

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT
KHOA HỆ THÔNG THÔNG TIN



PHÂN TÍCH DỮ LIỆU VỚI R/PYTHON

GVHD: Nguyễn Quang Phúc

NHÓM 2

1. Lâm Thị Hoài Thanh K184060802
2. Trần Khánh Duy K184060780
3. Ngô Hữu Tài K184060801
4. Hồ Thùy Tiên K184060808
5. Nguyễn Phạm Thùy Tiên K184060809

TP. HCM, ngày 22 tháng 08 năm 2021

MỤC LỤC

DANH MỤC HÌNH ẢNH.....	1
DANH MỤC BẢNG BIÊU	5
CHƯƠNG 1. MÔ HÌNH HỒI QUY TUYẾN TÍNH ĐƠN BIẾN	6
1.1. Cơ sở lý thuyết.....	6
1.2. Giới thiệu dữ liệu.....	8
1.3. Phân tích dự báo với Python.....	9
1.3.1. Cài đặt thư viện.....	9
1.3.2. Khám phá dữ liệu	9
1.3.3. Xây dựng mô hình	11
1.3.4. Tiến hành dự đoán	13
1.4. Phân tích dự báo với R	14
1.4.1. Load dữ liệu.....	14
1.4.2. Khám phá dữ liệu	15
1.4.3. Xây dựng mô hình	16
1.4.4. Tiến hành dự đoán	18
CHƯƠNG 2. MÔ HÌNH HỒI QUY TUYẾN TÍNH ĐA BIẾN	19
2.1. Cơ sở lý thuyết.....	19
2.1.1. Khái niệm	19
2.1.2. R bình phương và R bình phương hiệu chỉnh	19
2.1.2.1. R bình phương (R-square).....	19
2.1.2.2. Hệ số R bình phương hiệu chỉnh (Adjusted R-square)	20
2.1.2.3. Phản dư (Residuals).....	21
2.1.2.4. Variance Inflation Factor (VIF)	21
2.2. Giới thiệu dữ liệu.....	22
2.3. Phân tích dự báo với Python.....	24
2.3.1. Cài đặt thư viện.....	24
2.3.2. Khám phá dữ liệu	25

2.3.3. Tiền xử lý dữ liệu	30
2.3.4. Xây dựng mô hình	30
2.3.5. Đánh giá mô hình	32
2.3.6. Tiến hành dự đoán	36
2.4. Phân tích dự báo với R	37
2.4.1. Cài đặt thư viện.....	37
2.4.2. Khám phá dữ liệu	37
2.4.3. Tiền xử lý dữ liệu	41
2.4.4. Xây dựng mô hình	42
2.4.5. Kiểm định mô hình và dự báo	43
CHƯƠNG 3. MÔ HÌNH HỒI QUY LOGISTIC ĐƠN BIẾN	47
3.1. Cơ sở lý thuyết.....	47
3.1.1. Phân loại biến số.....	47
3.1.2. Khái niệm	48
3.1.3. Chỉ số chênh (Odds)	49
3.1.4. Tỷ số chênh (Odds Ratio - OR).....	50
3.1.5. Hàm Sigmoid.....	50
3.2. Giới thiệu dữ liệu.....	50
3.3. Phân tích dự báo với Python.....	52
3.3.1. Cài đặt thư viện.....	52
3.3.2. Khám phá dữ liệu	53
3.3.3. Xây dựng mô hình	54
3.3.3.1. Biến liên tục.....	55
3.3.3.2. Biến nhị phân.....	59
3.3.3.3. Biến thứ bậc.....	62
3.4. Phân tích dự báo với R	66
3.4.1. Cài đặt thư viện.....	66
3.4.2. Khám phá dữ liệu	66
3.4.3. Xây dựng mô hình	67

3.4.3.1. Biến liên tục.....	67
3.4.3.2. Biến nhị phân.....	70
3.4.3.3. Biến thứ bậc.....	72
CHƯƠNG 4. MÔ HÌNH HỒI QUY LOGISTIC ĐA BIẾN	74
4.1. Cơ sở lý thuyết.....	74
4.2. Phân tích dự báo với Python.....	74
4.2.1. Cài đặt thư viện.....	74
4.2.2. Xây dựng mô hình	75
4.2.2.1. Biến liên tục.....	75
4.2.2.2. Biến nhị phân.....	78
4.2.2.3. Biến thứ bậc.....	81
4.3. Phân tích dự báo với R	84
4.3.1. Cài đặt thư viện.....	84
4.3.2. Load dữ liệu.....	84
4.3.3. Xây dựng mô hình	85
4.3.3.1. Biến liên tục.....	85
4.3.3.2. Biến nhị phân.....	87
4.3.3.3. Biến thứ bậc.....	88
4.3.4. Đánh giá mô hình	89
CHƯƠNG 5. DỮ LIỆU CHUỖI THỜI GIAN & MÔ HÌNH ARIMA	91
5.1. Cơ sở lý thuyết.....	91
5.1.1. Khái niệm	91
5.1.2. Thành phần của chuỗi thời gian	92
5.1.2.1. Xu hướng (Trend - T)	92
5.1.2.2. Mùa vụ (Seasonal – S).....	92
5.1.2.3. Chu kỳ (Cyclical – C).....	93
5.1.2.4. Ngẫu nhiên (Irregular – I)	93
5.1.3. Mô hình ARIMA	94
5.2. Giới thiệu dữ liệu.....	95

5.3. Phân tích dự báo với Python.....	96
5.3.1. Cài đặt thư viện.....	96
5.3.2. Khám phá dữ liệu	97
5.3.3. Kiểm tra tính dừng.....	98
5.3.4. Phân tích bằng mô hình tự hồi quy.....	99
5.3.5. Phân tích bằng mô hình ARMA	101
5.3.6. Phân tích bằng mô hình ARIMA.....	103
5.4. Phân tích dự báo với R	104
5.4.1. Cài đặt thư viện.....	104
5.4.2. Kiểm tra tính dừng.....	104
5.4.3. Phân tích bằng mô hình tự hồi quy.....	105
5.4.4. Phân tích bằng mô hình ARMA	106
5.4.5. Phân tích bằng mô hình ARIMA.....	107
CHƯƠNG 6. RECURRENT NEURAL NETWORK.....	109
6.1. Cơ sở lý thuyết.....	109
6.1.1. Phương pháp máy học – học sâu (Deeplearning).....	109
6.1.2. Mô hình RNNs (Recurrent Neural Networks)	109
6.1.3. Vấn đề bị mất độ dốc (Vanishing Gradient Problem)	110
6.1.4. Mô hình LSTM (Long Short-Term Memory)	110
6.1.5. Phương pháp chính quy hóa Dropout (Dropout Regularization)	111
6.1.6. Thuật toán Adam	111
6.2. Phân tích dự báo bằng Python	112
6.2.1. Giới thiệu bộ dữ liệu.....	112
6.2.2. Cài đặt thư viện và Load dữ liệu	113
6.2.3. Khám phá dữ liệu	113
6.2.4. Xây dựng mô hình LSTM	114
6.2.5. Tiến hành dự đoán	116
6.3. Phân tích dự báo bằng R.....	117
6.3.1. Cài đặt thư viện.....	117

6.3.2. Chuẩn hóa dữ liệu.....	118
6.3.3. Xây dựng mô hình LSTM	118
6.3.4. Tiến hành dự đoán	118
CHƯƠNG 7. TỔNG QUAN CÁC MÔ HÌNH MÁY HỌC.....	120
7.1. Mô hình Decision Tree	120
7.1.1. Khái niệm	120
7.1.2. Thuật toán ID3.....	121
7.1.2.1. Hàm số Entropy	121
7.1.2.2. Information Gain	122
7.1.3. Thuật toán C.45	123
7.1.4. Thuật toán CART	124
7.1.5. Ưu và nhược điểm	125
7.1.5.1. Ưu điểm	125
7.1.5.2. Nhược điểm	125
7.2. Mô hình Random Forest	126
7.2.1. Khái niệm	126
7.2.2. Thuật toán máy học	126
7.2.3. Ưu và nhược điểm	127
7.2.3.1. Ưu điểm	127
7.2.3.2. Nhược điểm	127
7.3. Mô hình Neural Network.....	127
7.3.1. Khái niệm	127
7.3.2. Đặc điểm.....	128
7.3.3. Kiến trúc Neural Network	128
7.3.4. Quy tắc chuỗi (Chain Rule).....	129
7.3.5. Thuật toán Gradient Descent	130
7.3.6. Thuật toán Lan truyền ngược (Backpropagation)	131
7.3.7. Activation function	133
7.3.8. ArgMax và SoftMax.....	133

7.4. Mô hình Support Vector Machine	134
7.4.1. Khái niệm	134
7.4.2. Các tham số trong SVM	134
7.4.2.1. Tham số Kernel	134
7.4.2.2. Tham số Regularization.....	135
7.4.2.3. Tham số Gamma	135
7.4.2.4. Tham số Margin	136
7.4.3. Ưu và nhược điểm	137
7.4.3.1. Ưu điểm	137
7.4.3.2. Nhược điểm	137
7.5. Các phương pháp đánh giá mô hình	137
7.5.1. Precision và Recall	137
7.5.2. F1 Score	139
7.5.3. Accuracy	139
7.5.4. AUC – ROC	139
7.5.4.1. Khái niệm	139
7.5.4.2. Các chỉ số sử dụng trong AUC - ROC	140
7.5.4.3. Mối liên hệ giữa Specificity - Sensitivity, TPR - FPR.....	141
CHƯƠNG 8. ỨNG DỤNG CÁC MÔ HÌNH MÁY HỌC.....	142
8.1. Giới thiệu dữ liệu.....	142
8.2. Phân tích dự báo với Python.....	143
8.2.1. Cài đặt thư viện.....	143
8.2.2. Khám phá dữ liệu	144
8.2.3. Tiền xử lý dữ liệu	150
8.2.4. Xây dựng mô hình	151
8.2.4.1. Mô hình Decision Tree.....	151
8.2.4.2. Mô hình Random Forest.....	155
8.2.4.3. Mô hình Neural Network	157
8.2.4.4. Mô hình Support Vector Machine.....	161

8.2.5. Đánh giá.....	163
8.2.6. Hướng phát triển.....	164
8.3. Phân tích dự báo với R	165
8.3.1. Mô hình Decision Tree	165
8.3.1.1. Cài đặt thư viện	165
8.3.1.2. Tiền xử lý dữ liệu	165
8.3.1.3. Xây dựng mô hình.....	166
8.3.1.4. Đánh giá mô hình	167
8.3.2. Mô hình Random Forest.....	168
8.3.2.1. Cài đặt thư viện	168
8.3.2.2. Tiền xử lý dữ liệu	168
8.3.2.3. Xây dựng mô hình.....	169
8.3.2.4. Đánh giá mô hình	169
8.3.2.5. Xếp hạng thuộc tính	171
8.3.3. Mô hình Neural Network.....	172
8.3.3.1. Cài đặt thư viện	172
8.3.3.2. Tiền xử lý dữ liệu	173
8.3.3.3. Xây dựng mô hình.....	174
8.3.3.4. Đánh giá mô hình	175
8.3.4. Mô hình Support Vector Machine.....	178
8.3.4.1. Xây dựng mô hình.....	178
8.3.4.2. Đánh giá mô hình	179
CHƯƠNG 9. TỔNG KẾT	181
TÀI LIỆU THAM KHẢO	182

DANH MỤC HÌNH ẢNH

Hình 1.1. Mô hình hồi quy tuyến tính thể hiện mối quan hệ giữa chi tiêu và thu nhập	7
Hình 1.2. Các dòng đầu tiên của bộ dữ liệu “Salary.csv”	9
Hình 1.3. Các giá trị thống kê của bộ dữ liệu “Salary.csv”.....	10
Hình 1.4. Biểu đồ thể hiện tương quan giữa “YearsExperience” và “Salary”	11
Hình 1.5. Biểu đồ hồi quy tuyến tính giữa hai biến “YearsExperience” và “Salary”.....	13
Hình 1.6. Biểu đồ mô tả giá trị dự đoán so với giá trị thật trong bộ dữ liệu Salary.csv....	14
Hình 1.7. Biểu đồ thể hiện tương quan giữa “YearsExperience” và “Salary” bằng R	16
Hình 1.8. Các thông số của mô hình với biến độc lập “YearsExperience”.....	17
Hình 1.9. Biểu đồ hồi quy tuyến tính đơn biến bằng R	18
Hình 2.1. Các dòng đầu tiên của bộ dữ liệu “CarPrice_Assignment.csv”	25
Hình 2.2. Biểu đồ tần suất kèm đường biểu diễn ước lượng mật độ của biến “price”.....	26
Hình 2.3. Biểu đồ nhiệt thể hiện tương quan giữa các biến dữ liệu số với biến “price” ...	27
Hình 2.4. Datset sau khi chuyển đổi	28
Hình 2.5. Biểu đồ nhiệt thể hiện tương quan giữa tất cả các biến.....	29
Hình 2.6. Kết quả của mô hình OLS Regression	32
Hình 2.7. Biểu đồ tần suất kèm đường biểu diễn ước lượng mật độ chênh lệch giá xe thực tế so với giá xe dự đoán.....	33
Hình 2.8. Biểu đồ thể hiện mối liên hệ giữa kết quả dự đoán so với giá xe thực tế.....	35
Hình 2.9. Các giá trị thống kê trong bộ dữ liệu “CarPrice_Assignment.csv”	38
Hình 2.10. Biểu đồ tần suất kết hợp đường biểu diễn ước lượng giá xe bằng R	39
Hình 2.11. Biểu đồ nhiệt tương quan giữa các biến số so với nhau	40
Hình 2.12. Biểu đồ đường thể hiện sự thay đổi của thông số R-Squared	43
Hình 2.13. Biểu đồ thể hiện sự chênh lệch kết quả dự đoán và kết quả thực tế.....	44
Hình 2.14. Biểu đồ phân tán của kết quả dự đoán so với kết quả thực tế bằng R.....	45
Hình 3.1. Phân loại các biến số	47
Hình 3.2. Biểu đồ thể hiện sự phân chia hai lớp 0 và 1 của biến “target”	54
Hình 3.3. Biểu đồ phân tán thể hiện mối tương quan giữa 2 biến “thalach” và “target” ..	55

Hình 3.4. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “thalach”	56
Hình 3.5. Ma trận hỗn loạn dựa trên biến “thalach”	58
Hình 3.6. Biểu đồ thể hiện sự phân chia của biến “exang” với biến “target”	59
Hình 3.7. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “exang”	60
Hình 3.8. Ma trận hỗn loạn dựa trên biến “exang”	62
Hình 3.9. Biểu đồ thể hiện sự phân chia của biến “thal” với biến “target”	63
Hình 3.10. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “thal”	64
Hình 3.11. Ma trận hỗn loạn dựa trên biến “thal”	65
Hình 3.12. Các giá trị thống kê của từng biến trong bộ dữ liệu “heart.csv”	67
Hình 3.13. Số người mắc bệnh tim mạch dựa trên nhịp tim tối đa	68
Hình 3.14. Các thông số của mô hình với biến độc lập “thalach”	69
Hình 3.15. Đồ thị hồi quy logistic với biến độc lập “thalach”	69
Hình 3.16. Các thông số của mô hình với biến độc lập “exang”	71
Hình 3.17. Các thông số của mô hình với biến độc lập “thal”	72
Hình 4.1. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “thal” và “chol”	76
Hình 4.2. Ma trận hỗn loạn dựa trên 2 biến “thal” và “chol”	77
Hình 4.3. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “fbs” và “exang”	79
Hình 4.4. Ma trận hỗn loạn dựa trên 2 biến “fbs” và “exang”	80
Hình 4.5. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “slope” và “thal”	82
Hình 4.6. Ma trận hỗn loạn dựa trên 2 biến “slope” và “thal”	83
Hình 4.7. Load dữ liệu vào RStudio	85
Hình 4.8. Các thông số của mô hình với 2 biến độc lập “thalach” và “chol”	86
Hình 4.9. Biểu đồ phân tán ba chiều dựa trên các biến độc lập và mục tiêu	87
Hình 4.10. Các thông số của mô hình với 2 biến độc lập “fbs” và “exang”	88
Hình 4.11. Các thông số của mô hình với 2 biến độc lập “slope” và “thal”	89
Hình 5.1. Các thành phần trong chuỗi dữ liệu thời gian	92
Hình 5.2. Sơ đồ ACF và PACF	95
Hình 5.3. Số lượng ca nhiễm Covid 19 từ tháng 1 đến tháng 7 năm 2020	97
Hình 5.4. Tính chu kỳ, tính thời vụ, tính xu hướng và các điểm khác thường	98

Hình 5.5. Sơ đồ PACF.....	99
Hình 5.6. Kết quả của mô hình tự hồi quy	100
Hình 5.7. Kết quả dự đoán khi dùng mô hình AR so với dữ liệu thật.....	101
Hình 5.8. Sơ đồ ACF	101
Hình 5.9. Kết quả dự đoán khi dùng mô hình ARMA so với dữ liệu thật	102
Hình 5.10. Kết quả dự đoán bằng mô hình ARIMA so với dữ liệu thật	104
Hình 5.11. Kết quả mô hình AR trên R	106
Hình 5.12. Kết quả dự đoán mô hình ARMA	107
Hình 5.13. Kết quả dự đoán mô hình ARIMA bằng R.....	108
Hình 6.1. Mô hình RNNs.....	109
Hình 6.2. Các dạng mô hình RNNs	110
Hình 6.3. Cách thức mô hình LSTM hoạt động tại 1 lớp.....	111
Hình 6.4. Thuật toán đồ dốc Gradient Descent	112
Hình 6.5. Giá trị của cổ phiếu khi mở phiên giao dịch từ năm 2012 đến năm 2016	114
Hình 6.6. So sánh kết quả dự đoán và dữ liệu thật.....	117
Hình 6.7. Kết quả dự đoán mô hình LSTM trên R.....	119
Hình 7.1. Minh họa mô hình Decision Tree	120
Hình 7.2. Đồ thị hàm Entropy với $n = 2$	122
Hình 7.3. Thuật toán máy học của Random Forest	127
Hình 7.4. Kiến trúc mạng Neural Network	129
Hình 7.5. Ảnh hưởng của độ học.....	131
Hình 7.6. Mô tả cách thức hoạt động của thuật toán Backpropagation	132
Hình 7.7. Minh họa mô hình SVM.....	134
Hình 7.8. Tóm tắt kernel thông dụng	135
Hình 7.9. Minh họa 2 trường hợp của hệ số gamma (thấp và cao)	136
Hình 7.10. Margin tốt trong SVM	136
Hình 7.11. Margin xấu trong SVM	137
Hình 7.12. Cách tính Precision và Recall.....	138
Hình 7.13. Đường cong ROC	140

Hình 8.1. Biểu đồ thể hiện mối tương quan giữa các biến trong bộ dữ liệu online_shoppers_intention.csv	145
Hình 8.2. Biểu đồ phân chia hai lớp True - False trong “Revenue” và “Weekend”	146
<i>Hình 8.3. Biểu đồ tương quan giữa “Revenue - VisitType – Month”</i>	147
Hình 8.4. Biểu đồ miêu tả vị trí phân bố dữ liệu của các biến độc lập	148
Hình 8.5. Biểu đồ thể hiện sự phân phối dữ liệu của các biến trong bộ dữ liệu	149
Hình 8.6. Giá trị của các biến phân loại sau khi mã hóa	150
Hình 8.7. Mô hình cây quyết định	152
Hình 8.8. Biểu đồ ROC của mô hình Decision Tree	155
Hình 8.9. Biểu đồ ROC của mô hình Random Forest	157
Hình 8.10. Biểu đồ ROC của mô hình Neural Network.....	160
Hình 8.11. Biểu đồ ROC của mô hình SVM	163
Hình 8.12. Mô hình cây quyết định bằng R	167
Hình 8.13. Top 5 thuộc tính quan trọng trong mô hình Random Forest	171
Hình 8.14. Biểu đồ thể hiện đường cong Precision - Recall của mô hình mạng nơ ron .	177
Hình 8.15. Biểu đồ đường cong ROC của mô hình Neural Network bằng R	178
Hình 8.16. Biểu đồ đường cong ROC của mô hình SVM bằng R	180

DANH MỤC BẢNG BIỂU

Bảng 1.1. Dữ liệu về mức thu nhập và chi tiêu	7
Bảng 2.1. Các thuộc tính của bộ dữ liệu “CarPrice_Assignment.csv”	24
Bảng 3.1. Kết quả tiêm ngừa phòng bệnh PRRS của heo nái	49
Bảng 3.2. Các thuộc tính của bộ dữ liệu “heart.csv”	52
Bảng 5.1. Ví dụ về dữ liệu chuỗi thời gian (ĐVT: triệu đồng)	91
Bảng 5.2. Các thuộc tính trong bộ dữ liệu “day_wise.csv”	96
Bảng 6.1. Các thuộc tính trong bộ dữ liệu “Google_Stock_Price.csv”.....	113
Bảng 8.1. Các thuộc tính của bộ dữ liệu “online_shoppers_intention.csv”	143
Bảng 8.2. Bảng đánh giá kết quả của 4 mô hình trên lớp 1.....	163

CHƯƠNG 1. MÔ HÌNH HỒI QUY TUYẾN TÍNH ĐƠN BIỀN

1.1. Cơ sở lý thuyết

Hồi quy tuyến tính đơn biến (*Simple Linear Regression*) là phương pháp nghiên cứu mối quan hệ giữa biến phụ thuộc (biến mục tiêu) vào một biến độc lập (ảnh hưởng đến biến mục tiêu) để ước lượng hay dự đoán giá trị trung bình của biến phụ thuộc trên cơ sở các giá trị biết trước của biến độc lập.

Phương trình tổng quát của hồi quy tuyến tính đơn biến được biểu diễn như sau:

$$y = \beta_0 + \beta_1 X + \varepsilon$$

Trong đó:

- y là biến phụ thuộc (chịu ảnh hưởng của biến X).
- X là biến độc lập (biến tác động lên biến phụ thuộc).
- β_0 là hệ số chặn (*Intercept*) hay giá trị ước lượng của y khi $X = 0$.
- β_1 là độ dốc của đường hồi quy tuyến tính (*Slope*), nói cách khác là sự tăng hoặc giảm của y khi X thay đổi 1 đơn vị. Nếu $\beta_1 > 0$ thì X và y có mối quan hệ cùng chiều (X tăng, y tăng). Nếu $\beta_1 < 0$, thì X và y có mối quan hệ nghịch biến (X tăng, y giảm).
- ε là thành phần sai số ngẫu nhiên có phân phối chuẩn với trung bình bằng 0, phương sai bằng nhau, và độc lập không có liên hệ với biến nào khác.

Mô hình hồi quy bao gồm hai thành phần, phần xác định của mô hình $\beta_0 + \beta_1 X$, biểu diễn mối quan hệ đường thẳng và thành phần sai số ngẫu nhiên ε , thể hiện giá trị của các yếu tố khác không thể nghiên cứu hết và các yếu tố này vẫn tác động lên giá trị của y .

Trong thực tế, chúng ta không thể xác định chính xác α hay β mà chỉ ước lượng được nêu ở phương trình tổng quát có sai số nhất định. Đối với phương trình hồi quy tuyến tính đơn biến dùng cho dự báo, thông thường chúng ta sẽ xử lý dữ liệu mẫu và lấy kết quả

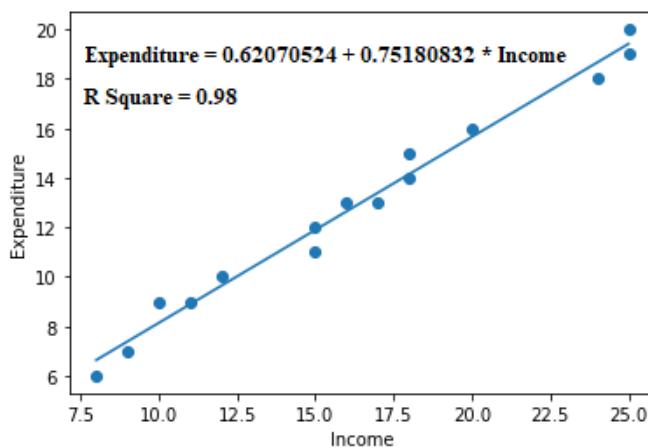
từ đây ước lượng cho tổng thể nên chúng ta loại bỏ sai số ϵ ra ngoài. Phương trình lúc này có công thức: $y = \beta_0 + \beta_1 X$

Ví dụ, ta có bảng dữ liệu về mức thu nhập và chi tiêu như sau: (ĐVT: 1 triệu VNĐ)

Income	Expenditure
8	6
9	7
10	9
11	9
12	1
15	12
15	11
16	13
17	13
18	15
18	14
20	16
24	18
25	2
25	19

Bảng 1.1. Dữ liệu về mức thu nhập và chi tiêu

Từ bảng trên ta có mô hình hồi quy tuyến tính thể hiện mối quan hệ giữa chi tiêu và thu nhập:



Hình 1.1. Mô hình hồi quy tuyến tính thể hiện mối quan hệ giữa chi tiêu và thu nhập

Phương trình hồi quy tương ứng:

$$\text{Expenditure} = 0.62070524 + 0.75180832 * \text{Income}$$

Từ phương trình trên ta có thể kết luận khi thu nhập tăng lên 1 đơn vị thì chi tiêu tăng trung bình 0.75180832 triệu đồng.

Ngoài ra tham số R bình phương (*R Squared*) cho biết mức độ (%) sự biến thiên của biến phụ thuộc được giải thích bởi biến độc lập. Trong ví dụ này, có thể nói 98% sự biến đổi chi tiêu có thể được giải thích bằng sự biến đổi về mức thu nhập của một người (ĐVT: 1 triệu VND).

Nhìn vào Hình 1.1, ta thấy các chấm tròn không nằm trên đường thẳng hồi quy thể hiện sự sai lệch trong dự đoán. Sự sai lệch này trong thống kê gọi là phần dư (*residual*) hoặc errors (ϵ). Như vậy nếu phần dư ϵ càng nhỏ sự liên hệ giữa X, y càng lớn và ngược lại.

1.2. Giới thiệu dữ liệu

Bộ dữ liệu “Salary_Data.csv” miêu tả lương của nhân viên dựa trên số năm kinh nghiệm của họ. Dữ liệu được lấy từ: <https://www.kaggle.com/vihansp/salary-data>

Bộ dữ liệu gồm 30 dòng với 2 thuộc tính như sau:

- YearsExperience: Số năm kinh nghiệm của nhân viên
- Salary: Lương của nhân viên

Dựa vào bộ dữ liệu này, chúng ta sẽ dùng phương pháp hồi quy tuyến tính đơn biến và triển khai trên ngôn ngữ Python và R để dự đoán mức lương của nhân viên dựa trên số năm kinh nghiệm của họ.

1.3. Phân tích dự báo với Python

1.3.1. Cài đặt thư viện

```
#Import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

- Thư viện Pandas: load và đọc dữ liệu.
- Thư viện Numpy: làm việc với dữ liệu dạng chuỗi.
- Thư viện Matplotlib: trực quan hóa dữ liệu.
- Thư viện Scikit-learn: sử dụng thuật toán LinearRegression để xây dựng mô hình hồi quy tuyến tính.

1.3.2. Khám phá dữ liệu

Để load dữ liệu trong Python ta sử dụng lệnh `read_csv` và sử dụng hàm `head()` để xem bộ dữ liệu. Trong bộ dữ liệu này, biến “YearsExperience” là biến phụ thuộc, biến “Salary” là biến độc lập.

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0

Hình 1.2. Các dòng đầu tiên của bộ dữ liệu “Salary.csv”

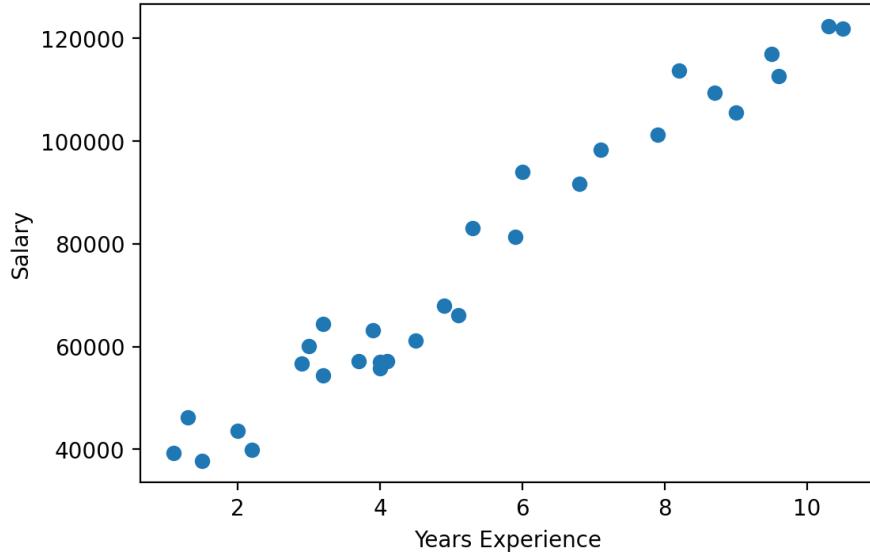
Tiếp theo ta sử dụng hàm describe() để xem các giá trị thống kê min, max, tứ phân vị của từng biến trong bộ dữ liệu.

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

Hình 1.3. Các giá trị thống kê của bộ dữ liệu “Salary.csv”

Để biểu diễn mối tương quan giữa 2 biến ta sử dụng biểu đồ phân tán trong thư viện Matplotlib.

```
#Scatter plot of 'YearsExperience' vs 'Salary'  
plt.scatter(df.YearsExperience, df.Salary)  
plt.ylabel('Salary')  
plt.xlabel('Years Experience')  
plt.show()
```



Hình 1.4. Biểu đồ thể hiện tương quan giữa “YearsExperience” và “Salary”

Từ hình trên ta thấy số năm kinh nghiệm của nhân viên càng cao thì mức lương cũng sẽ tăng theo.

1.3.3. Xây dựng mô hình

Để xây dựng mô hình hồi quy tuyến tính đơn biến ta sử dụng thuật toán LinearRegression trong bộ Scikit-learn, chúng ta thực hiện đoạn code trong Python như sau:

```
#Create model
x = df['YearsExperience']
x_ = np.array(x).reshape(-1,1)
model = LinearRegression().fit(x_, df.Salary)
```

Biến model chứa một đối tượng LinearRegression trong bộ thư viện Scikit-learn và tiếp theo là thực hiện phương thức fit() trên đối tượng này. Fit() thực hiện tính toán tối ưu hóa các tham số β_0 và β_1 . Sau khi thực hiện phương thức này, chúng ta đã có một đối tượng chứa đầy đủ thông tin kết quả của mô hình.

```
#Get results
intercept = model.intercept_
slope = model.coef_
score = model.score(x_, df.Salary)
print('Intercept:', intercept)
print('Slope:', slope)
print('R-squared:', score)
```

Intercept: 25792.200198668696
 Slope: [9449.96232146]
 R-squared: 0.9569566641435084

Từ kết quả trên ta suy ra được phương trình hồi quy có dạng:

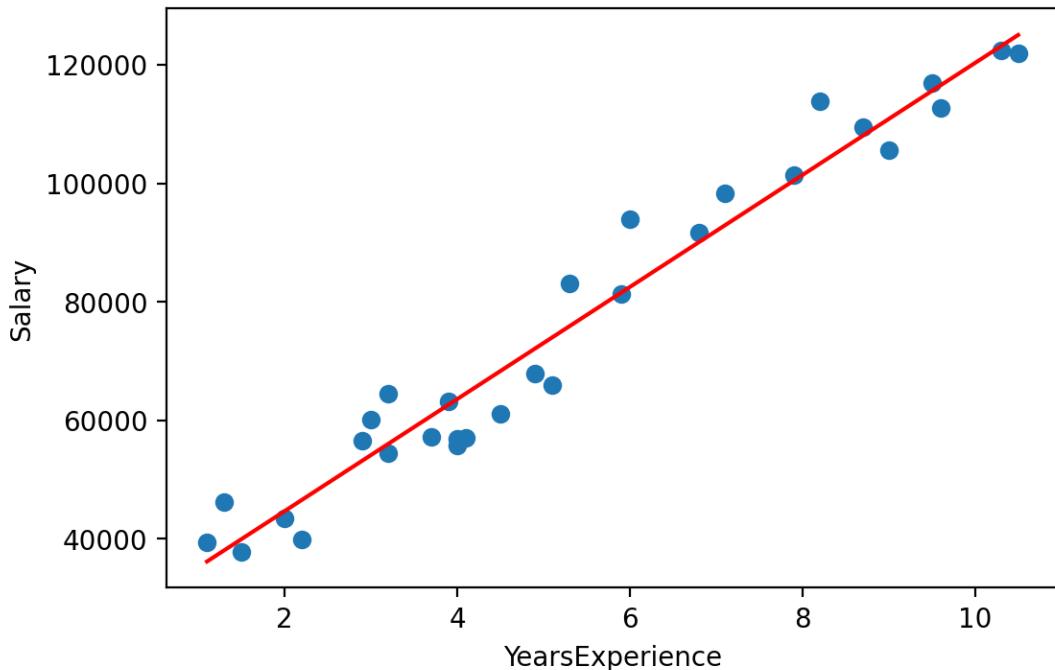
$$\text{Salary} = 25792.20 + 9449.96 * \text{YearsExperience}$$

Khi số năm kinh nghiệm của nhân viên tăng lên 1 đơn vị thì mức lương của họ tăng trung bình 9449.96 (đơn vị tiền tệ).

Hệ số R-squared bằng 0.95 cho thấy 95% sự biến đổi về mức lương của nhân viên có thể được giải thích bằng sự biến đổi dựa trên số năm kinh nghiệm của họ.

Với hai hệ số Intercept và Slope ta vẽ được đường thẳng mô tả mối quan hệ giữa mức lương và số năm kinh nghiệm.

```
#Visualization
plt.scatter(x_, df.Salary)
plt.plot(x_, intercept + slope * x, color='r')
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.show()
```

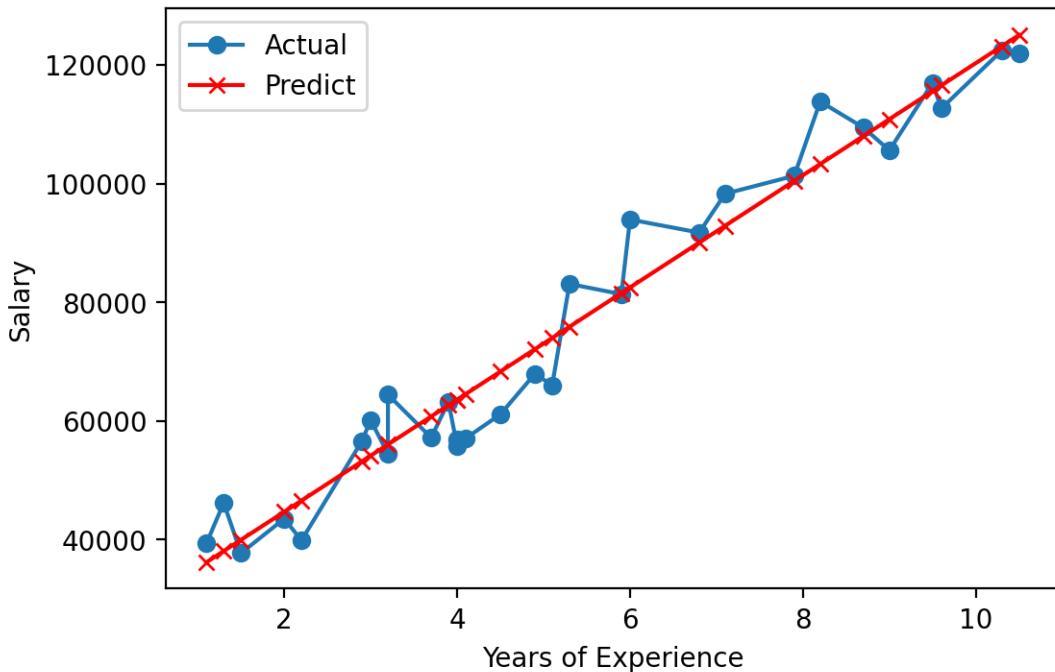


Hình 1.5. Biểu đồ hồi quy tuyến tính giữa hai biến “YearsExperience” và “Salary”

1.3.4. Tiến hành dự đoán

Để dự đoán các giá trị hiện tại so với bộ dữ liệu ta sử dụng hàm predict() và trực quan hóa bằng biểu đồ dưới đây.

```
#Predict for present values
pred_values = model.predict(x_)
plt.plot(df.YearsExperience, df.Salary, label='Actual', marker='o')
plt.plot(df.YearsExperience, pred_values,color='r', label='Predict', marker='x')
plt.legend()
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```



Hình 1.6. Biểu đồ mô tả giá trị dự đoán so với giá trị thật trong bộ dữ liệu Salary.csv

Ngoài ra chúng ta cũng có thể dự đoán các giá trị mới như đoạn code dưới đây.

```
#Predict for future values
x2 = np.array([5.5, 2]).reshape(-1,1)
y2 = model.predict(x2)
print(y2)

[77766.99296667 44692.12484158]
```

Với 1 người có 5.5 năm kinh nghiệm thì mức lương sẽ là 77766.99 (đơn vị tiền tệ), còn 1 người có 2 năm kinh nghiệm thì mức lương là 44692.12 (đơn vị tiền tệ).

1.4. Phân tích dự báo với R

1.4.1. Load dữ liệu

Để import dữ liệu ta chọn vào “Import Dataset” đã có sẵn trong RStudio hoặc sử dụng lệnh read_csv tương tự như ở Python.

1.4.2. Khám phá dữ liệu

Như ta thấy dữ liệu gồm có 30 dòng và 2 thuộc tính. Ta sử dụng hàm str() để xem một số thông tin của bộ dữ liệu.

```
> str(data)
'data.frame': 30 obs. of 2 variables:
 $ YearsExperience: num 1.1 1.3 1.5 2 2.2 2.9 3 3.2 3.2 3.7 ...
 $ Salary          : num 39343 46205 37731 43525 39891 ...
```

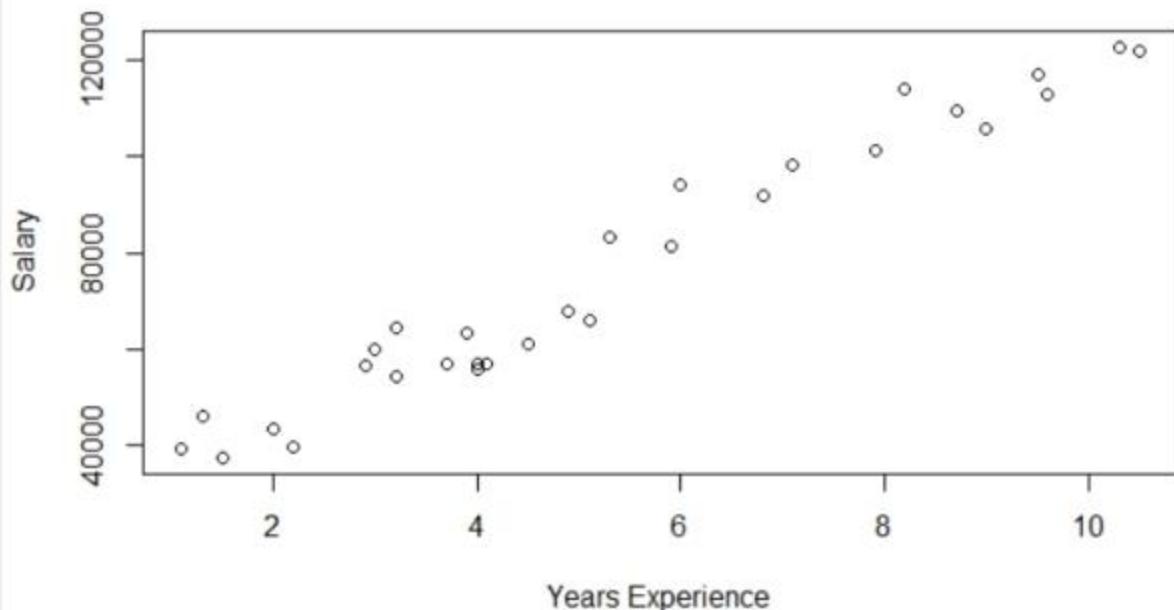
Tiếp theo ta sử dụng hàm summary() để xem một số thông tin cơ bản về giá trị min, max của từng biến trong bộ dữ liệu.

```
> summary(data)
   YearsExperience      Salary
Min.   : 1.100   Min.   : 37731
1st Qu.: 3.200   1st Qu.: 56721
Median : 4.700   Median : 65237
Mean   : 5.313   Mean   : 76003
3rd Qu.: 7.700   3rd Qu.:100545
Max.   :10.500   Max.   :122391
```

Trong bộ dữ liệu này, biến “YearsExperience” là biến phụ thuộc và biến “Salary” là biến độc lập. Ta sẽ xây dựng bài toán để dự đoán lương dựa theo năm kinh nghiệm của nhân viên bằng ngôn ngữ R.

```
7
8 #Visualize "YearsExperience" and "Salary"
9 plot(data$YearsExperience,data$Salary, xlab = 'Years Experience', ylab = 'Salary')
10
```

Để vẽ được biểu đồ thể hiện mối tương quan giữa 2 biến ta dùng hàm plot().



Hình 1.7. Biểu đồ thể hiện tương quan giữa “YearsExperience” và “Salary” bằng R

Sau khi chạy hàm ta thấy được giữa hai biến “YearsExperience” và “Salary” có sự tương quan với nhau.

