0
0.0	0.0	32765.0	12000.0	sedan	2.0	0	1	1

Hình 2: Chi tiết tập dữ liệu

Mô tả chi tiết về các cột trong tập dữ liệu:

1. **AGE**: Tuổi của khách hàng
2. **SEX**: Giới tính của khách hàng
3. **RACE**: Chủng tộc của khách hàng.
4. **DRIVING_EXPERIENCE**: Số năm kinh nghiệm lái xe của khách hàng
5. **EDUCATION**: Trình độ học vấn của khách hàng.
6. **INCOME**: Thu nhập của khách hàng
7. **CREDIT_SCORE**: Điểm tín dụng của khách hàng
8. **VEHICLE_OWNERSHIP**: Thông tin về quyền sở hữu xe
9. **VEHICLE_YEAR**: Năm sản xuất của xe
10. **MARRIED**: Tình trạng hôn nhân của khách hàng
11. **CHILDREN**: Số lượng con cái trong gia đình khách hàng
12. **POSTAL_CODE**: Mã bưu điện của khách hàng
13. **ANNUAL_MILEAGE**: Số dặm mà khách hàng lái xe hàng năm
14. **VEHICLE_TYPE**: Loại xe mà khách hàng đang sử dụng
15. **SPEEDING_VIOLATIONS**: Số lượng vi phạm tốc độ của khách hàng
16. **DUIS**: Số lần vi phạm luật giao thông (Driving Under the Influence) của khách hàng.
17. **PAST_ACCIDENTS**: Số vụ tai nạn trước đây của khách hàng.
18. **OUTCOME**: Cột kết quả cho biết liệu khách hàng đã yêu cầu bồi thường hay chưa. Giá trị sẽ là 1 nếu khách hàng đã yêu cầu bồi thường và 0 nếu không.

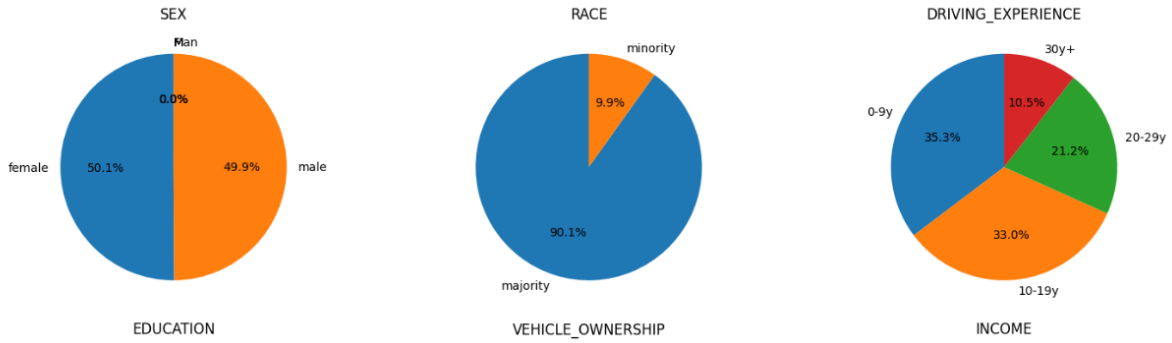
AGE	0	VEHICLE_YEAR	2
SEX	1	MARRIED	1
RACE	2	CHILDREN	1
DRIVING_EXPERIENCE	2	POSTAL_CODE	1
EDUCATION	2	ANNUAL_MILEAGE	958
INCOME	2	VEHICLE_TYPE	1
CREDIT_SCORE	984	SPEEDING_VIOLATIONS	1
SALARY	9976	DUIS	0
VEHICLE_OWNERSHIP	2	PAST_ACCIDENTS	0
VEHICLE_YEAR	2	OUTCOME	0

Hình 3: Các giá trị bị thiếu trong tập dữ liệu

Các cột có khá nhiều giá trị bị thiếu(cột salary có 9976 giá trị Null nên sẽ được loại bỏ)

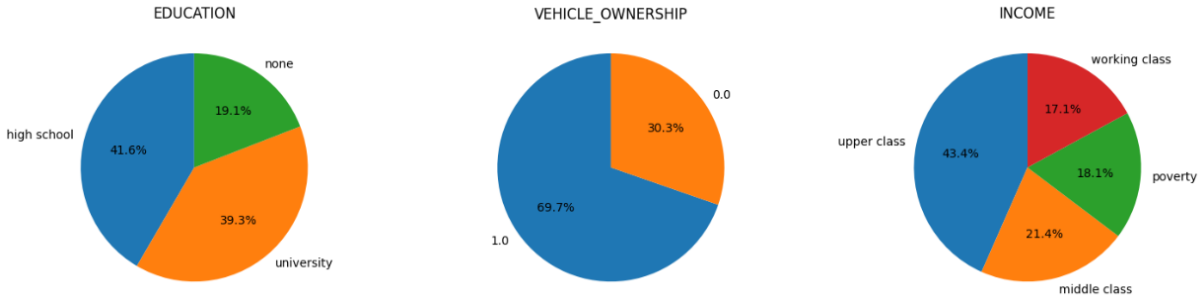
1.3 Trực quan hóa dữ liệu

- Ở đây chúng ta sẽ dùng một số biểu đồ như histogram, boxplot … để có cái nhìn tổng quan về tập dữ liệu



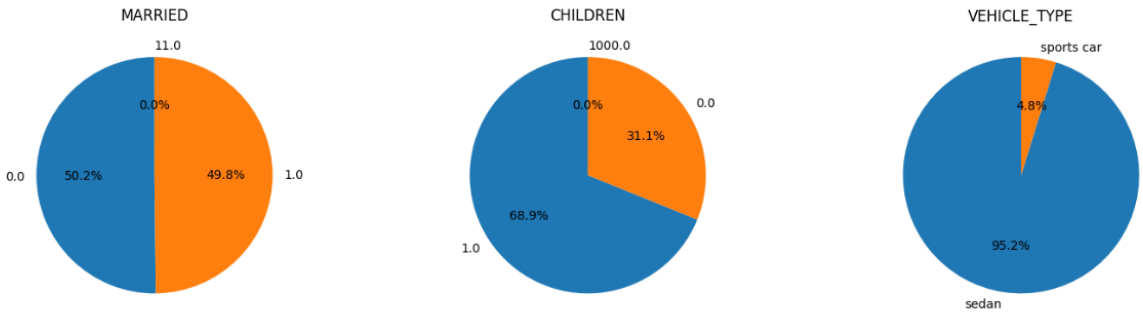
Hình 4:Biểu đồ cột biểu diễn Sex, Race, Driving_experience

- Cột **Sex**(giới tính): mang 2 giá trị cân bằng, female(nữ) và male(nam), bên cạnh có 1 một giá trị là Man gây nhầm , chúng ta sẽ chuẩn hóa giá trị này về male
- Cột **Race**(chủng tộc): mang 2 giá trị: majority(đa số) và minority(thiểu số), giá trị majority chiếm đa số
- Cột **Driving_experince**(số năm kinh nghiệm lái xe): có 4 lớp tuổi bao gồm 0-9y, 10-19y, 20-29y và 30+ chia đều nhau



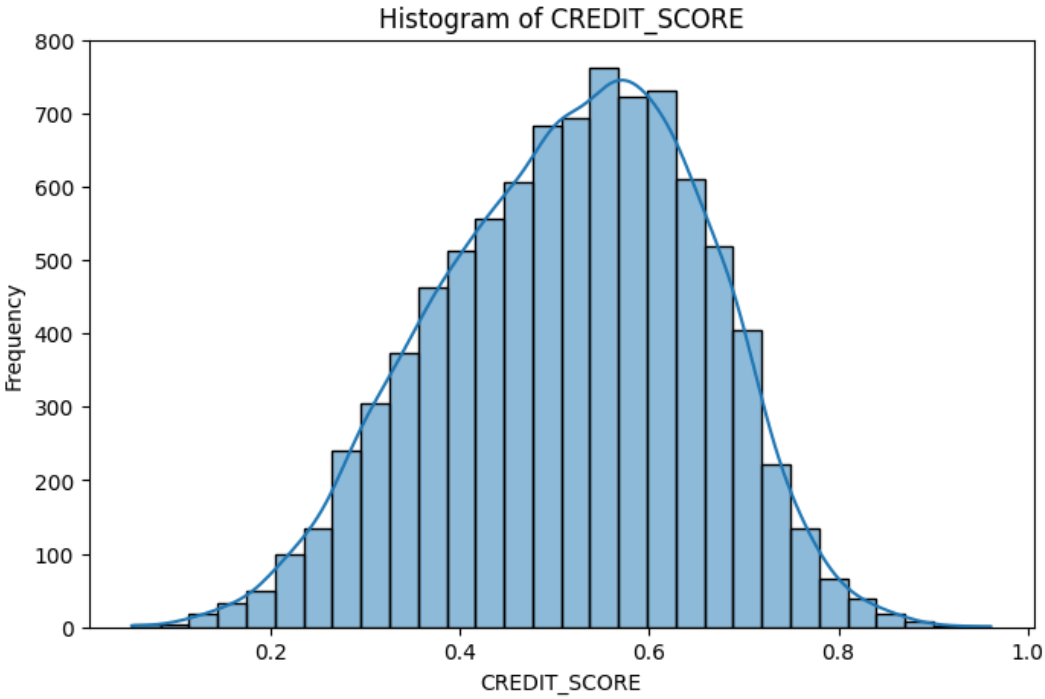
Hình 5: Biểu đồ cột biểu diễn Education, Vehicle_ownership, Income

- Cột **Education**(học vấn): có 3 giá trị là high school, university và none
- Cột **Vihicle_ownership**(sở hữu xe) : 1 nếu khách hàng sở hữu xe ô tô, ngược lại 0
- Cột **Income**(thu nhập): mang 4 giá trị là poverty, working class, middle class và upper class, giá trị upper class chiếm ưu thế