1.4.3. Xây dựng mô hình

```

11 #Model Building
12 Salary_Experience = lm(Salary ~ YearsExperience, data )
13 abline(Salary_Experience, col='red')
14 summary(Salary_Experience)

```

Hàm lm() trong R có thể tính toán các giá trị của β_0 và β_1 , cũng như R^2 một cách nhanh gọn. Sau khi chạy hàm lm() ta có kết quả như sau:

```

Call:
lm(formula = Salary ~ YearsExperience, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-7958.0 -4088.5 -459.9  3372.6 11448.0 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 25792.2    2273.1   11.35 5.51e-12 ***
YearsExperience 9450.0     378.8   24.95 < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5788 on 28 degrees of freedom
Multiple R-squared:  0.957,    Adjusted R-squared:  0.9554 
F-statistic: 622.5 on 1 and 28 DF,  p-value: < 2.2e-16

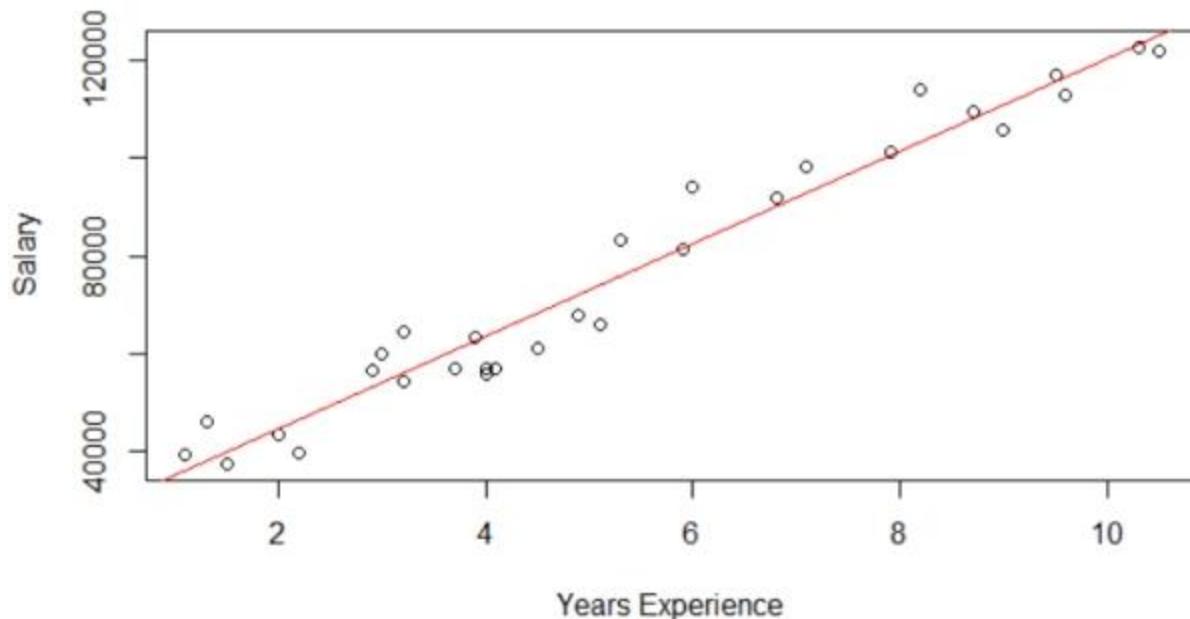
```

Hình 1.8. Các thông số của mô hình với biến độc lập “YearsExperience”

Trong lệnh trên, “Salary ~ YearsExperience” có nghĩa là mô tả Salary là một hàm số của YearsExperience. Kết quả tính toán của lm cho thấy $\beta_0 = 25792.2$ và $\beta_1 = 9450.0$. Nói cách khác, với hai thông số này, chúng ta có thể ước tính lương của một nhân viên thông qua năm kinh nghiệm bằng phương trình tuyến tính:

$$\text{Salary} = \mathbf{25792.2 + 9450.0 * YearsExperience}$$

Hàm abline() dùng để vẽ một đường biểu diễn hồi qui tuyến tính qua các điểm. Và đây là kết quả của hàm.



Hình 1.9. Biểu đồ hồi quy tuyến tính đơn biến bằng R

1.4.4. Tiên hành dự đoán

Ta dùng hàm predict() để dự đoán lương của nhân viên trong tương lai ở những năm xa hơn.

```

15
16 #Predict
17 predict(Salary_Experience)

```

Sau khi chạy hàm ta sẽ có kết quả mức lương của nhân viên dựa theo năm kinh nghiệm từ 1 năm đến 30 năm.

```

> predict(Salary_Experience)
   1      2      3      4      5      6      7      8      9
36187.16 38077.15 39967.14 44692.12 46582.12 53197.09 54142.09 56032.08 56032.08
  10     11     12     13     14     15     16     17     18
60757.06 62647.05 63592.05 63592.05 64537.05 68317.03 72097.02 73987.01 75877.00
  19     20     21     22     23     24     25     26     27
81546.98 82491.97 90051.94 92886.93 100446.90 103281.89 108006.87 110841.86 115566.84
  28     29     30
116511.84 123126.81 125016.80

```

CHƯƠNG 2. MÔ HÌNH HỒI QUY TUYẾN TÍNH ĐA BIẾN

2.1. Cơ sở lý thuyết

2.1.1. Khái niệm

Hồi quy tuyến tính đa biến (*Multiple Linear Regression*) là phương pháp nghiên cứu mối quan hệ giữa biến mục tiêu (biến phụ thuộc) với nhiều hơn 2 biến độc lập (biến đầu vào). Mô hình tổng quan với Y là biến phụ thuộc và những biến độc lập X₁, X₂, X₃,...,X_n.

$$Y = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \dots + \beta_n X_{ni} + \varepsilon_i$$

Trong đó:

- Y là biến phụ thuộc hoặc biến dự đoán.
- X_{1i}, X_{2i},..., X_{ni} là biến độc lập.
- β₀ là hệ số chặn (*Intercept*) hay giá trị ước lượng của y khi X = 0.
- β₁, β₂,..., β_n là độ dốc của đường hồi quy tuyến tính (*Slope*), nói cách khác là sự tăng hoặc giảm của y khi X thay đổi 1 đơn vị.
- ε là lỗi ngẫu nhiên (phản dư) của mô hình.

2.1.2. R bình phương và R bình phương hiệu chỉnh

2.1.2.1. R bình phương (*R-square*)

Hệ số xác định (*coefficient of determination*) thường kí hiệu là R², một con số thống kê tổng hợp khả năng giải thích của một phương trình. Nó biểu thị tỷ lệ biến thiên của biến phụ thuộc do tổng mức biến thiên của các biến giải thích gây ra và được tính bằng công thức:

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

Trong đó:

- RSS (*Residual Sum of Squares*) là tổng các độ lệch bình phương của phản dư.

- TSS (*Total Sum of Squares*) là tổng các độ lệch bình phương của toàn bộ các nhân tố nghiên cứu.
- Từ công thức trên, có thể thấy rằng R² phải nằm trong khoảng từ 0 đến 1. Khi R² càng gần 0, khả năng giải thích càng kém và điều ngược lại sẽ đúng khi các giá trị của nó tiến dần tới 1.

R-square hay R bình phương được sử dụng nhiều trong thống kê và được thực hiện bởi phương pháp gọi là hồi quy tuyến tính.

- R bình phương cho biết mô hình đó hợp với dữ liệu ở mức bao nhiêu %.
- R bình phương cũng cho biết độ phù hợp của mô hình.
- Giá trị R² càng cao thì mối quan hệ giữa nhân tố độc lập (biến độc lập) và nhân tố phụ thuộc càng chặt chẽ.

Hệ số xác định R² có ý nghĩa vô cùng quan trọng trong thống kê và nghiên cứu, đặc biệt là trong phương pháp hồi quy tuyến tính.

2.1.2.2. Hệ số R bình phương hiệu chỉnh (Adjusted R-square)

Bên cạnh R bình phương, R bình phương hiệu chỉnh cũng là một khái niệm không thể bỏ qua. Đây là một hệ số được sử dụng để hạn chế những nhược điểm của R bình phương.

Công thức tính R bình phương hiệu chỉnh:

$$R_{hc}^2 = 1 - \frac{ESS/(n-k)}{TSS/(n-1)}$$

Sau khi biến đổi ta được:

$$R_{hc}^2 = 1 - \frac{n-1}{n-k}(1 - R^2)$$

Trong đó:

- n là số lượng mẫu quan sát.

- k tham số của mô hình (bằng lượng biến độc lập cộng thêm 1).

Hạn chế nổi bật nhất của R bình phương là việc giảm tính chính xác của mô hình khi ta thêm một tham số trong quá trình tính toán. Vì vậy, R bình phương hiệu chỉnh được nghiên cứu giúp khắc phục nhược điểm của R bình phương thông thường. Hệ số này cho phép ta đo độ thích hợp khi ta thêm một tham số nữa. Qua đó giúp giảm sự phức tạp của mô hình.

2.1.2.3. Phần dư (Residuals)

Sự khác biệt giữa giá trị quan sát của biến phụ thuộc (y) và giá trị dự đoán (\hat{y}) được gọi là phần dư (e). Mỗi điểm dữ liệu có một phần dư.

Công thức tổng quát của Residual như sau:

$$\text{Residual} = \text{Giá trị quan sát} - \text{Giá trị dự đoán}$$

$$e = y - \hat{y}$$

Cả tổng và giá trị trung bình của các phần dư đều bằng 0, tức là $\sum e = 0$ và $e = 0$.

Biểu đồ Residual là đồ thị biểu diễn phần dư trên trục tung và biến số độc lập trên trục hoành. Nếu các điểm trong một biểu đồ Residual được phân tán ngẫu nhiên quanh trục hoành, thì mô hình hồi quy tuyến tính đang xét sẽ được xem là phù hợp với dữ liệu. Ngược lại, ta cần xem xét thay thế bằng một mô hình phi tuyến sẽ thích hợp hơn.

2.1.2.4. Variance Inflation Factor (VIF)

Variance Inflation Factor (VIF), hay còn được dịch là Hệ số lạm phát phương sai, là thước đo mức độ đa cộng tuyến trong một tập hợp nhiều biến hồi quy. Đa cộng tuyến là khi có mối tương quan giữa các yếu tố dự báo (tức là các biến độc lập) trong một mô hình; sự hiện diện của nó có thể ảnh hưởng xấu đến kết quả hồi quy. VIF ước tính phương sai của hệ số hồi quy bị thổi phồng lên bao nhiêu do đa cộng tuyến trong mô hình.

VIF có thể được tính theo công thức dưới đây:

$$VIF_i = \frac{1}{1 - R_i^2}$$

Trong đó:

- R_i^2 : là hệ số xác định chưa điều chỉnh để hồi quy biến độc lập thứ i trên các biến còn lại.
- Ngoài ra, nếu R_i^2 bằng 0 thì không thể dự đoán phương sai của các biến độc lập còn lại từ biến độc lập thứ i.

Quy tắc chung khi giải thích VIF:

- Bằng 1 = không tương quan.
- Trong khoảng từ 1 đến 5 = tương quan vừa phải.
- Lớn hơn 5 = tương quan cao.

Như vậy, nếu VIF càng tăng thì kết quả hồi quy càng kém tin cậy.

2.2. Giới thiệu dữ liệu

Bộ dữ liệu “CarPrice_Accident.csv” chứa thông tin về giá bán và thông số kỹ thuật các mẫu xe hơi trên thị trường Mỹ do công ty Geely Auto thu thập. Dữ liệu này được trích từ bộ dữ liệu “Automobile Dataset” tại UCI Machine Learning Repository.

Link dữ liệu: <https://www.kaggle.com/hellbuoy/car-price-prediction>

Bộ dữ liệu gồm 26 thuộc tính:

STT	Tên thuộc tính	Mô tả thuộc tính
1	car_ID	Số thứ tự của mẫu xe
2	symboling	Mức độ nguy hiểm của xe theo đánh giá bảo hiểm. Có 3 mức chính, trong đó mức 3 là mức nguy hiểm nhất, -2 là mức an toàn nhất
3	CarName	Tên của mẫu xe
4	fueltype	Dạng nhiên liệu mẫu xe sử dụng (xăng hoặc dầu diesel)
5	aspiration	Hệ thống thông khí xe sử dụng

6	doornumber	Số lượng cửa (hai (two) hay bốn cửa (four))
7	carbody	Các dạng thân xe, bao gồm: 1. Convertible 2. Sedan 3. Hatchback 4. Wagon 5. Hardtop
8	drivewheel	Dạng bánh lái: 1. fwd: Bánh lái trước 2. rwd: Bánh lái sau
9	enginelocation	Vị trí của động cơ (trước (front) hoặc sau (rear))
10	wheelbase	Khoảng cách giữa bánh trước và bánh sau
11	carlength	Chiều dài xe
12	carwidth	Chiều rộng xe
13	carheight	Chiều cao xe
14	curbweight	Cân nặng xe
15	enginetype	Loại động cơ
16	cylindernumber	Số xi lanh
17	enginesize	Kích cỡ động cơ
18	fuelsystem	Hệ thống nhiên liệu
19	boreratio	Tỉ lệ giữa khoảng chạy và đường kính xi lanh
20	stroke	Hành trình pít tông
21	compressionratio	Tỉ lệ nén
22	horsepower	Mã lực
23	peakrpm	Tốc độ vòng trên phút tối đa
24	citympg	Quãng đường chạy được thấp nhất với mỗi một gallon xăng

25	highwaympg	Quãng đường chạy được dài nhất với mỗi một gallon xăng
26	price	Giá xe. Đây là biến phụ thuộc vì bài toán đặt ra là liệu với các thông số kỹ thuật đề ra thì giá xe sẽ là bao nhiêu.

Bảng 2.1. Các thuộc tính của bộ dữ liệu “CarPrice_Accident.csv”

2.3. Phân tích dự báo với Python

2.3.1. Cài đặt thư viện

- Thư viện Pandas: nhằm nạp và thao tác trên tập dữ liệu.
- Thư viện Scikit-learn: xây dựng mô hình thuật toán hồi quy tuyến tính, kèm theo đó là các thuật toán xử lý dữ liệu và các phép đo, đánh giá mô hình.
- Thư viện Seaborn: trực quan hóa dữ liệu.
- Thư viện Statsmodels: xây dựng mô hình hồi quy tuyến tính kèm theo phân tích chuyên sâu về mô hình mà scikit-learn không cung cấp.

```
#Import Library

#Core libraries
import numpy as np
import pandas as pd

#Visualization libraries
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

#Preprocessing libraries
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split

#Models libraries
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
```

2.3.2. Khám phá dữ liệu

Đầu tiên ta sử dụng thư viện pandas để đọc vào file .csv chứa bộ dữ liệu.

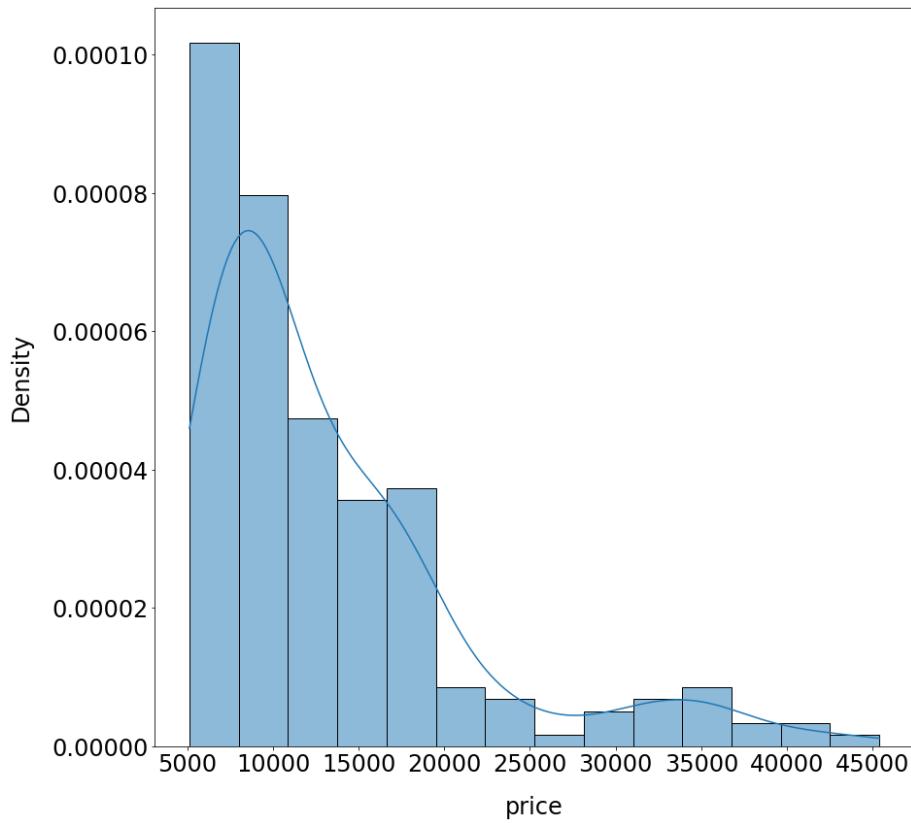
```
#Load data
cars = pd.read_csv("./data/CarPrice_Assignment.csv")
cars.head(10)
```

car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylin
0	1	3 alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	
1	2	3 alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	
2	3	1 alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	
3	4	2 audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	
4	5	2 audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	
5	6	2 audi fox	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	
6	7	1 audi 100ls	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7	2844	ohc	
7	8	1 audi 5000	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	55.7	2954	ohc	
8	9	1 audi 4000	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9	3086	ohc	
9	10	0 audi 5000s (diesel)	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9	52.0	3053	ohc	

Hình 2.1. Các dòng đầu tiên của bộ dữ liệu “CarPrice_Assignment.csv”

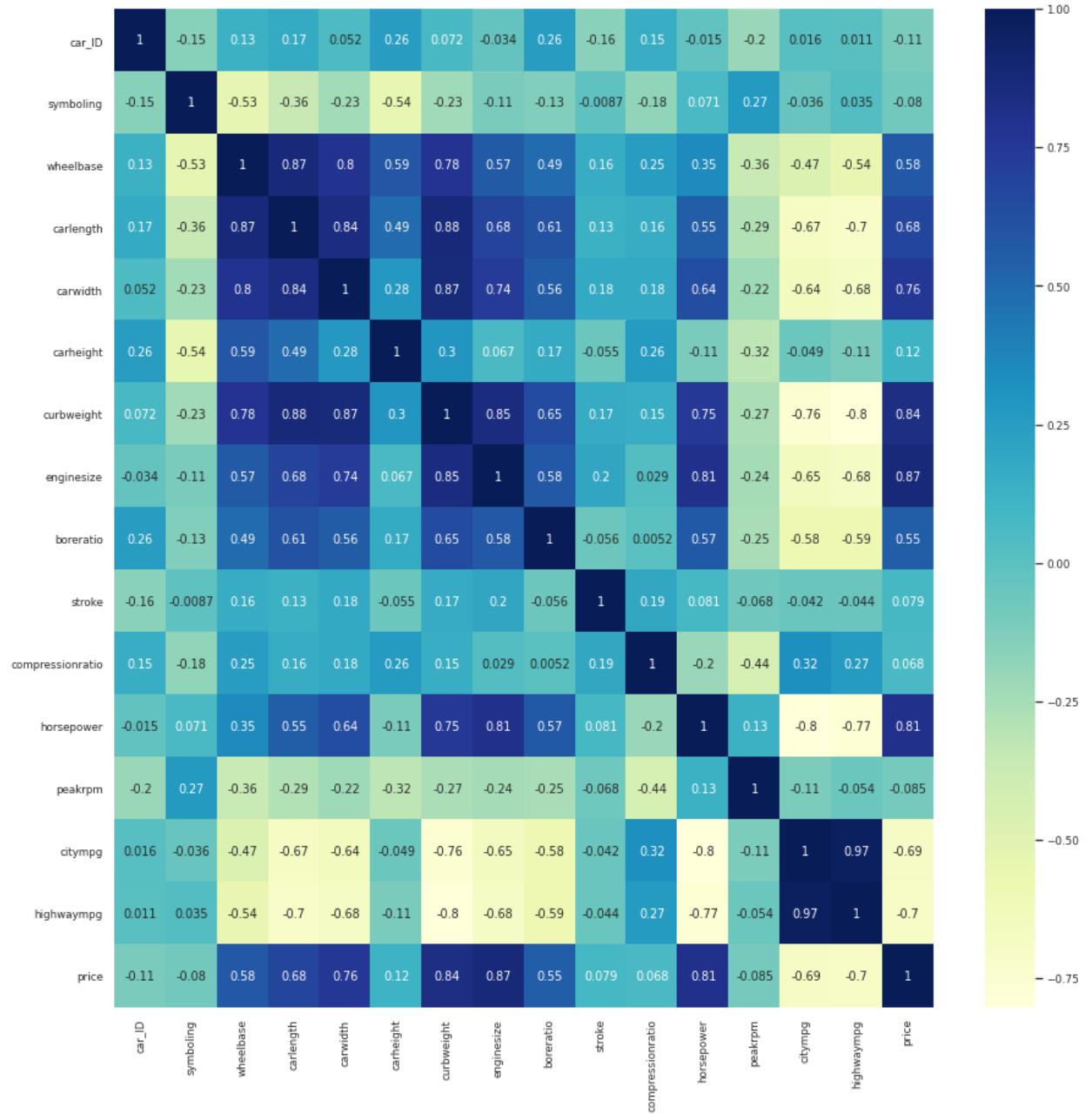
Tiếp đến sử dụng hàm info() để xem thông tin tóm tắt của dataset. Sau khi kiểm tra, ta nhận thấy dataset không có cột nào có giá trị rỗng.

Trong bộ dataset trên, biến “price” là biến phụ thuộc, và bài toán đặt ra là phải dự đoán biến này dựa vào danh sách các biến còn lại.



Hình 2.2. Biểu đồ tần suất kèm đường biểu diễn ước lượng mật độ của biến “price”

Từ Hình 2.2 ta đánh giá được mật độ xuất hiện của các xe giá rẻ nhiều hơn so với các xe sang trọng mắc tiền.



Hình 2.3. Biểu đồ nhiệt thể hiện tương quan giữa các biến dữ liệu số với biến “price”

Dựa vào Hình 2.3 có thể thấy được các biến có màu càng đậm càng thể hiện sự tương quan dương cao với biến “price” và các biến có màu càng nhạt càng thể hiện sự tương quan âm cao với biến price.

Ta tiến hành tách các biến mang kiểu dữ liệu phân lớp ra thành các cột dạng nhị phân (Dummies Column - Các cột chứa giá trị 0 hoặc 1) để chuyển hóa hoàn toàn các dữ liệu phân lớp thành dữ liệu số, thuận tiện hơn cho việc huấn luyện mô hình.

Đầu tiên, tách biến “price” ra khỏi dataset và lưu riêng và tiến hành thay thế các cột dữ liệu phân lớp thành dữ liệu số (chỉ mang 0 và 1).

```
#Extract categorical data
cars_lr = cars
price = cars['price']
cars_lr.drop(['price'], axis=1, inplace=True)
cars_lr.head()

# Define function to get dummies variables
cars_lr = pd.get_dummies(cars_lr, columns=['fueltype', 'aspiration', 'doornumber',
                                             'carbody', 'drivewheel', 'enginelocation',
                                             'enginetype', 'cylindernumber', 'fuelsystem'], drop_first=True)

      car_ID symboling          CarName wheelbase carlength \
0         1           3    alfa-romero giulia     88.6   168.8
1         2           3    alfa-romero stelvio     88.6   168.8
2         3           1  alfa-romero Quadrifoglio    94.5   171.2
3         4           2        audi 100 ls    99.8   176.6
4         5           2       audi 100ls    99.4   176.6

      carwidth carheight curbweight enginesize boreratio ... \
0       64.1     48.8      2548       130     3.47 ...
1       64.1     48.8      2548       130     3.47 ...
2       65.5     52.4      2823       152     2.68 ...
3       66.2     54.3      2337       109     3.19 ...
4       66.4     54.3      2824       136     3.19 ...

      cylindernumber_three cylindernumber_twelve cylindernumber_two \
0                  0                      0                      0
1                  0                      0                      0
2                  0                      0                      0
3                  0                      0                      0
4                  0                      0                      0

      fuelsystem_2bbl fuelsystem_4bbl fuelsystem_idi fuelsystem_mfi \
0                  0                  0                  0                  0
1                  0                  0                  0                  0
2                  0                  0                  0                  0
3                  0                  0                  0                  0
4                  0                  0                  0                  0

      fuelsystem_mpfi fuelsystem_spdi fuelsystem_spfi
0                  1                  0                  0
1                  1                  0                  0
2                  1                  0                  0
3                  1                  0                  0
4                  1                  0                  0

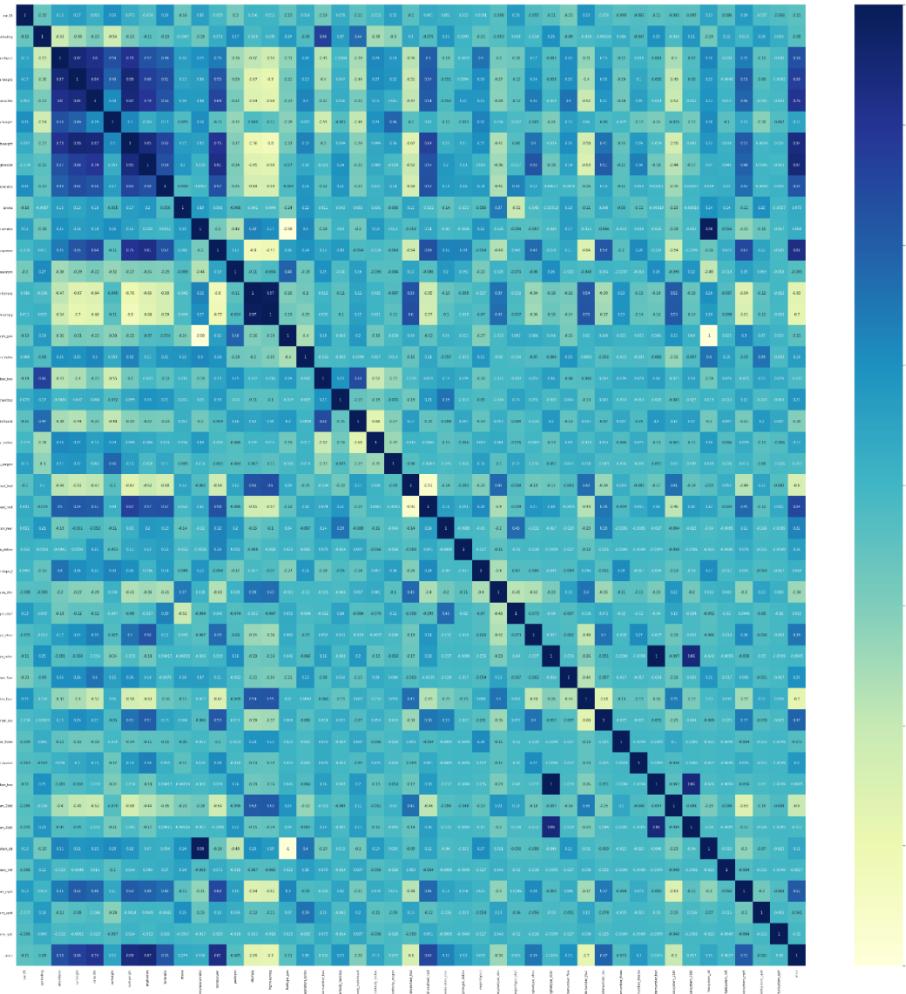
[5 rows x 45 columns]
```

Hình 2.4. Dataset sau khi chuyển đổi

Ở đây, lựa chọn “drop_first = True” thể hiện việc ta sẽ bỏ bớt một cột dummies trong các lớp được tách ra, nhằm giảm tính liên quan chéo lẫn nhau giữa các cột dummies. Điều này là quan trọng trong mô hình hồi quy tuyến tính đa biến, vì các biến không độc lập tuyến tính nhau sẽ gây nhiễu cho mô hình. Hơn nữa, việc bỏ đi một cột sẽ giúp dataset lúc huấn luyện ít đặc trưng hơn, tăng tốc độ huấn luyện cho mô hình.

Sau đó, ta ghép cột price vào dataset để tiếp tục vẽ biểu đồ tương quan nhằm đánh giá mức độ ảnh hưởng của các biến lên biến price nhằm chọn ra danh sách biến độc lập.

```
#Add "Price" back into dataset
cars_lr=pd.concat([cars_lr, price], axis=1)
cars_lr.head()
```



Hình 2.5. Biểu đồ nhiệt thể hiện tương quan giữa tất cả các biến

Dựa vào biểu đồ và thang màu, ta chọn được lần lượt các biến độc có tương quan dương và âm cao với biến “price”. Đó lần lượt là các biến “wheelbase”, “carlength”, “carwidth”, “curbweight”, “enginesize”, “boreratio”, “horsepower”, “rwd”, “mpfi”, “citympg”, “highwaympg”, và “four”.

2.3.3. Tiền xử lý dữ liệu

Ta tiến hành tách các biến liên quan liệt kê ở hình 18 ra khỏi dataset và sử dụng hàm MinMaxScaler() trong thư viện scikit-learn để chuẩn hóa các cột chứa dữ liệu số ban đầu.

```
#Extract highly related columns
related_columns = ['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize', 'boreratio',
                   'horsepower', 'drivewheel_rwd', 'fuelsystem_mpfi', 'citympg', 'highwaympg', 'cylindernumber_four', '']

#Extract numerical columns
num_vars = ['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize',
            'boreratio', 'horsepower', 'citympg', 'highwaympg', 'price']

#Scale numerical columns
scaler = MinMaxScaler()
df[num_vars] = scaler.fit_transform(df[num_vars])
```

Tiếp theo ta tách dataset đã chuẩn hóa thành tập train và test theo tỉ lệ 7:3, kèm theo đó là tách đặc trưng và nhãn của từng cả hai tập train, test ra.

```
#Split train and test set
df_train, df_test = train_test_split(df, train_size=0.7, test_size=0.3, random_state=0)
print(df_train.shape)
print(df_test.shape)

(143, 13)
(62, 13)

#Split X and Y
y_train = df_train.pop('price')
X_train = df_train
y_test = df_test.pop('price')
X_test = df_test
```

2.3.4. Xây dựng mô hình

Đầu tiên, sử dụng thuật toán Recursive Feature Elimination (RFE) cung cấp bởi thư viện scikit-learn để chọn ra 8 đặc trưng tốt nhất cho mô hình hồi quy tuyến tính.

```

#Create model
# Recursive Feature Selection using scikit-learn Linear Regression

# Choose best 8 variables using RFE
lm = LinearRegression()
rfe = RFE(lm, n_features_to_select=8)
rfe = rfe.fit(X_train, y_train)

print(list(zip(X_train.columns, rfe.support_, rfe.ranking_)))
[('wheelbase', False, 4), ('carlength', False, 2), ('carwidth', True, 1), ('curbweight',
print(X_train.columns[rfe.support_])

Index(['carwidth', 'curbweight', 'enginesize', 'horsepower', 'drivewheel_rwd',
       'citympg', 'highwaympg', 'cylindernumber_four'],
      dtype='object')

```

Kết quả này chính là các cột thể hiện tương quan tốt nhất với biến “price”. Ta sẽ xây dựng mô hình hồi quy tuyến tính dựa trên các cột này.

```

X_train_rfe = X_train[X_train.columns[rfe.support_]]
X_train_rfe.head()

   carwidth  curbweight  enginesize  horsepower  drivewheel_rwd  citympg  highwaympg  cylindernumber_four
40  0.183333    0.342901    0.184906    0.158333          0  0.388889    0.447368           1
60  0.516667    0.357642    0.230189    0.150000          0  0.361111    0.421053           1
56  0.450000    0.346005    0.033962    0.220833          1  0.111111    0.184211           0
101 0.516667    0.623351    0.452830    0.433333          0  0.111111    0.157895           0
86  0.425000    0.355702    0.230189    0.166667          0  0.333333    0.421053           1

```

Sử dụng hàm OLS() của statsmodels, tiến hành xây dựng mô hình hồi quy tuyến tính đa biến (MLR) trên tập X_train_rfe và nhận lại các thông tin đánh giá chi tiết của mô hình.

```

#OLS Linear Regression
X_train_rfe = sm.add_constant(X_train_rfe)
linear_model = sm.OLS(y_train, X_train_rfe).fit()

```

Ta sử dụng hàm summary() để xem kết quả của mô hình.

```
OLS Regression Results
Dep. Variable: price R-squared: 0.861
Model: OLS Adj. R-squared: 0.853
Method: Least Squares F-statistic: 104.2
Date: Thu, 19 Aug 2021 Prob (F-statistic): 1.07e-53
Time: 10:29:20 Log-Likelihood: 171.38
No. Observations: 143 AIC: -324.8
Df Residuals: 134 BIC: -298.1
Df Model: 8
Covariance Type: nonrobust

            coef  std err      t    P>|t| [0.025 0.975]
const      -0.0813  0.060   -1.358  0.177 -0.200  0.037
carwidth     0.1918  0.076    2.526  0.013  0.042  0.342
curbweight    0.0397  0.098    0.404  0.687 -0.154  0.234
enginesize    0.4487  0.104    4.302  0.000  0.242  0.655
horsepower    0.2607  0.108    2.418  0.017  0.047  0.474
drivewheel_rwd  0.0819  0.018    4.486  0.000  0.046  0.118
citympg      -0.1444  0.174   -0.830  0.408 -0.488  0.200
highwaympg     0.2504  0.168    1.491  0.138 -0.082  0.583
cylindernumber_four -0.0952  0.022   -4.346  0.000 -0.139 -0.052

Omnibus: 8.219 Durbin-Watson: 2.114
Prob(Omnibus): 0.016 Jarque-Bera (JB): 15.572
Skew: -0.117 Prob(JB): 0.000415
Kurtosis: 4.600 Cond. No. 61.0
```

Hình 2.6. Kết quả của mô hình OLS Regression

Ta thấy điểm R-Squared thể hiện độ chính xác của mô hình tương đối cao (0.857), cho nên mô hình có thể được coi là khá chính xác.

2.3.5. Đánh giá mô hình

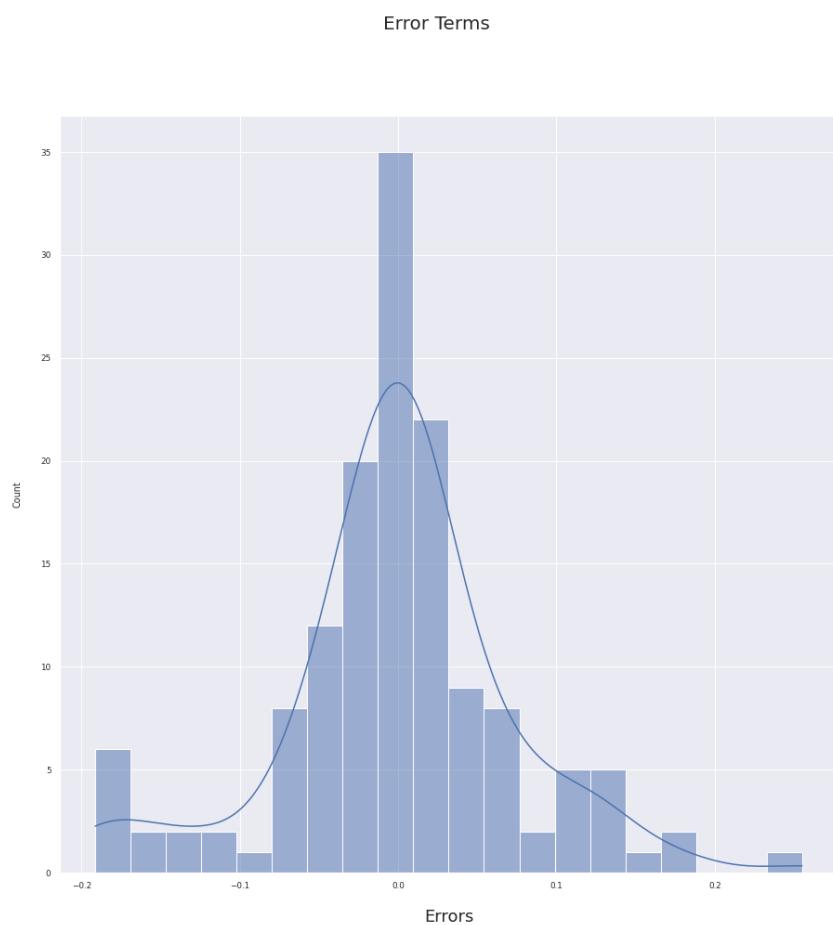
Sử dụng mô hình, ta dự đoán giá xe từ tập đặc trưng X_train_rfe, và trực quan hóa tần số lỗi của mô hình.

```

#Get predicted results
y_train_price = linear_model.predict(X_train_rfe)

#Analyze residual
fig = plt.figure(figsize=fig_size)
sns.histplot(x=(y_train - y_train_price), kde=True, bins=20)
fig.suptitle('Error Terms', fontsize=20) # Plot heading
plt.xlabel('Errors', fontsize=18)

```



Hình 2.7. Biểu đồ tần suất kèm đường biểu diễn ước lượng mật độ chênh lệch giá xe thực tế so với giá xe dự đoán

Có thể thấy mô hình dự đoán khá chính xác khi mức độ chênh lệch này tập trung chủ yếu xung quanh 0.0.

Để tiến hành đánh giá mô hình dựa trên tập test, trước tiên cần lọc các cột đã được chọn bởi RFE từ tập test.

```
# Extract relevant column from test set  
X_train_new = X_train_rfe.drop(['const'], axis=1)  
X_test_new = X_test[X_train_new.columns]  
print(X_test_new)
```

Sau đó tiến hành thêm hằng số vào trong tập test và sử dụng mô hình MLR đã huấn luyện dự đoán giá xe từ tập test.

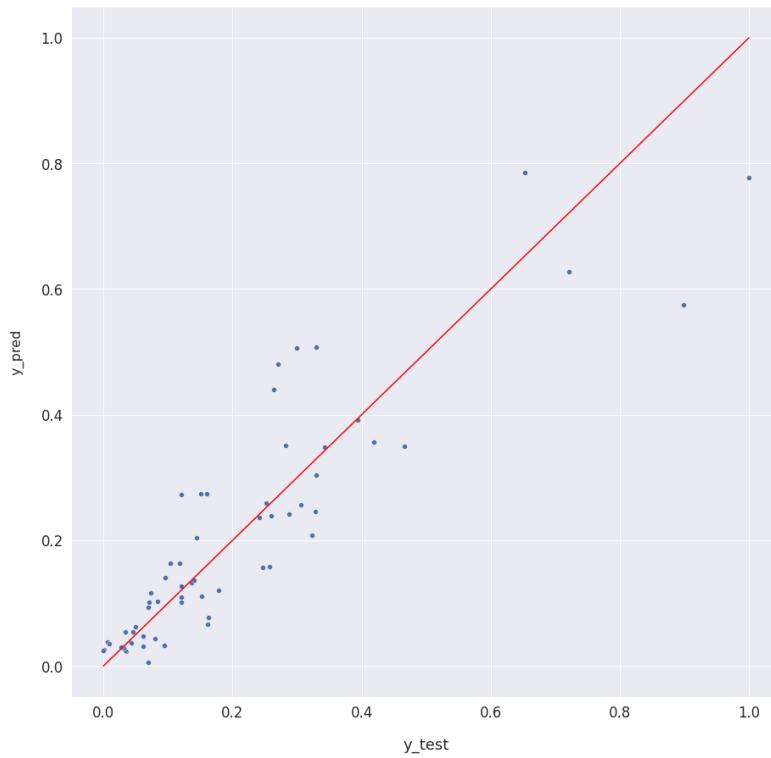
```
#Add constant to test set  
X_test_new = sm.add_constant(X_test_new)  
  
#Predict using training model  
y_pred = linear_model.predict(X_test_new)  
  
#Evaluate using R2 score  
print(r2_score(y_test, y_pred))
```

0.8006737842332712

Kết quả cho ra điểm R-Squared đạt 0.816, tức mô hình dự đoán khá chính xác, nhưng kém hơn so với tập train với R-Squared đạt 0.875.

Trực quan hóa mối liên hệ giữa kết quả dự đoán so với giá xe thực tế trong tập test để minh họa tính chính xác của mô hình bằng cách vẽ biểu đồ Residual của mô hình.

y_test vs y_pred



Hình 2.8. Biểu đồ thể hiện mối liên hệ giữa kết quả dự đoán so với giá xe thực tế

Ta thấy các điểm dự đoán tập trung rất gần đường chéo “ $y=x$ ” nghĩa là kết quả dự đoán rất gần với giá xe thực tế.

Cuối cùng, ta đánh giá nhân tố lạm phát phương sai (VIF) của từng biến độc lập trong mô hình.

```
#Get Variance Inflated Factor

vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by="VIF", ascending=False)
print(vif)

          Features      VIF
0           const  90.14
6       citympg  21.01
7    highwaympg  19.90
2     curbweight  10.10
4   horsepower   7.31
3    enginesize   6.89
1     carwidth   4.26
8 cylindernumber_four  2.05
5  drivewheel_rwd  2.00
```

Có thể thấy, biến “citympg”, “highwaympg” và “curbweight” có nhân tố lạm phát phương sai rất cao.

2.3.6. Tiết kiệm dự đoán

Sử dụng tập y_pred đã được tạo ra từ hàm predict của mô hình, ta lần lượt chuyển tập test và tập pred về lại dạng ban đầu trước khi chuẩn hóa để trích xuất giá tiền theo USD.

Đầu tiên, ta lần lượt ghép tập X_test với y_test và y_pred để tạo ra hai dataset là reverse_test và reverse_predict. Sau đó, sử dụng hàm inverse_transform() của MinMaxScaler, ta chuyển đổi ngược hai dataset trên về lại dạng trước chuẩn hóa.

```
reverse_test = X_test.assign(price=y_test.values)
reverse_pred = X_test.assign(price=y_pred.values)
reverse_pred.columns

Index(['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize',
       'boreratio', 'horsepower', 'drivewheel_rwd', 'fuelsystem_mpfi',
       'citympg', 'highwaympg', 'cylindernumber_four', 'price'],
      dtype='object')

reverse_test[num_vars] = scaler.inverse_transform(reverse_test[num_vars])
reverse_pred[num_vars] = scaler.inverse_transform(reverse_pred[num_vars])
```

Trích xuất cột “price” sau khi đảo ngược quá trình chuẩn hóa và in ra màn hình giá xe thực tế dự đoán. Có thể thấy, giá xe được dự đoán khá sát với giá xe thực tế.

```
print("Predicted:\n", y_pred_reverse.head())
print("Actual: \n", y_test_reverse.head())

Predicted:
52      6629.066928
181     22862.342798
5       15584.215046
18      6169.343963
188     9553.962891
Name: price, dtype: float64
Actual:
52      6795.0
181     15750.0
5       15250.0
18      5151.0
188     9995.0
Name: price, dtype: float64
```

2.4. Phân tích dự báo với R

2.4.1. Cài đặt thư viện

Sử dụng thư viện pacman để nạp các thư viện cần thiết khác một cách dễ dàng hơn. Các thư viện cần sử dụng là rio (nhập dữ liệu), ggplot2 (trực quan hóa dữ liệu), dplyr, reshape2, purrr, fastDummies (tiền xử lý dữ liệu), caret (mô hình hồi quy tuyến tính kết hợp với chọn lọc đặc trưng bằng đệ quy) và car (đánh giá hệ số lạm phát phương sai).

```
# Import pacman (package manager)
if (!require("pacman")) install.packages("pacman")

# Use pacman to load add-on packages as desired
pacman::p_load(pacman, rio, ggplot2, dplyr, reshape2,
  purrr, fastDummies, caret, car)
```

2.4.2. Khám phá dữ liệu

Sử dụng câu lệnh import từ thư viện rio để nạp dữ liệu từ tập tin .csv vào R. Sau đó sử dụng hàm head(), str() và summary() của R để in ra bảng tổng quan dataset.

```
# Import dataset
cars <- import("./datasets/CarPrice_Assignment.csv")
head(cars)

# Summarize dataset
str(cars)
summary(cars)
```

```

car_ID symboling CarName fueltypes aspiration doornumber
Min. : 1 Min. :-2.0000 Length:205 Length:205 Length:205 Length:205
1st Qu.: 52 1st Qu.: 0.0000 Class :character Class :character Class :character Class :character
Median :103 Median : 1.0000 Mode :character Mode :character Mode :character Mode :character
Mean :103 Mean : 0.8341
3rd Qu.:154 3rd Qu.: 2.0000
Max. :205 Max. : 3.0000

carbody drivewheel enginelocation wheelbase carlength carwidth carheight
Length:205 Length:205 Length:205 Min. : 86.60 Min. :141.1 Min. :60.30 Min. :47.80
Class :character Class :character Class :character 1st Qu.: 94.50 1st Qu.:166.3 1st Qu.:164.10 1st Qu.:52.00
Mode :character Mode :character Mode :character Median : 97.00 Median :173.2 Median :65.50 Median :54.10
Mean : 98.76 Mean :174.0 Mean :65.91 Mean :53.72
3rd Qu.:102.40 3rd Qu.:183.1 3rd Qu.:166.90 3rd Qu.:55.50
Max. :120.90 Max. :208.1 Max. :72.30 Max. :59.80

curbweight enginetype cylindernumber enginesize fuelsystem boreratio stroke
Min. :1488 Length:205 Length:205 Min. : 61.0 Length:205 Min. :2.54 Min. :2.070
1st Qu.:2145 Class :character Class :character 1st Qu.: 97.0 Class :character 1st Qu.:3.15 1st Qu.:3.110
Median :2414 Mode :character Mode :character Median :120.0 Mode :character Median :3.31 Median :3.290
Mean : 2556 Mean :126.9 Mean :3.33 Mean :3.255
3rd Qu.:2935 3rd Qu.:141.0 3rd Qu.:3.58 3rd Qu.:3.410
Max. :4066 Max. :326.0 Max. :3.94 Max. :4.170

compressionratio horsepower peakrpm citympg highwaympg price
Min. : 7.00 Min. : 48.0 Min. :4150 Min. :13.00 Min. :16.00 Min. : 5118
1st Qu.: 8.60 1st Qu.: 70.0 1st Qu.:4800 1st Qu.:19.00 1st Qu.:25.00 1st Qu.: 7788
Median : 9.00 Median : 95.0 Median :5200 Median :24.00 Median :30.00 Median :10295
Mean :10.14 Mean :104.1 Mean :5125 Mean :25.22 Mean :30.75 Mean :13277
3rd Qu.: 9.40 3rd Qu.:116.0 3rd Qu.:5500 3rd Qu.:30.00 3rd Qu.:34.00 3rd Qu.:16503
Max. :23.00 Max. :288.0 Max. :6600 Max. :49.00 Max. :54.00 Max. :45400

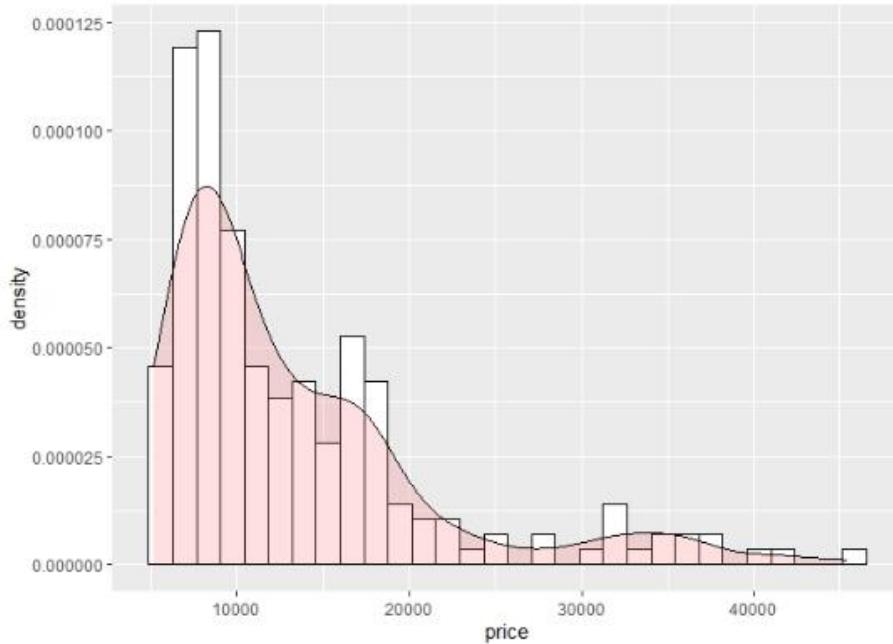
```

Hình 2.9. Các giá trị thống kê trong bộ dữ liệu “CarPrice_Assignment.csv”

Vẽ biểu đồ tần suất của cột “price” kết hợp đường biểu diễn ước lượng mật độ hạt nhân (Kernel Density Estimation - KDE).

```
# Visualize

ggplot(cars, aes(x=price)) +
  geom_histogram(aes(y=..density..),
                 colour="black", fill="white") +
  geom_density(alpha=.2, fill="#FF6666")
```



Hình 2.10. Biểu đồ tần suất kết hợp đường biểu diễn ước lượng giá xe bằng R

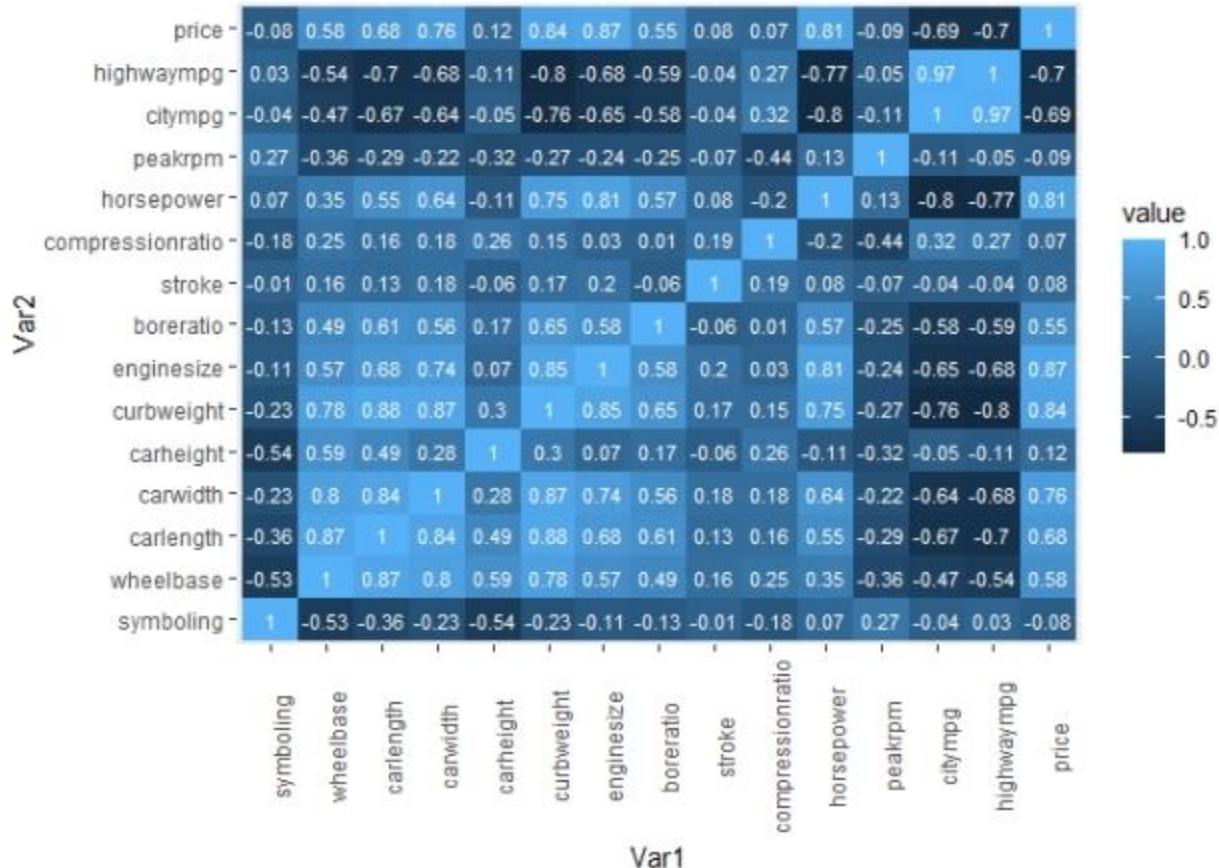
Tiến hành tách bỏ các cột “car_ID” và “CarName” ra khỏi dataset vì các cột này không liên quan quá nhiều đến giá thành của xe. Sau đó, ta tiến hành trích xuất các cột mang giá trị số vào một dataset mới thuận tiện cho việc trực quan hóa dữ liệu số học bằng biểu đồ nhiệt tương quan. Sử dụng hàm melt từ thư viện reshape, ta chuyển hóa được bảng giá trị tương quan có được từ hàm cor của R thành dataframe có thể được biểu diễn được bằng thư viện ggplot.

```
# Drop car_ID and names
cars <- select(cars, -c(car_ID, CarName))

# Extract Numeric value
cars_numeric <- cars[, purrr::map_lgl(cars, is.numeric)]|>

# Correlation Heat map
cormat <- round(cor(cars_numeric), 2)
melted_cormat <- melt(cormat)

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(Var2, Var1, label = value), color = "white", size = 4)
```



Hình 2.11. Biểu đồ nhiệt tương quan giữa các biến số so với nhau

Có thể nhận thấy các biến “carwidth”, “carlength”, “wheelbase”, “enginesize”, “curbweight” và “horsepower” có tương quan dương lớn so với biến “price”, trong khi đó các biến “citympg” và “highwaympg” có tương quan âm lớn so với biến “price”

Tiến hành chuyển hóa các cột chứa dữ liệu phân lớp sang dữ liệu số học để thuận lợi trong việc trực quan hóa dữ liệu và huấn luyện mô hình. Sau đó ta vẽ biểu đồ nhiệt tương quan của dataset mới tạo.

```

categorical_columns <- c("fueltype", "aspiration", "doornumber",
                        "carbody", "drivewheel", "enginelocation",
                        "enginetype", "cylindernumber", "fuelsystem")

cars <- dummy_cols(cars,
                     select_columns = categorical_columns,
                     remove_first_dummy = TRUE)

cars <- select (cars,-c(fueltype, aspiration, doornumber, carbody,
                       drivewheel, enginelocation, enginetype,
                       cylindernumber, fuelsystem))

cormat <- round(cor(cars), 2)
melted_cormat <- melt(cormat)

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  theme(axis.text.x=element_text(angle=90, hjust=1))

```

Tương tự như Python, có thể thấy các biến “carwidth”, “carlength”, “wheelbase”, “enginesize”, “curbweight” và “horsepower” có tương quan dương lớn so với biến “price”, trong khi đó các biến “drivewheel_rwd”, “fuelsystem_mpfi”, “cylindernumber_four”, “citympg” và “highwaympg” có tương quan âm cao so với biến “price”.

2.4.3. Tiền xử lý dữ liệu

Ta chọn được các cột có độ tương quan cao với cột price, bao gồm: “carwidth”, “carlength”, “wheelbase”, “enginesize”, “curbweight”, “horsepower”, “drivewheel_rwd”, “fuelsystem_mpfi”, “cylindernumber_four”, “citympg” và “highwaympg”. Từ đây, ta tiến hành chọn giữ lại các cột này và chuẩn hóa các cột chứa dữ liệu số ban đầu.

```

# Extracting Highly Related Columns
df <- select (cars, c(wheelbase, carlength, carwidth, curbweight,
                      enginesize, boreratio, horsepower, drivewheel_rwd,
                      fuelsystem_mpfi, citympg, highwaympg,
                      cylindernumber_four, price))

```

```

# Data standardization
df[c("wheelbase", "carlength", "carwidth",
     "curbweight", "enginesize", "boreratio",
     "horsepower", "citympg", "highwaympg", "price"
)] <- scale(df[c("wheelbase", "carlength", "carwidth", "curbweight",
                 "enginesize", "boreratio", "horsepower", "citympg",
                 "highwaympg", "price")])

```

Sau đó, sử dụng hàm `createDataPartition()` của thư viện caret, tiến hành chia tách dataframe mới thành tập train và tập set, kèm theo đó là tách tập kết quả và tập đặc trưng ra từ hai tập train và tập test.

```

# Train Test Split
set.seed(0)
trainIndex <- createDataPartition(df$price, p=0.7, list=FALSE)
df_train <- df[trainIndex,] #training data (75% of data)
df_test <- df[-trainIndex,] #testing data (25% of data)

drops <- c("price")
X_train <- df_train[, !(names(df_train) %in% drops)]
y_train <- df_train$price
X_test <- df_test[, !(names(df_test) %in% drops)]
y_test <- df_test$price

```

2.4.4. Xây dựng mô hình

Tiến hành xây dựng mô hình hồi quy tuyến tính kết hợp chọn lọc đặc trưng bằng đệ quy (RFE) sử dụng thư viện caret.

```

# Linear Regression
# Recursive Feature Elimination
subsets <- c(1:12)

ctrl <- rfeControl(functions = lmFuncs,
                     method = "repeatedcv",
                     repeats = 10,
                     number=10,
                     verbose = FALSE)

lmProfile <- rfe(X_train, y_train,
                  sizes = subsets,
                  rfeControl = ctrl,
                  metric="Rsquared")

predictors(lmProfile)
lmProfile$fit

```

```

> predictors(lmProfile)
[1] "highwaympg"      "cylindernumber_four" "citympg"          "enginesize"
[6] "horsepower"       "wheelbase"           "curbweight"        "drivewheel_rwd"
> lmProfile$fit

Call:
lm(formula = y ~ ., data = tmp)

Coefficients:
(Intercept)    highwaympg   cylindernumber_four   citympg      enginesize
               0.2011        0.4507        -0.4175       -0.3475        0.3504
drivewheel_rwd horsepower     wheelbase         curbweight
               0.2766        0.2354        0.1433        0.1165

```

2.4.5. Kiểm định mô hình và dự báo

Kiểm định độ chính xác của mô hình bằng cách quan sát các thông số kiểm tra như R-squared bên trong thuộc tính resample của mô hình lmProfile. Sau đó, Vẽ biểu đồ thể hiện độ thay đổi của chỉ số R-squared lần lượt qua từng lần huấn luyện với số lượng các biến khác nhau.

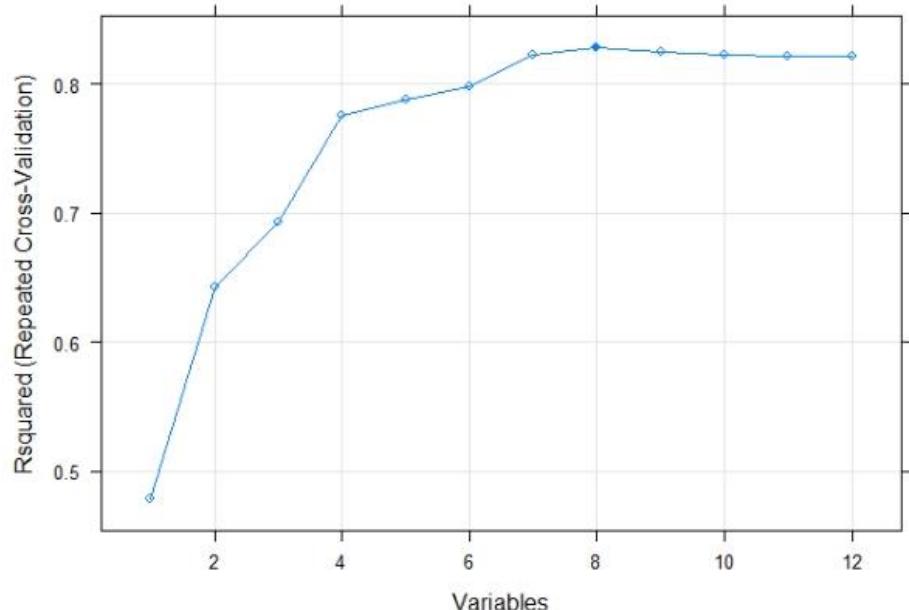
```

# Evaluation

head(lmProfile$resample)

plot(lmProfile, type=c("g", "o"))

```

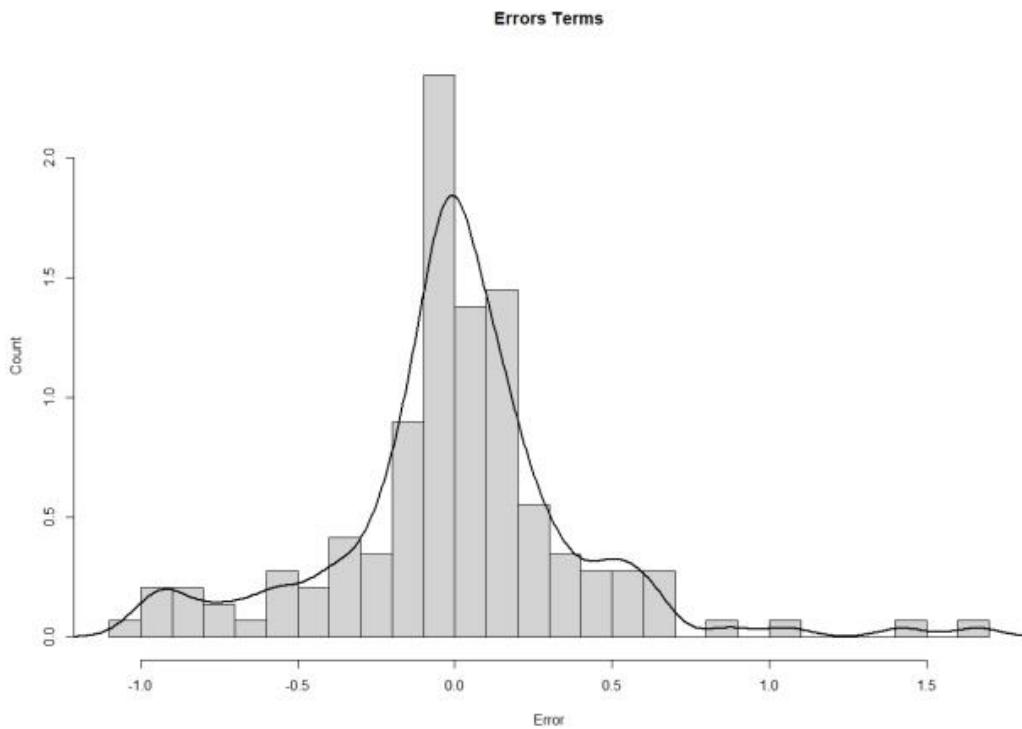


Hình 2.12. Biểu đồ đường thể hiện sự thay đổi của thông số R-Squared

Sử dụng hàm predict, ta thực hiện trích xuất kết quả dự đoán trên tập train và so sánh nó với kết quả thực tế của tập train. Ta trực quan hóa chênh lệch giữa dự đoán và thực tế với biểu đồ tần suất kết hợp KDE.

```
y_train_predicted <- predict(lmProfile, x_train)

hist(y_train - y_train_predicted, xlab="Error", breaks=20,
     main="Errors Terms", ylab="Count", prob=TRUE)
lines(density(y_train - y_train_predicted), lwd=2)
```



Hình 2.13. Biểu đồ thể hiện sự chênh lệch kết quả dự đoán và kết quả thực tế

Có thể thấy rằng, kết quả dự đoán chênh lệch kết quả thực tế rất ít (biểu thị ở việc các giá trị chênh lệch tập trung chủ yếu quanh giá trị 0.0).

Sử dụng mô hình và hàm predict, ta tiếp tục tiếp nhận kết quả dự đoán từ tập test. Ta sẽ tính toán chỉ số R2 của kết quả dự đoán này với kết quả thực tế của tập test.

```

# Prediction
new_predictions <- predict(lmProfile, X_test)

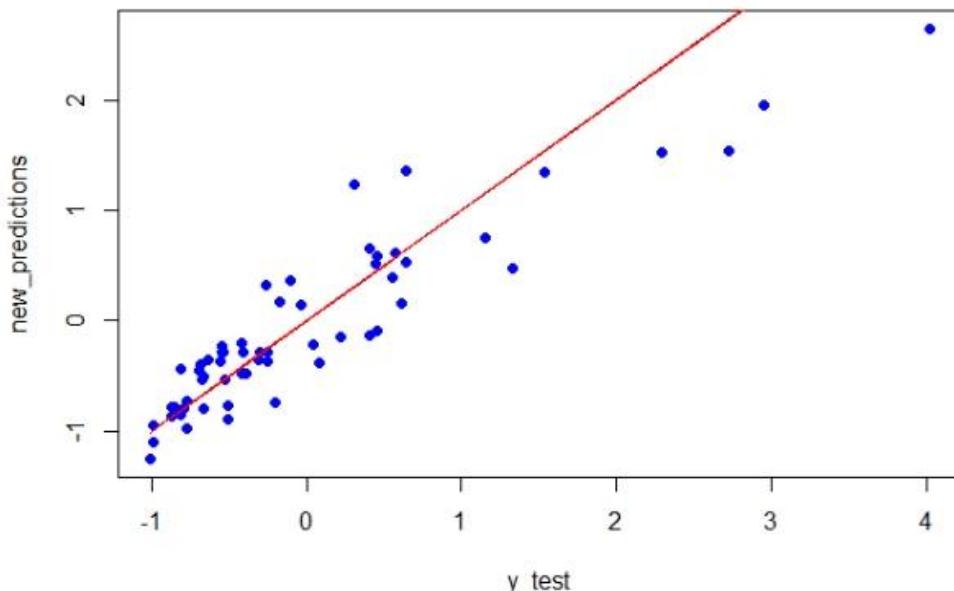
R2 <- function(x, y) cor(x, y) ^ 2
R2(new_predictions, y_test)

> R2(new_predictions, y_test)
[1] 0.8466528

```

Có thể thấy, mô hình cho chỉ số R-Squared khá cao (xấp xỉ 0.85).

Ta tiến hành phát họa các dự đoán của mô hình so với kết quả thực tế bằng biểu đồ phân tán.



Hình 2.14. Biểu đồ phân tán của kết quả dự đoán so với kết quả thực tế bằng R

Đường chéo đỏ thể hiện việc kết quả dự đoán khớp hoàn toàn với kết quả thực tế. Các điểm màu xanh tập trung gần đường màu đỏ chứng tỏ kết quả dự đoán rất gần với kết quả thực tế.

Cuối cùng, ta đánh giá chỉ số lạm phát phương sai của từng đặc trưng được huấn luyện bằng cách sử dụng hàm vif() thuộc thư viện car.

```
vif(lmProfile$fit)
highwaympg cylindernumber_four
27.580128      1.840885
citympg        enginesize
26.388440      6.648462
drivewheel_rwd horsepower
1.878107       7.351539
wheelbase       curbweight
4.136910       11.015423
```

Các đặc trưng “highwaympg”, “cylindernumber_four”, “citympg”, “curbweight” có chỉ số VIF rất cao chứng tỏ các biến này vẫn có sự phụ thuộc vào nhau.

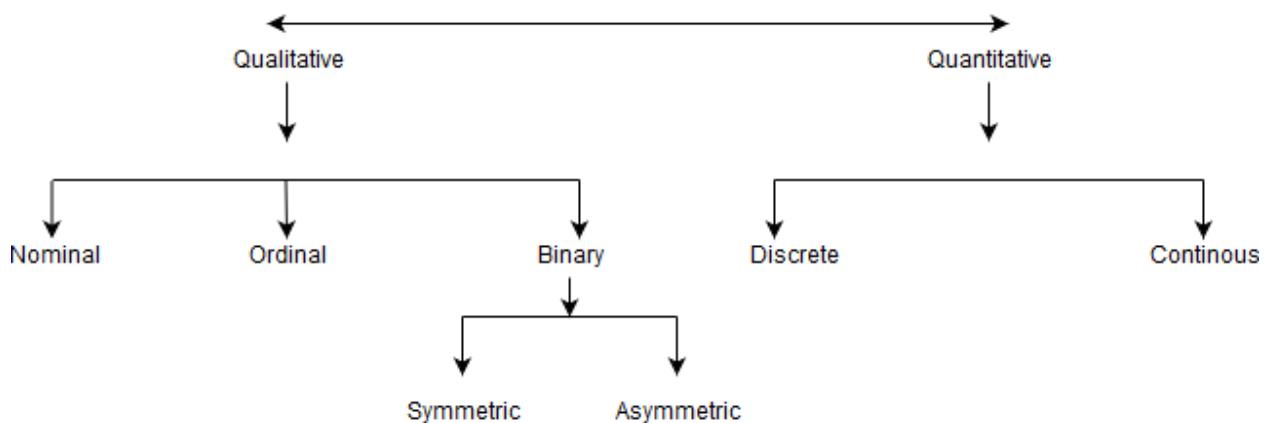
CHƯƠNG 3. MÔ HÌNH HỒI QUY LOGISTIC ĐƠN BIỀN

3.1. Cơ sở lý thuyết

3.1.1. Phân loại biến số

Biến (hay dữ liệu) thường có 2 dạng chính là định tính (*qualitative/categorical variable*), định lượng (*quantitative/numerical variable*), và biến nhị phân (*binary variable*).

- **Biến định tính** hay biến phân loại là biến phản ánh tính chất, hay loại hình, không có biểu hiện trực tiếp bằng con số. Có hai dạng Nominal (định danh) ví dụ như nghề nghiệp, tình trạng hôn nhân và Ordinal (thứ bậc) ví dụ như thứ hạng (nhất, nhì,...).
- **Biến định lượng** là biến biểu hiện trực tiếp bằng con số ví dụ tuổi, chiều cao, trọng lượng. Biến định lượng được chia làm 2 loại Discrete (biến định lượng rời rạc) như số học sinh một lớp và Continuous (biến định lượng liên tục), ví dụ như nhiệt độ.
- **Biến nhị phân** là loại biến chỉ có 2 giá trị (ví dụ như 0 và 1) và không trùng nhau của một đơn vị. Ví dụ như có hoặc không, sống hoặc chết, rời dịch vụ hoặc còn tiếp tục sử dụng dịch vụ. Biến nhị phân có 2 dạng: Symmetric (đối xứng) và Asymmetric (không đối xứng).



Hình 3.1. Phân loại các biến số

Với biến mục tiêu là biến định lượng liên tục thì chúng ta sử dụng phương pháp đã tìm hiểu qua chính là hồi quy tuyến tính như đã trình bày ở Chương 1 và Chương 2. Với

biến mục tiêu là biến định tính, hay biến nhị phân (hoặc biến rời rạc) thì phương pháp hồi quy chủ yếu, và thường là duy nhất chính là hồi quy logistic (*Logistic Regression*).

3.1.2. Khái niệm

Hồi quy logistic đơn biến là phương pháp nghiên cứu mối liên hệ giữa một biến độc lập (biến số, biến nhị phân hoặc biến phân loại) với biến phụ thuộc thường là biến nhị phân. Phương trình tổng quát của hồi quy logistic đơn biến có công thức như sau:

$$y = \beta_0 + \beta_1 X + \varepsilon$$

Trong đó:

- y là biến phụ thuộc, xác suất khả năng y xảy ra 0 hoặc 1.
- X là biến độc lập (biến tác động lên biến phụ thuộc).
- β_0 là giá trị ước lượng của y khi X bằng 0.
- β_1 dùng để xác định giá trị trung bình của y tăng hay giảm khi X tăng.
- ε là sai số thể hiện giá trị của các yếu tố khác không thể nghiên cứu hết và các yếu tố này vẫn tác động lên giá trị y .

Trong hồi quy logistic, biến phụ thuộc y chỉ có 2 trạng thái 1 (ví dụ tử vong) và 0 (ví dụ sống). Muốn đổi ra biến số liên tục người ta tính xác suất của 2 trạng thái này. Nếu gọi p là xác suất để một biến có xảy ra (ví dụ: tử vong), thì $1-p$ là xác suất để biến không xảy ra (ví dụ: sống). Phương trình hồi quy logistic lúc này có dạng:

$$\text{Log}\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X + \varepsilon$$

$$p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Tham số β_1 trong mô hình thể hiện giá trị trung bình của p tăng hay giảm khi X tăng. Khi $\beta_1 > 0$, xác suất p tăng khi X tăng. Khi $\beta_1 < 0$, xác suất p giảm khi X tăng. Nếu $\beta_1 = 0$, p không thay đổi khi X thay đổi, lúc này đường cong sẽ biến thành một đường thẳng nằm ngang. Độ dốc của đường cong tăng khi giá trị tuyệt đối của β_1 tăng. Tuy nhiên, không

giống như trong đồ thị đường thẳng trong hồi quy tuyến tính, β_1 không phải là độ dốc và do đó không thể được hiểu là sự thay đổi về giá trị trung bình p khi X thay đổi 1 đơn vị.

3.1.3. Chỉ số chênh (Odds)

Chỉ số chênh là tỷ số của 2 xác suất 0 và 1. Nếu p là xác suất bị bệnh thì $1-p$ là xác suất không bị bệnh. Chỉ số chênh được tính như sau:

$$Odds = \frac{p}{1-p} = \frac{\text{Xác suất sự kiện xảy ra}}{\text{Xác xuất sự kiện không xảy ra}}$$

- Nếu Odds > 1: khả năng bị bệnh cao hơn khả năng không bị bệnh.
- Nếu Odds < 1: khả năng bị bệnh thấp khả năng không bị bệnh .
- Nếu Odds = 1: khả năng bị bệnh bằng khả năng không bị bệnh.

Ta có ví dụ sau, một điều tra về hội chứng hô hấp sinh sản của heo nái (PRRS) liên quan đến việc tiêm ngừa vaccine phòng bệnh này cho 2000 heo nái được chọn ngẫu nhiên tại một trang trại chăn nuôi của thành phố H với kết quả qua bảng sau:

		PRRS	
		Có	Không
Chích ngừa bằng vaccine	Có	75	425
	Không	1200	300

Bảng 3.1. Kết quả tiêm ngừa phòng bệnh PRRS của heo nái

Gọi $p_1 = \frac{75}{500} = 0,15$ là tỉ lệ heo nái bị PRRS mặc dù có chích ngừa

$P_2 = \frac{1200}{1500} = 0,80$ là tỉ lệ heo nái bị PRRS do không có chích ngừa

Ta tính được chỉ số chênh như sau:

- Chỉ số chênh trong nhóm có chích ngừa là: $Odds_1 = \frac{p_1}{1-p_1} = 0,1767$

Nghĩa là heo nái có chích ngừa vẫn có nguy cơ bị PRRS với khoảng 0,1764 lần (17,64%) so với heo nái có chích ngừa mà không bị PRRS.

- Chỉ số chênh trong nhóm không chích ngừa là: $Odds_2 = \frac{p_2}{1 - p_2} = 4$

Nghĩa là heo nái không chích ngừa sẽ có nguy cơ bị PRRS gấp 4 lần (400%) so với heo nái không chích ngừa mà không bị PRRS.

3.1.4. Tỷ số chênh (Odds Ratio - OR)

Tỷ số chênh (OR) là tỉ số của 2 Odds. Odds ratio được dùng chủ yếu trong lĩnh vực thống kê mục đích để chuyển đổi giá trị xác suất trong khoảng (0; 1] thành những giá trị số thực nằm trong khoảng $(-\infty, +\infty)$, nghĩa là giá trị của hàm logit sẽ tiến đến $+\infty$ khi xác suất p tiến đến 1, và giá trị hàm logit sẽ tiến đến $-\infty$ khi xác suất p tiến đến 0.

Áp dụng vào ví dụ trên, ta có: $OR = \frac{Odds_1}{Odds_2} = 0,044175$.

Nghĩa là khả năng bị PRRS/không bị PRRS ở nhóm có chích ngừa thấp hơn 0,0441 (4,41%) so với khả năng bị PRRS/không bị PRRS ở nhóm không có chích ngừa.

3.1.5. Hàm Sigmoid

Hàm Sigmoid là một hàm số có dạng đường cong hình S (đường cong sigmoid). Trong hồi quy logistic, hàm sigmoid còn gọi là hàm logistic là hàm có công thức định nghĩa như sau:

$$y = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Căn cứ vào công thức, nếu giá trị của x chuyển đến dương vô cùng thì giá trị dự đoán của y sẽ trở thành 1 và nếu nó đi đến âm vô cùng thì giá trị dự đoán của y sẽ trở thành 0. Nếu kết quả của hàm sigmoid lớn hơn 0,5 thì chúng ta phân loại nhãn đó thành lớp 1 và nếu nó nhỏ hơn 0,5 thì chúng ta có thể phân loại nó thành lớp 0.

3.2. Giới thiệu dữ liệu

Bộ dữ liệu “heart.csv” chứa thông tin cá nhân và các chỉ số xét nghiệm lâm sàng của 303 bệnh nhân nhằm dự đoán họ có bị mắc bệnh tim mạch hay không. Dữ liệu này được trích từ bộ dữ liệu “Cleveland Heart Disease” tại UCI Machine Learning Repository.

Link dữ liệu: <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

Bộ dữ liệu gồm 14 thuộc tính sau:

STT	Tên thuộc tính	Mô tả thuộc tính
1	Age	Tuổi của bệnh nhân
2	Sex	Giới tính của bệnh nhân - 1: nam - 0: nữ
3	Chest-pain type (cp)	4 loại đau thắt ngực: - 0: đau thắt ngực điển hình - 1: đau thắt ngực không điển hình - 2: không đau thắt ngực - 3: đau thắt ngực không triệu chứng
4	Resting blood pressure (trestbps)	Số đo huyết áp khi nghỉ ngơi (đơn vị: mmHg)
5	Serum cholesterol (chol)	Lượng cholesterol trong máu (đơn vị: mg/dl)
6	Fasting blood sugar (fbs)	So sánh giá trị đường huyết lúc đói của bệnh nhân. Nếu nhỏ hơn 120 mg/dl: - 1: true - 0: false
7	Resting ECG (restecg)	Kết quả điện tâm đồ khi nghỉ ngơi: - 0: bình thường - 1: bất thường sóng ST-T - 2: phì đại tâm thắt trái
8	Max heart rate achieved (thalach)	Nhịp tim tối đa
9	Exercise induced angina (exang)	Đau thắt ngực do tập thể dục - 1: yes

		- 0: no
10	ST depression induced by exercise relative to rest (oldpeak)	Độ chênh xuống của ST khi tập thể dục
11	Peak exercise ST segment (slope)	Các dạng ST chênh: - 0: Chênh lên - 1: Bằng phẳng - 2: Chênh xuống
12	Number of major vessels (0–3) colored by flourosopy (ca)	Số lượng mạch chính có thể nhìn thấy bằng phương pháp soi huỳnh quang
13	Thal	Các mức độ bệnh thiếu máu huyết tán (Thalassemia): - 1: Bình thường - 2: Khuyết tật cố định - 3: Khuyết tật có thể khắc phục
14	Target	Kết quả mắc bệnh tim của bệnh nhân: - 1: có - 0: không

Bảng 3.2. Các thuộc tính của bộ dữ liệu “heart.csv”

3.3. Phân tích dự báo với Python

3.3.1. Cài đặt thư viện

- Thư viện Pandas, Numpy: đọc và xử lý dữ liệu.
- Thư viện Matplotlib, Seaborn: trực quan hóa dữ liệu.
- Thư viện Sklearn: chia dữ liệu thành 2 tập train test và xây dựng mô hình bằng thuật toán LogisticRegression.

```
#Import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
```

3.3.2. Khám phá dữ liệu

Sau khi load dữ liệu bằng lệnh `read_csv` ta sử dụng hàm `info` để xem thông tin của bộ dữ liệu. Ta thấy dữ liệu không có giá trị rỗng.

```
#Data information
df.info()

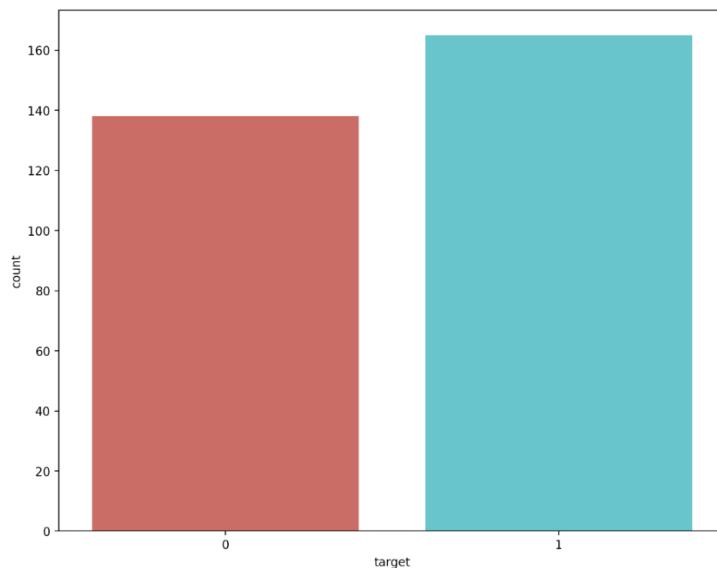
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   age         303 non-null    int64  
 1   sex          303 non-null    int64  
 2   cp            303 non-null    int64  
 3   trestbps     303 non-null    int64  
 4   chol          303 non-null    int64  
 5   fbs           303 non-null    int64  
 6   restecg       303 non-null    int64  
 7   thalach       303 non-null    int64  
 8   exang          303 non-null    int64  
 9   oldpeak        303 non-null    float64 
 10  slope          303 non-null    int64  
 11  ca             303 non-null    int64  
 12  thal           303 non-null    int64  
 13  target         303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Tiếp theo ta kiểm tra sự phân chia giữa các lớp trong biến phụ thuộc “target” bằng hàm `value_counts()` sau đó trực quan hóa bằng biểu đồ cột.