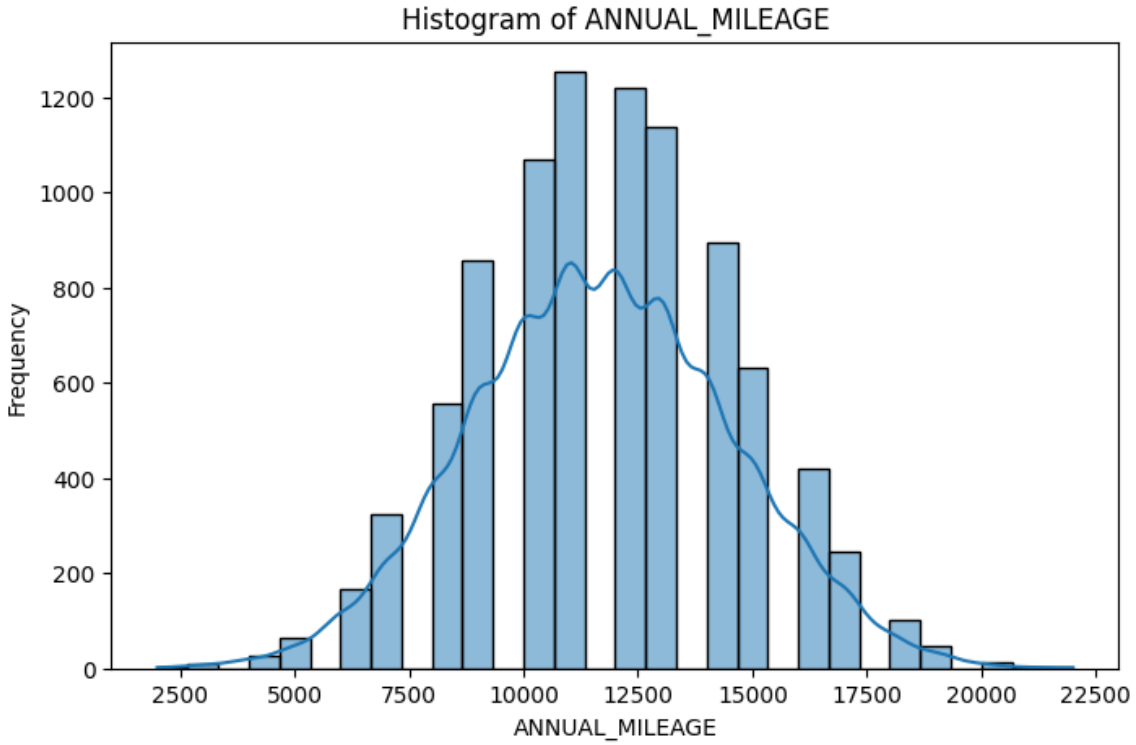
Hình 6: Biểu đồ cột biểu diễn Married, Children, Vehicle_type

- Cột **Married**(tình trạng hôn nhân): 0 nếu chưa kết hôn và ngược lại 1, có một giá trị 11 gây nhiễu sẽ bị loại bỏ.
- Cột **Children**(con cái): 0 nếu khách hàng chưa có con và ngược lại 1, có một giá trị là 1000 sẽ bị loại bỏ..
- Cột **Vehicle_type**(loại xe): 2 giá trị là sedan và sport car, sedan chiếm đa số.
- Cột **Vehicle_year**(năm sản xuất xe): 2 giá trị là before 2015 và after 2015



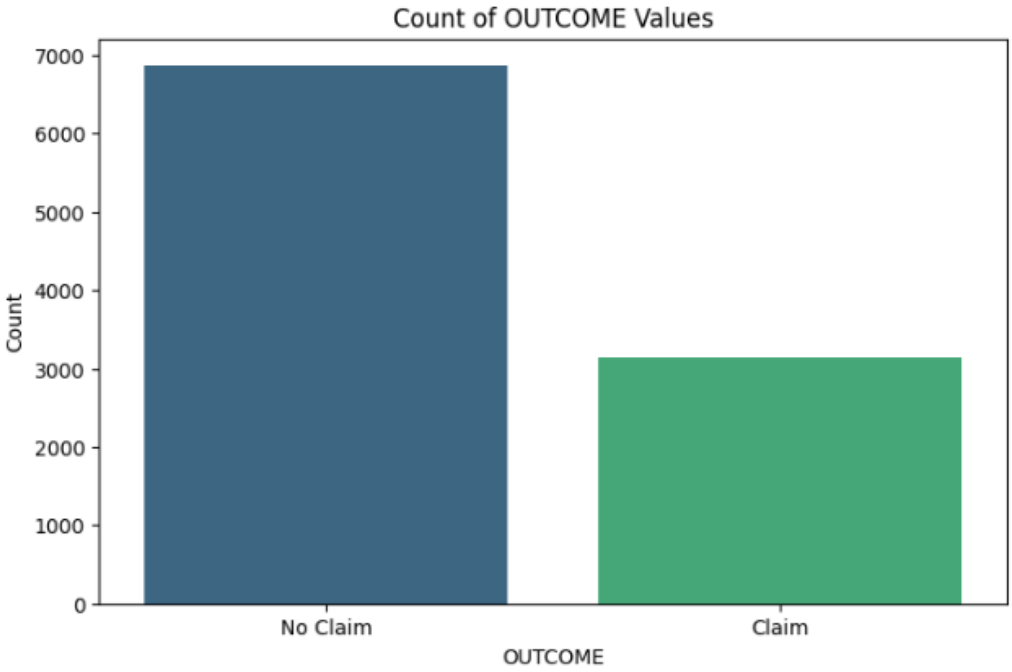
Hình 7: Biểu đồ Histogram biểu diễn Credit_score

- **Credit_score** là cột mang giá trị liên tục, phân phối từ 0 tới 1 trong đó 0,4 tới 0,65 là khoảng có nhiều giá trị nhất cho thấy một phần lớn khách hàng trong tập dữ liệu có điểm tín dụng nằm trong khoảng này



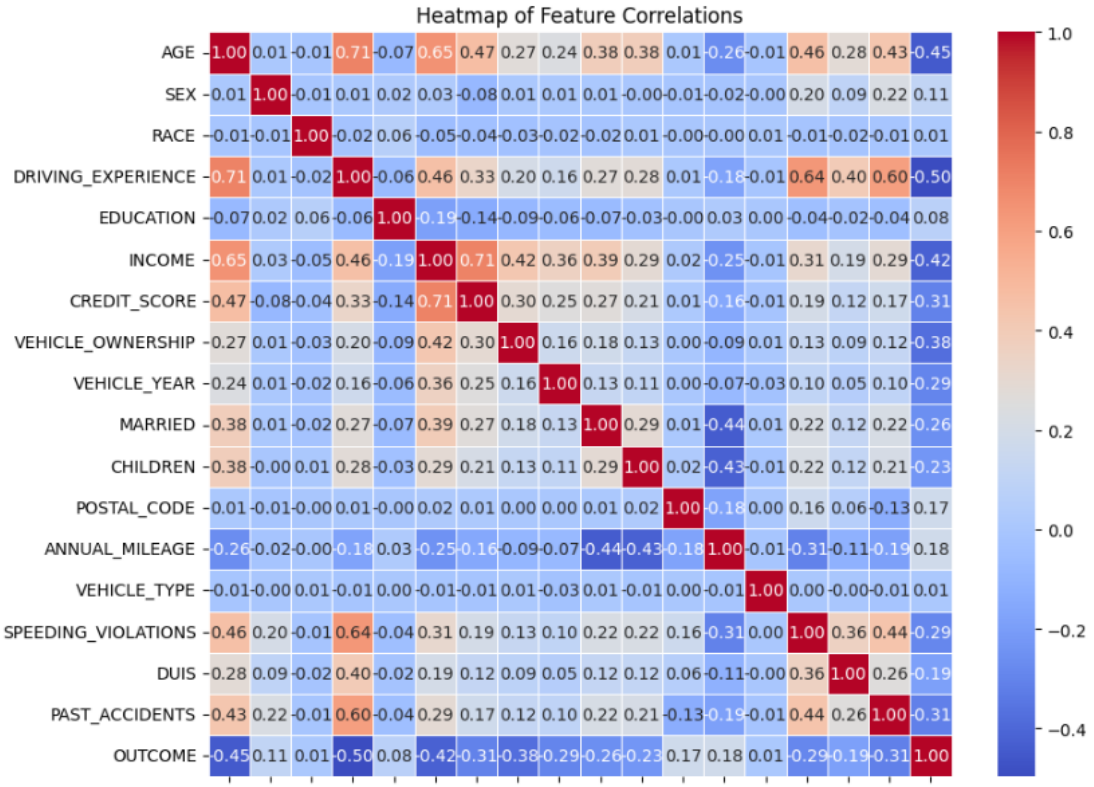
Hình 8: Biểu đồ Histogram biểu diễn Annuala_mileage

- Cột **Annual_mileage**: cho thấy rằng khoảng giá trị từ 10.000 đến 15.000 dặm là khoảng có số lượng quan sát cao nhất



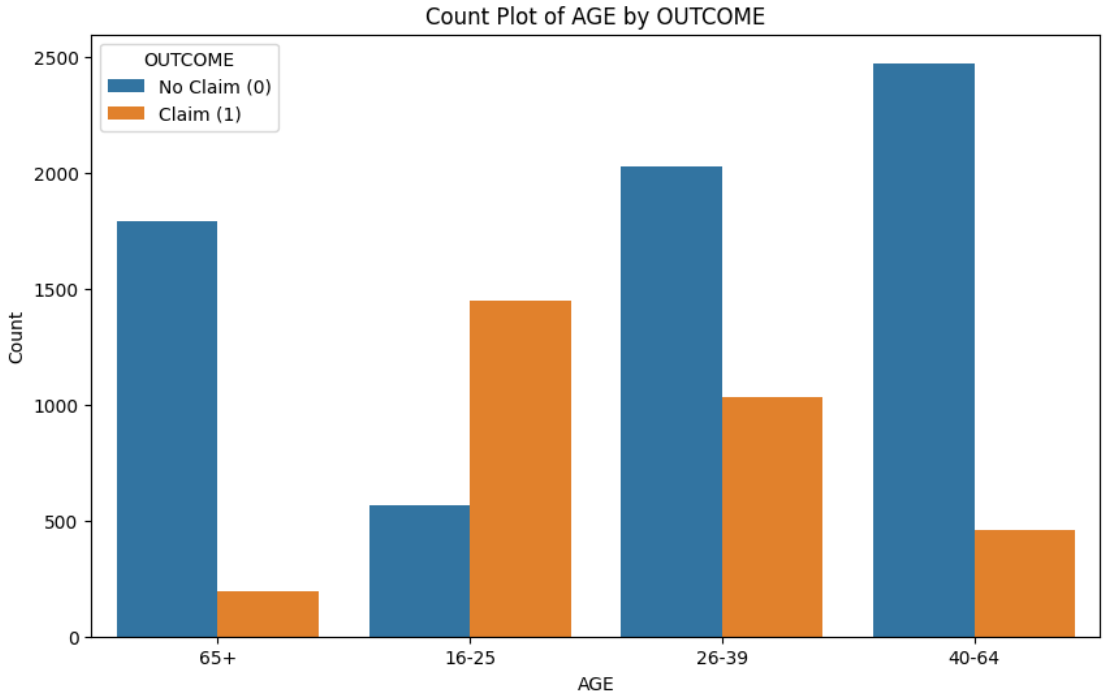
Hình 9: Biểu đồ cột biểu diễn Outcome

- Cột **nhãn** với hai giá trị 0 (không yêu cầu bồi thường) và 1 (yêu cầu bồi thường), nhìn chung cột nhãn không quá mất cân bằng



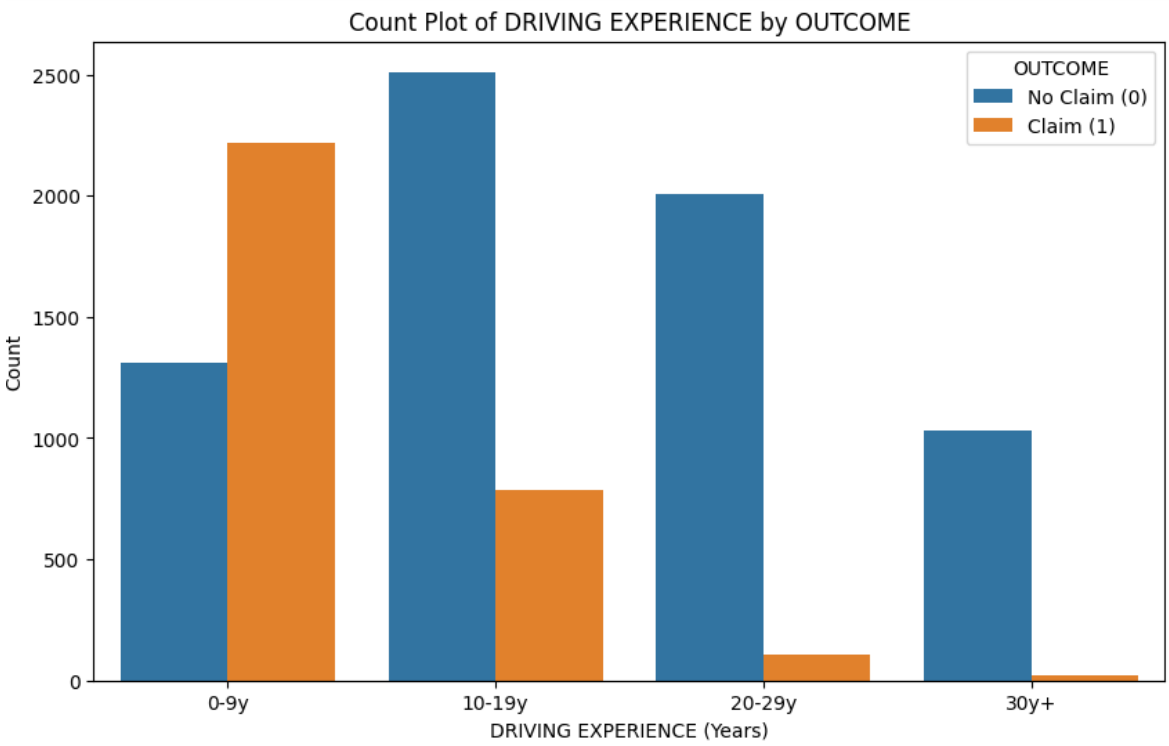
Hình 10: Biểu đồ Heatmap biểu diễn mối quan hệ giữa các đặc trưng

- **Age** và **Drving_experience** có liên quan mật thiết tới **Outcome**, bên cạnh đó các cột như Income hay Vehice_Ownership cũng có thể được xem xét



Hình 11: Biểu đồ cột biểu diễn mối quan hệ giữa Age và Outcome

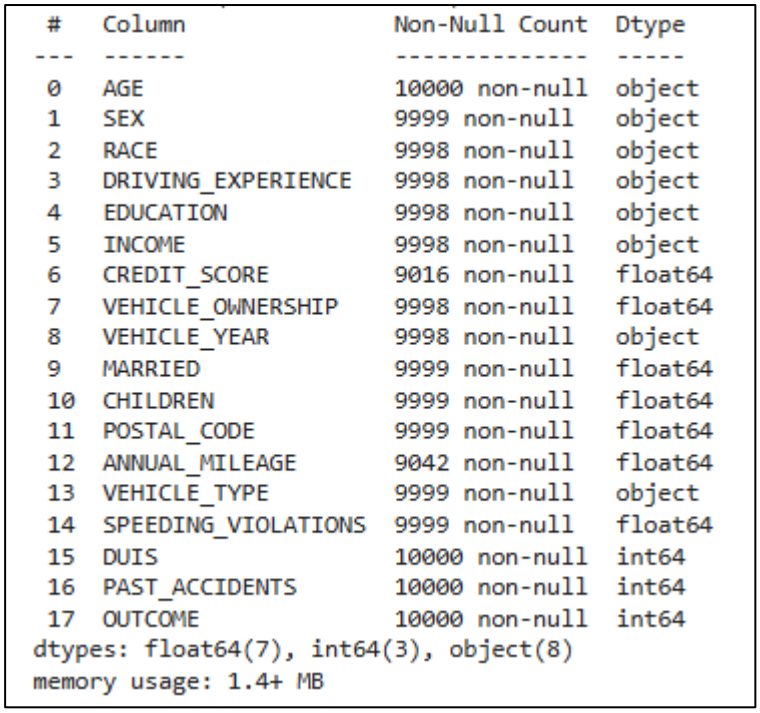
- Phần lớp tệp khách hàng đòi bồi thường trong khoảng 16-25 tuổi (tệp khách trẻ), một phần khác thuộc về nhóm 26-39 tuổi, các nhóm tuổi còn lại rất ít yêu cầu.



Hình 12: Biểu đồ cột biểu diễn mối quan hệ giữa Driving_experience và Outcome

- Nhóm khách có kinh nghiệm lái xe từ 0-9 và 10-19 có khuynh hướng thuộc lớp 1 cao hơn, trong khi nhóm khách có kinh nghiệm lái xe lâu năm hầu như sẽ không yêu cầu đền bù bảo hiểm

2.Tiền xử lí dữ liệu



#	Column	Non-Null Count	Dtype
0	AGE	10000 non-null	object
1	SEX	9999 non-null	object
2	RACE	9998 non-null	object
3	DRIVING_EXPERIENCE	9998 non-null	object
4	EDUCATION	9998 non-null	object
5	INCOME	9998 non-null	object
6	CREDIT_SCORE	9016 non-null	float64
7	VEHICLE_OWNERSHIP	9998 non-null	float64
8	VEHICLE_YEAR	9998 non-null	object
9	MARRIED	9999 non-null	float64
10	CHILDREN	9999 non-null	float64
11	POSTAL_CODE	9999 non-null	float64
12	ANNUAL_MILEAGE	9042 non-null	float64
13	VEHICLE_TYPE	9999 non-null	object
14	SPEEDING_VIOLATIONS	9999 non-null	float64
15	DUIS	10000 non-null	int64
16	PAST_ACCIDENTS	10000 non-null	int64
17	OUTCOME	10000 non-null	int64

dtypes: float64(7), int64(3), object(8)
memory usage: 1.4+ MB

Hình 13: Chi tiết tập dữ liệu

- Bộ dữ liệu đang có giá trị bị thiếu, một số cột mang kiểu Object … sẽ được xử lí ở bước tiếp theo

2.1 Xử lí dữ liệu bị thiếu

- Dùng phương pháp mode cho các cột mang giá trị rời rạc, mean cho các cột mang giá trị liên tục để điền các giá trị bị thiếu

```
# Điền các giá trị thiếu cho cột số bằng giá trị trung bình (mean)
df['CREDIT_SCORE'] = df['CREDIT_SCORE'].fillna(df['CREDIT_SCORE'].mean())
df['ANNUAL_MILEAGE'] = df['ANNUAL_MILEAGE'].fillna(df['ANNUAL_MILEAGE'].mean())

# Điền các giá trị thiếu cho các cột phân loại bằng giá trị thường gặp nhất (mode)
columns_mode = ['SEX', 'RACE', 'DRIVING_EXPERIENCE', 'EDUCATION', 'INCOME',
                'VEHICLE_OWNERSHIP', 'VEHICLE_YEAR', 'MARRIED', 'CHILDREN',
                'POSTAL_CODE', 'VEHICLE_TYPE', 'SPEEDING_VIOLATIONS']

for column in columns_mode:
    df[column] = df[column].fillna(df[column].mode()[0])
```

Hình 14: Xử lí dữ liệu bị thiếu

```
print(df.isnull().sum())
```

```
AGE                    0
SEX                    0
RACE                   0
DRIVING_EXPERIENCE     0
EDUCATION              0
INCOME                 0
CREDIT_SCORE           0
VEHICLE_OWNERSHIP      0
VEHICLE_YEAR           0
MARRIED                0
CHILDREN               0
POSTAL_CODE            0
ANNUAL_MILEAGE         0
VEHICLE_TYPE           0
SPEEDING_VIOLATIONS    0
DUIS                   0
PAST_ACCIDENTS         0
OUTCOME                0
dtype: int64
```

Hình 15: Kiểm tra dữ liệu sau khi xử lí dữ liệu bị thiếu

2.2 Chuẩn hóa dữ liệu

- Dùng phương pháp Label Encoding để mã hóa các cột mang kiểu dữ liệu Object

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

AGE	count
26-39	3063
40-64	2931
16-25	2016
65+	1990

dtype: int64

```
#Xử lí cột age
df['AGE'] = df['AGE'].replace({
    '16-25': 0,
    '26-39': 1,
    '40-64': 2,
    '65+': 3
}).astype(int)

df.head()
```

Hình 16: Mã hóa đặc trưng Age

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

SEX	count
female	5010
male	4987
Man	1
M	1
F	1

dtype: int64

Hình 17: Chi tiết đặc trưng Sex

```
[ ]  # Xử lí cột SEX

     df['SEX'] = df['SEX'].replace({
         'female': 'female',
         'male': 'male',
         'Man': 'male',  # Chuyển 'Man' thành 'male'
         'M': 'male',    # Chuyển 'M' thành 'male'
         'F': 'female'   # Chuyển 'F' thành 'female'
     })
```

```
[ ]  #Mã hóa cột SEX
     df['SEX'] = df['SEX'].map({'female': 0, 'male': 1})
```

Hình 18: Mã hóa đặc trưng Sex

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

RACE	count
majority	9012
minority	988

dtype: int64

```
[ ]  #Xử lí cột RACE
     df['RACE'] = df['RACE'].map({'majority': 0, 'minority': 1})
```