```
#Count values of the dependent variable  
df['target'].value_counts()
```

```
1    165  
0    138  
Name: target, dtype: int64
```

```
#Visualize values of "target"  
sns.countplot(x='target', data=df, palette='hls')  
plt.show()
```



Hình 3.2. Biểu đồ thể hiện sự phân chia hai lớp 0 và 1 của biến “target”

Ta thấy số người mắc bệnh tim mạch chiếm nhiều hơn số người không mắc bệnh và tỷ lệ giữa hai lớp khá đồng đều với nhau, không có sự chênh lệch quá nhiều.

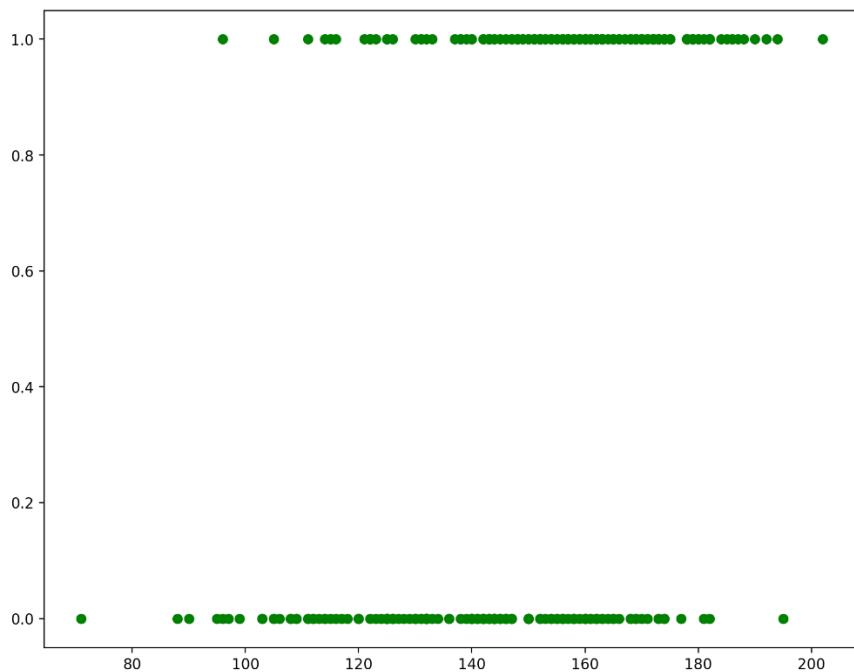
3.3.3. Xây dựng mô hình

Trong bộ dữ liệu này, biến “target” đóng vai trò là biến phụ thuộc và các biến còn lại là biến độc lập. Ở đây ta sẽ chọn ra ba biến độc lập lần lượt là “thalach”, “exang”, “thal” tương ứng với biến liên tục, nhị phân và thứ bậc để dự đoán bệnh nhân có mắc bệnh tim mạch hay không.

3.3.3.1. Biến liên tục

Đầu tiên ta dùng hàm mean() để xem giá trị trung bình của nhịp tim tối đa trong hai lớp 0 và 1. Kết quả cho thấy nhịp tim tối đa của những bệnh bị mắc bệnh tim mạch có giá trị trung bình là 159 bpm, còn những người không mắc bệnh thì nhịp tim trung bình là 139 bpm.

#Mean values of "target" with other attributes df.groupby('target').mean()													
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212



Hình 3.3. Biểu đồ phân tán thể hiện mối tương quan giữa 2 biến “thalach” và “target”

Từ Hình 3.4, những người mắc bệnh tim mạch có nhịp tim tối đa tập trung nhiều trong khoảng 140-190 bpm.

Để chuẩn bị cho việc xây dựng mô hình ta sẽ chia bộ dữ liệu thành 2 tập train và test với tỷ lệ 7:3. Sau đó dùng hàm LogisticRegression() với phương thức fit() để tạo và và huấn luyện mô hình trên tập train.

```

#Split train and test dataset
X_train, X_test, y_train, y_test = train_test_split(x_, df.target, test_size=0.3, random_state=0)

#Create model
model = LogisticRegression().fit(X_train, y_train)

#Get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)

Intercept: [-6.27109675]
Coef: [[0.04355535]]

```

Từ mô hình ta có được các hệ số intercept và coef với phương trình như sau:

$$\text{Heart disease} = -6.27109675 + 0.04355535 * \text{thalach}$$

Như vậy ta có tỉ lệ Odds(1):Odds(0) là $0.00292 : 0.00189$. Tỉ số chênh sê là 1.54. Tức nghĩa là mỗi khi biến thalach tăng 1 đơn vị thì nguy cơ bị mắc tim mạch sẽ tăng thêm 1.54 lần.

Sau đó ta dùng biểu đồ phân tán để trực quan hóa kết quả dự đoán so với kết quả thực tế trên tập train.



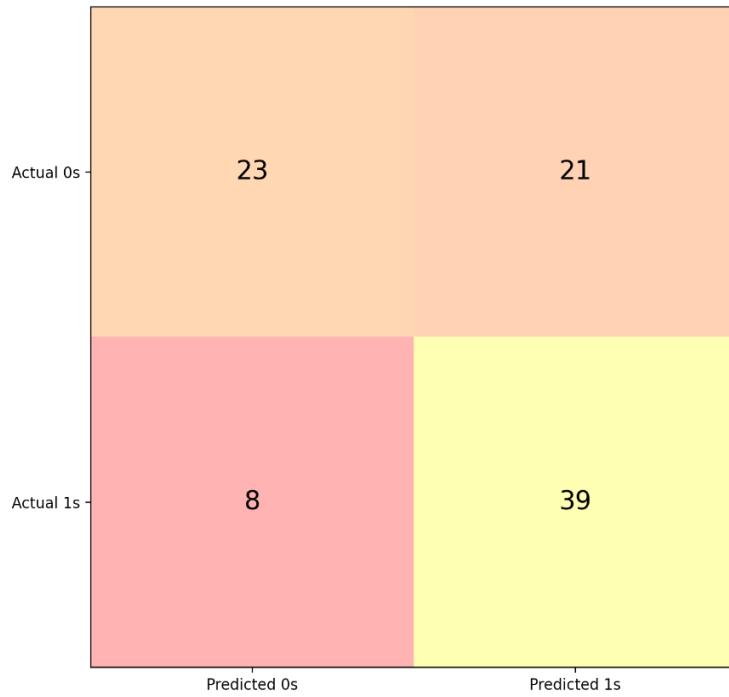
Hình 3.4. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “thalach”

Ta tiến hành dự đoán trên tập test bằng cách sử dụng hàm predict() với tham số đầu vào X_test, sau đó in kết quả dự đoán được ra màn hình.

```
#Predict on the test set  
y_pred = model.predict(X_test)  
  
#Actual value and predicted value  
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})  
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	1
228	0	1
201	0	0
52	1	1
245	0	1
175	0	0
168	0	1
223	0	0
217	0	0

Tiếp đến dùng ma trận hỗn loạn để kiểm tra sự phân chia giữa các kết quả dự đoán với nhau.



Hình 3.5. Ma trận hỗn loạn dựa trên biến “thalach”

Dựa vào Hình 3.6, trong tổng số 31 người không mắc bệnh thì mô hình dự đoán đúng 23 người. Tương tự với trường hợp còn lại, mô hình đã dự đoán đúng 39 người mắc bệnh tim mạch.

Để kiểm tra độ mức độ hiệu quả của mô hình ta sử dụng hàm `classification_report()` để in ra các độ đo đánh giá như Precision, Recall và F1-score. Dựa vào kết quả, ta thấy độ chính xác của mô hình là 0.68 ở mức tương đối thấp.

```
#Evaluation metrics
print(metrics.classification_report(y_test, y_pred))

precision    recall  f1-score   support
0            0.74    0.52     0.61      44
1            0.65    0.83     0.73      47

accuracy                           0.68      91
macro avg       0.70    0.68     0.67      91
weighted avg    0.69    0.68     0.67      91
```

Nếu chỉ dựa vào nhịp tim tối đa để dự đoán một người có mắc bệnh tim mạch hay không thì kết quả sẽ không có độ chính xác cao. Bởi lẽ điều này còn phụ thuộc vào nhiều

yếu tố khác như độ tuổi, hút thuốc, tiểu đường,... Ta cần kết hợp thêm những yếu tố đó để cải thiện kết quả của mô hình.

3.3.3.2. Biến nhị phân

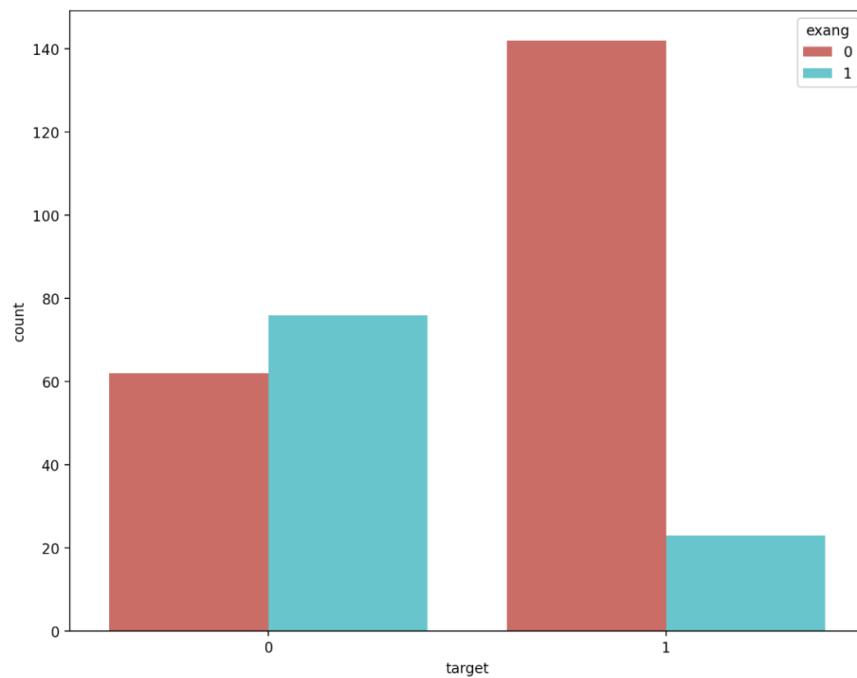
Biến nhị phân được dùng để xây dựng mô hình hồi quy logistic là biến “exang” cho biết bệnh nhân có bị đau thắt ngực khi tập thể dục hay không.

Ta dùng hàm value_counts() để đếm số giá trị giữa hai lớp 0 và 1 trong biến “exang”, sau đó trực quan hóa để kiểm tra sự phân chia trong biến “target”.

```
#Count values of "exang"  
df['exang'].value_counts()
```

```
0    204  
1     99  
Name: exang, dtype: int64
```

```
#Visualize values of "exang" and "target"  
sns.countplot(x='target', hue='exang', data=df, palette='hls')  
plt.show()
```



Hình 3.6. Biểu đồ透视 hiển thị sự phân chia của biến “exang” với biến “target”

Nhìn vào Hình 3.7 có thể thấy số người không bị đau thắt ngực khi tập thể dục lại chiếm tỉ lệ nhiều hơn ở trường hợp mắc bệnh tim mạch.

Tương tự như biến liên tục, ta cũng chia tập train test theo tỷ lệ 7:3 và xây dựng mô hình.

```
#Split train and test dataset
X_train, X_test, y_train, y_test = train_test_split(x_, df.target, test_size=0.3, random_state=0)

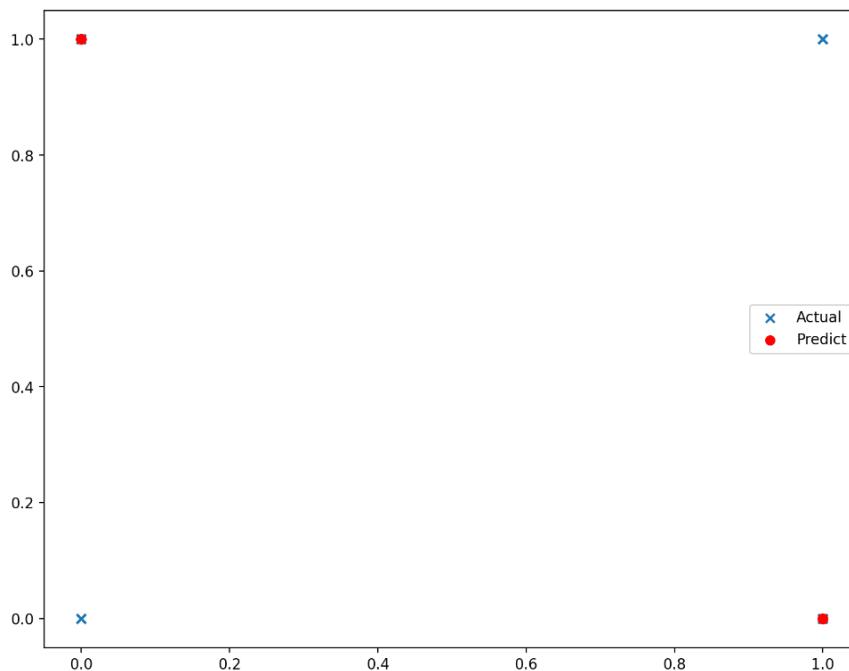
#Create model
model = LogisticRegression().fit(X_train, y_train)

#Get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)

Intercept: [0.87622878]
Coef: [[-1.86042193]]
```

Như vậy, ta có được phương trình sau:

$$\text{Heart disease} = 0.87622878 - 1.86042193 * \text{exang}$$



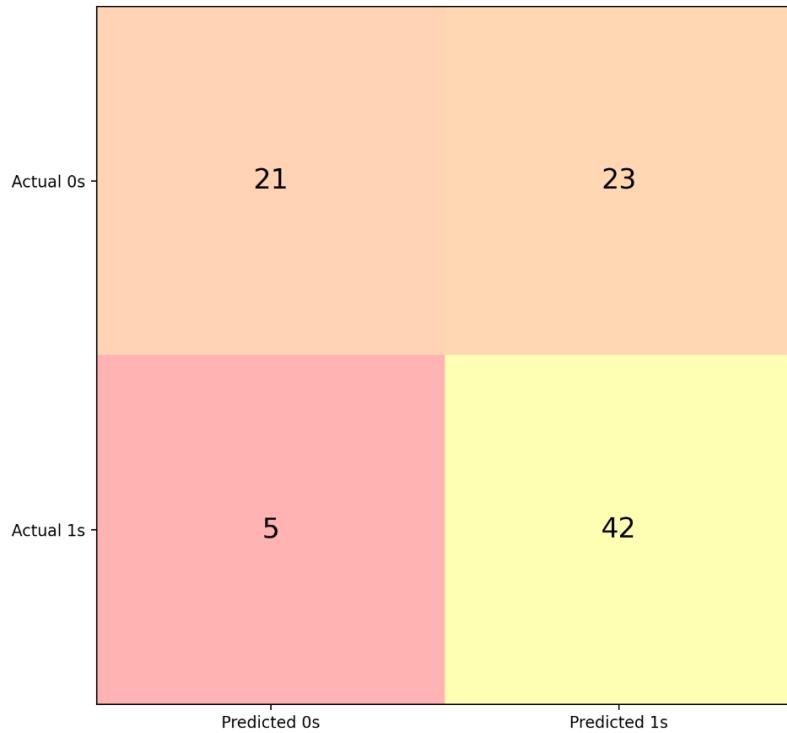
Hình 3.7. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “exang”

Dựa vào biểu đồ ta thấy ở trường hợp mắc bệnh tim mạch thì mô hình chỉ dự đoán đúng với những người không bị đau thắt ngực khi tập thể dục, đối với trường hợp mắc bệnh thì mô hình dự đoán đúng với những người bị đau thắt ngực.

Ta cũng sẽ dự đoán trên tập test và kiểm tra sự phân chia kết quả dự đoán bằng ma trận hỗn loạn, và cuối cùng đánh giá mô hình.

```
#Predict on the test set  
y_pred = model.predict(X_test)  
  
#Actual value and predicted value  
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})  
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	1
228	0	1
201	0	0
52	1	1
245	0	1
175	0	0
168	0	1
223	0	0
217	0	0



Hình 3.8. Ma trận hỗn loạn dựa trên biến “exang”

Ta thấy độ chính xác của mô hình là 0.69, điều này thể hiện mức độ chính xác của mô hình tương đối thấp.

```
#Evaluation metrics
print(metrics.classification_report(y_test, y_pred))

precision    recall  f1-score   support

      0       0.81      0.48      0.60       44
      1       0.65      0.89      0.75       47

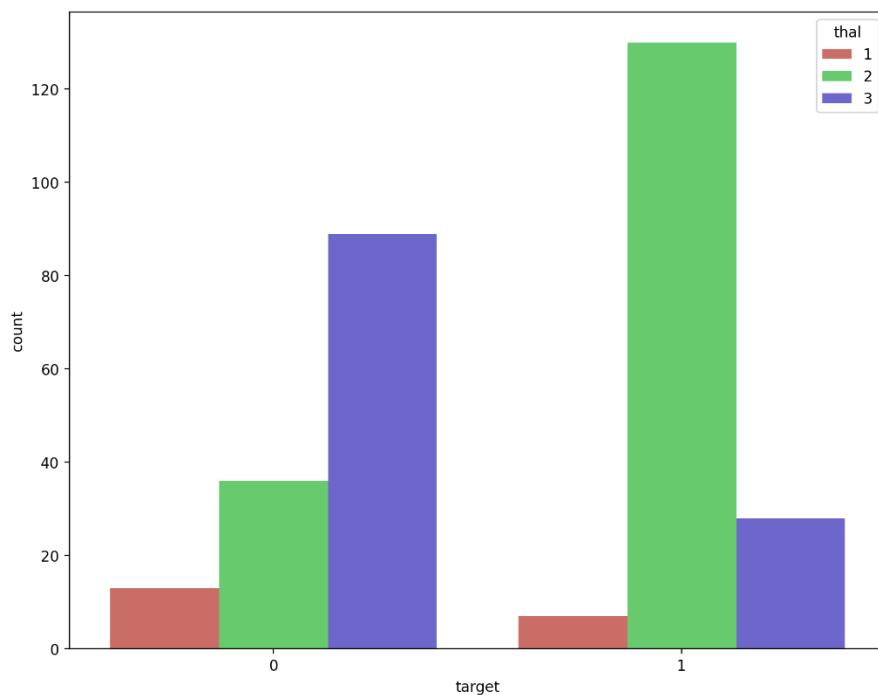
  accuracy                           0.69      91
  macro avg       0.73      0.69      0.68      91
weighted avg       0.72      0.69      0.68      91
```

3.3.3.3. Biến thứ bậc

Biến thứ bậc “thal” cho biết các mức độ của bệnh thiếu máu huyết tán. Đây là tình trạng các tế bào hồng cầu của người bệnh bị phá hủy nhanh hơn được tạo ra. Hồng cầu là tế bào mang oxy đi khắp cơ thể, nếu lượng hồng cầu thấp hơn mức bình thường, người

bệnh bị thiếu máu. Khi đó, máu không cung cấp đủ oxy cho các mô và cơ quan để hoạt động hiệu quả. Trong bộ dữ liệu này, biến “thal” được chia thành 3 loại:

- 1: bình thường (cơ thể khỏe mạnh, không mắc bệnh).
- 2: khuyết tật cố định (do di truyền).
- 3: khuyết tật có thể khắc phục (mắc bệnh trong quá trình sinh sống và có thể chữa kịp thời).



Hình 3.9. Biểu đồ thể hiện sự phân chia của biến “thal” với biến “target”

Từ Hình 3.10 ta thấy những người thuộc mức độ khuyết tật cố định mắc bệnh tim mạch nhiều nhất so với 2 mức độ còn lại.

Tương tự như các bước trên, ta cũng tiến hành chia tập train test theo tỷ lệ 7:3 và xây dựng mô hình.

```

#Split train and test dataset
X_train, X_test, y_train, y_test = train_test_split(x_, df.target, test_size=0.3, random_state=0)

#Create model
model = LogisticRegression().fit(X_train, y_train)

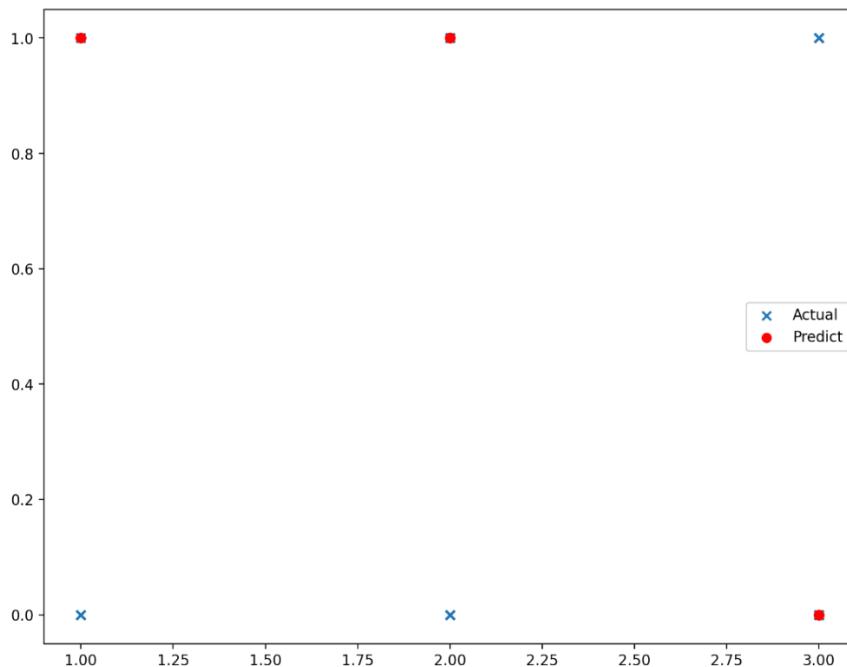
#Get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)

Intercept: [2.71054627]
Coef: [[-1.07582551]]

```

Với kết quả trên có được phương trình:

$$Heart\ disease = 2.71054627 - 1.07582551 * thal$$



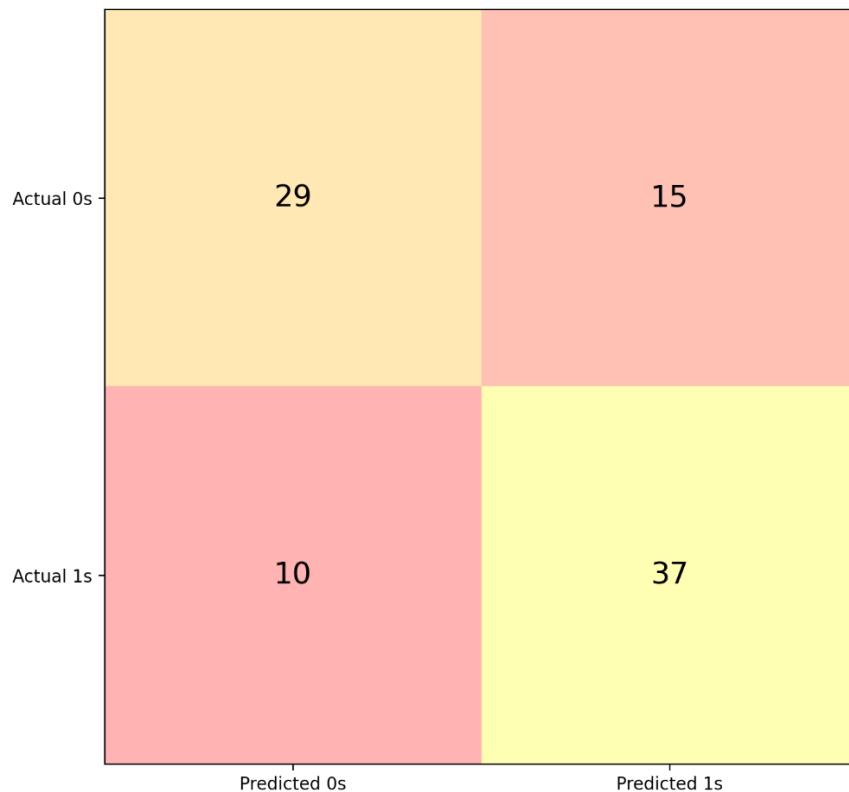
Hình 3.10. Biểu đồ thể hiện kết quả dự đoán trên tập train dựa trên biến “thal”

Ta thấy ở trường hợp mắc bệnh tim mạch thì mô hình dự đoán đúng với những bệnh nhân thuộc mức độ 1 và 2, còn mức độ 3 thuộc trường hợp không mắc bệnh.

Tiếp theo, ta sẽ dự đoán trên tập test và kiểm tra sự phân chia kết quả dự đoán bằng ma trận hỗn loạn, và cuối cùng cùng đánh giá mô hình.

```
#Actual value and predicted value
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	0
228	0	0
201	0	0
52	1	0
245	0	0
175	0	0
168	0	0
223	0	0
217	0	0



Hình 3.11. Ma trận hỗn loạn dựa trên biến “thal”

Ta thấy độ chính xác của mô hình là 0.73, với độ chính xác ở lớp 0 là 74% và lớp 1 là 71%, cho thấy mức độ chính xác của mô hình tương đối cao.

```
#Evaluation metrics
print(metrics.classification_report(y_test, y_pred))

precision    recall  f1-score   support

          0       0.74      0.66      0.70      44
          1       0.71      0.79      0.75      47

   accuracy                           0.73      91
  macro avg       0.73      0.72      0.72      91
weighted avg       0.73      0.73      0.72      91
```

3.4. Phân tích dự báo với R

3.4.1. Cài đặt thư viện

Để trực quan hóa dữ liệu trong R, ta sẽ sử dụng thư viện ggplot2. Ta có thể chọn thư viện này trong phần “Packages” của RStudio. Nếu thư viện này không có sẵn trong Packages, ta sử dụng lệnh:

```
#Import library
install.packages('ggplot2')
library(ggplot2)
```

3.4.2. Khám phá dữ liệu

Để import dữ liệu ta chọn vào “Import Dataset” đã có sẵn trong RStudio hoặc sử dụng lệnh read_csv tương tự như ở Python. Tiếp theo ta sử dụng hàm summary() để xem một số thông tin cơ bản về giá trị min, max của từng biến trong bộ dữ liệu.

```

> #Summary statistics
> summary(df)
      age          sex          cp          trestbps        chol
Min. :29.00  Min. :0.0000  Min. :0.000  Min. : 94.0  Min. :126.0
1st Qu.:47.50 1st Qu.:0.0000  1st Qu.:0.000  1st Qu.:120.0  1st Qu.:211.0
Median :55.00  Median :1.0000  Median :1.000  Median :130.0  Median :240.0
Mean   :54.37  Mean   :0.6832  Mean   :0.967  Mean   :131.6  Mean   :246.3
3rd Qu.:61.00  3rd Qu.:1.0000  3rd Qu.:2.000  3rd Qu.:140.0  3rd Qu.:274.5
Max.  :77.00   Max.  :1.0000  Max.  :3.000  Max.  :200.0  Max.  :564.0
      fbs          restecg        thalach       exang        oldpeak
Min. :0.0000  Min. :0.0000  Min. : 71.0  Min. :0.0000  Min. : 0.00
1st Qu.:0.0000 1st Qu.:0.0000  1st Qu.:133.5  1st Qu.:0.0000  1st Qu.:0.00
Median :0.0000  Median :1.0000  Median :153.0  Median :0.0000  Median :0.80
Mean   :0.1485  Mean   :0.5281  Mean   :149.6  Mean   :0.3267  Mean   :1.04
3rd Qu.:0.0000  3rd Qu.:1.0000  3rd Qu.:166.0  3rd Qu.:1.0000  3rd Qu.:1.60
Max.  :1.0000   Max.  :2.0000  Max.  :202.0  Max.  :1.0000  Max.  :6.20
      slope         ca          thal        target
Min. :0.000  Min. :0.0000  Min. : 1.00  Min. :0.0000
1st Qu.:1.000 1st Qu.:0.0000  1st Qu.:2.00  1st Qu.:0.0000
Median :1.000  Median :0.0000  Median :2.00  Median :1.0000
Mean   :1.399  Mean   :0.7294  Mean   :2.32  Mean   :0.5446
3rd Qu.:2.000  3rd Qu.:1.0000  3rd Qu.:3.00  3rd Qu.:1.0000
Max.  :2.000   Max.  :4.0000  Max.  :3.00  Max.  :1.0000

```

Hình 3.12. Các giá trị thống kê của từng biến trong bộ dữ liệu “heart.csv”

Sau khi kiểm tra, ta nhận thấy bộ dữ liệu không có giá trị rỗng.

```

> #Check null values
> sum(is.na(df))
[1] 0

```

3.4.3. Xây dựng mô hình

3.4.3.1. Biến liên tục

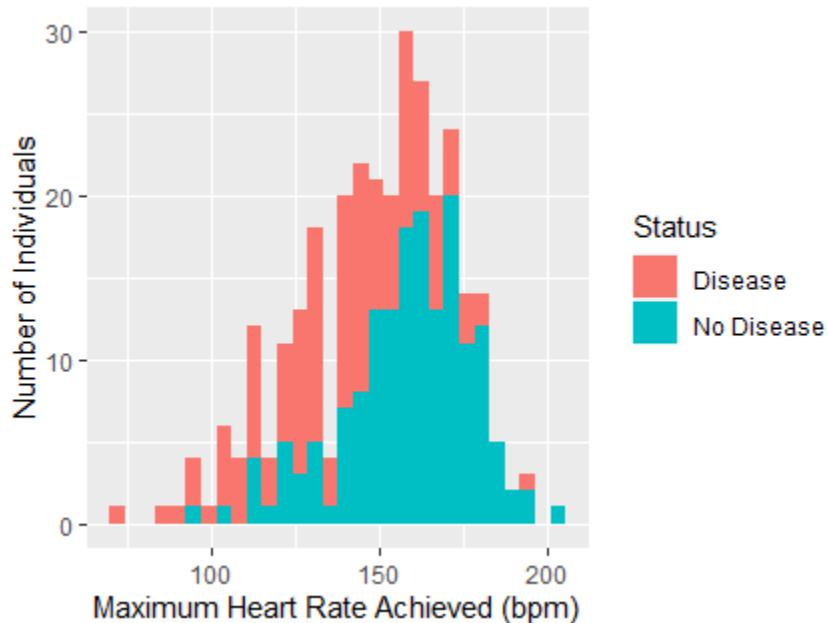
Trong bộ dữ liệu này, biến “target” là biến phụ thuộc, các biến còn lại là biến độc lập. Ta sẽ chọn biến “thalach” để dự đoán khả năng mắc bệnh tim mạch của các bệnh nhân.

```

#Visualize "target" vs "thalach"
ggplot(df, aes(x = thalach, fill = factor(target))) +
  geom_histogram() +
  labs(x = "Maximum Heart Rate Achieved (bpm)",
       y = "Number of Individuals") +
  scale_fill_discrete(name = "Status",
                      labels = c("Disease", "No Disease"))

```

Để vẽ biểu đồ thể hiện số người bị mắc bệnh tim dựa trên nhịp tim tối đa ta sẽ sử dụng thư viện ggplot2 và dùng fill = factor(target) để thêm màu vào các lớp dựa trên biến “target”.



Hình 3.13. Số người mắc bệnh tim mạch dựa trên nhịp tim tối đa

Ta sẽ chia tập dữ liệu thành 2 tập train và test với tỉ lệ 7:3. Tập train sẽ dùng để xây dựng và huấn luyện mô hình, tập test sẽ dùng để kiểm định lại mô hình.

```
#Split data
dt = sort(sample(nrow(df), nrow(df)*.70))
train<-df[dt,]
test<-df[-dt,]
```

Trong R, để xây dựng mô hình hồi quy logistic ta sử dụng hàm glm() với family = binomial. Phần tử family dùng để mô tả phân phối của sai số và hàm liên kết được sử dụng trong model.

```
#Create model
model<-glm(target ~ thalach, family = 'binomial', data = train)
summary(model)
```

```

Call:
glm(formula = target ~ thalach, family = "binomial", data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.0484 -1.0709  0.6637  0.9535  2.0770 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -5.914010  1.151545 -5.136 2.81e-07 ***
thalach      0.040416  0.007603  5.316 1.06e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 292.97 on 211 degrees of freedom
Residual deviance: 257.84 on 210 degrees of freedom
AIC: 261.84

Number of Fisher Scoring iterations: 4

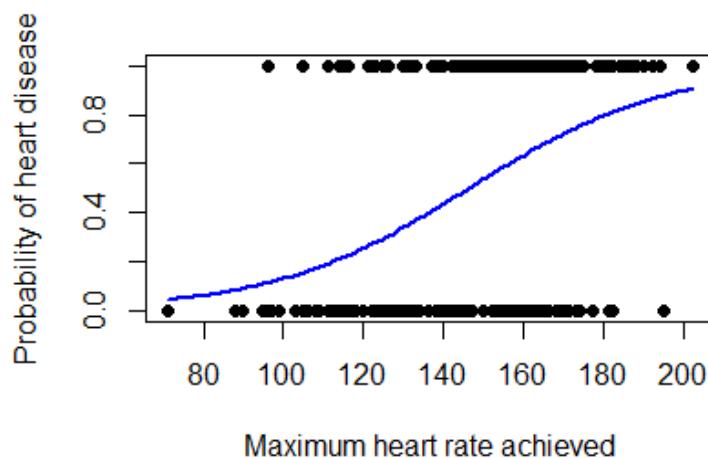
```

Hình 3.14. Các thông số của mô hình với biến độc lập “thalach”

Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = -5.914010 + 0.040416 * \text{thalach}$$

Tiếp theo ta dùng hàm plot() và curve() để vẽ đồ thị hồi quy logistic. Từ đồ thị ta thấy kết quả dự đoán trên tập train vẫn còn nhiều điểm bị bỏ lỡ.



Hình 3.15. Đồ thị hồi quy logistic với biến độc lập “thalach”

Để kiểm tra lại mô hình ta sẽ sử dụng tập dữ liệu test và hàm predict() để dự đoán kết quả. Ngoài ra, ta sẽ đặt ngưỡng giá trị lớn hơn 0.5 để cải thiện dự đoán của mô hình.

```
#Predict on test set  
pred_values = predict(model, newdata=test, type='response')  
pred_classes <- ifelse(pred_values > 0.5, "1", "0")
```

Ta có ma trận hỗn loạn sau:

```
> #Confusion matrix  
> table(test$target, pred_values > 0.5)  
  
 FALSE TRUE  
 0    24   15  
 1    12   40
```

Trong 39 người được dự đoán không mắc bệnh tim thì kết quả có 24 người thật sự không mắc bệnh. Đối với trường hợp còn lại có 40 người thật sự mắc bên trên tổng số 52 người được dự đoán là mắc bệnh.

Độ chính xác của mô hình đạt 70%, mô hình có độ chính xác tương đối cao.

```
> mean(pred_classes == test$target)  
[1] 0.7032967
```

3.4.3.2. Biến nhị phân

Đối với biến nhị phân ta cũng làm tương tự như biến liên tục. Sau khi tạo model ta dùng hàm summary() để xem kết quả của mô hình.

```

Call:
glm(formula = target ~ exang, family = "binomial", data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5390 -0.6776  0.8547  0.8547  1.7800 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.8190    0.1875   4.369 1.25e-05 ***
exang       -2.1736    0.3373  -6.444 1.16e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 293.72 on 211 degrees of freedom
Residual deviance: 244.20 on 210 degrees of freedom
AIC: 248.2

Number of Fisher Scoring iterations: 4

```

Hình 3.16. Các thông số của mô hình với biến độc lập “exang”

Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = 0.8190 - 2.1736 * \text{exang}$$

Ta cũng dùng hàm predict() để dự đoán trên tập test.

```
#Predict on test set
pred_values = predict(model, newdata=test, type='response')
pred_classes <- ifelse(pred_values > 0.5, "1", "0")
```

Ta có ma trận hỗn loạn với kết quả dự đoán đúng 14 người không mắc bệnh và 49 người mắc bệnh.

```
> #Confusion matrix
> table(test$target, pred_values > 0.5)

    FALSE TRUE
0      14   21
1       7   49
```

Độ chính xác của mô hình là 69%, một kết quả không quá cao.

```
> #Model accuracy
> mean(pred_classes == test$target)
[1] 0.6923077
> |
```

3.4.3.3. Biến thứ bậc

Ta cũng chia dữ liệu thành hai tập train test với tỉ lệ 7:3 và sử dụng hàm glm() để xây dựng mô hình hồi quy logistic.

```
#Split data
dt = sort(sample(nrow(df), nrow(df)*.70))
train<-df[dt,]
test<-df[-dt,]

#Create model
model<-glm(target ~ thal, family = 'binomial', data = train)
summary(model)

Call:
glm(formula = target ~ thal, family = "binomial", data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.9821 -0.9409  0.9302  0.9302   1.4340 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 3.0126    0.6392   4.713 2.44e-06 ***
thal        -1.1994    0.2655  -4.518 6.25e-06 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 291.17 on 211 degrees of freedom
Residual deviance: 268.15 on 210 degrees of freedom
AIC: 272.15

Number of Fisher Scoring iterations: 4
```

Hình 3.17. Các thông số của mô hình với biến độc lập “thal”

Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = 3.0126 - 1.1994 * \text{thal}$$

Tiếp theo ta dùng hàm predict() để dự đoán kết quả trên tập test và dùng ma trận hỗn loạn để kiểm tra sự phân chia của các kết quả.

```
#Predict on test set  
pred_values = predict(model, newdata=test, type='response')  
pred_classes <- ifelse(pred_values > 0.5, "1", "0")
```

Từ ma trận hỗn loạn thấy mô hình dự đoán đúng 31 người không mắc bệnh tim và 39 người mắc bệnh trên tổng số 91 người.

```
> #Confusion matrix  
> table(test$target, pred_values > 0.5)  
  
      FALSE TRUE  
0       31   13  
1        8   39  
> |
```

Độ chính xác của mô hình đạt 76% tương đối cao. Điều này cho thấy biến độc lập “thal” là một biến có độ ảnh hưởng đến việc dự đoán bệnh nhân mắc bệnh tim mạch hay không.

CHƯƠNG 4. MÔ HÌNH HỒI QUY LOGISTIC ĐA BIẾN

4.1. Cơ sở lý thuyết

Tương tự như hồi quy logistic đơn biến, hồi quy logistic đa biến chỉ khác ở chỗ khi bộ dữ liệu có 1 biến phụ thuộc và yêu cầu phân tích trên nhiều biến độc lập khác nhau. Trên thực tế, các yêu cầu phân tích hồi quy logistic chủ yếu là đa biến do các biến phụ thuộc được ghi nhận thông thường sẽ chịu ảnh hưởng của rất nhiều yếu tố khác nhau nên việc phân tích chỉ theo một biến độc lập duy nhất sẽ làm kém đi tính khách quan của kết quả thu được.

Kết quả trong phân tích hồi quy logistic thường được mã hóa là 0 hoặc 1. Nếu chúng ta định nghĩa p là xác suất mà kết quả là 1, thì mô hình hồi quy logistic đa biến có thể được viết như sau:

$$p = \frac{e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n}}{1 + e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n}}$$

Với:

- p là xác xuất để kết quả trả về 1.
- X_1 đến X_n là giá trị các biến phụ thuộc.
- β_0 đến β_n là các hệ số hồi quy.

4.2. Phân tích dự báo với Python

4.2.1. Cài đặt thư viện

Ngoài các thư viện cơ bản như Pandas, Numpy... ta sẽ cài đặt thêm thư việ Sklearn để sử dụng thuật toán LogisticRegression trong việc xây dựng mô hình hồi quy logistic.

```

#Import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

```

4.2.2. Xây dựng mô hình

Ta vẫn sẽ sử dụng lại bộ dữ liệu “heart.csv” đã giới thiệu ở Chương 3. trong bộ dữ liệu này, biến “target” là biến phụ thuộc, các biến còn lại là biến độc lập. Như yêu cầu đặt ra, chúng ta sẽ chạy lần lượt trên các biến độc lập là liên tục, nhị phân và cả thứ bậc. Vì vậy chúng ta sẽ tiến hành chia bộ dữ liệu 3 lần.

4.2.2.1. Biến liên tục

Ở đây, chúng ta sẽ chọn biến độc lập liên tục để chạy là biến “thalach” và “chol”. Nói cách khác, chúng ta sẽ chẩn đoán một người có bị bệnh tim dựa lượng cholesterol và nhịp tim tối đa của họ. Trước hết chúng ta import thư viện `train_test_split` từ `sklearn` và tiến hành chia bộ dữ liệu, tỉ lệ `train:test` ở đây sẽ là 7:3.

```

#Split train and test set
X_train, X_test, y_train, y_test = train_test_split(df[['thalach', 'chol']], df.target, test_size=0.3, random_state = 0)

```

Tại đây, chúng ta xây dựng mô hình để dự đoán từ bộ train và in ra các hệ số quan trọng như `coef` và `intercept`.

```

#create model
model = LogisticRegression().fit(X_train, y_train)

```

```

#get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)

```

```

Intercept: [-5.77981461]
Coef: [[ 0.04374532 -0.00207879]]

```

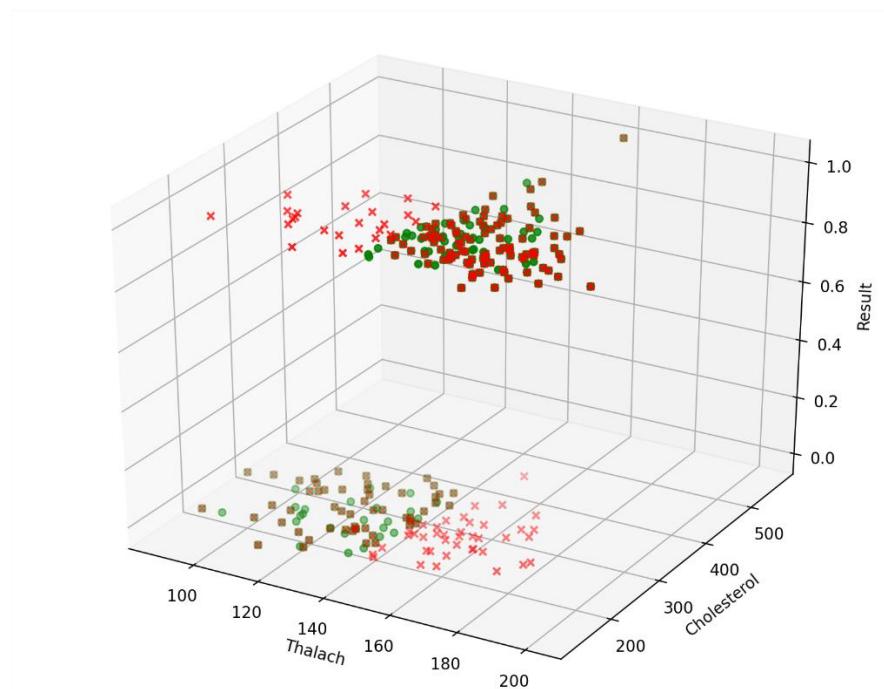
Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = -5.77981461 + 0.043742 * \text{thalach} - 0.002078 * \text{chol}$$

Ta cũng tính ra được chỉ số chênh giữa 1:0 là 0.00473:0.00412, như vậy ta được tỉ số chênh là 1.52. Nghĩa là khi ta tăng nhịp tim tối đa và lượng cholesterol trong máu lên 1 đơn vị mỗi biến thì xác suất bị bệnh tim mạch sẽ tăng lên thêm 1.52 lần.