Hình 19: Mã hóa đặc trưng Race

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

DRIVING_EXPERIENCE	count
0-9y	3531
10-19y	3298
20-29y	2119
30y+	1052

dtype: int64

```
[ ] #Mã hóa cột driving experince
    df['DRIVING_EXPERIENCE'] = df['DRIVING_EXPERIENCE'].replace({
        '0-9y': 0,
        '10-19y': 1,
        '20-29y': 2,
        '30y+': 3
    })
```

Hình 20: Mã hóa đặc trưng Driving_experience

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

EDUCATION	count
high school	4158
university	3928
none	1914

dtype: int64

```
[ ] # Mã hóa cột EDUCATION

    # Đảm bảo cột EDUCATION là chuỗi
    df['EDUCATION'] = df['EDUCATION'].astype(str)


    df['EDUCATION'] = df['EDUCATION'].replace({
        'high school': 0,
        'university': 1,
        'none': 2
    }).astype(int)  # Chuyển đổi thành kiểu int
```

Hình 21: Mã hóa đặc trưng Education

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

INCOME	count
upper class	4338
middle class	2137
poverty	1814
working class	1711

dtype: int64

```
#Mã hóa cột Income
df['INCOME'] = df['INCOME'].replace({
    'poverty': 0,
    'working class': 1,
    'middle class': 2,
    'upper class': 3
})
```

Hình 22: Mã hóa đặc trưng Income

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

VEHICLE_YEAR	count
before 2015	6968
after 2015	3032

dtype: int64

```
#Mã hóa cột Vihice_year
df['VEHICLE_YEAR'] = df['VEHICLE_YEAR'].replace({
    'before 2015': 0,
    'after 2015': 1
})
```

Hình 23: Mã hóa đặc trưng Vehicle_year

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

MARRIED	count
0.0	5018
1.0	4981
11.0	1

dtype: int64

```
# Xử lí cột married
# Loại bỏ các hàng có giá trị MARRIED là 11.0
df = df[df['MARRIED'] != 11.0]
```

Hình 24: Xử lí đặc trưng Married

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

CHILDREN	count
1.0	6887
0.0	3111
1000.0	1

dtype: int64

```
#Xu li cot Chilred
# Loại bỏ các hàng có giá trị CHILDREN là 1000.0
df = df[df['CHILDREN'] != 1000.0]
```

Hình 25: Xử lí đặc trưng Children

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

POSTAL_CODE	count
10238.0	6940
32765.0	2456
92101.0	482
21217.0	120

dtype: int64

```
# Xử lí cột postal_code, đây là cột mang tính phân loại nên chuyển sang 0,1,2 cho đơn giản
df['POSTAL_CODE'] = df['POSTAL_CODE'].replace({
    10238: 0,
    32765: 1,
    92101: 2,
    21217: 3
})
```

Hình 26: Mã hóa đặc trưng Postal_Code

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

VEHICLE_TYPE	count
sedan	9522
sports car	476

dtype: int64

```
#Mã hóa vihecie _type
# Ánh xạ VEHICLE_TYPE sang 0 và 1
df['VEHICLE_TYPE'] = df['VEHICLE_TYPE'].replace({
    'sedan': 0,
    'sports car': 1
})
```

Hình 27: Mã hóa đặc trưng Vehicle_type

```
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   AGE                  9998 non-null   int64
 1   SEX                  9998 non-null   int64
 2   RACE                 9998 non-null   int64
 3   DRIVING_EXPERIENCE   9998 non-null   int64
 4   EDUCATION            9998 non-null   int64
 5   INCOME               9998 non-null   int64
 6   CREDIT_SCORE         9998 non-null   float64
 7   VEHICLE_OWNERSHIP    9998 non-null   float64
 8   VEHICLE_YEAR         9998 non-null   int64
 9   MARRIED              9998 non-null   float64
 10  CHILDREN             9998 non-null   float64
 11  POSTAL_CODE          9998 non-null   float64
 12  ANNUAL_MILEAGE       9998 non-null   float64
 13  VEHICLE_TYPE         9998 non-null   int64
 14  SPEEDING_VIOLATIONS  9998 non-null   float64
 15  DUIS                 9998 non-null   int64
 16  PAST_ACCIDENTS       9998 non-null   int64
 17  OUTCOME              9998 non-null   int64
dtypes: float64(7), int64(11)
memory usage: 1.4 MB
```

Hình 28: Chi tiết tập dữ liệu sau xử lí

CHƯƠNG 3: HUẤN LUYỆN MÔ HÌNH

1. Tiêu chí đánh giá

Để đánh giá hiệu quả của các mô hình phân loại, chúng ta sẽ sử dụng hai tiêu chí chính là **Ma trận nhầm lẫn** và **Độ chính xác (Accuracy)**:

1. **Ma trận nhầm lẫn (Confusion Matrix)**: Ma trận nhầm lẫn là công cụ trực quan để đánh giá kết quả phân loại của mô hình, cung cấp các chỉ số về số lượng dự đoán đúng và sai cho từng lớp nhãn. Ma trận này bao gồm 4 thành phần chính:
    - **True Positives (TP)**: Số mẫu dự đoán đúng thuộc lớp 1 (OUTCOME = 1).
    - **True Negatives (TN)**: Số mẫu dự đoán đúng thuộc lớp 0 (OUTCOME = 0).
    - **False Positives (FP)**: Số mẫu thuộc lớp 0 nhưng bị dự đoán sai thành lớp 1.
    - **False Negatives (FN)**: Số mẫu thuộc lớp 1 nhưng bị dự đoán sai thành lớp 0.

Ma trận nhầm lẫn giúp hiểu rõ mô hình phân loại tốt đến đâu, đặc biệt là trong các trường hợp có sự chênh lệch giữa hai lớp.

2. **Độ chính xác (Accuracy)**: Độ chính xác là tỉ lệ giữa số lượng dự đoán đúng trên tổng số dự đoán. Độ chính xác cho ta biết tỉ lệ dự đoán đúng của mô hình, từ đó có cái nhìn tổng quan về hiệu quả của mô hình trong việc phân loại các trường hợp yêu cầu bảo hiểm và không yêu cầu bảo hiểm.

2 Xây dựng mô hình phân lớp

- Hold-out là phương pháp chia dữ liệu thành các phần riêng (thường là tập huấn luyện và kiểm tra) để huấn luyện và đánh giá mô hình nhằm kiểm tra khả năng tổng quát hóa. Ở đây, ta sẽ chia tập dữ liệu thành 2 phần: 75% để huấn luyện và 25% để kiểm tra

```
[ ]
    # Tách các đặc trưng và nhãn
    X = df.drop('OUTCOME', axis=1)  # Tất cả các cột ngoại trừ cột OUTCOME
    y = df['OUTCOME']  # Cột OUTCOME

    #Hold-out 75-25
    # Chia dữ liệu thành tập huấn luyện và tập kiểm tra
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Hình 29: Chia tập dữ liệu

- Để đảm bảo dữ liệu đầu vào có phạm vi nhất quán, chúng ta sẽ chuẩn hóa dữ liệu bằng phương pháp **MinMaxScaler**. Phương pháp này giúp đưa tất cả các giá trị của các biến về cùng một khoảng (thường là từ 0 đến 1), từ đó cải thiện hiệu suất của mô hình và giúp tăng tốc độ hội tụ trong quá trình huấn luyện.

```
[ ]

    # Khởi tạo MinMaxScaler
    scaler = MinMaxScaler()

    # Chuẩn hóa dữ liệu
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

Hình 30: Chuẩn hóa Minmax

2.1 Xây dựng mô hình phân lớp với KNN

K-Nearest Neighbors (KNN) là thuật toán phân loại dựa trên khoảng cách, xác định nhãn của một điểm mới dựa vào nhãn của kkk điểm dữ liệu gần nhất. Với mỗi điểm mới, KNN tính khoảng cách đến tất cả các điểm trong tập huấn luyện, tìm kkk láng giềng gần nhất, rồi quyết định nhãn dựa trên đa số. KNN dễ hiểu và hiệu quả cho dữ liệu có phân cụm rõ, nhưng có thể chậm với dữ liệu lớn và nhạy cảm với giá trị ngoại lai.

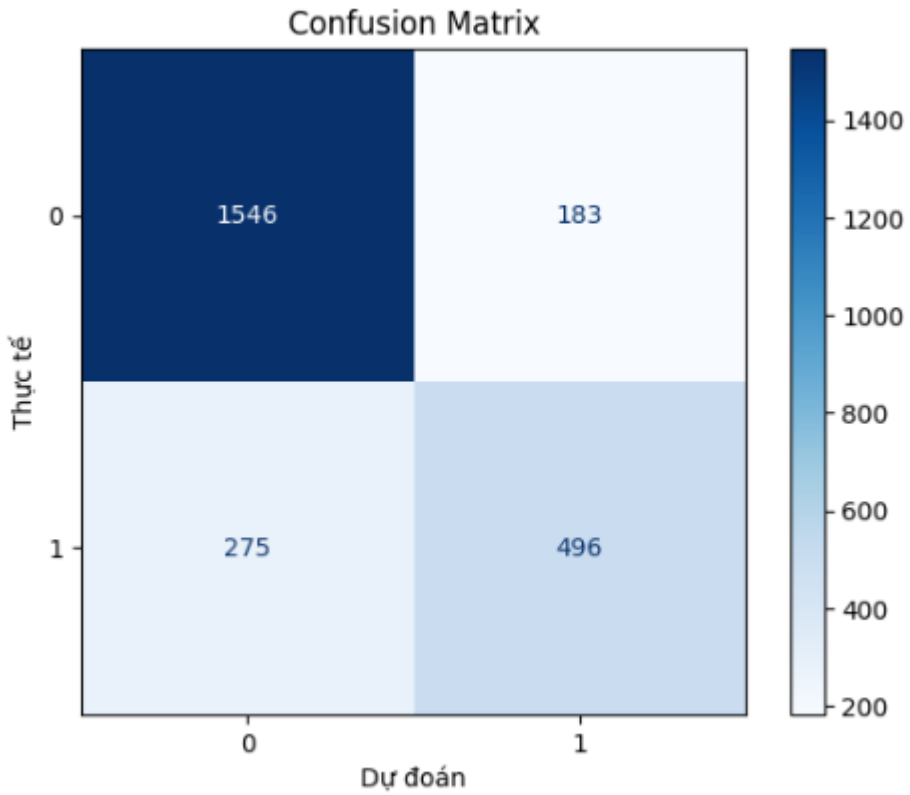
```
from sklearn.neighbors import KNeighborsClassifier

# Huấn luyện mô hình KNN
knn = KNeighborsClassifier(n_neighbors=15)  # Số lượng láng giềng
knn.fit(X_train_scaled, y_train)

# Dự đoán với mô hình KNN
y_pred_knn = knn.predict(X_test_scaled)

# Đánh giá mô hình
print("\nKNN:")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
plot_confusion_matrix(y_test, y_pred_knn, labels=[0, 1])
```

Hình 31: Xây dựng mô hình phân lớp với KNN

```
KNN:
Accuracy: 0.8168
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1729
           1       0.73      0.64      0.68       771

    accuracy                           0.82      2500
   macro avg       0.79      0.77      0.78      2500
weighted avg       0.81      0.82      0.81      2500
```

Hình 32: Kết quả huấn luyện với KNN

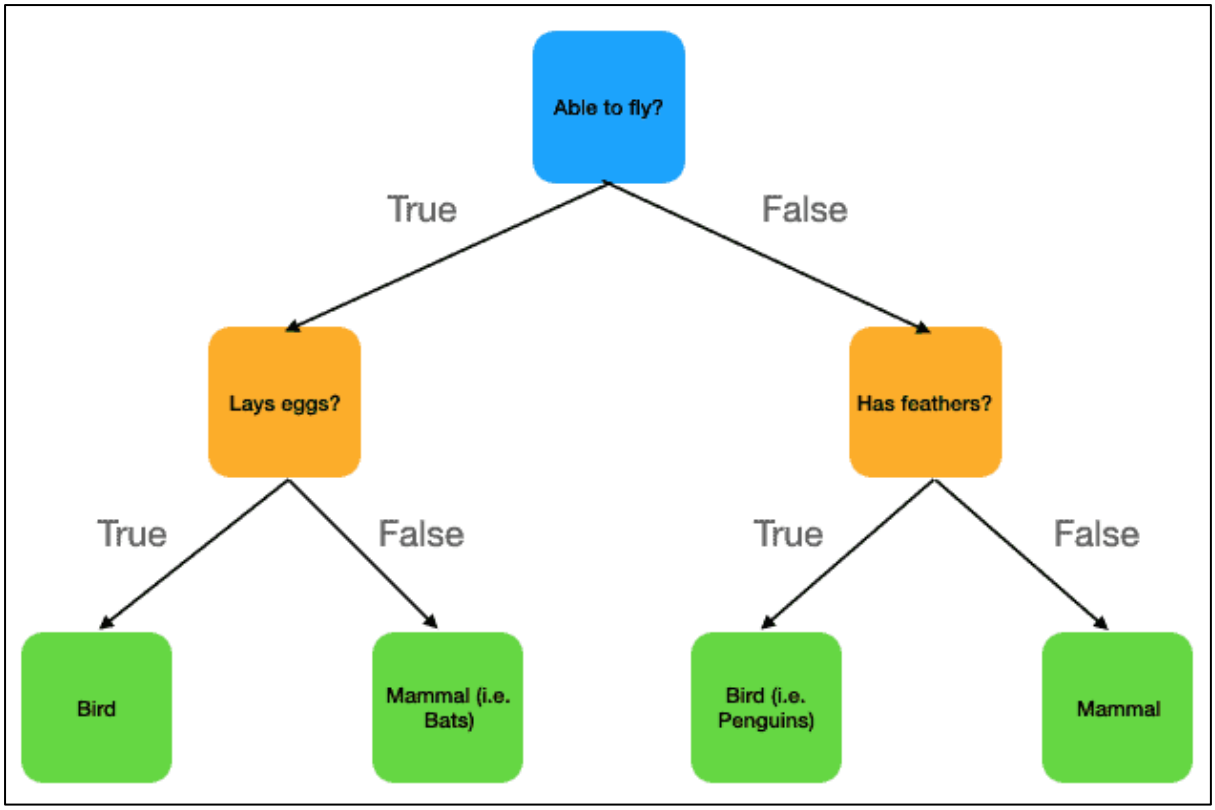
- Độ chính xác (Accuracy): Mô hình đạt độ chính xác 81.68%, cho thấy khả năng phân loại khá tốt nhưng còn có thể cải thiện, đặc biệt với lớp thứ hai.
- Hiệu suất trên các lớp:
    - Lớp 0 : Precision, recall, và F1-score của lớp này đều khá cao (0.85, 0.89, và 0.87), cho thấy mô hình dự đoán tốt hơn cho lớp này.
    - Lớp 1 : Precision và recall của lớp này thấp hơn (lần lượt là 0.73 và 0.64), và F1-score cũng chỉ đạt 0.68. Điều này cho thấy mô hình gặp khó khăn hơn trong việc dự đoán chính xác lớp 1.



Hình 33: Ma trận nhầm lẫn KNN

2.2 Xây dựng mô hình phân lớp với Decision Tree

Thuật toán Cây Quyết Định (Decision Tree) là một kỹ thuật học máy sử dụng cấu trúc cây phân nhánh để đưa ra quyết định phân loại. Trong quá trình huấn luyện, thuật toán chia dữ liệu theo các ngưỡng giá trị của từng đặc trưng, tạo thành các nút phân nhánh dựa trên tiêu chí như "độ giảm thông tin" hoặc "độ lợi thông tin" (information gain). Mục tiêu của quá trình này là tìm ra các phân nhánh giúp tối ưu hóa khả năng phân loại dữ liệu vào các lớp tương ứng.



Hình 34: Ví dụ về giải thuật Decision Tree

Ở đây, GridSearchCV được dùng để tối ưu hóa các siêu tham số của mô hình cây, như max_depth (chiều sâu tối đa của cây) và min_samples_split (số mẫu tối thiểu để tách nút). Những tham số này được điều chỉnh nhằm tăng độ chính xác và khả năng tổng quát của mô hình.

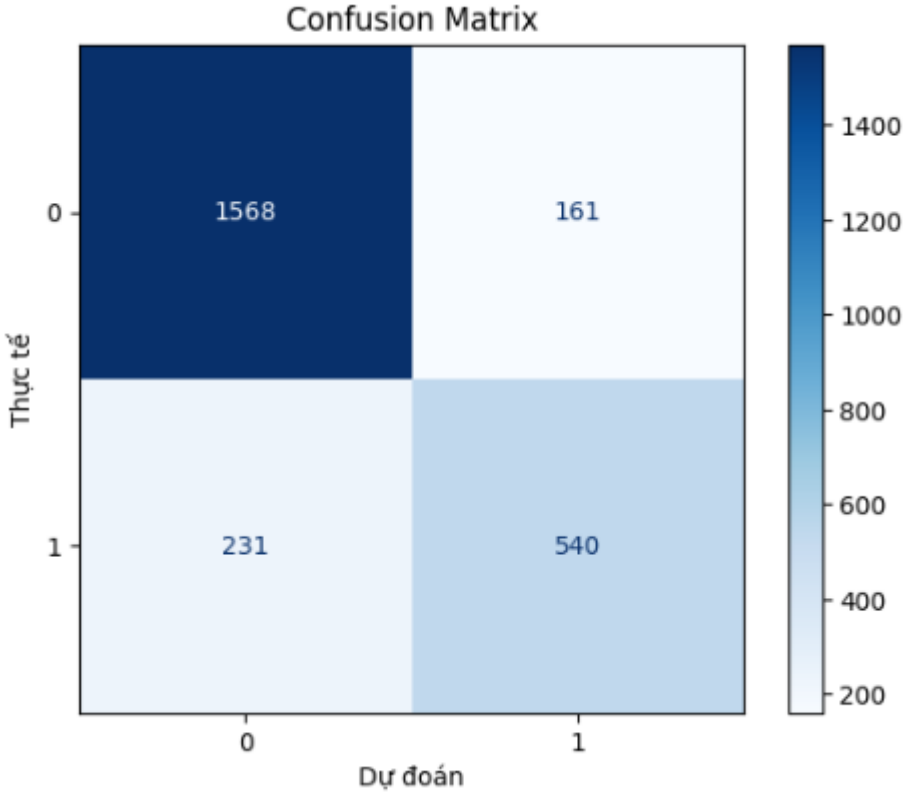
```python
# Định nghĩa mô hình Decision Tree
decision_tree = DecisionTreeClassifier(random_state=42)

# Xác định các tham số để điều chỉnh
param_grid = {
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2']
}

# Sử dụng GridSearchCV để tìm tham số tốt nhất
grid_search = GridSearchCV(estimator=decision_tree, param_grid=param_grid,
                           scoring='accuracy', cv=5, n_jobs=-1, verbose=1)

# Huấn luyện mô hình với GridSearchCV
grid_search.fit(X_train_scaled, y_train)

# In ra tham số tốt nhất và độ chính xác
print("Best parameters:", grid_search.best_params_)
print("Best cross-validation accuracy:", grid_search.best_score_)

# Dự đoán với mô hình Decision Tree đã tối ưu hóa
y_pred_tree = grid_search.predict(X_test_scaled)

# Đánh giá mô hình
print("\nDecision Tree:")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))

# Áp dụng ma trận nhầm lẫn
plot_confusion_matrix(y_test, y_pred_tree, labels=[0, 1])
```

Hình 35: Xây dựng mô hình phân lớp với Decision Tree

```
Fitting 5 folds for each of 135 candidates, totalling 675 fits
Best parameters: {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 2}
Best cross-validation accuracy: 0.8472949077162554

Decision Tree:
Accuracy: 0.8432
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      1729
           1       0.77      0.70      0.73       771

    accuracy                           0.84      2500
   macro avg       0.82      0.80      0.81      2500
weighted avg       0.84      0.84      0.84      2500
```