Sau đó ta trực quan hóa dữ liệu để dự đoán những người bị mắc bệnh tim mạch như

Hình 4.1.



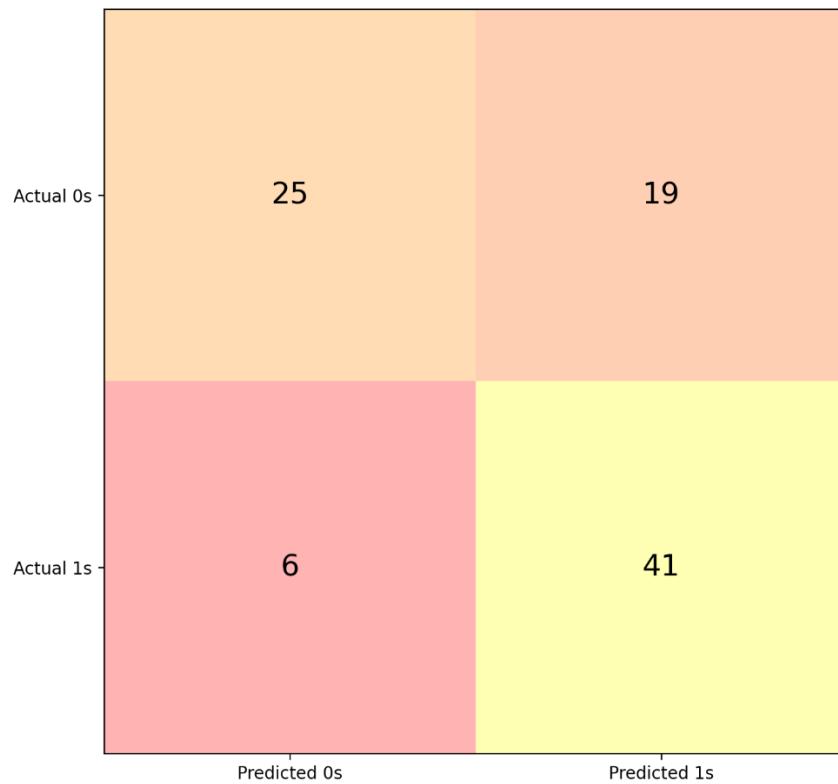
Hình 4.1. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “thal” và “chol”

Để kiểm tra lại mô hình ta sẽ sử dụng tập dữ liệu test và hàm predict() để dự đoán kết quả.

```
#Predict on test set
y_pred = model.predict(X_test)
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	1
228	0	1
201	0	0
52	1	1
245	0	1
175	0	0
168	0	1
223	0	0
217	0	0

Sau đó, ta dùng ma trận hỗn loạn để kiểm tra sự phân chia của các lớp.



Hình 4.2. Ma trận hỗn loạn dựa trên 2 biến “thal” và “chol”

Cuối cùng ta đánh giá mô hình bằng các phương pháp đánh giá như Precision, Recall, F1-Score, Accuracy.

```
#Evaluation metrics
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

precision    recall  f1-score   support

          0       0.81      0.57      0.67      44
          1       0.68      0.87      0.77      47

   accuracy                           0.73      91
macro avg       0.74      0.72      0.72      91
weighted avg    0.74      0.73      0.72      91
```

Như vậy, độ chính xác chung của mô hình này là 73% với 81% dự đoán đúng số người không bệnh và 68% dự đoán đúng số người bị bệnh.

4.2.2.2. Biến nhị phân

Lần này chúng ta sẽ chọn biến độc lập nhị phân để chạy là biến “exang” và “fbs”. Tương tự lần trước, lần này chúng ta vẫn lấy tỉ lệ train:test ở đây sẽ là 7:3.

```
#Split train and test set
X_train, X_test, y_train, y_test = train_test_split(df[['exang', 'fbs']], df.target, test_size=0.3, random_state = 0)
```

Tại đây, chúng ta xây dựng mô hình để dự đoán từ bộ train in ra các hệ số quan trọng như coef và intercept.

```
#Get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)

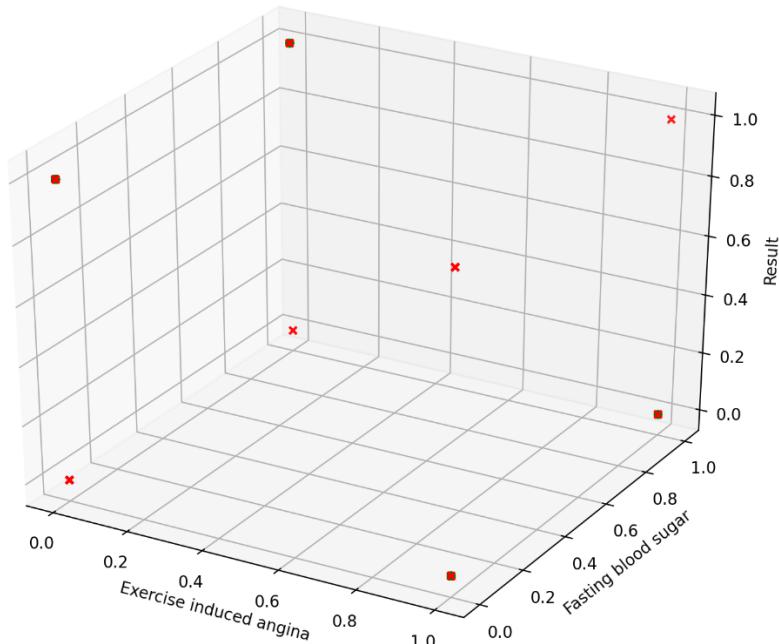
Intercept: [0.9559421]
Coef: [[-1.86119766 -0.48928988]]
```

Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = 0.955 - 1.8611 * \text{exang} - 0.4892 * \text{fbs}$$

Ta cũng tính ra được chỉ số chênh giữa 1:0 là 0.25:2.6, như vậy ta được tỉ số chênh là 0.096. Nghĩa là người bị đau ngực khi vận động (exang) và có đánh giá mức đường huyết lúc đói là nhỏ hơn 120 mg/dl, tức là thấp thì có tỉ lệ mắc bệnh tim chỉ bằng 9.5% so với nhóm còn lại.

Tiếp tục vẽ biểu đồ để dự đoán những người bị mắc bệnh tim mạch.



Hình 4.3. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “fbs” và “exang”

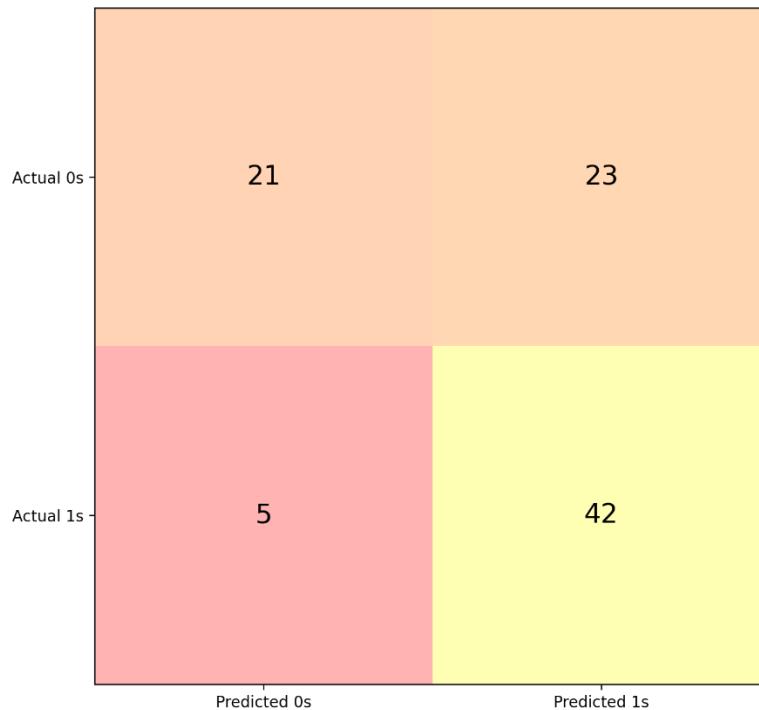
Như vậy, chúng ta thấy, đa phần mô hình chỉ dự đoán đúng những người có: exang = 1 & fbs = 0; exang = 1 & fbs = 1 là không bị bệnh; exang = 0 & fbs = 0; exang = 0 & fbs = 1 là bị bệnh.

Để kiểm tra lại mô hình ta sẽ sử dụng tập dữ liệu test và hàm predict() để dự đoán kết quả.

```
#Predict on test set
y_pred = model.predict(X_test)
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	1
228	0	1
201	0	0
52	1	1
245	0	1
175	0	0
168	0	1
223	0	0
217	0	0

Sau đó, ta dùng ma trận hỗn loạn để kiểm tra sự phân chia của các lớp.



Hình 4.4. Ma trận hỗn loạn dựa trên 2 biến “fbs” và “exang”

Cuối cùng ta đánh giá mô hình bằng các phương pháp đánh giá như Precision, Recall, F1-Score, Accuracy.

```
#Evaluation metrics
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

precision    recall  f1-score   support

          0       0.81      0.48      0.60      44
          1       0.65      0.89      0.75      47

   accuracy                           0.69      91
  macro avg       0.73      0.69      0.68      91
weighted avg       0.72      0.69      0.68      91
```

Như vậy, độ chính xác chung của mô hình này là 69% với 81% dự đoán đúng số người không bệnh và 65% dự đoán đúng số người bị bệnh.

4.2.2.3. Biến thứ bậc

Lần này chúng ta sẽ chọn biến độc lập thứ bậc để chạy là biến “slope” và “thal”. Tương tự hai lần trước, lần này chúng ta vẫn lấy tỉ lệ train:test ở đây sẽ là 7:3.

```
#Split train and test set
X_train, X_test, y_train, y_test = train_test_split(df[['slope', 'thal']], df.target, test_size=0.3, random_state = 0)
```

Tại đây, chúng ta xây dựng mô hình để dự đoán từ bộ train và in ra các hệ số quan trọng như coef và intercept.

```
#Get results
intercept = model.intercept_
coefs = model.coef_
print('Intercept:', intercept)
print('Coef:', coefs)
```

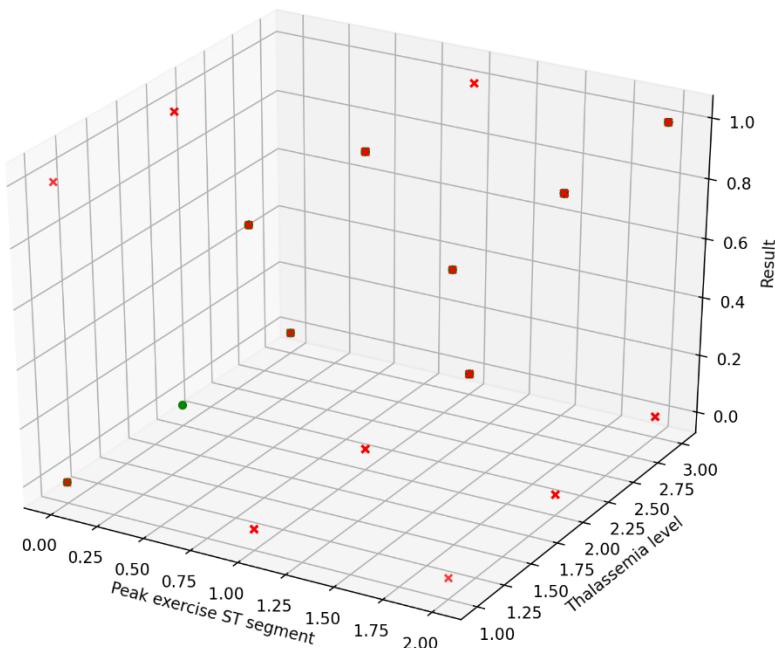
```
Intercept: [1.03214521]
Coef: [[ 1.14903782 -1.05295546]]
```

Từ kết quả trên ta có thể viết được phương trình hồi quy logistic như sau:

$$\text{Heart disease} = 1.03214521 + 1.14903782 * \text{slope} - 1.05295546 * \text{thal}$$

Ta cũng tính ra được chỉ số chênh giữa 1:0 là 1.12:1.03, như vậy ta được tỉ số chênh là 1.08. Nghĩa là người có độ cong đường RT (slope) và mức độ thiếu máu huyết tán (thal) tăng lên 1 bậc thì tỉ lệ mắc tim mạch tăng thêm 1.08 lần so với nhóm còn lại.

Tiếp tục vẽ biểu đồ để dự đoán những người bị mắc bệnh tim mạch như Hình 4.5.



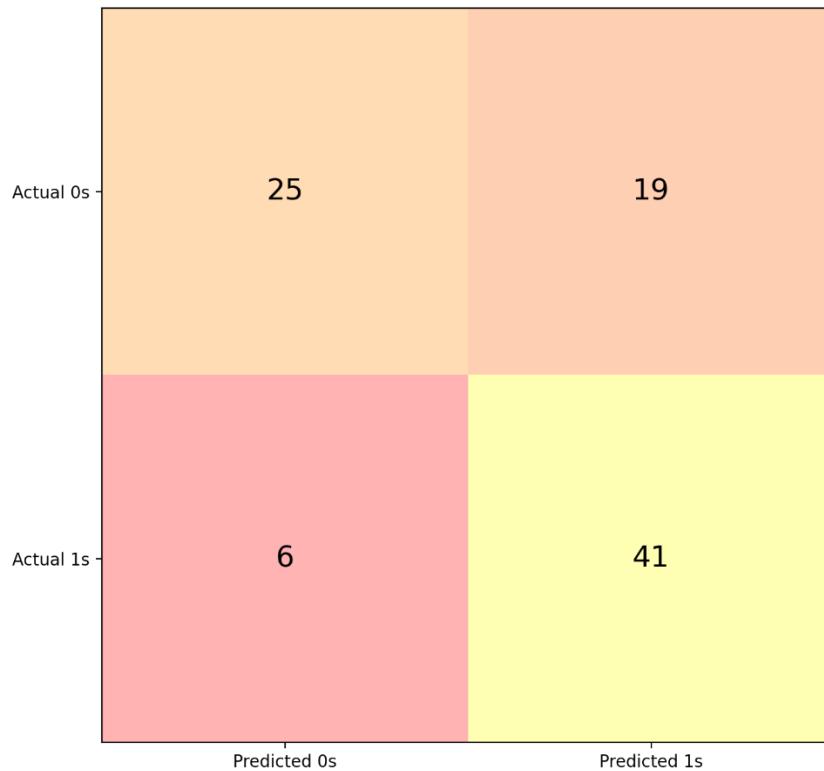
Hình 4.5. Biểu đồ dự đoán mắc bệnh tim mạch dựa trên 2 biến “slope” và “thal”

Để kiểm tra lại mô hình ta sẽ sử dụng tập dữ liệu test và hàm predict() để dự đoán kết quả.

```
#Predict on test set
y_pred = model.predict(X_test)
new = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred})
new.head(10)
```

	Actual value	Predicted value
225	0	0
152	1	0
228	0	0
201	0	0
52	1	0
245	0	0
175	0	0
168	0	0
223	0	0
217	0	1

Sau đó, ta xây dựng ma trận hỗn loạn và visualize nó lên.



Hình 4.6. Ma trận hỗn loạn dựa trên 2 biến “slope” và “thal”

Cuối cùng ta đánh giá mô hình bằng các phương pháp đánh giá như Precision, Recall, F1-Score, Accuracy.

```
#Evaluation metrics
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.81	0.57	0.67	44
1	0.68	0.87	0.77	47
accuracy			0.73	91
macro avg	0.74	0.72	0.72	91
weighted avg	0.74	0.73	0.72	91

Như vậy, độ chính xác chung của mô hình này là 73% với 81% dự đoán đúng số người không bệnh và 68% dự đoán đúng số người bị bệnh.

4.3. Phân tích dự báo với R

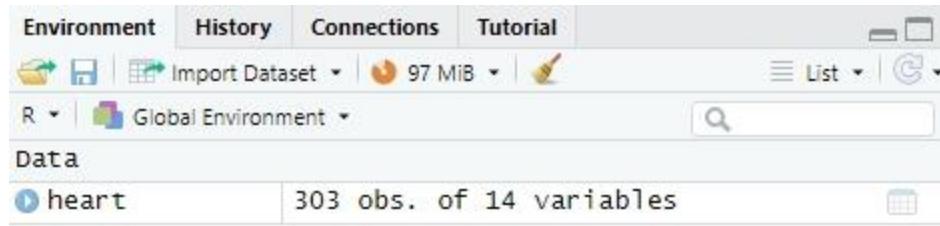
4.3.1. Cài đặt thư viện

Để trực quan hóa dữ liệu trong phần này, ta sẽ sử dụng thư viện scatterplot3d, để vẽ biểu đồ phân tán 3 chiều để trực quan số liệu dựa trên phương pháp hồi quy đa biến. Ta có thể chọn thư viện này trong phần “Packages” của RStudio hoặc nhập đoạn lệnh này vào trực tiếp và nhấn Run.

```
#Import Library
library(scatterplot3d)
library(dplyr)
```

4.3.2. Load dữ liệu

Để load dữ liệu trong R ta có thể chọn “Import Dataset” và chọn file “heart.csv” từ máy tính.



Hình 4.7. Load dữ liệu vào RStudio

Hoặc có thể chạy trực tiếp đoạn code dưới đây:

```
#ImportData  
df <- read_csv("heart.csv")
```

4.3.3. Xây dựng mô hình

Ta sẽ chia tập dữ liệu thành 2 tập train và test với tỉ lệ 75:25. Tập train sẽ dùng để xây dựng và huấn luyện mô hình, tập test sẽ dùng để kiểm định lại mô hình.

```
#Split train & test  
dt = sort(sample(nrow(df), nrow(df)*.75))  
train<-df[dt,]  
test<-df[-dt,]
```

Trong bộ dữ liệu này, biến “target” là biến phụ thuộc, các biến còn lại là biến độc lập. Như yêu cầu đặt ra, chúng ta sẽ chạy lần lượt trên các biến độc lập là liên tục, nhị phân và cả thứ bậc.

4.3.3.1. Biến liên tục

Ở đây, chúng ta sẽ chọn biến độc lập liên tục để chạy là biến “thalach” và “chol”. Nói cách khác, chúng ta sẽ chẩn đoán một người có bị bệnh tim dựa lượng cholesterol và nhịp tim tối đa của họ.

Trong R, để xây dựng mô hình hồi quy logistic ta sử dụng hàm `glm()` với `family = binomial`. Phần tử `family` dùng để mô tả phân phối của sai số và hàm liên kết được sử dụng trong model.

```
#CreateModel
mod1 <- glm(target ~ thalach + chol, family = binomial, data = train)
summary(mod1)

Call:
glm(formula = target ~ thalach + chol, family = binomial, data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.0435 -1.0816  0.6539  0.9731  2.0866 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -5.955e+00 1.307e+00 -4.555 5.25e-06 ***
thalach     4.042e-02 7.515e-03  5.379 7.49e-08 ***
chol        9.895e-05 2.778e-03  0.036   0.972    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

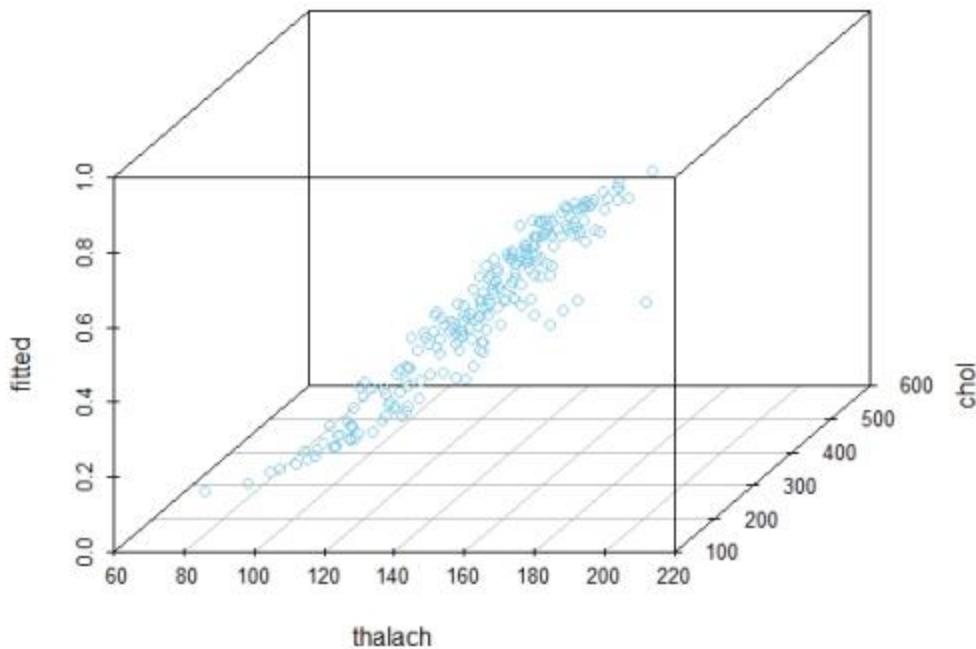
Null deviance: 314.16 on 226 degrees of freedom
Residual deviance: 278.03 on 224 degrees of freedom
AIC: 284.03

Number of Fisher Scoring iterations: 4
```

Hình 4.8. Các thông số của mô hình với 2 biến độc lập “thalach” và “chol”

Tiếp theo, ta vẽ biểu đồ phân tán ba chiều để dự đoán xu hướng mắc bệnh tim mạch của các đối tượng trong tập train.

```
#BuildPlot
train %>%
  mutate(fitted = fitted(mod1)) %>%
  select(thalach, chol, fitted) %>%
  scatterplot3d(color = "skyblue")
```



Hình 4.9. Biểu đồ phân tán ba chiều dựa trên các biến độc lập và mục tiêu

Quan sát biểu đồ, ta thấy những người có hàm lượng cholesterol cao và tim đập nhanh thường có xu hướng mắc bệnh tim mạch cao hơn so với những người có 2 chỉ số này ở mức thấp.

4.3.3.2. Biến nhị phân

Chúng ta tiếp tục chọn biến độc lập có dạng nhị phân để chạy hồi quy logistic cho mô hình này. Ở đây chúng ta sẽ chọn biến “exang” và “fbs”, biểu thị cho hiện tượng đau thắt ngực khi vận động và đánh giá mức đường huyết khi đói của bệnh nhân, tất cả đều ở dạng nhị phân 0 và 1.

Ta tiếp tục sử dụng hàm `glm()` như ở trên, chỉ việc thay đổi tên các biến độc lập lại mà thôi.

```
#CreateModel
mod2 <- glm(target ~ exang + fbs, family = binomial, data = train)
summary(mod2)
```

```

Call:
glm(formula = target ~ exang + fbs, family = binomial, data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.5633 -0.6656  0.8355  0.8355  1.9041 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.8729    0.1896   4.603 4.17e-06 ***
exang       -2.2675    0.3384  -6.700 2.08e-11 ***
fbs        -0.2401    0.4209  -0.571   0.568    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 314.16 on 226 degrees of freedom
Residual deviance: 258.59 on 224 degrees of freedom
AIC: 264.59

Number of Fisher Scoring iterations: 4

```

Hình 4.10. Các thông số của mô hình với 2 biến độc lập “fbs” và “exang”

Từ mô hình trên, ta dễ dàng thu được hệ số intercept là 0.8729 và coef của “exang” và “fbs” lần lượt là -2.2675 và -0.2401.

4.3.3.3. Biến thứ bậc

Cuối cùng là đến biến độc lập có dạng thứ bậc. Biến “slope” và “thal”, biểu thị cho mức độ dốc của đoạn ST cao điểm và đánh giá mức độ bệnh thiếu máu huyết tán Thalassemia là những biến thứ bậc sẽ được áp dụng cho lần chạy này.

Chúng ta lại thao tác trên hàm glm() như cũ:

```
#CreateModel
mod3 <- glm(target ~ slope + thal, family = binomial, data = train)
summary(mod3)
```

```

Call:
glm(formula = target ~ slope + thal, family = binomial, data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.1626 -1.1594  0.7468  0.7468  2.1672 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.0602    0.6829   1.552   0.121    
slope        1.1398    0.2550   4.470 7.81e-06 ***
thal        -1.1027    0.2514  -4.386 1.16e-05 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 314.16 on 226 degrees of freedom
Residual deviance: 268.92 on 224 degrees of freedom
AIC: 274.92

Number of Fisher Scoring iterations: 4

```

Hình 4.11. Các thông số của mô hình với 2 biến độc lập “slope” và “thal”

Từ mô hình trên, ta lấy được hệ số intercept là 1.0602 và coef của “slope” và “thal” lần lượt là 1.1398 và -1.1027.

4.3.4. Đánh giá mô hình

Để kiểm tra lại mô hình ta sẽ sử dụng tập dữ liệu test và hàm predict() để dự đoán kết quả. Ngoài ra, ta sẽ đặt ngưỡng giá trị lớn hơn 0.5 để dự đoán những người có kết quả mắc bệnh tim.

```

> #Evaluation
> predTest <- predict(mod1,test,type="response")
> pred_classes <- ifelse(predTest >0.5, '1','0')
> table(test$target, predTest>0.5)

    FALSE TRUE
0      19   11
1       8   38

> predTest2 <- predict(mod2,test,type="response")
> pred_classes2 <- ifelse(predTest2 >0.5, '1','0')
> table(test$target, predTest2>0.5)

    FALSE TRUE
0      13   17
1       8   38

```

```

> predTest3 <- predict(mod3,test,type="response")
> pred_classes3 <- ifelse(predTest3 >0.5, '1','0')
> table(test$target, predTest3>0.5)

    FALSE  TRUE
0      25     5
1      14    32

```

Từ đó ta lần lượt thu được các độ chính xác của từng mô hình một lần lượt từ biến liên tục, biến nhị phân và biến thứ bậc là:

```

> #ModelAccuracy
> mean(pred_classes == test$target)
[1] 0.75
> mean(pred_classes2 == test$target)
[1] 0.6710526
> mean(pred_classes3 == test$target)
[1] 0.75
> |

```

Ta thấy cả 3 cách lựa chọn biến đều cho độ chính xác khá cao, từ 67% đến gần 75%. Trong đó biến liên tục và thứ bậc cho kết quả cao nhất, song, thứ tự các mức chính xác giữa 3 cách chọn biến có thể thay đổi vì sự phân chia ngẫu nhiên tập train test giữa các lần chạy khác nhau nhưng nhìn chung sẽ không có khác biệt quá lớn vì tập dữ liệu cũng tương đối nhỏ.

CHƯƠNG 5. DỮ LIỆU CHUỖI THỜI GIAN & MÔ HÌNH ARIMA

5.1. Cơ sở lý thuyết

5.1.1. Khái niệm

Dữ liệu chuỗi thời gian có cấu trúc của một bảng dữ liệu thông thường, nhưng bắt buộc phải có một cột thể hiện chuỗi thời gian và các giá trị phải có sự thay đổi theo nó. Thời gian đo lường có thể tính bằng giờ, ngày, tháng, quý, đến năm hoặc bất cứ khoảng thời gian nào được quy ước trước đó như theo quý, 6 tháng, 3 tháng,...

Năm	2015	2016	2017	2018	2019	2020
Doanh thu sp A	1400	1323	1500	1670	1620	1430

Bảng 5.1. Ví dụ về dữ liệu chuỗi thời gian (ĐVT: triệu đồng)

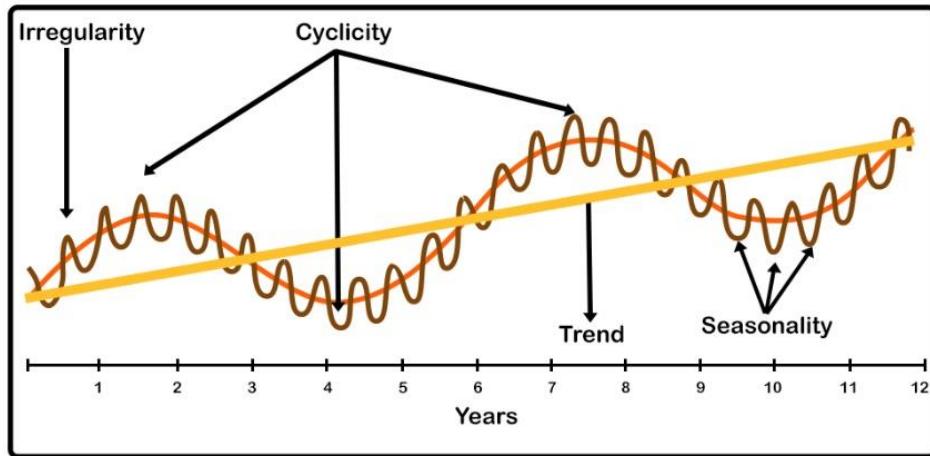
Ví dụ doanh thu sản phẩm A của công ty được ghi nhận theo thời kỳ mà ở đây là năm, tổng doanh thu các tháng trong năm 2015, hay doanh thu trong năm 2015 là 1400 triệu VND = 1.4 tỷ.

Chuỗi dừng là dữ liệu thời gian đã được độc lập ra khỏi sự thay đổi của thời gian. Vì các thông số của dữ liệu chuỗi thời gian sẽ thay đổi dần theo thời gian nên chúng ta cần phải chuyển dữ liệu thành chuỗi dừng để thuận tiện cho việc phân tích.

Một dữ liệu chuỗi thời gian được gọi là nhiễu trắng khi các giá trị độc lập và phân phối trên giá trị trung bình là 0. Nếu như chuỗi dữ liệu là nhiễu trắng thì nó sẽ không thể dùng để phân tích vì nó hoàn toàn ngẫu nhiên.

$$y(t) = signal(t) + noise(t)$$

5.1.2. Thành phần của chuỗi thời gian



Hình 5.1. Các thành phần trong chuỗi dữ liệu thời gian

5.1.2.1. Xu hướng (Trend - T)

Thể hiện chiều hướng biến động, tăng hoặc giảm của đối tượng nghiên cứu trong một khoảng thời gian dài. Mặc dù dữ liệu chuỗi thời gian nói chung thể hiện các biến động ngẫu nhiên, nó cũng có thể cho thấy sự thay đổi hoặc chuyển động dần dần đến các giá trị cao hơn hoặc thấp hơn trong một khoảng thời gian dài hơn. Nếu một biểu đồ chuỗi thời gian có biểu hiện như vậy, chúng ta có thể nói có một “xu hướng” tồn tại. Không nhất thiết phải luôn luôn tăng hoặc luôn luôn giảm thì mới gọi là xu hướng. Nó có thể tăng, hoặc giảm, hoặc ổn định trong 1 khoảng thời gian nào đó. Nhưng trong tổng thể, nhìn trên đồ thị từ điểm bắt đầu đến kết thúc chúng ta nhìn thấy có một xu hướng đi lên, hoặc đi xuống, hoặc đứng yên.

Xu hướng thường là kết quả của các yếu tố tác động dài hạn như dân số tăng hoặc giảm, thay đổi đặc điểm nhân khẩu học của dân số, sự thay đổi của công nghệ và / hoặc sở thích của người tiêu dùng, năng lực sản xuất,... Xu hướng có 2 dạng tuyến tính và phi tuyến tính.

5.1.2.2. Mùa vụ (Seasonal – S)

Xu hướng của một chuỗi thời gian có thể được xác định bằng cách phân tích các chuyển động qua nhiều năm trong dữ liệu lịch sử. Các biểu hiện theo mùa vụ thì được nhận

biết bằng cách quan sát các dấu hiệu tăng, giảm của đối tượng nghiên cứu lặp lại giống nhau trong các khoảng thời gian liên tiếp. Ví dụ, một nhà bán lẻ dự kiến hoạt động bán hàng thấp trong các tháng mùa thu và mùa hè, với doanh số bán hàng cao điểm vào các tháng mùa xuân và mùa đông.

Biểu đồ chuỗi thời gian có dấu hiệu lặp lại trong các khoảng thời gian trong một năm (theo giờ, ngày, tuần, tháng, quý, năm và không cao hơn 1 năm) do ảnh hưởng của mùa được gọi là dấu hiệu mùa vụ.

Mặc dù chúng ta thường nghĩ về sự chuyển động theo mùa trong một chuỗi thời gian xảy ra trong vòng một năm, nhưng dữ liệu chuỗi thời gian cũng có thể thể hiện các mô hình theo mùa dưới một năm. Ví dụ: dữ liệu lưu lượng giao thông hàng ngày cho thấy hành vi “theo mùa” trong ngày, với mức cao điểm xảy ra vào giờ cao điểm, lưu lượng vừa phải vào thời gian còn lại trong ngày và đầu giờ tối và lưu lượng nhẹ từ nửa đêm đến sáng sớm.

Những biến động do thời vụ là do tự nhiên như thời tiết, khí hậu hoặc do chính con người tạo ra. Các mùa hoặc điều kiện khí hậu khác nhau đóng một vai trò quan trọng trong sự thay đổi theo mùa. Chẳng hạn như sản xuất cây trồng phụ thuộc vào mùa vụ, việc bán ô và áo mưa vào mùa mưa, và việc bán quạt điện và máy lạnh tăng lên vào mùa hè. Ảnh hưởng do con người tạo ra như một số lễ hội, phong tục, tập quán, tuần sales, khuyến mãi... có thể dễ dàng nhận thấy.

5.1.2.3. Chu kỳ (Cyclical – C)

Mô hình chu kỳ tồn tại nếu biểu đồ chuỗi thời gian hiển thị một chuỗi xen kẽ các điểm bên dưới và bên trên đường xu hướng (tăng và giảm lặp lại) kéo dài hơn một năm hay nói cách khác nếu biến động theo thời vụ kéo dài hơn 1 năm, tối đa 10 năm, hoặc nhiều hơn thì có thể gọi là biến động theo chu kỳ.

5.1.2.4. Ngẫu nhiên (Irregular – I)

Các biến động ngẫu nhiên trong ngắn hạn không lường trước hay dự báo được. Các biến động ngẫu nhiên có thể xuất hiện trong thời gian ngắn, và không có dấu hiệu lặp lại,

không có quy luật nào cả tuy nhiên trong khoảng thời gian dài có thể xảy ra nhiều lần, và với nhiều hình thức khác nhau, mức độ tác động khác nhau, tạo ra độ nhiễu nhất định ví dụ thiên tai, khủng bố, các tin tức tiêu cực...

5.1.3. Mô hình ARIMA

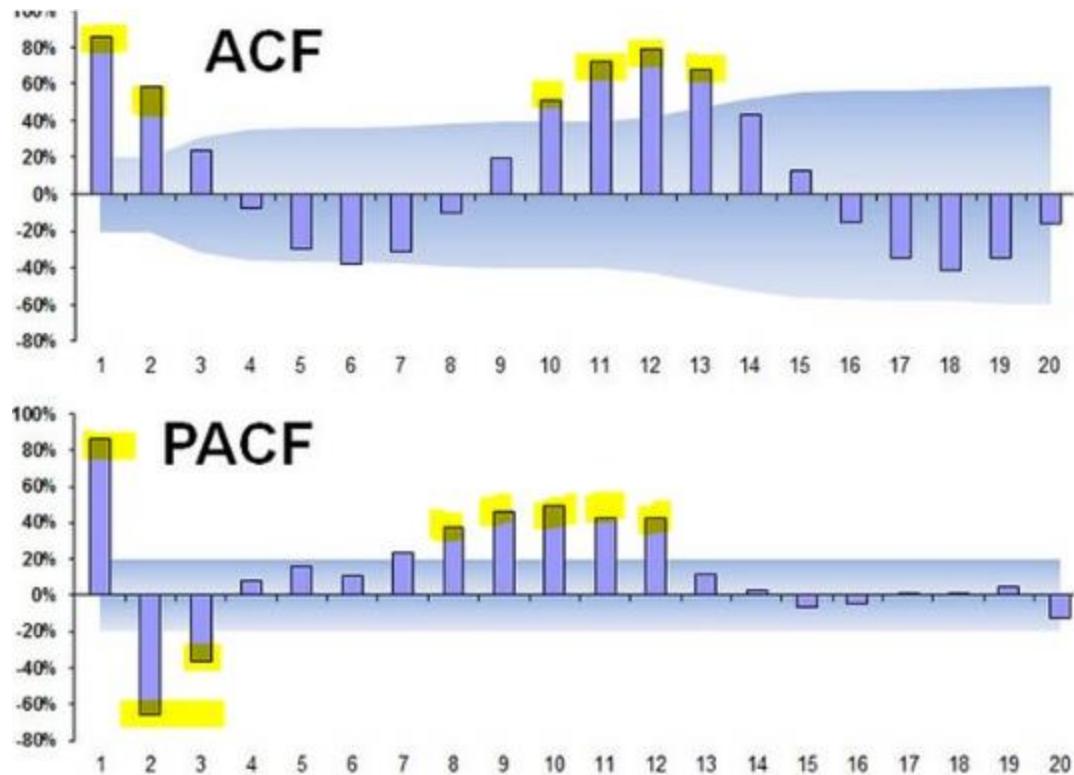
Mô hình tự hồi quy (*AutoRegression*) là mô hình dự đoán giá trị tương lai dựa vào giá trị quá khứ. Nó được dùng để mô hình hóa mối quan hệ giữa dữ liệu và độ trễ của chính nó. Trong mô hình tự hồi quy, ta xác định số p bằng cách vẽ sơ đồ PACF (*Partial Auto Correlation Factor*).

$$AR(p): Y_t = \alpha + \beta_1 Y_{t-1} + \dots + \beta_p Y_{t-p} + \varepsilon_t$$

Trung bình trượt (*MovingAverage*) mô hình hóa mối quan hệ giữa dữ liệu và sai số so với trung bình các độ trễ khác. Để xác định chỉ số q trong mô hình trung bình trượt ta dùng sơ đồ ACF (*Auto Correlation Factor*).

$$\text{residual}(\mu) = \text{expected} - \text{predicted}$$

$$MA(q): Y_t = \mu + \beta_1 \varepsilon_{t-1} + \dots + \beta_q \varepsilon_{t-q} + \varepsilon_t$$



Hình 5.2. Sơ đồ ACF và PACF

Mô hình là ARMA là kết quả của sự kết hợp của mô hình AR và mô hình MA.

$$ARMA(1, 1): Y_t = \alpha + \beta Y_{t-1} + \theta \varepsilon_{t-1} + \varepsilon_t$$

Mô hình ARIMA, cũng giống như mô hình ARMA nhưng có thêm yếu tố tích hợp I (*Integrated*). Việc sử dụng yếu tố này nhằm loại bỏ tính xu hướng trong dữ liệu, thể hiện qua chỉ số d.

5.2. Giới thiệu dữ liệu

Bộ dữ liệu “day_wise.csv” chứa dữ liệu về tình hình diễn biến của đại dịch Covid 19 trên toàn thế giới tính từ 22/01/2020 đến ngày 27/07/2020. Dữ liệu này được thu thập từ các nguồn báo chí chính thống từ các quốc gia.

Link dữ liệu: <https://www.kaggle.com/imdevskp/corona-virus-report>

Bộ dữ liệu có 12 thuộc tính:

STT	Tên thuộc tính	Mô tả thuộc tính
1	Date	Ngày tháng năm
2	Confirmed	Số ca nhiễm đã ghi nhận
3	Deaths	Số ca tử vong đã ghi nhận
4	Recovered	Số ca hồi phục được ghi nhận
5	Active	Số ca đang điều trị
6	New cases	Số ca nhiễm mới
7	New deaths	Số người tử vong trong ngày
8	New recovered	Số ca đã hồi phục trong ngày
9	Deaths / 100 Cases	Lượng người tử vong trên 100 ca nhiễm
10	Recovered / 100 Cases	Lượng người hồi phục trên 100 ca nhiễm
11	Deaths / 100 Recovered	Số ca tử vong trên 100 ca hồi phục
12	No. of countries	Số lượng quốc gia được ghi nhận

Bảng 5.2. Các thuộc tính trong bộ dữ liệu “day_wise.csv”

5.3. Phân tích dự báo với Python

5.3.1. Cài đặt thư viện

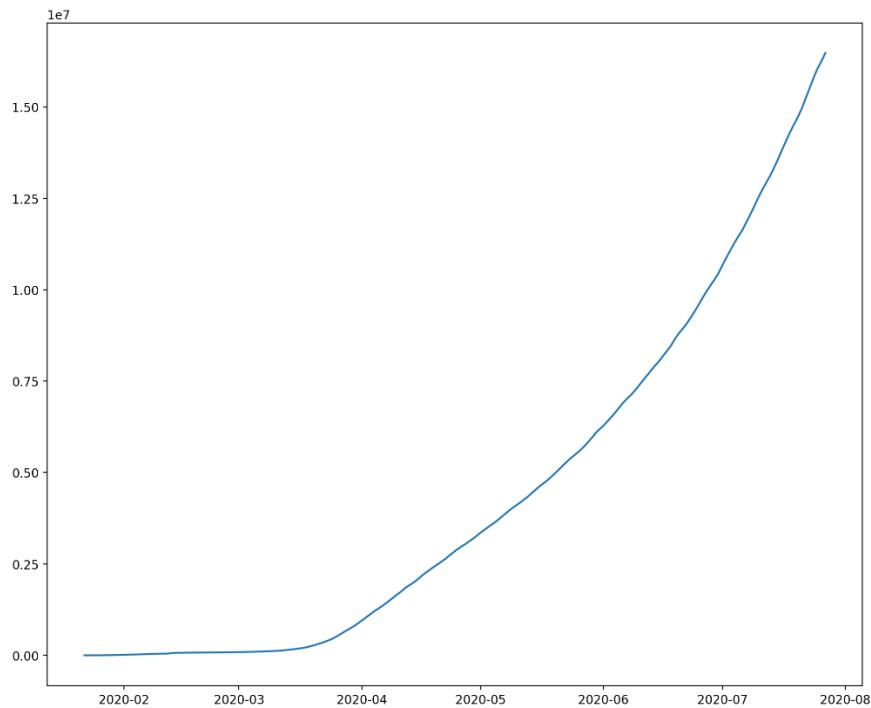
Để tiến hành phân tích dữ liệu chuỗi thời gian, chúng ta cần cài đặt các bộ thư viện như: pandas, numpy, matplotlib,... Đồng thời trong file mã nguồn chương trình, nhóm còn định nghĩa các chức năng để thể hiện rõ các kết quả của các kiểm định tính dừng.