Hình 36: Kết quả huấn luyện Decision Tree

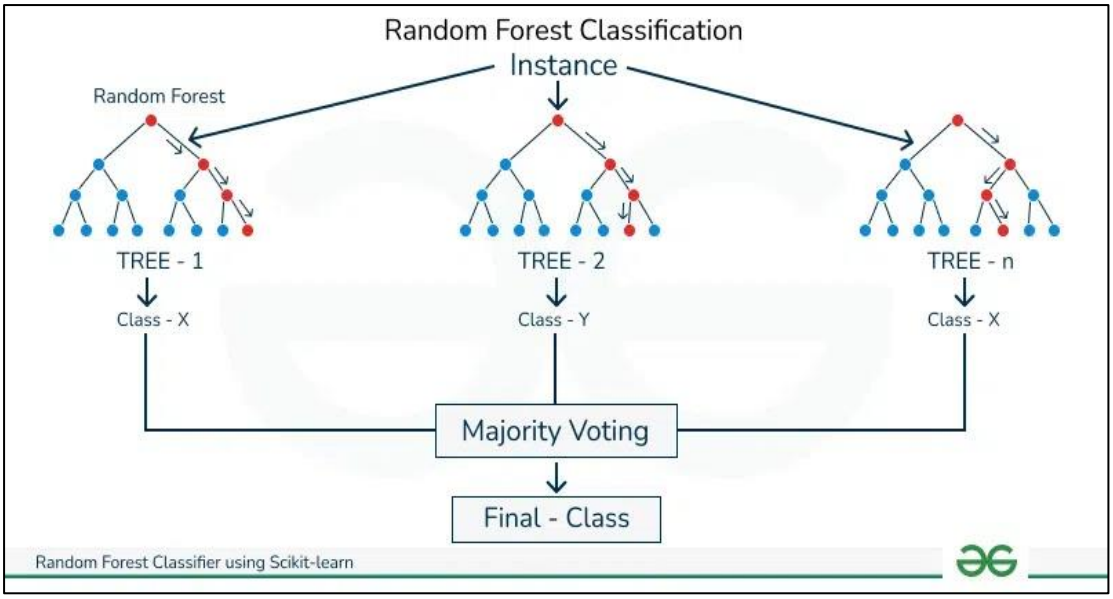
- Độ chính xác (Accuracy): Mô hình đạt độ chính xác 84.32%, thể hiện khả năng phân loại tổng thể tốt.
- Hiệu suất theo từng lớp:
    - Lớp 0: Mô hình hoạt động tốt hơn đối với lớp 0, với precision là 0.87 và recall là 0.91, dẫn đến F1-score 0.89. Điều này cho thấy mô hình rất hiệu quả trong việc phân loại đúng các mẫu thuộc lớp 0.
    - Lớp 1: Precision 0.77 và recall 0.70 ở lớp 1, với F1-score 0.73, cho thấy mô hình gặp nhiều khó khăn hơn trong việc nhận diện chính xác lớp này, dẫn đến một số mẫu của lớp 1 bị phân loại nhầm.



Hình 37: Ma trận nhầm lẫn Decision Tree

2.3 Xây dựng mô hình phân lớp với Random Forest

Thuật toán Rừng Ngẫu Nhiên (Random Forest) là một mô hình học máy dựa trên tập hợp nhiều cây quyết định (Decision Trees), kết hợp dự đoán của các cây này để tăng độ chính xác và giảm khả năng quá khớp (overfitting). Mỗi cây trong rừng được xây dựng từ các mẫu ngẫu nhiên khác nhau từ tập dữ liệu huấn luyện và chỉ sử dụng một phần ngẫu nhiên của các đặc trưng để phân nhánh tại mỗi nút.



Hình 38: Ví dụ về giải thuật Random Forest

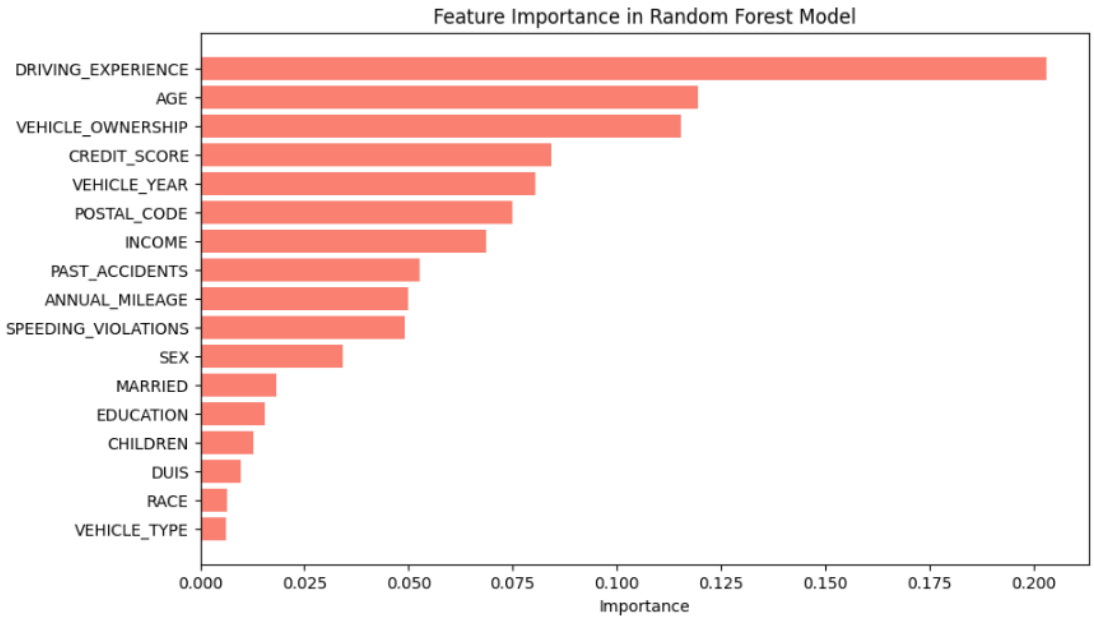
```python
# Định nghĩa mô hình Random Forest
rf_model = RandomForestClassifier(random_state=42)

# Xác định các tham số để điều chỉnh với ít giá trị hơn
param_grid = {
    'n_estimators': [50, 100],  # Giảm số lượng cây để kiểm tra
    'max_depth': [None, 10, 20]  # Giảm độ sâu cây để kiểm tra
}

# Sử dụng GridSearchCV để tìm tham số tốt nhất
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                              scoring='accuracy', cv=3, n_jobs=-1, verbose=1)

# Huấn luyện mô hình với GridSearchCV
grid_search_rf.fit(X_train_scaled, y_train)

# In ra tham số tốt nhất và độ chính xác
print("Best parameters:", grid_search_rf.best_params_)
print("Best cross-validation accuracy:", grid_search_rf.best_score_)

# Dự đoán với mô hình Random Forest đã tối ưu hóa
y_pred_rf = grid_search_rf.predict(X_test_scaled)

# Đánh giá mô hình
print("\nRandom Forest:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

# Áp dụng ma trận nhầm lẫn
plot_confusion_matrix(y_test, y_pred_rf, labels=[0, 1])
```

Hình 39: Xây dựng mô hình phân lớp với Random Forest

```
Fitting 3 folds for each of 6 candidates, totalling 18 fits
Best parameters: {'max_depth': 10, 'n_estimators': 100}
Best cross-validation accuracy: 0.8486268107242897

Random Forest:
Accuracy: 0.8448
              precision    recall  f1-score   support

           0       0.88      0.90      0.89      1729
           1       0.76      0.72      0.74       771

    accuracy                           0.84      2500
   macro avg       0.82      0.81      0.82      2500
weighted avg       0.84      0.84      0.84      2500
```

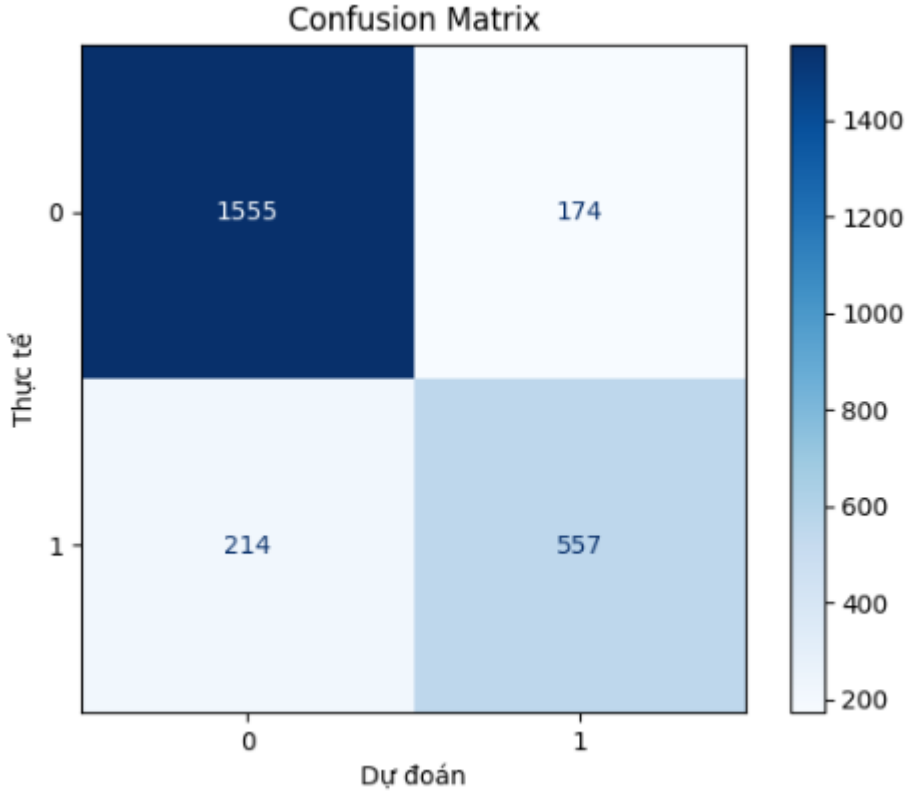
Hình 40: Kết quả huấn luyện Random Forest

- Độ chính xác (Accuracy): Mô hình đạt độ chính xác 84.48%, cho thấy khả năng phân loại tổng thể tốt.
- Hiệu suất theo từng lớp:
    - Lớp 0: Precision 0.88 và recall 0.90, với F1-score 0.89. Điều này cho thấy mô hình hoạt động tốt trong việc nhận diện chính xác các mẫu thuộc lớp 0.
    - Lớp 1: Precision 0.76 và recall 0.72, với F1-score 0.74, cho thấy hiệu suất thấp hơn một chút trong việc nhận diện chính xác các mẫu thuộc lớp 1, nhưng vẫn tốt hơn so với Decision Tree.



Hình 41: Biểu đồ thể hiện tầm quan trọng giữa các đặc trưng Random Forest

Với Random Forest, ta có thể biết được những đặc trưng nào ảnh hưởng trực tiếp tới mô hình, ở đây driving_experience, age, vehicle_ownership là các cột ảnh hưởng trực tiếp tới mô hình



Hình 42: Ma trận nhầm lẫn Ranadom Forest

2.4 Xây dựng mô hình phân lớp với AdaBoosting

Thuật toán AdaBoost (Adaptive Boosting) là một phương pháp học máy kết hợp nhiều mô hình học yếu (weak learners), thường là cây quyết định, để tạo ra một mô hình mạnh mẽ hơn. Mục tiêu của AdaBoost là cải thiện độ chính xác của mô hình bằng cách điều chỉnh trọng số của các mẫu dữ liệu trong quá trình huấn luyện. Cụ thể, nó sẽ tăng trọng số cho các mẫu mà các mô hình trước đó phân loại sai, giúp các mô hình mới tập trung vào các mẫu khó hơn.

GridSearchCV được sử dụng để tìm kiếm các siêu tham số tốt nhất cho mô hình AdaBoost, bao gồm số lượng cây cơ sở (n_estimators) và tốc độ học (learning_rate). Qua việc tối ưu hóa các tham số này, mô hình AdaBoost có thể đạt được hiệu suất tối ưu hơn trên tập dữ liệu. Sau khi hoàn thành việc tối ưu hóa, mô hình sẽ được đánh giá thông qua độ chính xác, báo cáo phân loại và ma trận nhầm lẫn để kiểm tra hiệu quả phân loại của nó.

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import GridSearchCV

# Định nghĩa mô hình AdaBoost
ada_model = AdaBoostClassifier(random_state=42)

# Xác định các tham số để điều chỉnh
param_grid = {
    'n_estimators': [50, 100, 200],  # Số lượng cây cơ sở
    'learning_rate': [0.01, 0.1, 1.0]  # Tốc độ học
}

# Sử dụng GridSearchCV để tìm tham số tốt nhất
grid_search_ada = GridSearchCV(estimator=ada_model, param_grid=param_grid,
                               scoring='accuracy', cv=3, n_jobs=-1, verbose=1)

# Huấn luyện mô hình với GridSearchCV
grid_search_ada.fit(X_train_scaled, y_train)

# In ra tham số tốt nhất và độ chính xác
print("Best parameters:", grid_search_ada.best_params_)
print("Best cross-validation accuracy:", grid_search_ada.best_score_)

# Dự đoán với mô hình AdaBoost đã tối ưu hóa
y_pred_ada = grid_search_ada.predict(X_test_scaled)

# Đánh giá mô hình
print("\nAdaBoost:")
print("Accuracy:", accuracy_score(y_test, y_pred_ada))
print(classification_report(y_test, y_pred_ada))

# Áp dụng ma trận nhầm lẫn
plot_confusion_matrix(y_test, y_pred_ada, labels=[0, 1])  # Gọi hàm ma trận nhầm lẫn đã định nghĩa
```

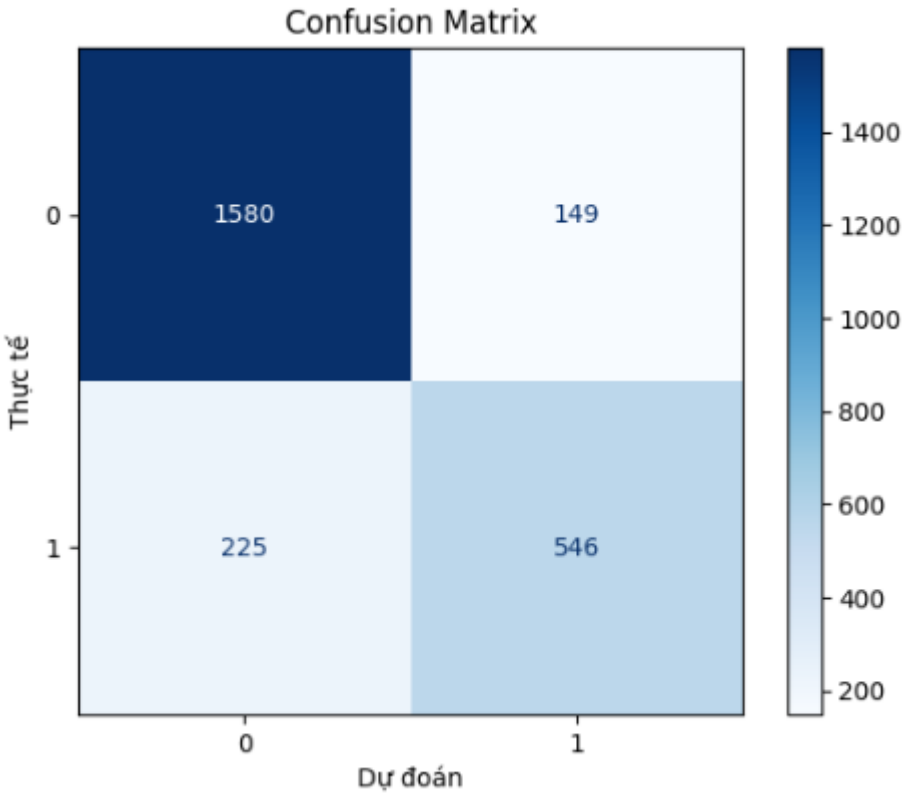
Hình 43: Xây dựng mô hình phân lớp với AdaBoosting

```
Best parameters: {'learning_rate': 0.1, 'n_estimators': 200}
Best cross-validation accuracy: 0.8506270241429905

AdaBoost:
Accuracy: 0.8504
              precision    recall  f1-score   support

           0       0.88      0.91      0.89      1729
           1       0.79      0.71      0.74       771

    accuracy                           0.85      2500
   macro avg       0.83      0.81      0.82      2500
weighted avg       0.85      0.85      0.85      2500
```

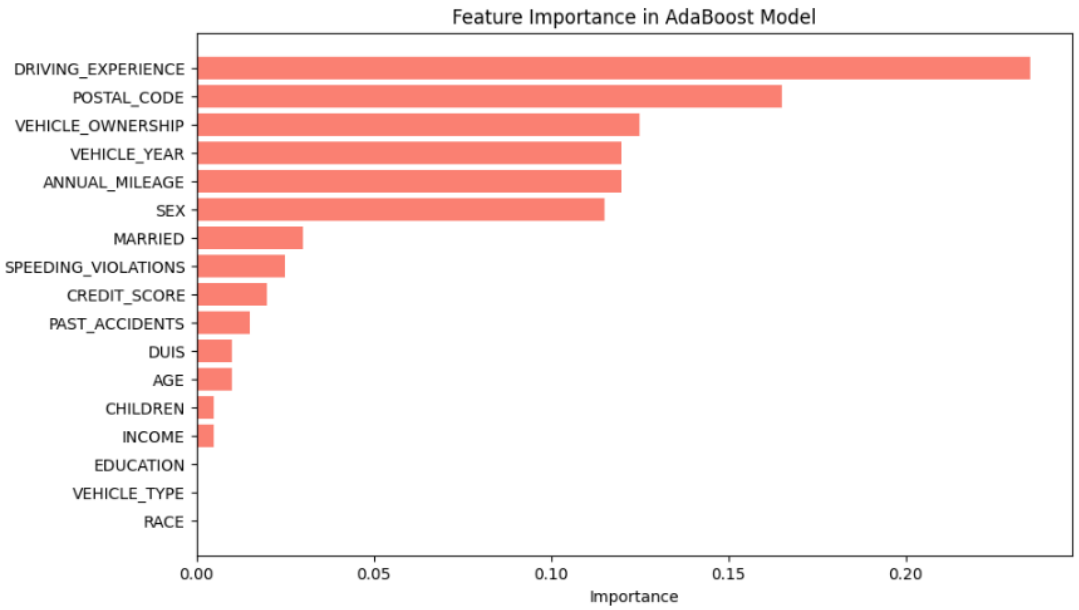
Hình 44: Kết quả huấn luyện AdaBoosting

- Độ chính xác (Accuracy): Mô hình đạt độ chính xác 85.04%, cao hơn so với Decision Tree và Random Forest, cho thấy AdaBoost có hiệu suất tổng thể tốt hơn trên tập dữ liệu này.
- Hiệu suất theo từng lớp:

- Lớp 0: Precision 0.88 và recall 0.91, với F1-score 0.89. Mô hình dự đoán rất tốt cho lớp 0, thể hiện khả năng nhận diện đúng các mẫu thuộc lớp này.
- Lớp 1: Precision 0.79 và recall 0.71, với F1-score 0.74. Mặc dù hiệu suất dự đoán cho lớp 1 vẫn thấp hơn so với lớp 0



Hình 45: Ma trận nhầm lẫn AdaBoosting



Hình 46: Biểu đồ thể hiện tầm quan trọng các đặc trưng AdaBoosting

Driving_experience và postala_code là 2 đặc trưng quan trọng nhất với mô hình AdaBoosting

2.4 Xây dựng mô hình phân lớp với Gradient Boosting

```python
# Định nghĩa mô hình Gradient Boosting
gbm_model = GradientBoostingClassifier(random_state=42)

# Xác định các tham số để điều chỉnh
param_grid = {
    'n_estimators': [100, 200],  # Số lượng cây
    'learning_rate': [0.01, 0.1, 0.2],  # Tốc độ học
    'max_depth': [3, 5, 7]  # Độ sâu tối đa của cây
}

# Sử dụng GridSearchCV để tìm tham số tốt nhất
grid_search_gbm = GridSearchCV(estimator=gbm_model, param_grid=param_grid,
                               scoring='accuracy', cv=3, n_jobs=-1, verbose=1)

# Huấn luyện mô hình với GridSearchCV
grid_search_gbm.fit(X_train_scaled, y_train)

# In ra tham số tốt nhất và độ chính xác
print("Best parameters:", grid_search_gbm.best_params_)
print("Best cross-validation accuracy:", grid_search_gbm.best_score_)

# Dự đoán với mô hình Gradient Boosting đã tối ưu hóa
y_pred_gbm = grid_search_gbm.predict(X_test_scaled)

# Đánh giá mô hình
print("\nGradient Boosting:")
print("Accuracy:", accuracy_score(y_test, y_pred_gbm))
print(classification_report(y_test, y_pred_gbm))