```
from my_function import *
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import r2_score, mean_squared_error
from pmdarima import auto_arima

import warnings
warnings.filterwarnings('ignore')
```

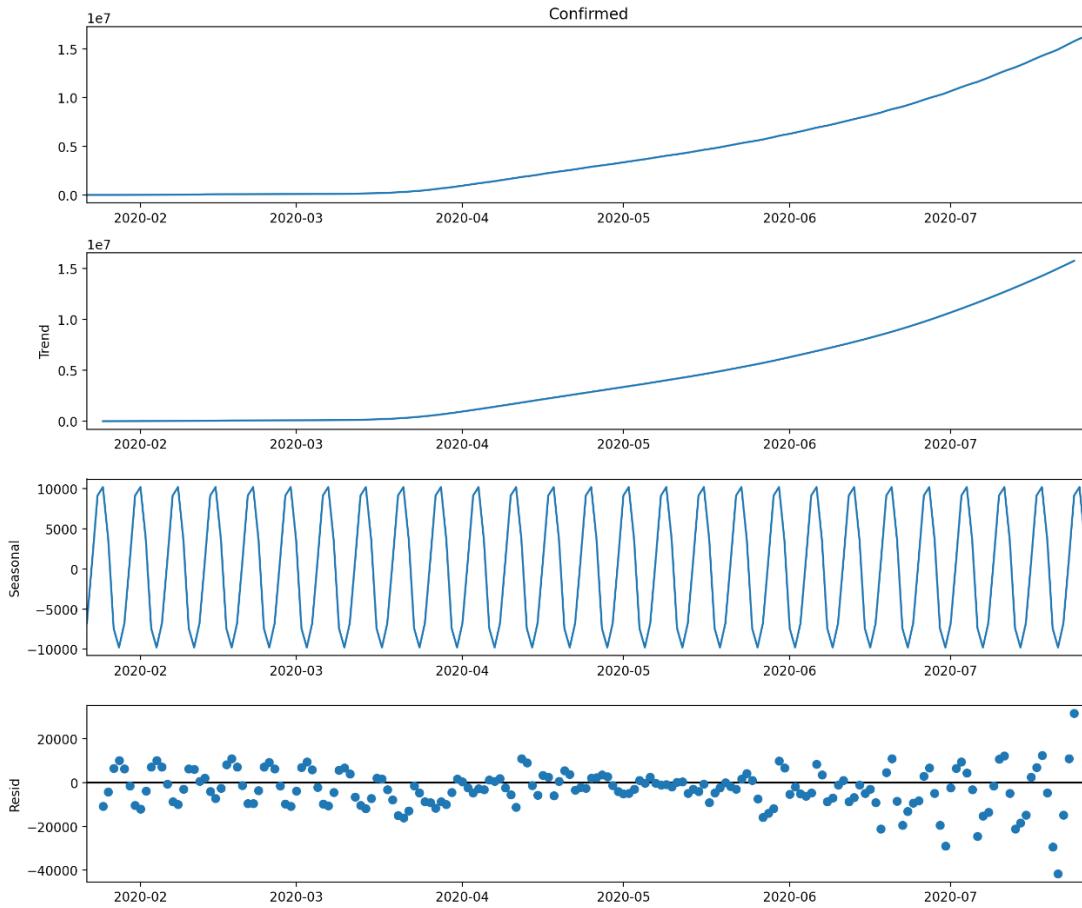
5.3.2. Khám phá dữ liệu

Từ Hình 5.3, ta nhận thấy được số lượng các ca nhiễm tăng đột biến từ giai đoạn tháng 4 trở đi. Điều này, cho thấy tốc độ lây lan của dịch bệnh khi chỉ trong vòng 2 tháng số lượng các quốc gia ghi nhận có ca nhiễm tăng từ 6 lên 179 quốc gia.



Hình 5.3. Số lượng ca nhiễm Covid 19 từ tháng 1 đến tháng 7 năm 2020

Để hiểu rõ hơn về dữ liệu, ta tiến hành kiểm tra các thành phần của một dữ liệu chuỗi thời gian.



Hình 5.4. Tính chu kỳ, tính thời vụ, tính xu hướng và các điểm khác thường

5.3.3. Kiểm tra tính dừng

Trước khi tiến hành đưa dữ liệu vào các mô hình phân tích, ta thực hiện kiểm tra tính dừng của bộ dữ liệu. Kết quả, chạy kiểm định ADF cho thấy điểm trị tuyệt đối kiểm định 2.06 không lớn hơn giá trị tuyệt đối critical value (5%) là -2.88, nên ta kết luận chuỗi này chưa dừng.

```
print(ADF(y))
```

ADF	2.056470
p_value	0.998742
lags	11.000000
oservation	176.000000
Critical value (1%)	-3.468062
Critical value (5%)	-2.878106
Critical value (10%)	-2.575602
dtype: float64	

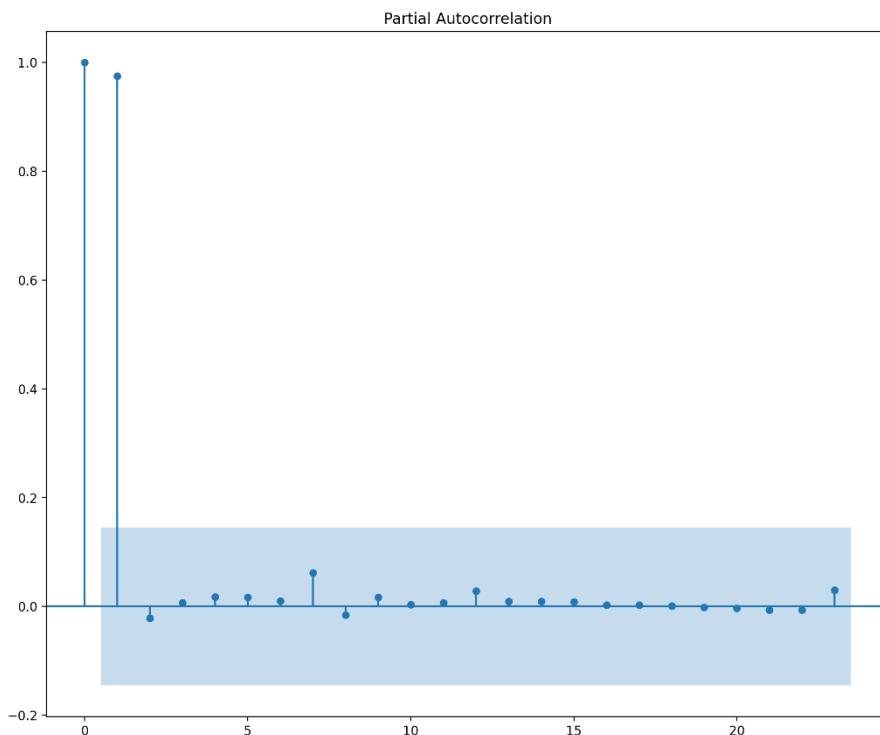
Để đổi chuỗi dữ liệu thành chuỗi dừng, ta sử dụng chức năng log và tiến hành kiểm định lại một lần nữa. Kết quả sau khi, đổi chuỗi ta thấy điểm trị tuyệt đối kiểm định là -3.51 nhỏ hơn điểm tuyệt đối tại critical value (5%) là -2.88, ta kết luận chuỗi đã dừng.

```
y_log = np.log(y)
print(ADF(y_log))
```

```
ADF           -3.509179
p_value       0.007754
lags          15.000000
oservation   172.000000
Critical value (1%) -3.468952
Critical value (5%)  -2.878495
Critical value (10%) -2.575809
dtype: float64
```

5.3.4. Phân tích bằng mô hình tự hồi quy

Mô hình AR được tính toán dựa trên chỉ số p, vì vậy ta sử dụng sơ đồ PACF để xác định chỉ số này. Hình 5.5, cho thấy ta có thể cho p bằng 1.



Hình 5.5. Sơ đồ PACF

Sau đó, ta tiến hành chạy mô hình AR với $p = 1$. Kết quả, của mô hình được thể hiện dưới đây.

```
AR_model = AutoReg(train, lags=1).fit()
AR_model.summary()
```

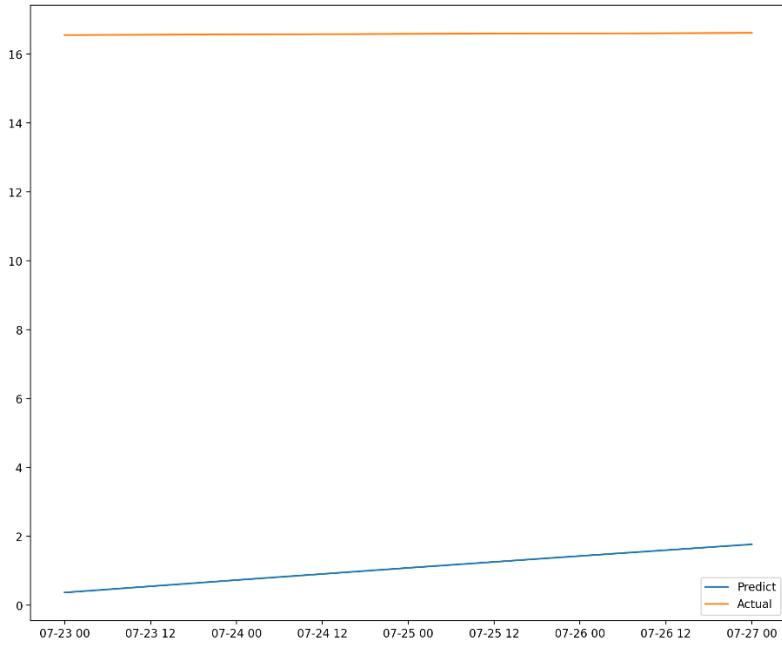
```
AutoReg Model Results
Dep. Variable: Confirmed      No. Observations: 183
Model: AutoReg(1)            Log Likelihood   252.943
Method: Conditional MLE     S.D. of innovations 0.060
Date: Sat, 14 Aug 2021       AIC             -5.585
Time: 13:04:57                BIC             -5.532
Sample: 01-23-2020           HQIC            -5.563
- 07-22-2020
            coef  std err      z      P>|z| [0.025 0.975]
intercept  0.3706  0.026  14.479  0.000  0.320  0.421
Confirmed.L1 0.9772  0.002  533.668  0.000  0.974  0.981
Roots
Real    Imaginary Modulus Frequency
AR.1  1.0234 +0.0000j  1.0234  0.0000
```

Hình 5.6. Kết quả của mô hình tự hồi quy

Sau đó, ta tiến hành dự đoán dữ liệu kiểm tra để kiểm tra tính chính xác của mô hình.

```
predict_AR = AR_model.predict(start=len(train), end=len(y_log)-1, dynamic=True)
plt.plot(predict_AR,label='Predict')
plt.plot(test,label='Actual')
plt.legend()
```

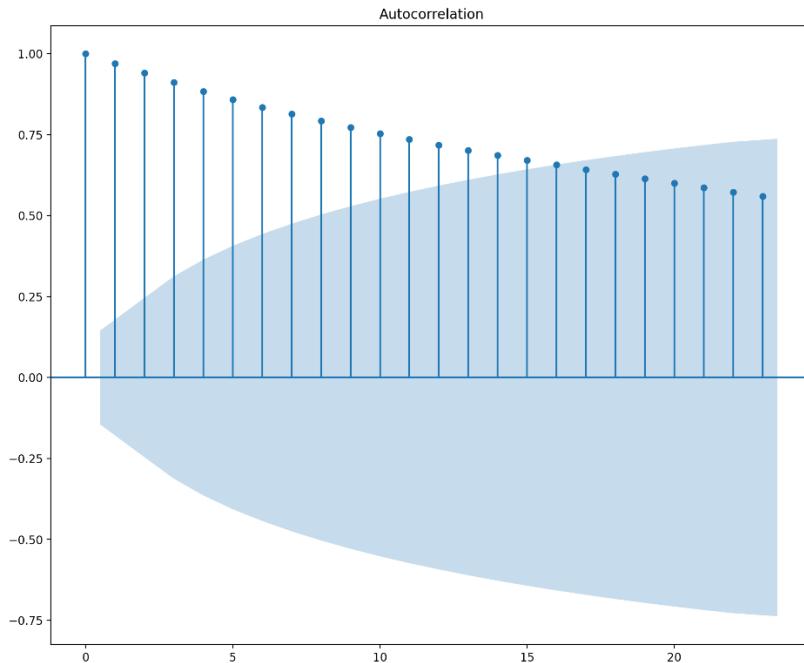
Hình 5.7 cho thấy dữ liệu dự đoán có xu hướng tăng, nhưng không thể hiện nhiều, do biểu đồ vẫn còn thể hiện ở giá trị log. Để kết luận, ta có thể nói mô hình AR, cho kết quả dữ liệu có xu hướng tương đồng với dữ liệu gốc, thế nhưng độ chênh lệch dữ liệu là rất lớn, nên mô hình AR không phù hợp để dự đoán dữ liệu này.



Hình 5.7. Kết quả dự đoán khi dùng mô hình AR so với dữ liệu thật

5.3.5. Phân tích bằng mô hình ARMA

Để dự đoán bằng mô hình ARMA, bên cạnh chỉ số p , ta còn cần xác định chỉ số q . Vì vậy, ta sử dụng sơ đồ ACF để xác định chỉ số này.



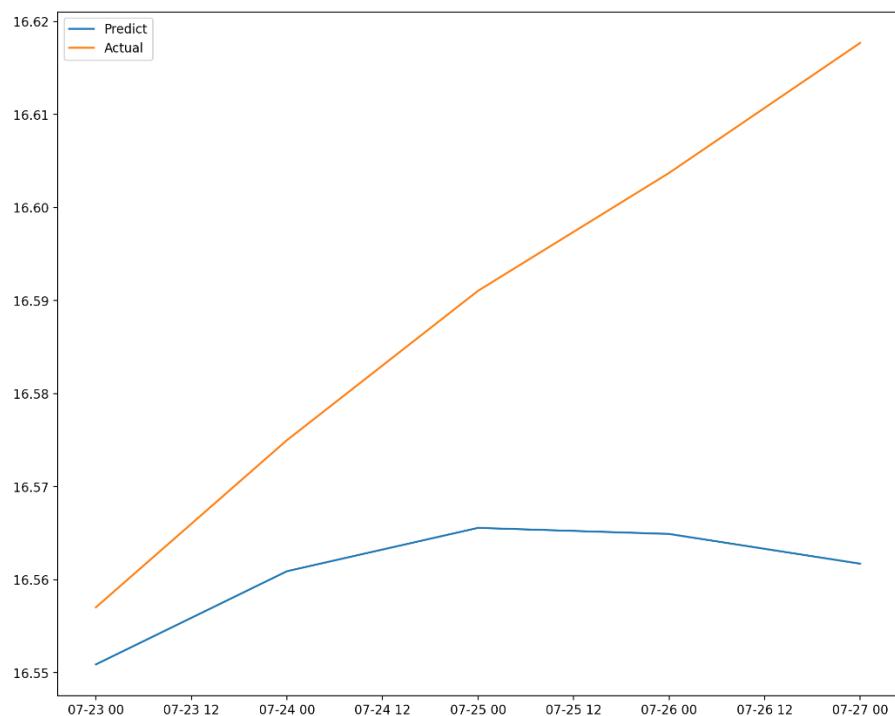
Hình 5.8. Sơ đồ ACF

Sau đó, tiến hành chạy mô hình dự đoán với $p = 1$ và $q = 4$. Rồi dự đoán dữ liệu để so sánh tương tự như với mô hình AR.

```
ARMA_model = ARIMA(train, order=(1,0,4)).fit()
ARMA_model.summary()

predict_ARMA = ARMA_model.predict(start=len(train), end=len(y_log)-1, dynamic=True)
plt.plot(predict_ARMA,label='Predict')
plt.plot(test,label='Actual')
plt.legend()
```

Hình 5.9 cho thấy độ chênh lệch giữa dữ liệu thật và dữ liệu dự đoán đã giảm đáng kể thế nhưng xu hướng dữ liệu vẫn chưa khớp với dữ liệu thật. Điều này xảy ra có thể do bộ dữ liệu được sử dụng để đào tạo mô hình khá ít, hoặc do dữ liệu được tổng hợp từ nhiều quốc gia nên lượng dữ liệu gia tăng không theo quy luật.



Hình 5.9. Kết quả dự đoán khi dùng mô hình ARMA so với dữ liệu thật

5.3.6. Phân tích bằng mô hình ARIMA

Trước khi chạy phân tích dữ liệu bằng mô hình ARIMA, ta cần chạy thuật toán auto arima để xác định lại các hệ số p, d, q tương ứng. Kết quả, của thuật toán cho kết quả p, d, q lần lượt là 0, 2, 3.

```
auto_arima(train,trace=True, suppress_warnings=True)

Performing stepwise search to minimize aic
ARIMA(2,2,2)(0,0,0)[0] : AIC=-591.806, Time=0.68 sec
ARIMA(0,2,0)(0,0,0)[0] : AIC=-508.556, Time=0.04 sec
ARIMA(1,2,0)(0,0,0)[0] : AIC=-582.015, Time=0.08 sec
ARIMA(0,2,1)(0,0,0)[0] : AIC=-587.487, Time=0.07 sec
ARIMA(1,2,2)(0,0,0)[0] : AIC=-592.834, Time=0.36 sec
ARIMA(0,2,2)(0,0,0)[0] : AIC=-593.467, Time=0.24 sec
ARIMA(0,2,3)(0,0,0)[0] : AIC=-594.600, Time=0.58 sec
ARIMA(1,2,3)(0,0,0)[0] : AIC=-589.730, Time=0.60 sec
ARIMA(0,2,4)(0,0,0)[0] : AIC=-593.295, Time=0.57 sec
ARIMA(1,2,4)(0,0,0)[0] : AIC=-592.123, Time=0.39 sec
ARIMA(0,2,3)(0,0,0)[0] intercept : AIC=-594.026, Time=0.67 sec

Best model: ARIMA(0,2,3)(0,0,0)[0]
Total fit time: 4.302 seconds
ARIMA(order=(0, 2, 3), scoring_args={}, suppress_warnings=True,
      with_intercept=False)
```

Sau đó, ta đưa 3 giá trị trên vào tiến hành chạy mô hình dự đoán ARIMA.

```
ARIMA_model = ARIMA(train,order=(0,2,3)).fit()
ARIMA_model.summary()

predict_ARIMA = ARIMA_model.predict(start=len(train), end=len(y_log)-1,
                                      dynamic=True)
plt.plot(predict_ARIMA,label='Predict')
plt.plot(test,label='Actual')
plt.legend()
```

Hình 5.10 cho thấy kết quả dự đoán còn tệ hơn so với 2 mô hình ban đầu. Ta có thể kết luận chắc chắn rằng mô hình ARIMA hoàn toàn không phù hợp để phân tích bộ dữ liệu này. Đồng thời các yếu tố khác trong bộ dữ liệu có ảnh hưởng đến kết quả phân tích.



Hình 5.10. Kết quả dự đoán bằng mô hình ARIMA so với dữ liệu thật

5.4. Phân tích dự báo với R

5.4.1. Cài đặt thư viện

Để có thể chạy các mô hình phân tích AR, ARMA, và ARIMA ta cần cài đặt thư viện tseries – thư viện chuyên xử lý dữ liệu chuỗi thời gian, thư viện tidyverse – thư viện có các phương thức để tiền xử lý dữ liệu, và thư viện forecast – để thực thi các mô hình.

```
# import library
library(tseries)
library(tidyverse)
library(forecast)
```

5.4.2. Kiểm tra tính dừng

Tương tự như khi sử dụng ngôn ngữ Python, ta cần kiểm tra tính dừng của bộ dữ liệu. Rõ ràng khi kiểm định ADF trên R các giá trị kết quả có sự thay đổi, nhưng vẫn thể hiện rằng bộ dữ liệu vẫn chưa thàng chuỗi dừng.

```
> adf.test(df$confirmed)
Augmented Dickey-Fuller Test

data: df$confirmed
Dickey-Fuller = 1.4336, Lag order = 5, p-value = 0.99
alternative hypothesis: stationary
```

Tiến hành log dữ liệu và kiểm tra lại. Kết quả có thay đổi thế nhưng so với Python, kiểm định ADF trên R vẫn cho kết quả là chuỗi chưa dừng, hoặc dừng yếu. Ta sẽ dùng kiểm định tại đây để kiểm tra một giả thuyết rằng có khả năng kết quả kiểm định trên R là chính xác, và vì vậy ảnh hưởng đến kết quả dự đoán mô hình.

```
> y_log = log(y)
> adf.test(y_log)
Augmented Dickey-Fuller Test

data: y_log
Dickey-Fuller = -1.6665, Lag order = 5, p-value = 0.7156
alternative hypothesis: stationary
```

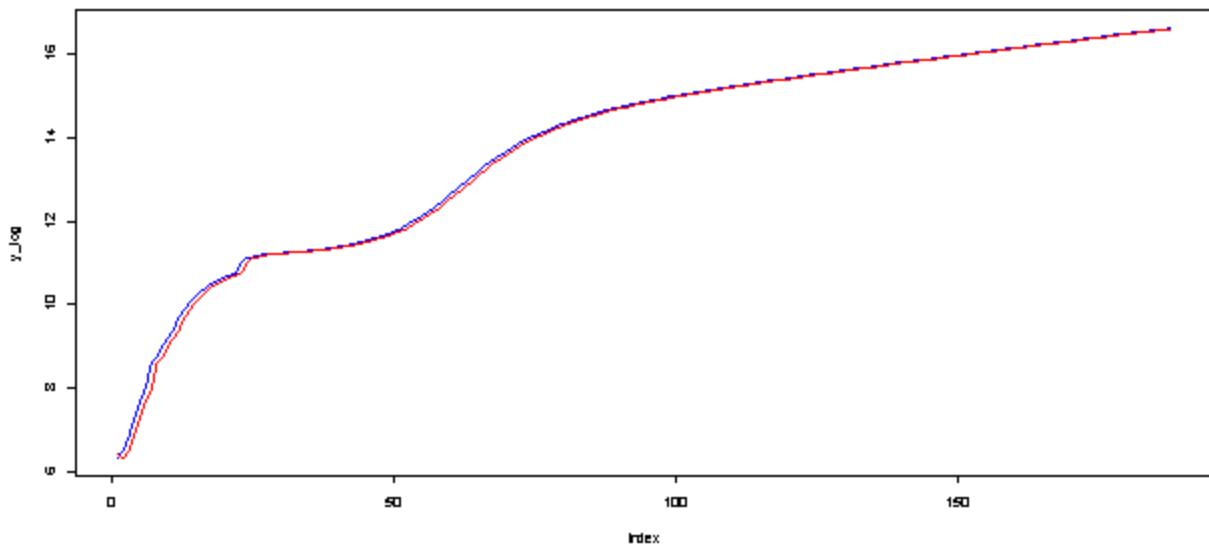
5.4.3. Phân tích bằng mô hình tự hồi quy

Ta thực hiện xây dựng và dự đoán kết quả của mô hình theo đoạn code như sau:

```
#Xây dựng model AR
AR_model = arima(y_log, order =c(1,0,0))

#Visualize
plot(y_log, type="l", col="blue")
pre = y_log - residuals(AR_model)
lines(pre, type="l", col="red")
```

Hình 5.11 cho thấy một kết quả khá thú vị, việc kết quả kiểm định ADF bằng ngôn ngữ R khác hoàn toàn với ngôn ngữ Python thế nhưng mô hình dự đoán có dấu hiệu xảy ra tình trạng overfitting. Điều này có thể dùng để lý giải cho kết quả khi dùng Python.



Hình 5.11. Kết quả mô hình AR trên R

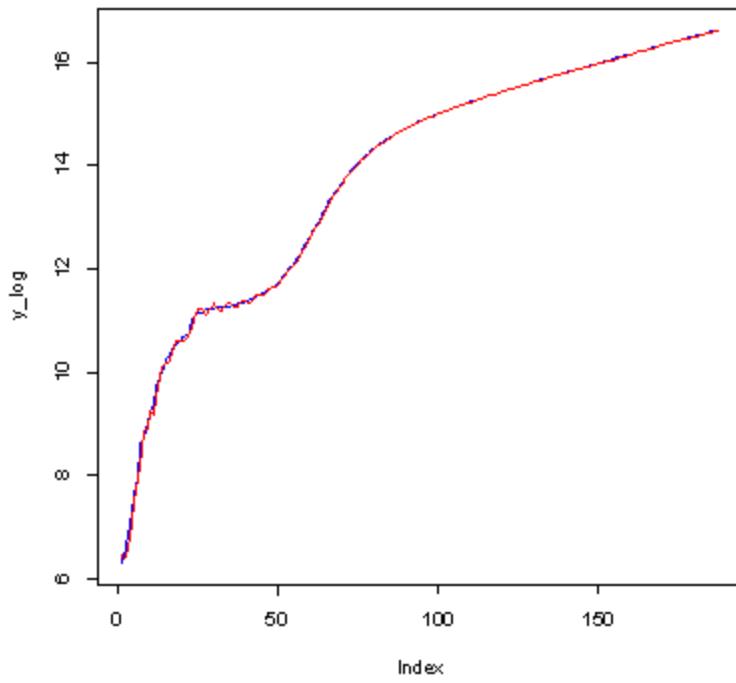
5.4.4. Phân tích bằng mô hình ARMA

Ta sẽ thực hiện phân tích mô hình ARMA với p và q tương tự như trên ngôn ngữ Python.

```
#xây dựng model ARMA
ARMA_model = arima(y_log, order =c(1,0,3))

#visualize
plot(y_log, type="l",col="blue")|
pre = y_log - residuals(ARMA_model)
lines(pre, type="l",col="red")
```

Hình 5.12 đã chứng minh giả thuyết về tình trạng “overfitting” khi dự đoán trên mô hình ARMA. Thế nhưng, ta vẫn cần tiếp tục sử dụng mô hình ARIMA với bộ dữ liệu này.



Hình 5.12. Kết quả dự đoán mô hình ARMA

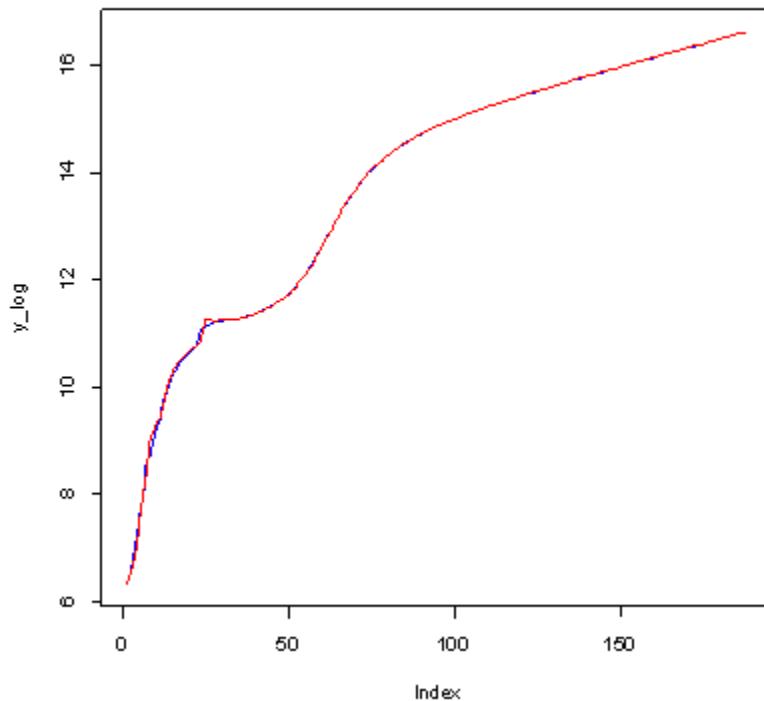
5.4.5. Phân tích bằng mô hình ARIMA

Để việc kiểm tra kết quả một cách chính xác, thay vì sử dụng p, d, q như ở ngôn ngữ Python, ta sẽ tiến hành phương thức auto arima để xác định 3 chỉ số này 1 lần nữa. Và kết quả 3 chỉ số này đã được thay đổi từ (0, 2, 3) thành (2, 2, 0).

```
> auto_arima(y_log, trace=TRUE)
Fitting models using approximations to speed things up...
ARIMA(2,2,2) : -629.659
ARIMA(0,2,0) : -516.3296
ARIMA(1,2,0) : -603.4216
ARIMA(0,2,1) : -585.3282
ARIMA(1,2,2) : -630.5308
ARIMA(0,2,2) : -590.3965
ARIMA(1,2,1) : -624.9986
ARIMA(1,2,3) : -628.917
ARIMA(0,2,3) : -591.2431
ARIMA(2,2,1) : -631.7693
ARIMA(2,2,0) : -632.2183
ARIMA(3,2,0) : -632.1702
ARIMA(3,2,1) : Inf

Now re-fitting the best model(s) without approximations...
ARIMA(2,2,0) : -617.3158
Best model: ARIMA(2,2,0)
```

Từ Hình 5.13, việc dự đoán trên mô hình ARIMA cho thấy dữ liệu dự đoán và dữ liệu thật không có quá nhiều sự chênh lệch. Điều này, cho thấy các kết luận và giả thiết ta đã đưa ra trong suốt quá trình phân tích là hoàn toàn chính xác.



Hình 5.13. Kết quả dự đoán mô hình ARIMA bằng R

CHƯƠNG 6. RECURRENT NEURAL NETWORK

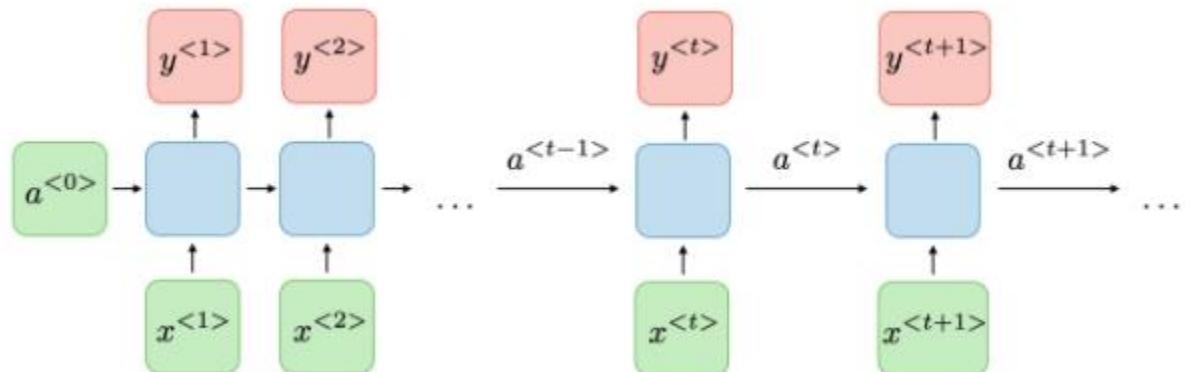
6.1. Cơ sở lý thuyết

6.1.1. Phương pháp máy học – học sâu (*Deep learning*)

Phương pháp máy học – học sâu là một nhánh con của các phương pháp máy học. Cụm tự học sâu tượng trưng cho việc nhiều lớp xử lý dữ liệu hơn so với các phương pháp máy học thông thường. Ta có thể nói một thuật toán học sâu được mô tả sơ lược sẽ gồm một bộ dữ liệu, hàm mất mát, một phương thức cải thiện mô hình và một mô hình bất kỳ.

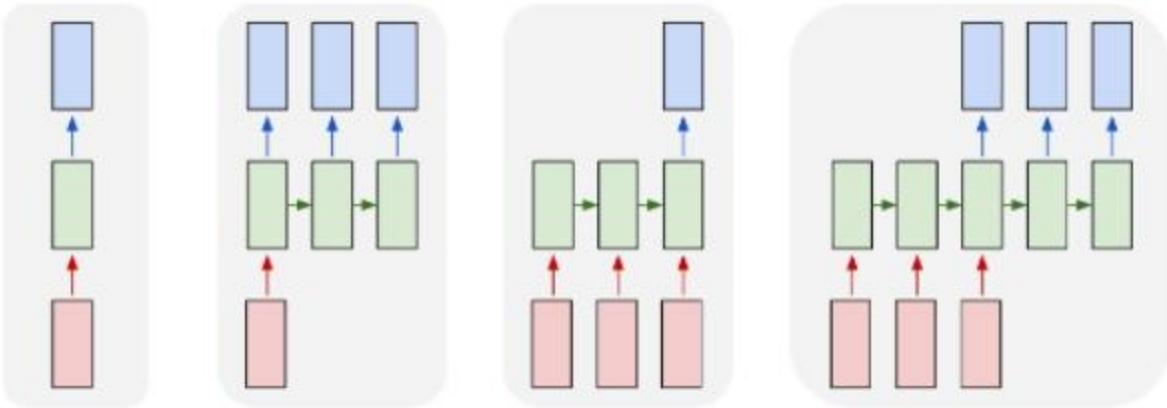
6.1.2. Mô hình RNNs (Recurrent Neural Networks)

Mô hình RNNs là một mô hình mạng nơ-tron cho phép các dữ liệu đầu ra được sử dụng như là dữ liệu đầu vào trong các lớp ẩn.



Hình 6.1. Mô hình RNNs

Mô hình RNNs có 4 dạng: 1 – 1, 1 – nhiều, nhiều – 1, nhiều -nhiều. Được thể hiện trong Hình 6.2.



Hình 6.2. Các dạng mô hình RNNs

6.1.3. Vấn đề bị mất độ dốc (Vanishing Gradient Problem)

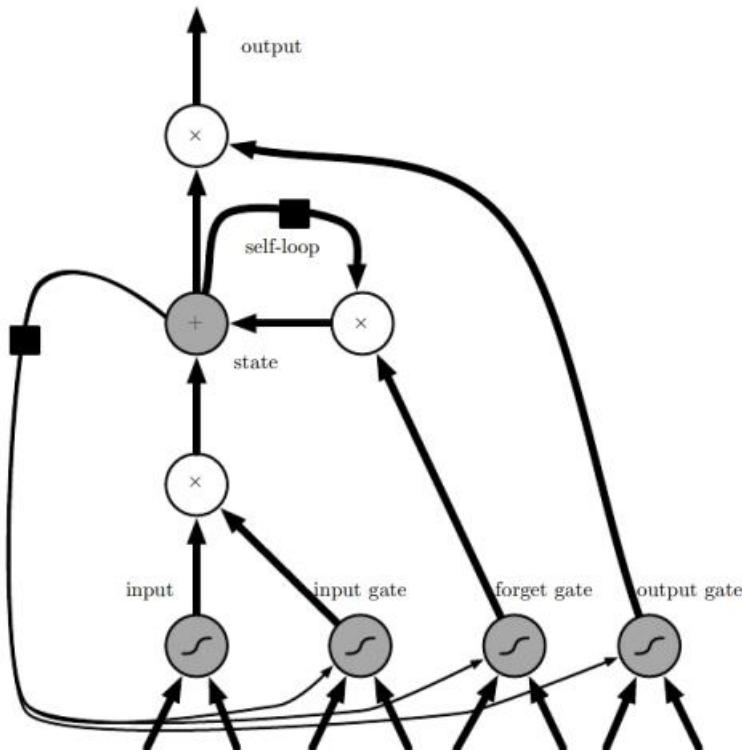
Vấn đề bị mất độ dốc có liên quan đến các thuật toán tính độ dốc của phương pháp học sâu. Thuật toán tính độ dốc sẽ kết hợp cũng với thuật toán lan truyền ngược để tính chỉ số weight trong mô hình mạng nơ-tron. Thế nhưng trong mô hình RNNs, thuật toán này lại được sử dụng để huấn luyện cho lớp tiếp theo liền kề. Và xảy ra vấn đề khi thuật toán lan truyền ngược cập nhật chỉ số weight.

6.1.4. Mô hình LSTM (Long Short-Term Memory)

Mô hình LSTM là một dạng đặc biệt của mô hình RNNs, được tạo nên để tránh các vấn đề phụ thuộc xa. Trong mô hình LSTM có 4 cổng: cổng cập nhật để so sánh với kết quả trước đó, cổng tương quan để loại bỏ giá trị trước đó, cổng quên lảng để quyết định có xóa dữ liệu đó hay không, và cổng xuất dữ liệu.

$$\Gamma = \sigma(W_x^{<t>} + U_a^{<t-1>} + b)$$

Với σ là hàm sigmoid và W, U và b là các hệ số tương ứng với các cổng.



Hình 6.3. Cách thức mô hình LSTM hoạt động tại 1 lớp

6.1.5. Phương pháp chính quy hóa Dropout (*Dropout Regularization*)

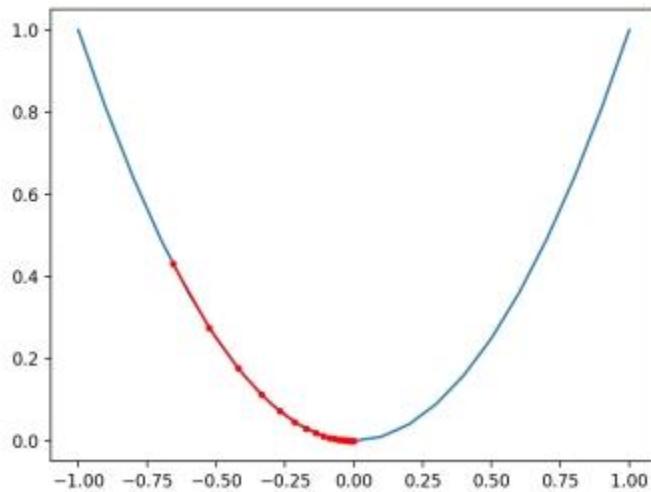
Một trong những vấn đề khi ta sử dụng mạng nơ-tron cho dữ liệu có kích thước nhỏ, là mô hình dự đoán có thể xảy ra tình trạng overfitting, dẫn đến việc khó khăn khi dự đoán các dữ liệu nằm ngoài tập huấn luyện. Vì vậy, chúng ta thực hiện phương pháp chính quy hóa Dropout, để loại bỏ ngẫu nhiên giá trị khi đưa dữ liệu qua các lớp để giảm tầm ảnh hưởng của các giá trị trong việc xây dựng mô hình.

Phương pháp Dropout có thể được sử dụng trong hầu như tất cả các lớp trong mạng nơ-tron, ngoại trừ lớp xuất giá trị. Thông thường, người ta sẽ loại bỏ xác suất 50% của dữ liệu đầu vào hoặc lớn hơn.

6.1.6. Thuật toán Adam

Chúng ta đã từng nghe qua thuật toán đồ dốc ngẫu nhiên khi tìm hiểu về mô hình hồi quy. Ở mô hình hồi quy, thuật toán này được sử dụng để tính toán tổng của R bình phương để vẽ đường hồi quy đi qua các điểm, nhưng sang mạng nơ-tron thuật toán này

được dùng để tính toán các hệ số weight. Còn thuật toán adam là một thuật toán tối ưu khác cũng được dùng để cập nhật các trị số weight trong mạng nơ-tron.



Hình 6.4. Thuật toán đồ dốc Gradient Descent

Điểm khác biệt của 2 thuật toán này là: thuật toán đồ dốc ngẫu nhiên, có hệ số tốc độ học tập gần như không đổi. Trong khi thuật toán adam sẽ điều chỉnh hệ số tốc độ học tập của mình dựa trên lũy thừa trung bình trượt và bình phương độ dốc.

6.2. Phân tích dự báo bằng Python

6.2.1. Giới thiệu bộ dữ liệu

Bộ dữ liệu “Google_Stock_Price.csv” là bộ dữ liệu về cổ phiếu của Google. Bộ dữ liệu này được chia thành 2 phần, phần train để xây dựng mô hình và phần test để kiểm tra độ chính xác của mô hình.

Link dữ liệu: <https://www.kaggle.com/medharawat/google-stock-price>

Bộ dữ liệu có 6 thuộc tính:

STT	Tên thuộc tính	Mô tả thuộc tính
1	Date	Ngày tháng năm
2	Open	Giá trị khi mở phiên giao dịch

3	High	Giá trị cao nhất
4	Low	Giá trị thấp nhất
5	Close	Giá trị khi kết thúc phiên giao dịch
6	Volume	Khối lượng giao dịch

Bảng 6.1. Các thuộc tính trong bộ dữ liệu “Google_Stock_Price.csv”

6.2.2. Cài đặt thư viện và Load dữ liệu

Để có thể xây dựng mô hình LSTM bên cạnh các thư viện thường sử dụng, ta cần cài đặt thêm bộ thư viện keras và thư viện tensorflow.

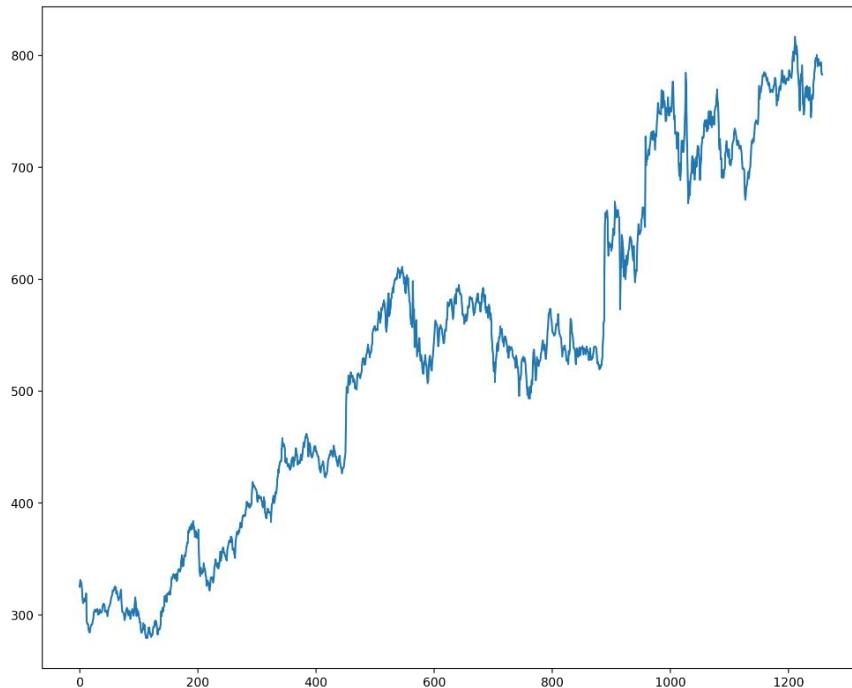
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense,Dropout,LSTM
```

Nhập file dữ liệu “Google_Stock_Price_Train.csv” để tiến hành xây dựng mô hình. Bởi vì chúng ta chỉ quan tâm giá trị khi mở phiên giao dịch nên tách ra khỏi bộ dữ liệu.

```
train = pd.read_csv('./data/Google_Stock_Price_Train.csv')
y = train.iloc[:,1:2].values
```

6.2.3. Khám phá dữ liệu

Hình 6.5 cho thấy giá trị cổ phiếu khi mở phiên giao dịch có sự thay đổi theo thời gian. Điều này cũng tương đồng với thời điểm Google ra mắt nhiều sản phẩm mới trên thị trường.



Hình 6.5. Giá trị của cổ phiếu khi mở phiên giao dịch từ năm 2012 đến năm 2016

Để dễ dàng trong việc xây dựng mô hình, ta sẽ tiến hành chuẩn hóa giá trị dữ liệu trên thang từ 0 tới 1. Việc chuẩn hóa được thực hiện bằng phương thức MinMaxScaler.

```
sc = MinMaxScaler(feature_range=(0,1))
y_scaled = sc.fit_transform(y)
```

6.2.4. Xây dựng mô hình LSTM

Đối với mô hình RNNs, để xây dựng mô hình ra cần phải chuẩn bị cả dữ liệu đầu vào và đầu ra. Dữ liệu đầu vào sẽ có dạng bảng có chiều rộng là 60, và một dãy dữ liệu đầu ra, tương ứng với mô hình RNNS dạng nhiều mảng.

```
X_train = []
y_train = []
for i in range(60, 1258):
    X_train.append(y_scaled[i-60:i,0])
    y_train.append(y_scaled[i,0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

Ta sử dụng phương thức Sequential để khởi tạo việc xây dựng mô hình. Sau đó, ta tiến hành xây dựng lớp ẩn đầu tiên của mô hình với dạng dữ liệu đầu ra là bảng 50 dòng, giữ nguyên chuỗi giá trị và dạng dữ liệu đầu vào. Và sử dụng chính quy hóa để loại bỏ 20% dữ liệu đầu vào để tránh tình trạng việc dự đoán quá khớp.

```
LSTM_model = Sequential()  
  
LSTM_model.add(LSTM(units = 50, return_sequences = True,  
                     input_shape = (X_train.shape[1], 1)))  
LSTM_model.add(Dropout(0.2))
```

Tiếp theo ta xây dựng 3 lớp ẩn còn lại tương tự như lớp đầu tiên.

```
LSTM_model.add(LSTM(units = 50, return_sequences = True))  
LSTM_model.add(Dropout(0.2))
```

```
LSTM_model.add(LSTM(units = 50, return_sequences = True))  
LSTM_model.add(Dropout(0.2))
```

```
LSTM_model.add(LSTM(units = 50))  
LSTM_model.add(Dropout(0.2))
```

Cuối cùng ta xây dựng lớp xuất dữ liệu với 1 dòng dữ liệu đầu ra. Sau khi xây dựng xong các lớp của mô hình LSTM, ta sẽ cho dữ liệu huấn luyện vào và bắt đầu hoàn thiện mô hình.

Ta sẽ cải thiện mô hình bằng thuật toán Adam, và hàm thuật toán tính theo trung bình khoảng cách sai số. Và tiến hành đổ dữ liệu đầu vào, dữ liệu mong muốn, số lần huấn luyện mô hình, và số lần cập nhật.

```
LSTM_model.add(Dense(units = 1))  
LSTM_model.compile(optimizer='adam', loss='mean_squared_error')  
LSTM_model.fit(X_train,y_train,epochs=100, batch_size=32)
```

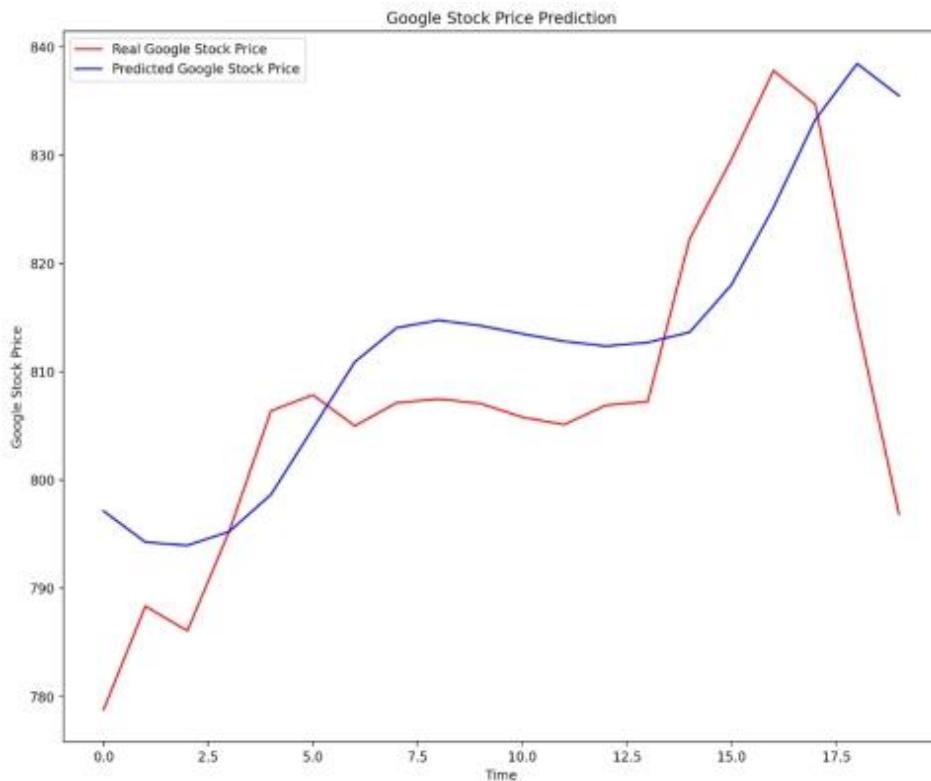
6.2.5. Tiến hành dự đoán

Hoàn thành việc xây dựng mô hình, ta tiến hành nhập bộ dữ liệu kiểm tra và tiến hành các bước chuẩn hóa, xây dựng bảng dữ liệu đầu vào tương tự như bộ dữ liệu huấn luyện trước đó. Tiếp theo ta đổ dữ liệu kiểm tra sau khi đã biến đổi vào mô hình để tiến hành dự đoán. Lưu ý, để tiện cho mô họa, ta sẽ trả dữ liệu dự đoán về dạng ban đầu trước khi chuẩn hóa.

```
test = pd.read_csv('./data/Google_Stock_Price_Test.csv')
real_stock_price = test.iloc[:, 1:2].values

total = pd.concat((train['Open'], test['Open']), axis = 0)
inputs = total[len(total) - len(test) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 80):
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = LSTM_model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

Hình 6.6 cho ta thấy dạng dữ liệu dự đoán gần giống với dạng dữ liệu thật mặc dù có độ chênh lệch giữa 2 giá trị khá đáng kể. Điều này, hoàn toàn nằm trong dự tính do khi xây dựng mô hình ta cố gắng để nó không xảy ra tình trạng overfitting.



Hình 6.6. So sánh kết quả dự đoán và dữ liệu thật

6.3. Phân tích dự báo bằng R

6.3.1. Cài đặt thư viện

Trước khi tiến hành phân tích dữ liệu bằng các phương pháp máy học – học sâu, ta cần sử dụng bộ thư viện keras và tensorflow. Cách hoạt động của cả 2 thư viện tương tự như ở ngôn ngữ Python, vì vậy cần kiểm tra kỹ càng, mình đã hoàn thành việc cài đặt hay chưa. Trong khi đó thư viện scales và dplyr được dùng để xử lý và chuẩn hóa bộ dữ liệu.

```
#import library
library(keras)
library(tensorflow)
library(scales)
library(dplyr)

#Load train data
train = read.csv("./data/Google_Stock_Price_Train.csv")
```

6.3.2. Chuẩn hóa dữ liệu

Để có thể hỗ trợ cho mô hình dự đoán, ta cũng cần chuẩn hóa bộ dữ liệu. Ở ngôn ngữ R, ta sử dụng giá trị trung bình và độ lệch chuẩn để chuẩn hóa bộ dữ liệu.

```
#Normalize data
scale = c(mean(train$open),sd(train$open))
y = (train$open - scale[1])/scale[2]
y = data.matrix(y)
```

6.3.3. Xây dựng mô hình LSTM

Cấu trúc dữ liệu đầu vào và đầu ra khi dùng ngôn ngữ R vẫn được giữ nguyên.

```
#Create data structure
x_train = t(sapply(1:(length(y)-60),
                    function(x) y[x:(x+60-1),1]))
x_train = array(data = as.numeric(unlist(x_train)),
                dim = c(nrow(x_train),ncol(x_train),1))

y_train = t(sapply((1 + 60):(length(y)),
                    function(x) y[x]))
y_train = array(data = as.numeric(unlist(y_train)))
```

Khác với ngôn ngữ Python, khi xây dựng các lớp của mô hình LSTM, ta không có một phương thức add, thay vào đó ta sử dụng toán tử pipe line để lồng các lớp vào mô hình.

```
#Build LSTM model
LSTM = keras_model_sequential() %>%
  layer_lstm(units = 50, return_sequences = TRUE, input_shape =c(60,1)) %>%
  layer_dropout(rate = 0.2) %>%
  layer_lstm(units = 50, return_sequences = TRUE) %>%
  layer_dropout(rate=0.2) %>%
  layer_lstm(units = 50, return_sequences = TRUE)%>%
  layer_dropout(rate=0.2)%>%
  layer_lstm(units = 50)%>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units = 1)
compile(LSTM,optimizer = 'adam',loss='mean_squared_error')
LSTM %>% fit(x_train,y_train,epochs = 100,batch_size = 32)
```

6.3.4. Tiến hành dự đoán

Hoàn thành việc xây dựng mô hình, ta sẽ nhập bộ dữ liệu test và cấu trúc lại bộ dữ liệu này cho phù hợp với yêu cầu cấu trúc dữ liệu đầu vào của mô hình để tiến hành dữ đoán. Sau đó ta sẽ vẽ biểu đồ để so sánh kết quả.

```

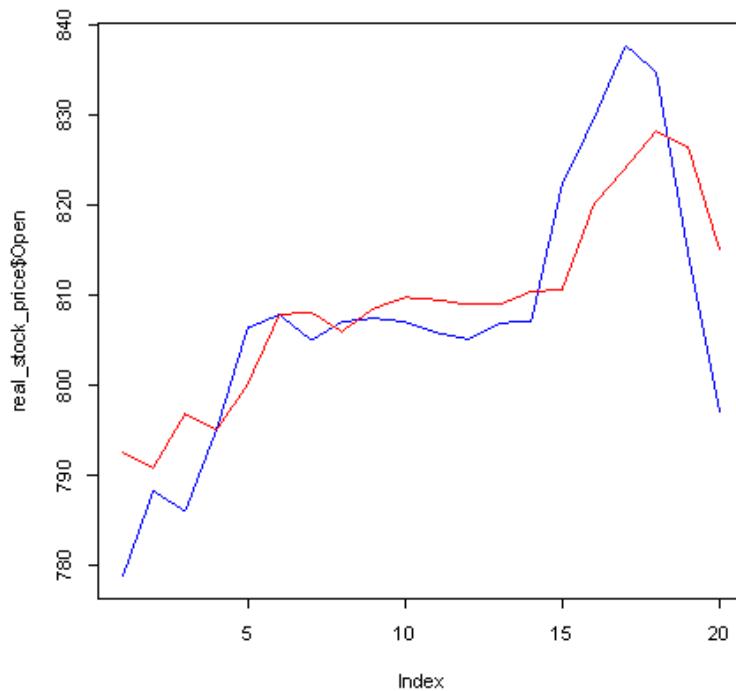
# import text data
test = read.csv("./data/Google_Stock_Price_Test.csv")
real_stock_price = test[2]
total = rbind(train[2],test[2])
total = (total-scale[1])/scale[2]
inputs = total[(lengths(total)-lengths(test[2])-60+1):lengths(total),]
inputs = data.matrix(inputs)
x_test = t(sapply(1:(length(inputs)-60),
                  function(x) inputs[x:(x+60-1),1]))
x_test = array(data = as.numeric(unlist(x_test)),
               dim = c(nrow(x_test),ncol(x_test),1))

#visualize the prediction

predict_stock_price = predict(LSTM,x_test,batch_size = 32)
predict_stock_price = predict_stock_price*scale[2]+scale[1]
plot(real_stock_price$Open,type='l',col="blue")
lines(predict_stock_price,col="red")

```

Khi so sánh giữa kết quả của Hình 6.7 khi phân tích bằng ngôn ngữ R và Hình 6.6 khi phân tích bằng Python, ta thấy kết quả có sự thay đổi rõ rệt. Nguyên do có thể là bởi vì trong suốt quá trình phân tích dữ liệu ta sử dụng phương pháp chuẩn hóa dữ liệu khác so với khi phân tích trên Python.



Hình 6.7. Kết quả dự đoán mô hình LSTM trên R

CHƯƠNG 7. TỔNG QUAN CÁC MÔ HÌNH MÁY HỌC

7.1. Mô hình Decision Tree

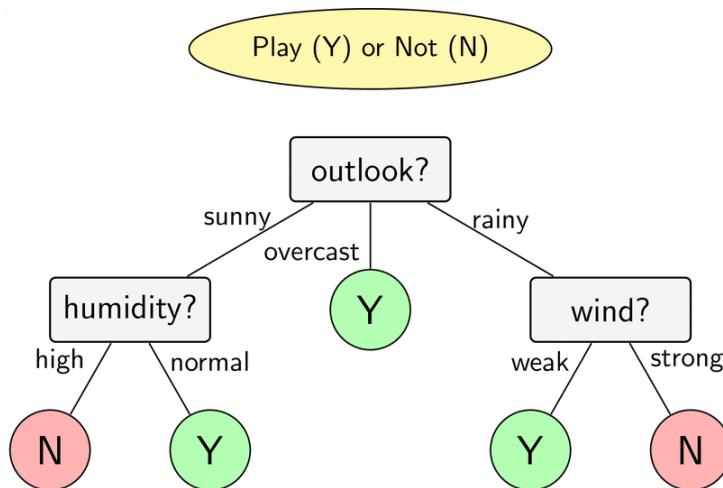
7.1.1. Khái niệm

Decision Tree là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (*Binary*), Định danh (*Nominal*), Thứ bậc (*Ordinal*), Số lượng (*Quantitative*) trong khi đó kết quả hay biến mục tiêu của Decision Tree chủ yếu là biến phân loại như biến Nhị phân hoặc biến Thứ bậc.

Decision Tree là một thuật toán thuộc phương pháp học có giám sát (*Supervised Learning*), có thể được áp dụng vào cả hai bài toán phân loại (*Classification*) và hồi quy (*Regression*). Tuy nhiên bài toán phân loại được sử dụng phổ biến hơn. Việc xây dựng một mô hình cây quyết định dựa trên tập dữ liệu cho trước là việc xác định các câu hỏi và thứ tự của chúng, để đánh giá xác suất đưa ra quyết định đúng đắn.

Lấy một ví dụ điển hình về Decision Tree, giả sử dựa theo thời tiết mà các bạn nam sẽ quyết định đi đá bóng hay không? Những đặc điểm ban đầu gồm: Thời tiết, Độ ẩm, Gió.

Dựa vào những thông tin trên, ta có thể xây dựng được mô hình như sau:



Hình 7.1. Minh họa mô hình Decision Tree

Từ mô hình trên, ta thấy: Nếu trời nắng, độ ẩm bình thường thì khả năng các bạn nam đi đá bóng sẽ cao. Còn nếu trời nắng, độ ẩm cao thì khả năng các bạn nam sẽ không đi đá bóng. Mỗi giá trị trong một biến độc lập đều có thể tác động lên quyết định sau cùng tạo nên sự liên kết, hay mối liên hệ với biến mục tiêu.

7.1.2. Thuật toán ID3

Thuật toán Iterative Dichotomiser 3 (ID3) được phát triển nhà khoa học Ross Quynlan và được công bố vào cuối thập niên 70 của thế kỷ 20. Đây là thuật toán nổi tiếng để xây dựng Decision Tree, áp dụng cho các bài toán phân loại mà tất cả các thuộc tính đều ở dạng categorical. Trong thuật toán ID3, các thuộc tính được đánh giá dựa trên Hàm số Entropy và Độ lợi thông tin (*Information Gain*).

7.1.2.1. Hàm số Entropy

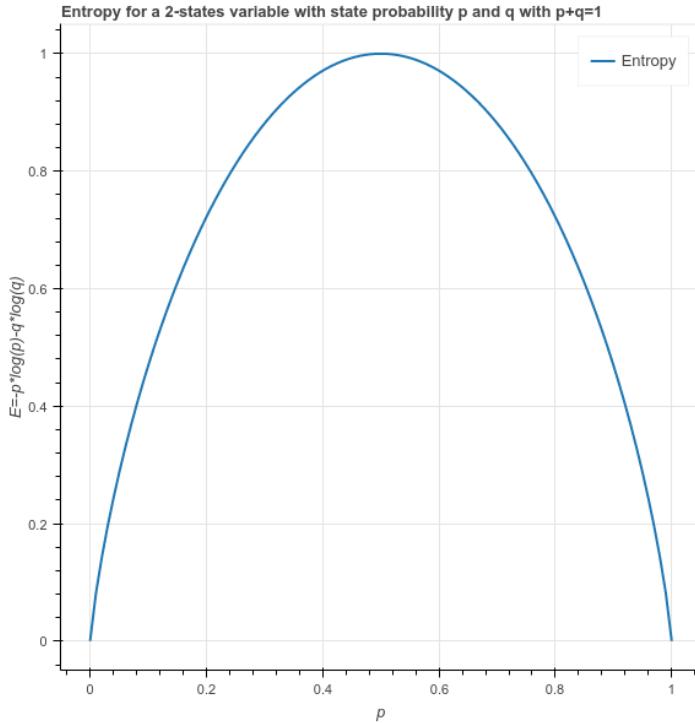
Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x_1, x_2, \dots, x_n . Giả sử rằng xác suất để x nhận các giá trị này $p_i = p(x = x_i)$ với $0 \leq p_i \leq 1$. Ký hiệu của phân phối này là $p = (p_1, p_2, \dots, p_n)$.

Entropy của phân phối này là:

$$H(p) = - \sum_{i=1}^n p_i \log_2(p_i)$$

Giả sử ta tung một đồng xu với 2 mặt sấp và ngửa. Gọi p và $q = 1 - p$ lần lượt là xác suất của 2 mặt đồng xu. Công thức Entropy được tính như sau:

$$H = -p \log_2(p) - q \log_2(q)$$



Hình 7.2. Đồ thị hàm Entropy với $n = 2$

Từ đồ thị Hình 6.2, ta có thể thấy hàm Entropy bằng 0 khi giá trị $p_i = 0$ hoặc $p_i = 1$, trong trường hợp này p là tinh khiết nhất. Nói cách khác, hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi nút con (*child node*) nằm trong cùng một lớp.

Khi cả hai giá trị p_i cùng bằng 0.5 thì hàm Entropy đạt giá trị cực đại. Lúc này p vẫn đúc nhất, hay mỗi nút con có chứa dữ liệu thuộc nhiều lớp khác nhau.

7.1.2.2. Information Gain

Information Gain còn được hiểu là “độ lợi thông tin”, dùng để đánh giá các thuộc tính nhằm xác định thuộc tính nào đem lại nhiều thông tin hữu ích nhất, đầy đủ cơ sở nhất cho mô hình để chúng ta phân loại đối tượng theo các giá trị, các lớp có sẵn của biến mục tiêu.

Information Gain dựa trên sự giảm của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính. Bài toán của chúng ta trở thành, tại mỗi tầng của cây quyết định cần chọn thuộc tính nào có độ giảm Entropy là thấp nhất để thu được Information Gain cao nhất.

Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

- **Bước 1:** Tính toán hệ số Entropy của tập dữ liệu S có N phần tử với N_i phần tử thuộc lớp c cho trước:

$$H(S) = - \sum_{i=1}^c \frac{N_i}{N} \log_2 \left(\frac{N_i}{N} \right)$$

- **Bước 2:** Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node S_1, S_2, \dots, S_K với số điểm trong mỗi child node lần lượt là m_1, m_2, \dots, m_K , ta có:

$$H(x, S) = - \sum_{k=1}^K \frac{m_k}{N} H(S_k)$$

- **Bước 3:** Chỉ số Information Gain được tính bằng:

$$Gain(x, S) = H(S) - H(x, S)$$

7.1.3. Thuật toán C.45

Thuật toán C4.5 là thuật toán mở rộng của thuật toán ID3 của nhà nghiên cứu khoa học máy tính Ross Quynlan, ra đời vào năm 1986. ID3 cũng sử dụng công thức Entropy và Information Gain để xây dựng mô hình cây quyết định có thể phân nhánh hơn 2 nhánh, tuy nhiên ID3 lại ưu tiên những thuộc tính có số lượng lớn các giá trị mà ít xét tới những giá trị nhỏ hơn. Do đó mặc dù ID3 là thuật toán cây quyết định ra đời từ rất lâu nhưng không còn được sử dụng phổ biến cũng chính vì lí do trên.

C4.5 được xem là phiên bản nâng cấp của ID3 với khả năng xử lý được cả dữ liệu định lượng dạng liên tục (*Continuous data*) và cả dữ liệu định tính, và là thuật toán Decision Tree tiêu biểu. C4.5 sử dụng thêm công thức Gain Ratio để khắc phục các khuyết điểm của Information Gain trong việc lựa chọn cách phân nhánh tối ưu.

Gain Ratio được tính bằng công thức:

$$GainRatio = \frac{Gain}{SplitInfo}$$

Với SplitInfo có công thức như sau:

$$SplitInfo = - \sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Splitinfo chính là “thông tin có được” về số lượng đối tượng dữ liệu trong mỗi nhánh. Ví dụ chúng ta có 15 khách hàng với ID khác nhau tức có thể có 15 nhánh mỗi nhánh có 1 đối tượng dữ liệu, và mỗi khách hàng có thể mang lại rủi ro tín dụng hoặc không mang lại rủi ro tín dụng.

7.1.4. Thuật toán CART

Thuật toán CART (*Classification and Regression Tree*) là một kĩ thuật học máy có giám sát phổ biến được sử dụng cho cả hai vấn đề phân loại và hồi quy. CART thường sử dụng phương pháp Gini để tạo các điểm phân chia.

Chỉ số Gini (*Gini Index*) là chỉ số đo lường mức độ đồng nhất hay nhiễu loạn của thông tin, hay sự khác biệt về các giá trị mà mỗi điểm dữ liệu trong một tập con, hay một nhánh của cây quyết định. Nói một cách khác, chỉ số Gini là phương pháp hướng đến đo lường tần suất của một đối tượng dữ liệu ngẫu nhiên trong tập dữ liệu ban đầu được phân loại không chính xác

Công thức tổng quát của hệ số Gini:

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

t là một node bất kỳ có chứa các điểm dữ liệu mang thuộc tính j của biến mục tiêu và p chính là xác suất để một điểm dữ liệu trong t có thuộc tính j, nếu tất cả các điểm dữ liệu đều mang thuộc tính j thì lúc này p = 1 và Gini(t) sẽ bằng 0, node t chính thức được công nhận là “pure” node.

Công thức trên để tính độ đồng nhất của một node, vậy khi chúng ta có nhiều cách phân nhánh, mỗi cách có thể phân ra một số node nhất định, tức có thể chia tập dữ liệu thành các tập con khác nhau theo các giá trị của biến dữ liệu, lúc này chúng ta có thêm công thức thứ 2 để tìm ra cách “split” tối ưu nhất.

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(t)$$

Với n_i là số quan sát, số điểm dữ liệu có trong “child node”, node của nhánh được phân, còn n là số quan sát, số điểm dữ liệu có trong “parent node”, node được dùng để phân nhánh. Hệ số Gini split càng nhỏ tức cách phân nhánh càng tối ưu.

7.1.5. Ưu và nhược điểm

7.1.5.1. Ưu điểm

- Thuật toán Decision Tree đơn giản, trực quan, không quá phức tạp để hiểu ngay lần đầu tiên. Đồng thời bộ dữ liệu training không nhất thiết phải quá lớn để tiến hành xây dựng mô hình phân tích.
- Một số thuật toán Decision Tree có khả năng xử lý dữ liệu bị trống và dữ liệu bị lỗi mà không cần áp dụng chuẩn hóa hay tạo biến giả. Bên cạnh đó Decision Tree ít bị ảnh hưởng bởi các dữ liệu ngoại lệ (outliers).
- Thuật toán Decision Tree mang lại kết quả dự báo có độ chính xác cao, dễ dàng thực hiện, nhanh chóng trong việc huấn luyện, không cần phải chuyển đổi các biến vì kết quả sẽ như nhau với bất kể loại biến dữ liệu biến đổi ra sao.
- Thuật toán Decision Tree vẫn nói lên được mối liên hệ giữa các biến, các thuộc tính dữ liệu một cách trực quan nhất mặc dù không thể hiện được rõ mối quan hệ tuyến tính, hay mức độ liên hệ giữa chúng như phương pháp phân tích hồi quy có được.

7.1.5.2. Nhược điểm

- Thuật toán Decision Tree khi được áp dụng với bộ dữ liệu phức tạp, nhiều biến và thuộc tính khác nhau có thể dẫn đến mô hình bị overfitting, quá khớp với dữ liệu

train dẫn đến vấn đề không đưa ra kết quả phân loại chính xác khi áp dụng cho dữ liệu test, và dữ liệu mới.

- Đối với thuật toán Decision Tree khi có sự thay đổi nhỏ trong bộ dữ liệu có thể gây ảnh hưởng đến cấu trúc của mô hình. Nghĩa là khi chúng ta điều chỉnh dữ liệu, cách thức phân nhánh, ngắt cây sẽ bị thay đổi, có thể dẫn đến kết quả sẽ khác so với ban đầu, phức tạp hơn.
- Thuật toán Decision Tree yêu cầu bộ dữ liệu train và test phải được cân đối theo các lớp, các nhóm trong biến mục tiêu, ví dụ có sự chênh lệch không quá lớn giữa số lượng đối tượng dữ liệu thuộc lớp A của biến mục tiêu và số lượng đối tượng dữ liệu thuộc lớp B của biến mục tiêu.

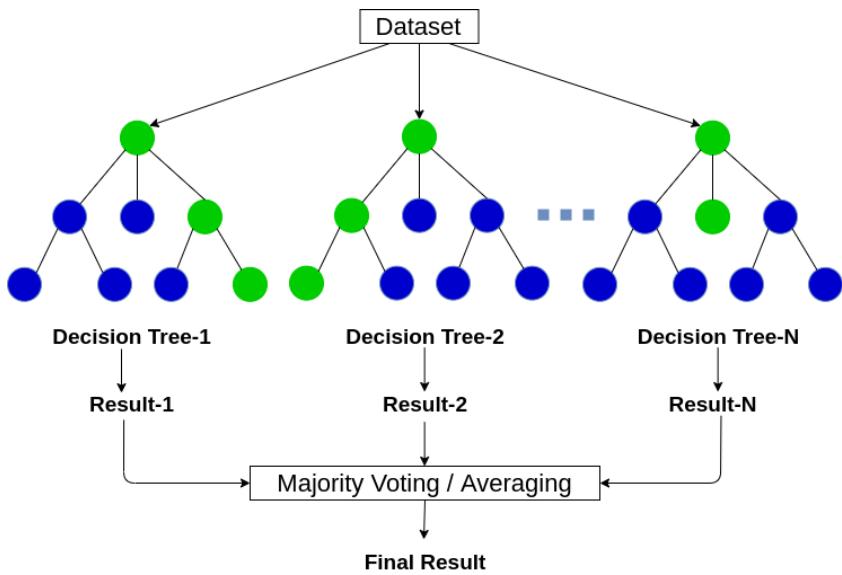
7.2. Mô hình Random Forest

7.2.1. Khái niệm

Random Forest - Rừng ngẫu nhiên là một thuật toán học có giám sát. Như tên gọi của nó, Random Forest sử dụng các cây để làm nền tảng. Nó là một phương pháp tổng hợp của các cây quyết định (*Decision Tree*) được tạo ra trên một tập dữ liệu được chia ngẫu nhiên. Bộ sưu tập phân loại cây quyết định này còn được gọi là rừng. Mỗi cây quyết định dự đoán một kết quả và kết quả nào được nhiều cây quyết định dự đoán thì đó là kết quả cuối cùng.

7.2.2. Thuật toán máy học

- Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
- Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
- Tính toán số lượng vote trên toàn bộ Forest cho từng kết quả.
- -Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Hình 7.3. Thuật toán máy học của Random Forest

7.2.3. Ưu và nhược điểm

7.2.3.1. Ưu điểm

Thuật toán Random Forest giải quyết tốt các bài toán có nhiều dữ liệu nhiễu, thiếu giá trị. Do cách chọn ngẫu nhiên thuộc tính nên các giá trị nhiễu, thiếu ảnh hưởng không lớn đến kết quả. Khi rừng có nhiều cây hơn, chúng ta có thể tránh được việc overfitting với tập dữ liệu. Ngoài ra, thuật toán có thể được sử dụng trong cả bài toán phân loại và hồi quy.

7.2.3.2. Nhược điểm

Dữ liệu huấn luyện cần được đa dạng hóa và cân bằng về số nhãn lớp. Việc không cân bằng giữa các lớp khiến kết quả dự đoán của thuật toán có thể lệch về số nhãn lớp chiếm ưu thế. Thời gian huấn luyện của rừng có thể kéo dài tùy số cây và số thuộc tính phân chia.

7.3. Mô hình Neural Network

7.3.1. Khái niệm

Neural Network đọc tiếng việt là Mạng nơ-ron nhân tạo, đây là một chuỗi những thuật toán được đưa ra để tìm kiếm các mối quan hệ cơ bản trong tập hợp các dữ liệu. Thông qua việc bắt bước cách thức hoạt động từ não bộ con người. Neural Network có khả

năng thích ứng được với mọi thay đổi từ đầu vào. Do vậy, nó có thể đưa ra được mọi kết quả một cách tốt nhất có thể mà bạn không cần phải thiết kế lại những tiêu chí đầu ra.

7.3.2. Đặc điểm

Mạng nơ ron nhân tạo có thể hoạt động như mạng nơ ron của con người. Mỗi một nơ ron thần kinh trong nơ ron nhân tạo là hàm toán học với chức năng thu thập và phân loại các thông tin dựa theo cấu trúc cụ thể.

Neural Network có sự tương đồng chuẩn mực với những phương pháp thống kê như đồ thị đường cong và phân tích hồi quy. Neural Network có chứa những lớp bao hàm các nút được liên kết lại với nhau. Mỗi nút lại là một tri giác có cấu tạo tương tự với hàm hồi quy đa tuyến tính.

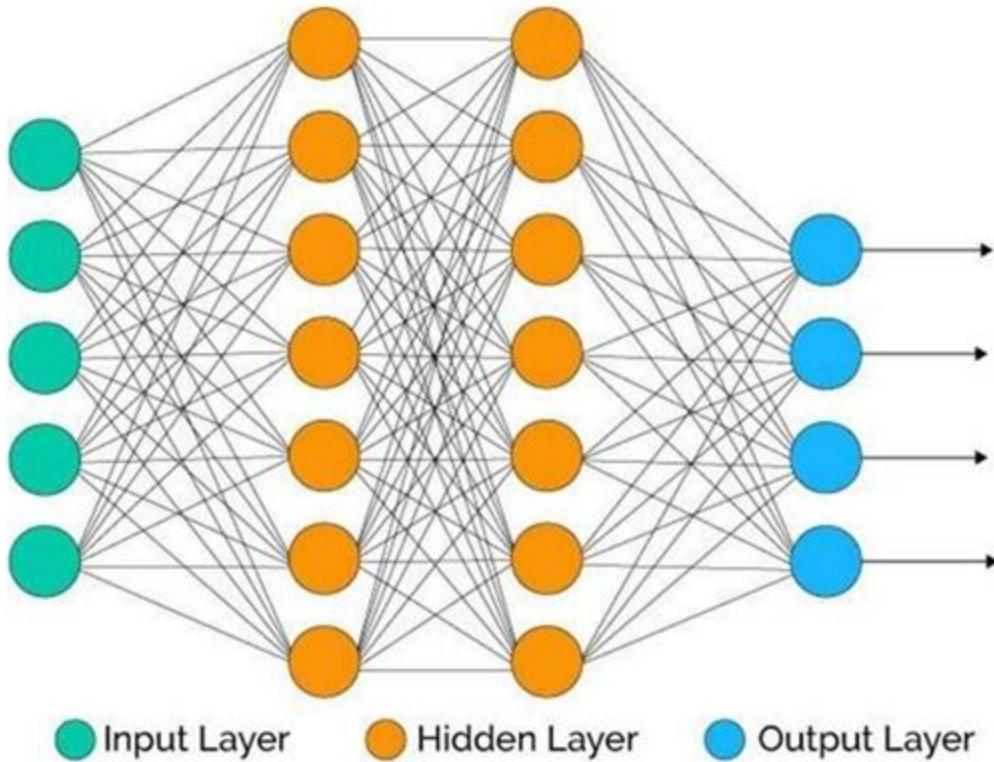
Bên trong một lớp tri giác đa lớp, chúng sẽ được sắp xếp dựa theo các lớp liên kết với nhau. Lớp đầu vào sẽ thu thập các mẫu đầu vào và lớp đầu ra sẽ thu nhận các phân loại hoặc tín hiệu đầu ra mà các mẫu đầu vào có thể phản ánh lại.

7.3.3. Kiến trúc Neural Network

Mạng Neural Network là sự kết hợp của những tầng perceptron hay còn gọi là perceptron đa tầng. Và mỗi một mạng Neural Network thường bao gồm 3 kiểu tầng là:

- Tầng input layer (tầng vào): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
- Tầng output layer (tầng ra): Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.
- Tầng hidden layer (tầng ẩn): Tầng này nằm giữa tầng vào và tầng ra nó thể hiện cho quá trình suy luận logic của mạng.

Lưu ý: Mỗi một Neural Network chỉ có duy nhất một tầng vào và 1 tầng ra nhưng lại có rất nhiều tầng ẩn.



Hình 7.4. Kiến trúc mạng Neural Network

Với mạng Neural Network thì mỗi nút mạng là một sigmoid nơ-ron nhưng chúng lại có hàm kích hoạt khác nhau. Thực tế, người ta thường sử dụng có cùng loại với nhau để việc tính toán thuận lợi hơn.

Tại mỗi tầng, số lượng nút mạng có thể khác nhau còn tùy vào bài toán hoặc cách giải quyết. Tuy nhiên, khi làm việc người ta sẽ để các tầng ăn số với số lượng nơ-ron khác nhau. Ngoài ra, những nơ-ron nằm ở tầng thường sẽ liên kết đôi với nhau để tạo thành mạng kết nối dày đủ nhất. Khi đó, người dùng có thể tính toán được kích cỡ của mạng dựa vào tầng và số lượng nơ-ron.

7.3.4. Quy tắc chuỗi (Chain Rule)

Quy tắc chuỗi (Chain rule) tính toán bất kỳ thành phần nào của phân phối chung của một tập hợp các biến ngẫu nhiên chỉ sử dụng xác suất có điều kiện. Lý thuyết xác suất này được sử dụng làm nền tảng cho việc Lan truyền ngược và trong việc tạo ra các mạng Bayes.

Chuỗi xác suất và biến ngẫu nhiên đơn giản này được biểu thị như sau:

$$P(A, B) = P(B|A) P(A)$$

Một ví dụ đơn giản minh họa cách thức hoạt động của Chain rule như sau: Xác suất chọn vé xổ số trúng thưởng từ các nón khác nhau. Nón 1 có 1 vé thắng và 2 vé thua bên trong. Nón 2 có 1 vé trúng thưởng và 3 vé trúng không.

- Lần đầu ta chọn một chiếc mũ ngẫu nhiên sẽ là sự kiện A. Tỷ lệ bạn chọn chiếc mũ 1 là: $P(A) = P(\sim A) = \frac{1}{2}$
- Qua sự kiện B là cơ hội giành được vé trúng thưởng. Trong trường hợp chiếc mũ thứ nhất, cơ hội lấy được chiếc vé trúng thưởng là: $P(B|A) = \frac{1}{3}$

Sự kiện “A, B” là giao điểm của việc chọn mũ 1 và vé trúng thưởng. Sử dụng quy tắc chuỗi về xác suất cho thấy rằng tỷ lệ bạn có được tấm vé chiến thắng với Sự kiện A là 16,5% (33% x 50%).

7.3.5. Thuật toán Gradient Descent

Gradient Descent là một thuật toán tối ưu lặp (*iterative optimization algorithm*) được sử dụng trong các bài toán Machine Learning và Deep Learning (thường là các bài toán tối ưu lồi — Convex Optimization) với mục tiêu là tìm một tập các biến nội tại (*internal parameters*) cho việc tối ưu models. Trong đó:

- Gradient: là tỷ lệ độ nghiêng của đường dốc (rate of inclination or declination of a slope). Về mặt toán học, Gradient của một hàm số là đạo hàm của hàm số đó tương ứng với mỗi biến của hàm. Đối với hàm số đơn biến, chúng ta sử dụng khái niệm Derivative thay cho Gradient.
- Descent: là từ viết tắt của descending, nghĩa là giảm dần.

Gradient Descent có nhiều dạng khác nhau như Stochastic Gradient Descent (SGD), Mini-batch SDG. Nhưng về cơ bản thì đều được thực thi như sau:

- Khởi tạo biến nội tại.

- Đánh giá model dựa vào biến nội tại và hàm mất mát (Loss function).
- Cập nhật các biến nội tại theo hướng tối ưu hàm mất mát (finding optimal points).
- Lặp lại bước 2, 3 cho tới khi thỏa điều kiện dừng.

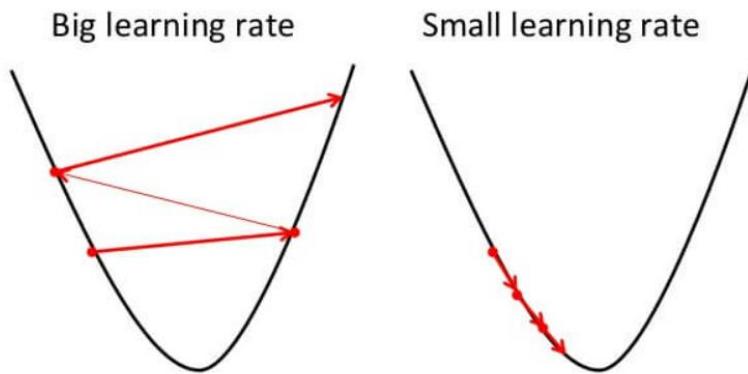
Công thức tổng quát của Gradient Descent được viết như sau:

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_{\theta} J(\theta^{(k)})$$

trong đó θ là tập các biến cần cập nhật, η là tốc độ học (*learning rate*), $\eta \nabla_{\theta} J(\theta)$ là Gradient của hàm mất mát f theo tập θ .