# Áp dụng ma trận nhầm lẫn
plot_confusion_matrix(y_test, y_pred_gbm, labels=[0, 1])  # Gọi hàm ma trận nhầm lẫn đã định nghĩa
```

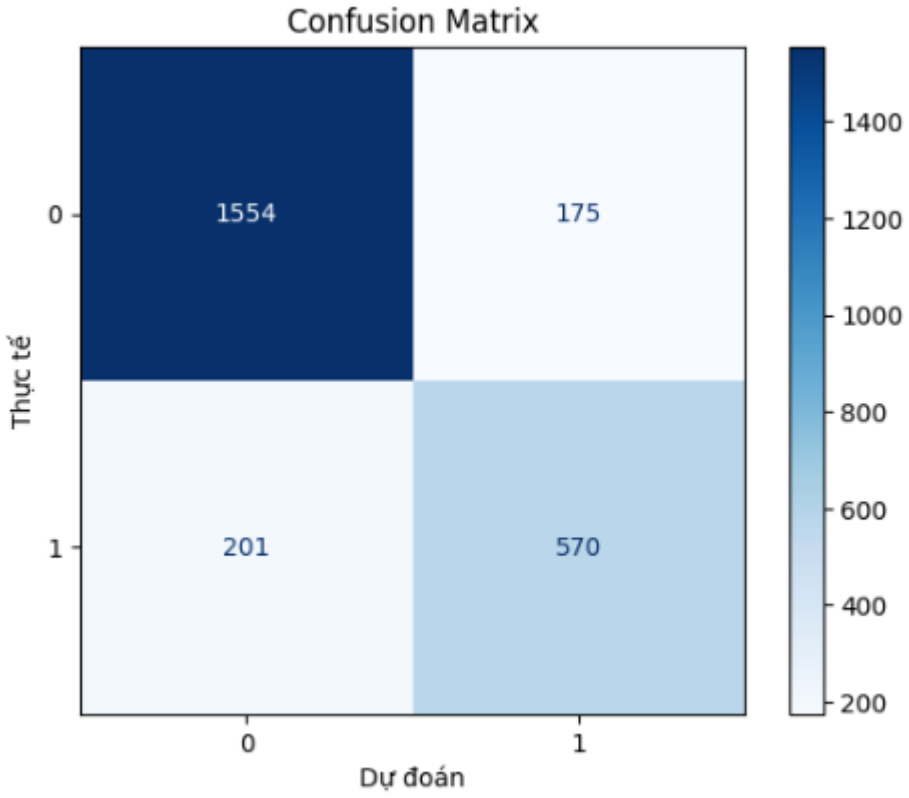
Hình 47: Xây dựng mô hình phân lớp với GBM

```
Best parameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
Best cross-validation accuracy: 0.8516942777110844

Gradient Boosting:
Accuracy: 0.8496
              precision    recall  f1-score   support

           0       0.89      0.90      0.89      1729
           1       0.77      0.74      0.75       771

    accuracy                           0.85      2500
   macro avg       0.83      0.82      0.82      2500
weighted avg       0.85      0.85      0.85      2500
```

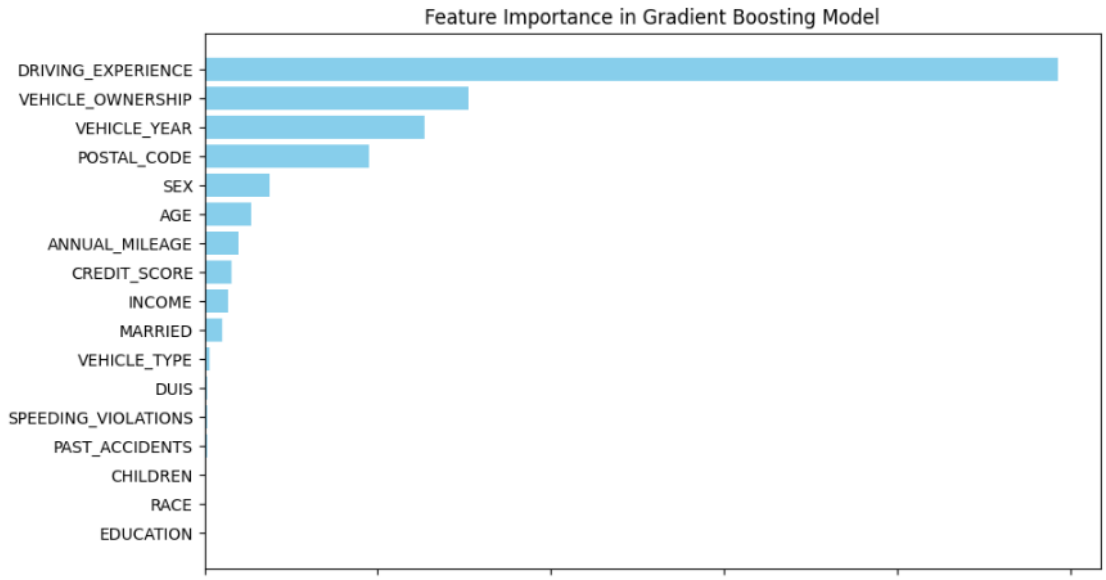
Hình 48: Kết quả huấn luyện GBM

- Độ chính xác (Accuracy): Mô hình đạt độ chính xác 84.96%, tương đương với AdaBoost, cho thấy Gradient Boosting có khả năng phân loại tốt trên tập dữ liệu này.

- Hiệu suất theo từng lớp:
    - Lớp 0: Precision 0.89 và recall 0.90, với F1-score 0.89. Mô hình hoạt động rất hiệu quả trong việc phân loại các mẫu thuộc lớp 0, với hiệu suất cao tương tự như AdaBoost.
    - Lớp 1: Precision 0.77 và recall 0.74, với F1-score 0.75. Hiệu suất của mô hình đối với lớp 1 khá tốt và có sự cân bằng giữa precision và recall, dù vẫn thấp hơn so với lớp 0.



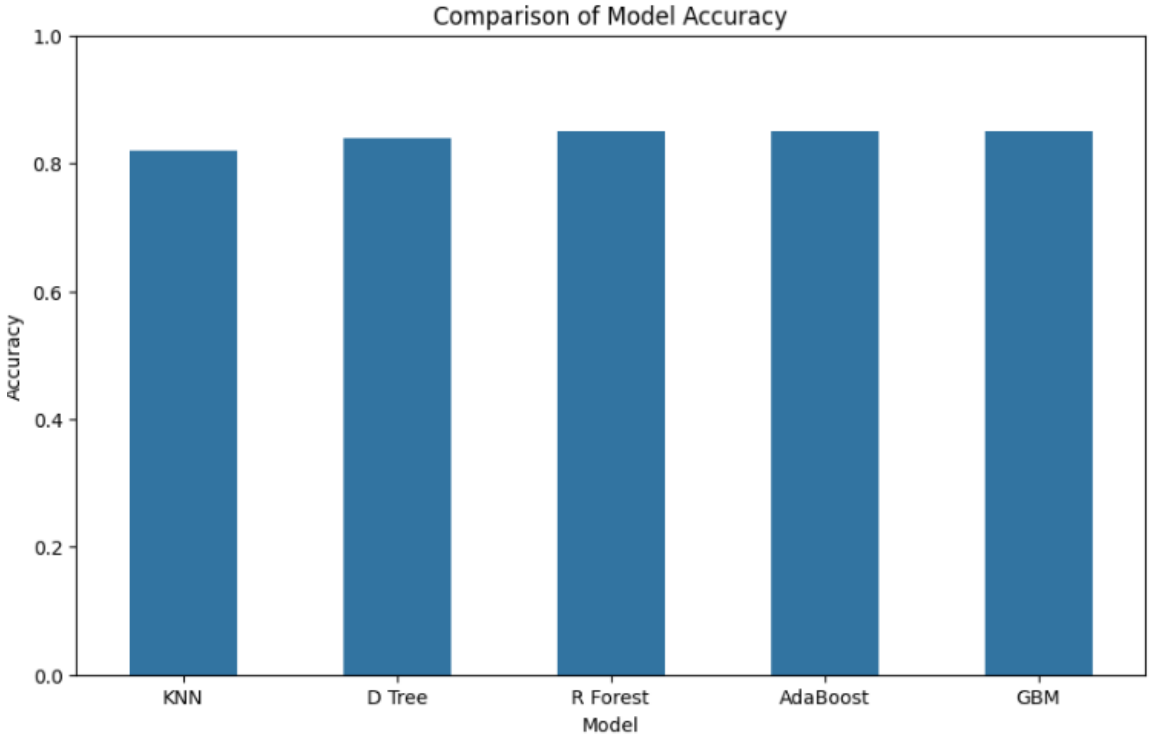
Hình 49 Ma trận nhầm lẫn GBM



Hình 50: Biểu đồ thể hiện tầm quan trọng giữa các đặc trưng GBM

- Giống với các mô hình khác, Dring_experience là đặc trưng quan trọng nhất trong mô hình GBM

2.5 So sánh giữa các mô hình máy học



Hình 51: So sánh giữa các mô hình

Nhìn chung, các mô hình máy học hoạt động khá tốt trên tập dữ liệu (độ chính xác trên 80%), với KNN có độ chính xác thấp nhất là 82%, trong khi Decision Tree đạt 84%.

Các mô hình Random Forest, AdaBoost, và Gradient Boosting đều thể hiện hiệu suất cao nhất, đồng thời đồng điểm với độ chính xác là 85%. Điều này cho thấy rằng các phương pháp boosting và rừng ngẫu nhiên có khả năng phân loại mạnh mẽ hơn và có thể phù hợp hơn cho các bài toán phức tạp trong thực tiễn, trong khi KNN có thể chưa tận dụng được tối đa thông tin từ dữ liệu.

Kết Luận Chung

Thông qua việc nghiên cứu và so sánh hiệu suất của nhiều mô hình học máy khác nhau trong bài toán phân loại dữ liệu. Qua quá trình phân tích, các mô hình **KNN**, **Decision Tree**, **Random Forest**, **AdaBoost**, và **Gradient Boosting** đã được áp dụng và đánh giá.

Kết quả cho thấy rằng các mô hình boosting, cụ thể là **AdaBoost** và **Gradient Boosting**, cùng với **Random Forest**, đạt được độ chính xác cao nhất là **85%**, thể hiện khả năng phân loại mạnh mẽ và tính ổn định. Mặc dù **KNN** có độ chính xác thấp hơn (**82%**), nhưng vẫn cung cấp một cái nhìn tổng quát về khả năng phân loại của mô hình.

Sự thành công của các mô hình này trong việc nhận diện các lớp khác nhau cho thấy rằng việc lựa chọn và tối ưu hóa mô hình là yếu tố quan trọng quyết định hiệu suất trong các bài toán phân loại. Kết quả nghiên cứu khuyến nghị rằng trong những bài toán phân loại phức tạp, các phương pháp boosting và Random Forest nên được ưu tiên do khả năng xử lý và hiệu suất tốt hơn.

Nhìn chung, nghiên cứu này đã cung cấp cái nhìn sâu sắc về ứng dụng của các mô hình học máy trong thực tiễn, mở ra hướng đi cho các nghiên cứu và ứng dụng trong tương lai nhằm cải thiện độ chính xác và hiệu quả trong việc giải quyết các bài toán phân loại.

TÀI LIỆU THAM KHẢO

https://www.kaggle.com/code/thuongtuandang/lightgbm-for-imputing-and-modeling

https://chatgpt.com/

https://www.kaggle.com/datasets/sagnik1511/car-insurance-data