Việc chọn η có ý nghĩa rất lớn trong phương pháp này vì nó quyết định tới tính sống còn của giải thuật. Nếu η quá lớn thì ta không hội tụ được về đích, nhưng nếu η quá nhỏ thì ta lại mất rất nhiều thời gian để chạy giải thuật này.

Gradient Descent



Hình 7.5. Ảnh hưởng của độ học

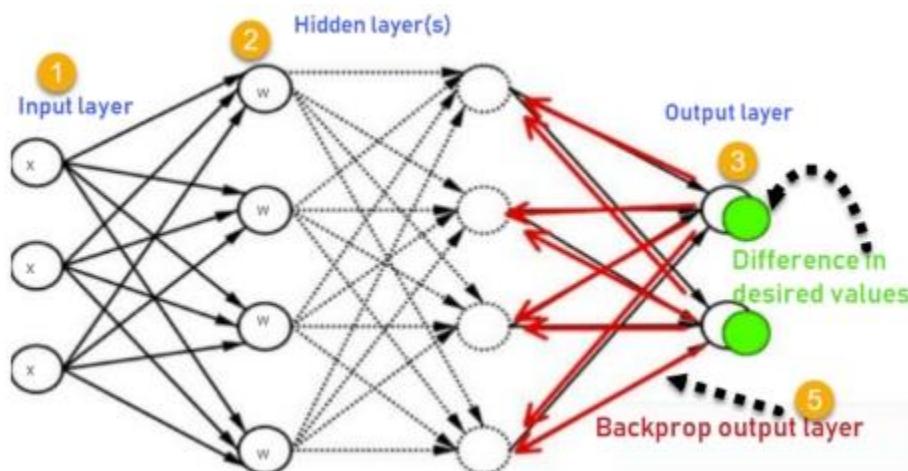
7.3.6. Thuật toán Lan truyền ngược (*Backpropagation*)

Thuật toán Lan truyền ngược sử dụng một tập hợp các giá trị đầu vào và đầu ra để tìm ra mạng nơron thần kinh mong muốn. Một tập hợp đầu vào được đưa vào một hệ thống giả định trước đó để tính ra giá trị đầu ra, sau đó giá trị đầu ra này được so sánh với giá trị giá trị thực đo. Nếu không có sự khác biệt nào, thì không cần thực hiện một quá trình

kiểm tra nào, ngược lại các trọng số sẽ được thay đổi trong quá trình lan truyền ngược trong mạng thần kinh để giảm sự khác biệt đó.

Lan truyền ngược trong mạng nơ-ron là một dạng viết tắt của “Lan truyền ngược của các lỗi”. Đây là một phương pháp tiêu chuẩn để đào tạo mạng nơ-ron nhân tạo. Phương pháp này giúp tính toán gradient của một hàm mất mát đối với tất cả các trọng số trong mạng.

Thuật toán Lan truyền ngược trong mạng nơron tính toán độ dốc của hàm mất mát cho một trọng số theo quy tắc chuỗi. Thuật toán này tính toán hiệu quả từng lớp một, không giống như tính toán trực tiếp gốc. Thuật toán này tính toán gradient, nhưng nó không xác định cách sử dụng gradient. Nó tổng quát hóa việc tính toán trong quy tắc delta.



Hình 7.6. Mô tả cách thức hoạt động của thuật toán Backpropagation

- Bước 1: Các biến đầu vào x , đến qua đường dẫn được kết nối trước.
- Bước 2: Biến đầu vào được mô hình hóa bằng cách sử dụng trọng số thực w . Các trọng số thường được chọn ngẫu nhiên
- Bước 3: Tính toán đầu ra cho mọi nơ-ron từ lớp đầu vào, đến các lớp ẩn, đến lớp đầu ra.
- Bước 4: Tính toán sai số trong kết quả đầu ra, bằng giá trị đầu ra thực trừ đi giá trị đầu ra mong muốn.

- Bước 5: Đi ngược lại từ lớp đầu ra đến lớp ẩn để điều chỉnh trọng số sao cho giảm sai số.
- Tiếp tục lặp lại quy trình cho đến khi đạt được đầu ra mong muốn.

7.3.7. Activation function

Activation functions là những hàm phi tuyến được áp dụng vào đầu ra của các nơ-ron trong tầng ẩn của một mô hình mạng, và được sử dụng làm input data cho tầng tiếp theo. Nếu không có các activation functions, khả năng dự đoán của mạng neural sẽ bị giới hạn và giảm đi rất nhiều, sự kết hợp của các activation functions giữa các tầng ẩn là để giúp mô hình học được các quan hệ phi tuyến phức tạp tiềm ẩn trong dữ liệu.

Một số Activation function phổ biến: Sigmoid function, Tanh function, ReLU function, Leaky ReLU, Maxout.

7.3.8. ArgMax và SoftMax

Argmax là hàm toán học tìm đối số cho giá trị lớn nhất từ một hàm mục tiêu. Nó được sử dụng phổ biến nhất trong học máy để tìm lớp có xác suất dự đoán lớn nhất. Argmax có thể được thực hiện theo cách thủ công, mặc dù trong thực tế, hàm Argmax() NumPy được ưu tiên hơn.

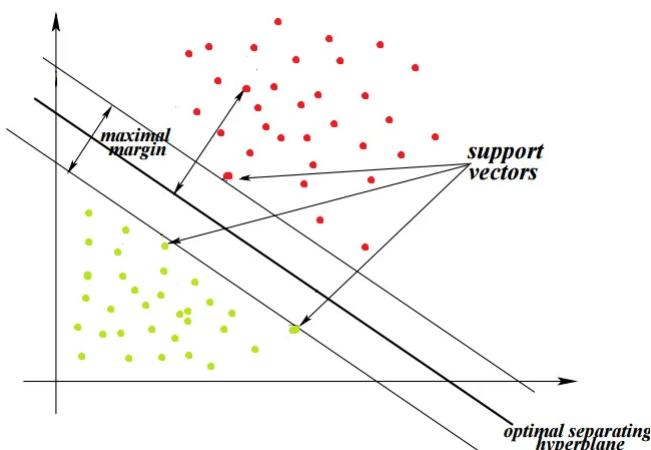
Softmax là một hàm toán học chuyển đổi một vectơ số thành một vectơ xác suất, trong đó xác suất của mỗi giá trị tỷ lệ với tỷ lệ tương đối của mỗi giá trị trong vectơ. Việc sử dụng phổ biến nhất của hàm Softmax trong học máy ứng dụng là nó được sử dụng như một hàm kích hoạt trong mô hình mạng nơ-ron. Cụ thể, mạng được cấu hình để xuất ra N giá trị, một giá trị cho mỗi lớp trong nhiệm vụ phân loại và hàm softmax được sử dụng để chuẩn hóa kết quả đầu ra, chuyển đổi chúng từ giá trị tổng có trọng số thành xác suất tổng bằng một. Mỗi giá trị trong đầu ra của hàm Softmax được hiểu là xác suất thành viên của mỗi lớp. Softmax có thể được coi là một phiên bản nhẹ hơn của hàm Argmax trả về chỉ mục của giá trị lớn nhất trong danh sách.

7.4. Mô hình Support Vector Machine

7.4.1. Khái niệm

SVM là một thuật toán học có giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đề quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại.

Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều (với n là số lượng các tính năng đang có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm “đường bay” (*hyper-plane*) phân chia các lớp. Hyper-plane chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



Hình 7.7. Minh họa mô hình SVM

Mục đích của phương pháp SVM là tìm ra khoảng cách biên (khoảng cách từ điểm dữ liệu gần nhất của mỗi lớp đến mặt phẳng này) lớn nhất. Trên Hình 7.5, mặt phẳng chia 2 lớp vàng và đỏ với khoảng cách biên lớn nhất. Các điểm gần nhất có mũi tên trỏ vào là các Support Vector.

7.4.2. Các tham số trong SVM

7.4.2.1. Tham số Kernel

Kernel là một phương thức được sử dụng để lấy dữ liệu làm đầu vào và chuyển đổi thành dạng dữ liệu xử lý yêu cầu. “Kernel” được sử dụng do tập hợp các hàm toán học được sử dụng trong SVM để thao tác dữ liệu. Ý tưởng cơ bản của Kernel SVM và các

phương pháp kernel nói chung là tìm một phép biến đổi sao cho dữ liệu ban đầu là không phân biệt tuyến tính được biến sang không gian mới. Ở không gian mới này, dữ liệu trở nên phân biệt tuyến tính. Ở đây, chúng ta có một số hàm kernel thông dụng như sau.

Tên	Công thức	kernel	Thiết lập hệ số
linear	$\mathbf{x}^T \mathbf{z}$	'linear'	không có hệ số
polynomial	$(r + \gamma \mathbf{x}^T \mathbf{z})^d$	'poly'	d : degree, γ : gamma, r : coef0
sigmoid	$\tanh(\gamma \mathbf{x}^T \mathbf{z} + r)$	'sigmoid'	γ : gamma, r : coef0
rbf	$\exp(-\gamma \ \mathbf{x} - \mathbf{z}\ _2^2)$	'rbf'	$\gamma > 0$: gamma

Hình 7.8. Tóm tắt kernel thông dụng

7.4.2.2. Tham số Regularization

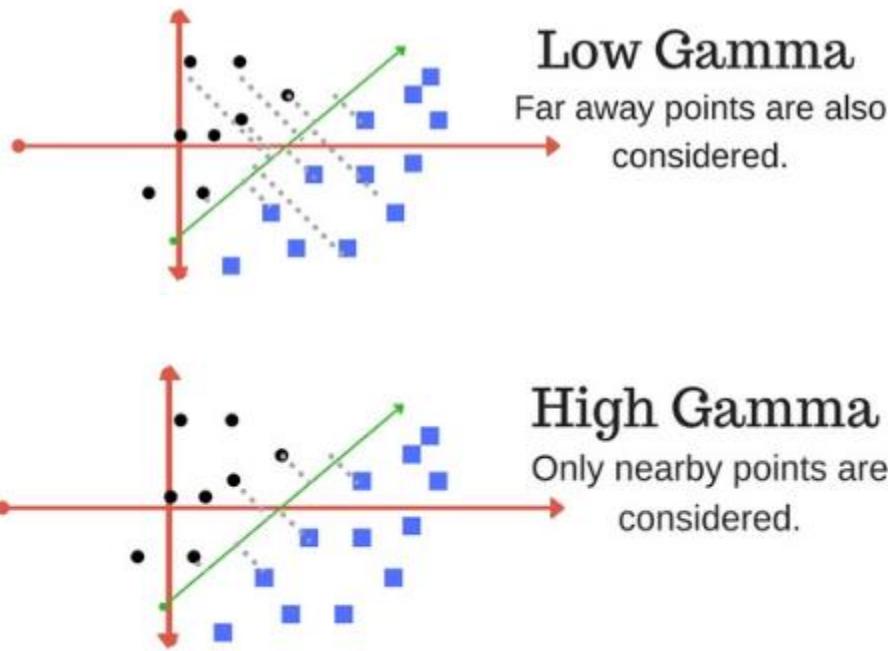
Tham số Regularization (tham số C) điều chỉnh việc có nên bỏ qua các điểm dữ liệu bất thường trong quá trình tối ưu mô hình SVM.

Nếu tham số này có giá trị lớn, quá trình tối ưu sẽ chọn một siêu phẳng sao cho siêu phẳng này phân cách tất cả các điểm dữ liệu một cách tốt nhất, từ đó khoảng cách giữa siêu phẳng tới các điểm dữ liệu của 2 lớp sẽ có giá trị nhỏ (*small-margin*).

Ngược lại, khi tham số này có giá trị nhỏ, siêu phẳng sẽ được xây dựng sao cho khoảng cách với các điểm dữ liệu của 2 lớp có giá trị lớn (*large-margin*), kể cả khi siêu phẳng này sẽ phân loại sai nhiều điểm dữ liệu hơn.

7.4.2.3. Tham số Gamma

Tham số gamma xác định việc sử dụng bao nhiêu điểm dữ liệu cho việc xây dựng siêu phẳng phân cách. Với giá trị gamma nhỏ, các điểm dữ liệu nằm xa đường phân cách sẽ được sử dụng trong việc tính toán đường phân cách. Ngược lại, với giá trị gamma lớn, chỉ những điểm nằm gần đường phân cách mới được sử dụng để tính toán.

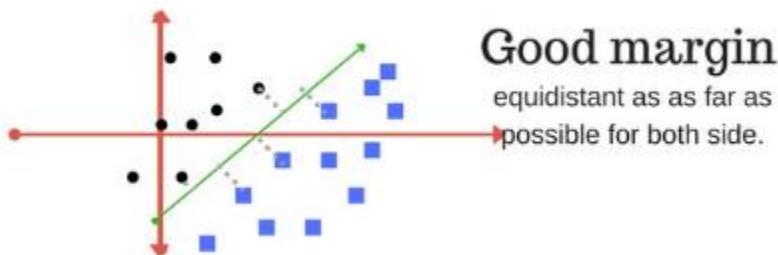


Hình 7.9. Minh họa 2 trường hợp của hệ số gamma (thấp và cao)

7.4.2.4. Tham số Margin

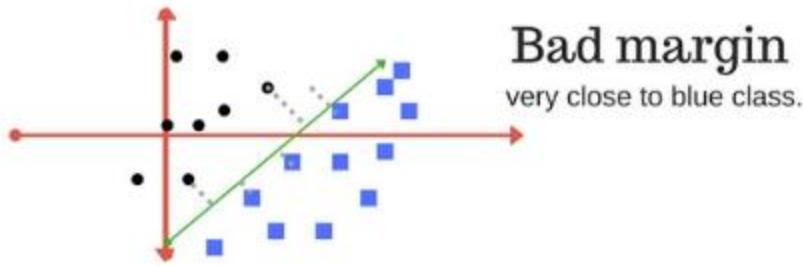
Margin trong SVM là khoảng cách giữa siêu phẳng phân cách với các điểm dữ liệu gần nó nhất. Khoảng cách này đối với các điểm dữ liệu gần nhất của cả 2 lớp càng lớn thì mô hình càng phân loại chính xác.

- SVM có margin tốt: khoảng cách lớn và cân bằng giữa siêu phẳng và các điểm dữ liệu của 2 lớp.



Hình 7.10. Margin tốt trong SVM

- SVM có margin xấu: khoảng cách không cân bằng & nghiêng hẳn về 1 phía.



Hình 7.11. Margin xâu trong SVM

7.4.3. Ưu và nhược điểm

7.4.3.1. Ưu điểm

SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn. SVM sử dụng được cho cả bài toán phân lớp lẩn hồi quy.

Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định.

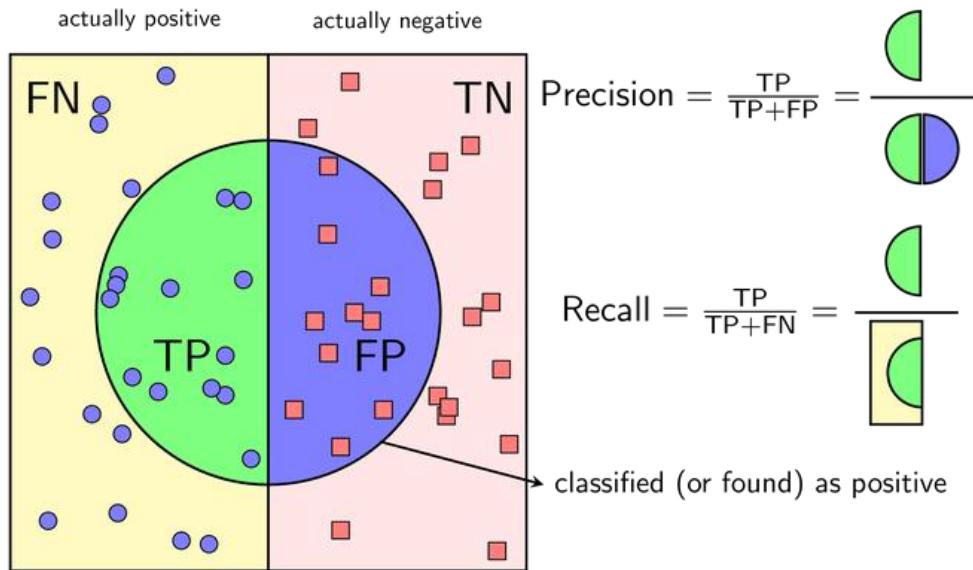
7.4.3.2. Nhược điểm

Thuật toán SVM không phù hợp với các tập dữ liệu lớn vì thời gian training rất lâu. SVM hoạt động không tốt khi tập dữ liệu có độ nhiễu lớn vì các lớp sẽ bị chồng chéo. Trong trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tệ.

7.5. Các phương pháp đánh giá mô hình

7.5.1. Precision và Recall

Trước hết xét bài toán phân loại nhị phân. Ta cũng coi một trong hai lớp là Positive, lớp còn lại là Negative. Xét Hình 7.10 dưới đây:



Hình 7.12. Cách tính Precision và Recall

Khi đó, **Precision** được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive. **Recall** được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu).

- Precision càng cao, tức là số điểm mô hình dự đoán là positive đều là positive càng nhiều. Precision = 1, tức là tất cả số điểm mô hình dự đoán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.
- Recall càng cao, tức là số điểm là positive bị bỏ sót càng ít. Recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.

Xét ví dụ sau, ta có hai trường hợp về tình trạng bệnh là mắc bệnh hoặc không mắc bệnh và hai trường hợp về kết quả chẩn đoán là dương tính và âm tính.

Khi đó, Precision là tỉ lệ người được chẩn đoán là dương tính thật sự mắc bệnh trên tổng số người được chẩn đoán là dương tính. Nếu Precision = 0.9, thì cứ 100 người được chẩn đoán là dương tính thì sẽ thật sự có 90 người mắc bệnh. Precision càng cao thì xác suất người được chẩn đoán là dương tính có khả năng mắc bệnh càng cao.

Recall là tỉ lệ người được chẩn đoán là dương tính thật sự mắc bệnh trên tổng số người thật sự mắc bệnh. Nếu Recall = 0.9, thì cứ 100 người mắc bệnh thì sẽ chẩn đoán 90 người dương tính. Recall càng cao thì xác suất người mắc bệnh được chẩn đoán là dương tính càng cao.

7.5.2. F1 Score

F1-score là trung bình điều hòa (*harmonic mean*) của Precision và Recall (giả sử hai đại lượng này khác 0). F1-score được tính theo công thức:

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

Căn cứ vào F1 ta sẽ chọn model, F1 càng cao thì càng tốt. Khi lý tưởng nhất thì F1 = 1 (khi Recall = Precision=1).

7.5.3. Accuracy

Khả năng mô hình phân loại dự báo chính xác, phân loại chính xác, hay xác định đúng lớp cho dữ liệu cần phân loại. Khi ai đó nói rằng các phép đo là chính xác thì các phép đo đó rất gần với giá trị đích. Sự phân tán của các phép đo chính xác cao có thể tạo ra nhóm các kết quả đo xa nhau, hoặc dày đặc gần nhau.

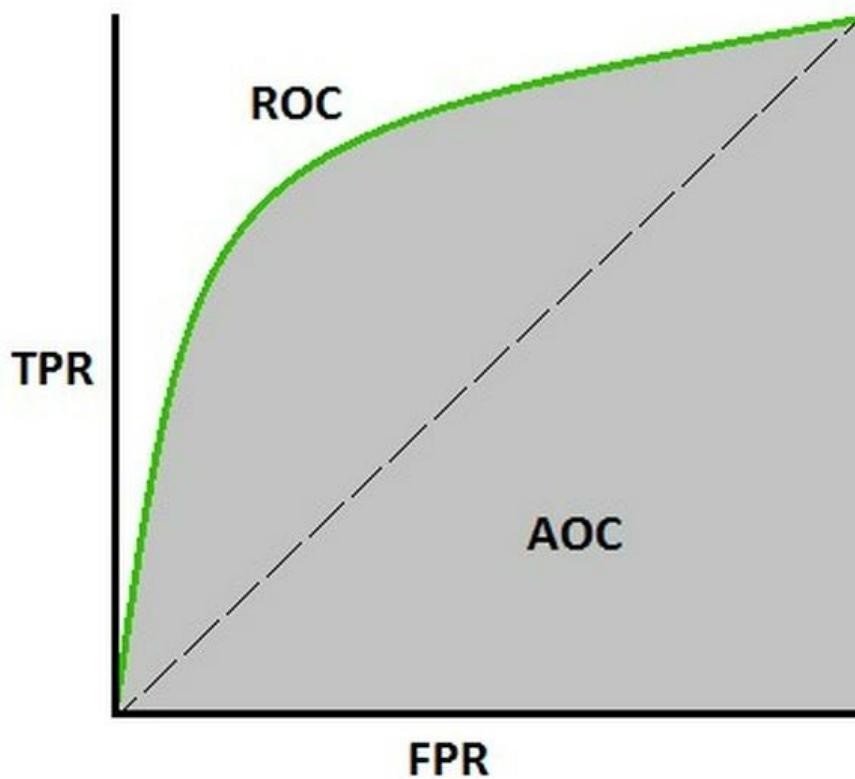
7.5.4. AUC – ROC

7.5.4.1. Khái niệm

AUC - ROC là một phương pháp tính toán hiệu suất của một mô hình phân loại theo các ngưỡng phân loại khác nhau. AUC là từ viết tắt của Area Under the Curve còn ROC viết tắt của Receiver Operating Characteristics. ROC là một đường cong biểu diễn xác suất và AUC biểu diễn mức độ phân loại của mô hình. AUC-ROC còn được biết đến dưới cái tên AUROC (*Area Under the Receiver Operating Characteristics*).

Ý nghĩa của AUROC có thể diễn giải như sau: Là xác suất rằng một mẫu dương tính được lấy ngẫu nhiên sẽ được xếp hạng cao hơn một mẫu âm tính được lấy ngẫu nhiên. Biểu diễn theo công thức, ta có $AUC = P(score(x+) > score(x-))$. Chỉ số AUC càng cao thì mô hình càng chính xác trong việc phân loại các lớp.

Đường cong ROC biểu diễn các cặp chỉ số (TPR, FPR) tại mỗi ngưỡng với TPR là trục tung và FPR là trục hoành.



Hình 7.13. Đường cong ROC

7.5.4.2. Các chỉ số sử dụng trong AUC - ROC

- **TPR (True Positive Rate/Sensitivity/Recall):** Biểu diễn tỷ lệ phân loại chính xác các mẫu dương tính thực sự trên tất cả các mẫu dương tính. TPR càng cao thì các mẫu dương tính càng được phân loại chính xác, được tính theo công thức:

$$TPR/Recall/Sensitivity = \frac{TP}{TP + FN}$$

- **Specificity:** Biểu diễn tỷ lệ phân loại chính xác các mẫu âm tính thực sự trên tất cả các mẫu âm tính được dự đoán là âm tính, được tính theo công thức:

$$Specificity = \frac{TN}{TN + FP}$$

- **FPR (False Positive Rate/Fall-out):** Biểu diễn tỷ lệ gănh nhän sai các mẫu âm tính thành dương tính trên tất cả các mẫu âm tính, được tính theo công thức:

$$FPR = 1 - Specificity = \frac{FP}{TN + FP}$$

Có thể thấy Specificity tỷ lệ nghịch với FPR. FPR càng cao thì Specificity càng giảm và số lượng các mẫu âm tính bị gănh nhän sai càng lớn.

Đây chính là các chỉ số dùng để tính toán hiệu suất phân loại của mô hình. Để hợp chúng lại thành 1 chỉ số duy nhất, ta sử dụng đường cong ROC để hiển thị từng cặp (TPR, FPR) cho các ngưỡng khác nhau với mỗi điểm trên đường cong biểu diễn 1 cặp (TPR, FPR) cho 1 ngưỡng, sau đó tính chỉ số AUC cho đường cong này. Chỉ số AUC chính là con số thể hiện hiệu suất phân loại của mô hình.

7.5.4.3. Mối liên hệ giữa Specificity - Sensitivity, TPR - FPR

- Sensitivity và Specificity là 2 chỉ số tỷ lệ nghịch với nhau. Khi chỉ số Sensitivity tăng thì chỉ số Specificity giảm và ngược lại.
- Khi tăng ngưỡng phân loại, số lượng mẫu được gănh nhän âm tính sẽ tăng lên, từ đó chỉ số Specificity tăng và chỉ số Sensitivity giảm. Điều ngược lại cũng đúng.
- Vì Sensitivity/TPR và FPR đều tỉ lệ nghịch với Specificity nên TPR tỷ lệ thuận với FPR.

CHƯƠNG 8. ỨNG DỤNG CÁC MÔ HÌNH MÁY HỌC

8.1. Giới thiệu dữ liệu

Bộ dữ liệu “online_shoppers_intention.csv” bao gồm nhiều thông tin khác nhau liên quan đến hành vi của khách hàng khi sử dụng các trang web mua sắm trực tuyến.

Link dữ liệu: <https://www.kaggle.com/roshansharma/online-shopper-s-intention>

Bộ dữ liệu gồm 12,330 dòng với 10 thuộc tính Numeric và 8 thuộc tính Categorical. Trong đó, thuộc tính Revenue đóng vai trò là nhãn dữ liệu dùng để phân lớp.

STT	Tên thuộc tính	Ý nghĩa thuộc tính
1	Administrative	Số lượng trang khách truy cập về quản lý tài khoản.
2	Administrative_Duration	Tổng thời gian (tính bằng giây) mà khách truy cập đã dành để quản lý tài khoản các trang liên quan.
3	Informational	Số lượng trang khách truy cập về trang web, thông tin liên lạc và địa chỉ của trang web mua sắm.
4	Informational_Duration	Tổng thời gian (tính bằng giây) khách truy cập trên các trang thông tin.
5	ProductRelated	Số trang khách truy cập về các trang liên quan đến sản phẩm.
6	ProductRelated_Duration	Tổng thời gian (tính bằng giây) khách truy cập trên các trang liên quan đến sản phẩm.
7	BounceRates	Tỷ lệ trung bình khách truy cập vào trang web và sau đó rời khỏi trang mà không thực hiện bất kỳ yêu cầu nào khác đến máy chủ phân tích trong phiên đó.

8	ExitRates	Tỷ lệ phần trăm thoát khỏi trang web của người dùng.
9	PageValues	Thời gian trung bình truy cập vào trang web của người dùng trước khi hoàn tất giao dịch thương mại điện tử.
10	SpecialDay	Thời gian truy cập gần nhất vào trang web trong ngày các ngày lễ.
11	Month	Tháng tương ứng của ngày truy cập.
12	OperatingSystems	Hệ điều hành của khách truy cập.
13	Browser	Trình duyệt của khách truy cập.
14	Region	Khu vực địa lý nơi khách hàng truy cập vào trang web.
15	TrafficType	Mô tả số lượt người dùng truy cập và hoạt động trên website.
16	VisitorType	Loại khách truy cập là “Khách truy cập mới”, “Khách truy cập quay lại” và “Khác”.
17	Weekend	Giá trị Boolean cho biết ngày của chuyến thăm có phải là cuối tuần hay không.
18	Revenue	Nhận dữ liệu cho biết với mỗi lượt truy cập khách hàng có mua hàng hay không.

Bảng 8.1. Các thuộc tính của bộ dữ liệu “online_shoppers_intention.csv”

Với bộ dữ liệu này chúng ta sẽ sử dụng các phương pháp máy học để phân lớp và dự đoán khách hàng có thực hiện giao dịch khi truy cập vào trang web hay không.

8.2. Phân tích dự báo với Python

8.2.1. Cài đặt thư viện

Ngoài các thư viện cơ bản dùng để xử lí và trực quan hóa dữ liệu, chúng ta sẽ cài đặt thư viện Sklearn để xây dựng mô hình và đánh giá kết quả của từng mô hình.

```

#Import library

#Data processing
import pandas as pd
import numpy as np

#Visualization
import seaborn as sns
import matplotlib.pyplot as plt

#Split train, test
from sklearn.model_selection import train_test_split

#Feature scaling
from sklearn.preprocessing import StandardScaler

#Create model
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

#Evaluation metrics
from sklearn import metrics

```

8.2.2. Khám phá dữ liệu

Sau khi load dữ liệu bằng lệnh `read_csv` ta sử dụng hàm `info()` để xem các thông tin cơ bản của bộ dữ liệu. Ta có thể thấy bộ dữ liệu không có giá trị rỗng.

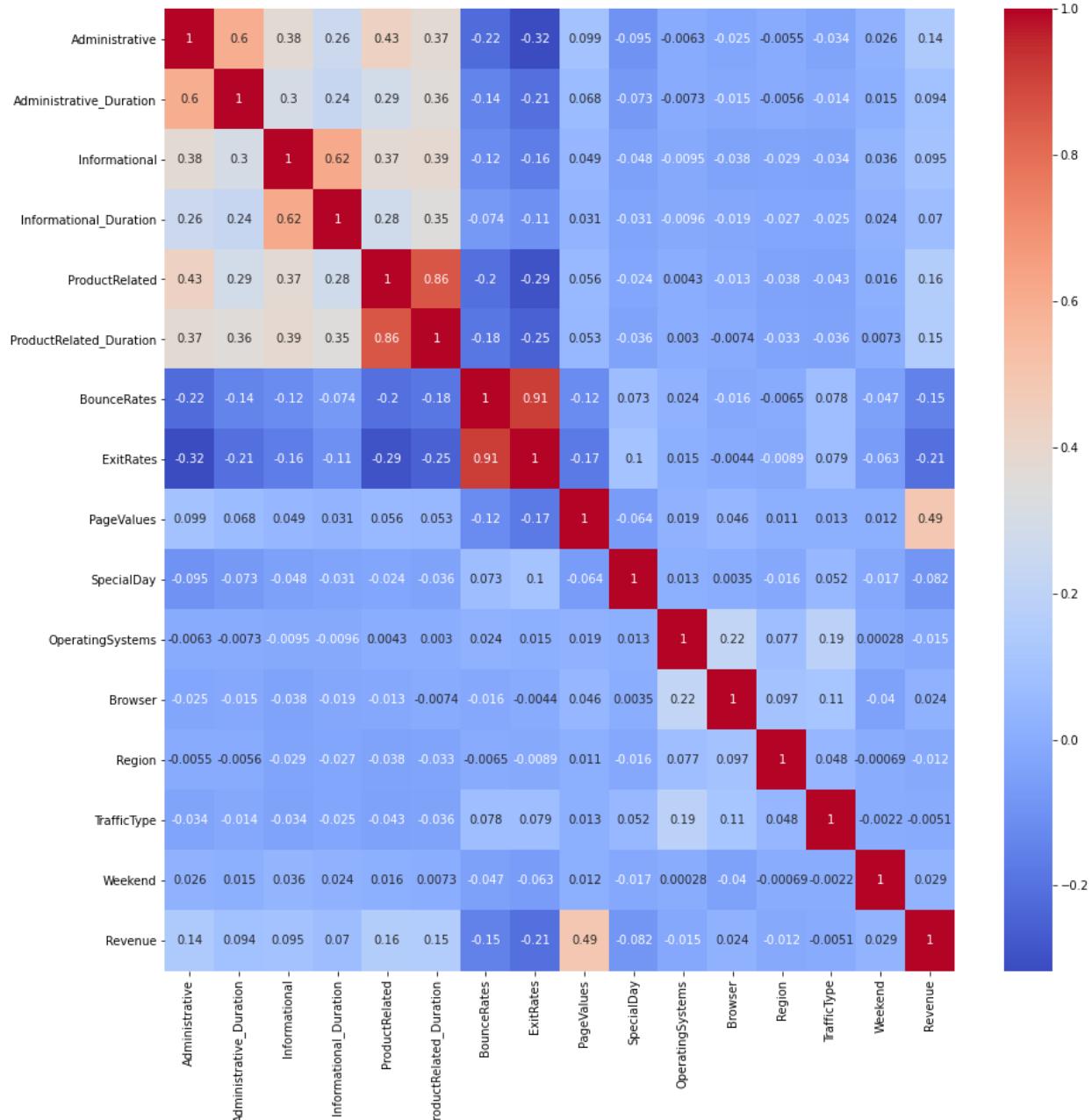
```

#Data information
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Administrative    12330 non-null   int64  
 1   Administrative_Duration 12330 non-null   float64 
 2   Informational     12330 non-null   int64  
 3   Informational_Duration 12330 non-null   float64 
 4   ProductRelated    12330 non-null   int64  
 5   ProductRelated_Duration 12330 non-null   float64 
 6   BounceRates       12330 non-null   float64 
 7   ExitRates         12330 non-null   float64 
 8   PageValues        12330 non-null   float64 
 9   SpecialDay        12330 non-null   float64 
 10  Month             12330 non-null   object  
 11  OperatingSystems  12330 non-null   int64  
 12  Browser           12330 non-null   int64  
 13  Region            12330 non-null   int64  
 14  TrafficType       12330 non-null   int64  
 15  VisitorType       12330 non-null   object  
 16  Weekend           12330 non-null   bool   
 17  Revenue           12330 non-null   bool  
dtypes: bool(2), float64(7), int64(7), object(2)

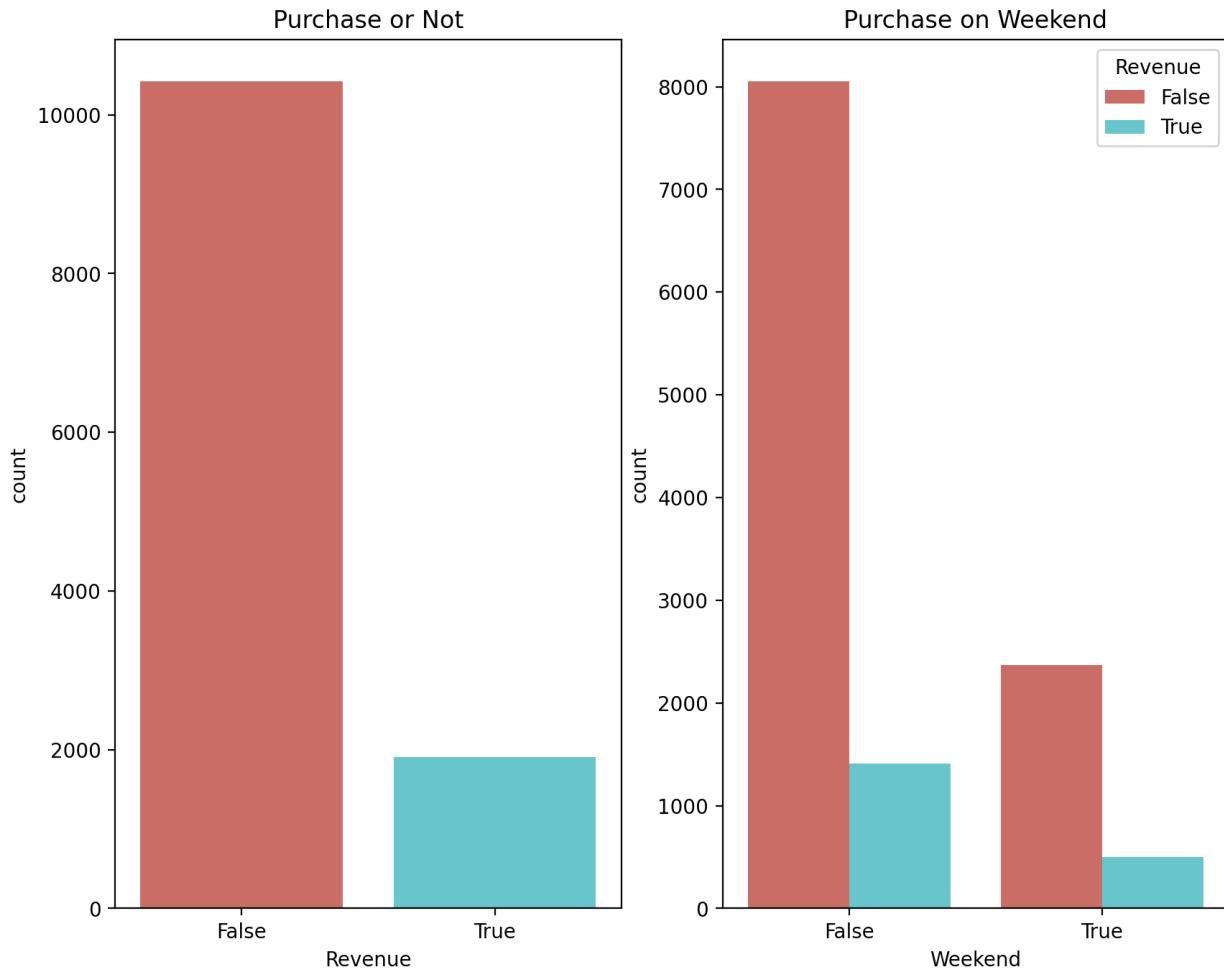
```

Từ Hình 8.1 ta có thể thấy thuộc tính “PageValues” có sự ảnh hưởng cao nhất đến quyết định mua hàng của khách hàng, kế tiếp là các thuộc tính “ProductRelated”, “ProductRelated_Duration”, “Administrative”, “ExitRates” (tương quan âm).



Hình 8.1. Biểu đồ thẻ hiện mối tương quan giữa các biến trong bộ dữ liệu online_shoppers_intention.csv

Khi tiến hành kiểm tra độ tương quan của 2 cột là Revenue và cột Weekend, ta nhận thấy vào thời điểm cuối tuần, thì tỷ lệ không có doanh thu giảm 2.51% (từ 85.11% lên 82.6%).



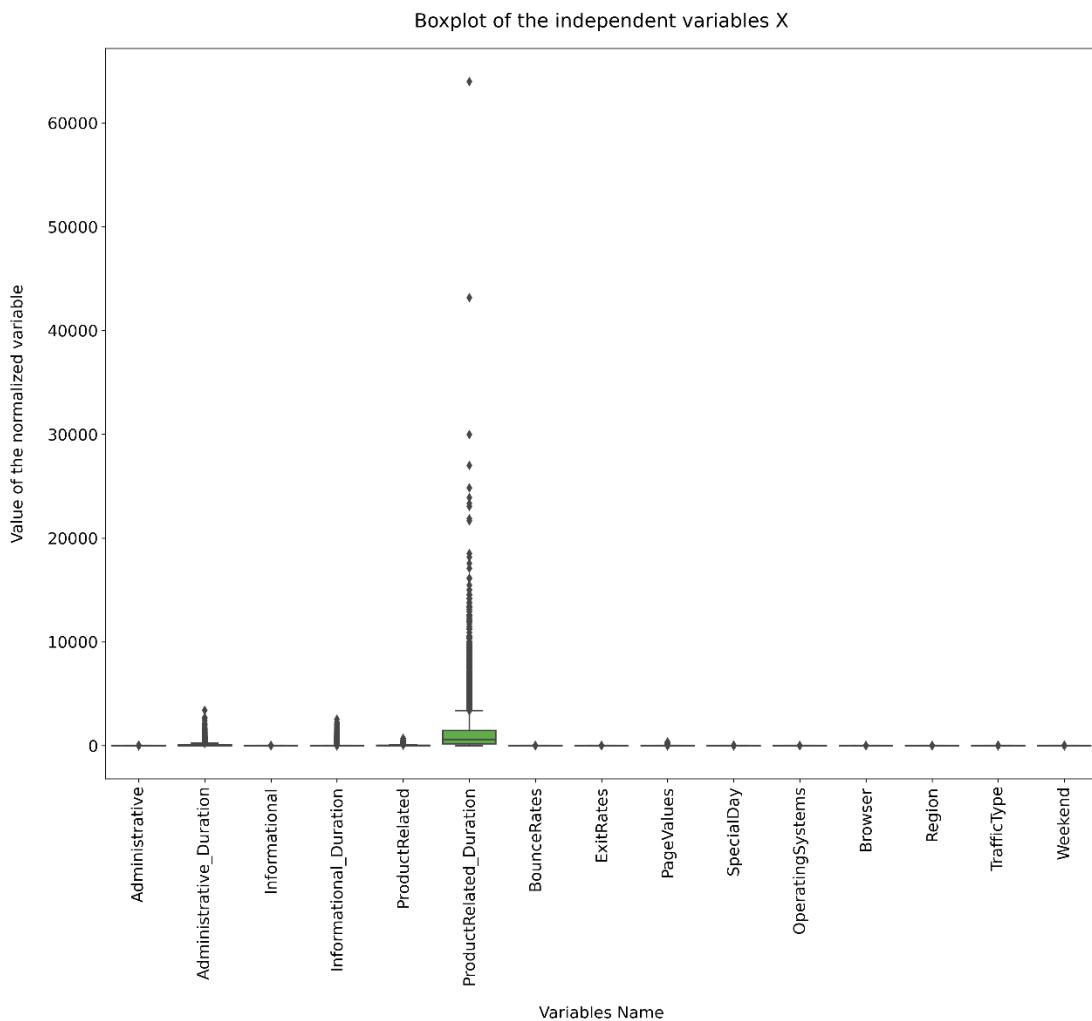
Hình 8.2. Biểu đồ phân chia hai lớp True - False trong “Revenue” và “Weekend”

Tiếp tục sử dụng cột Visitor Type để kiểm tra độ tương quan với cột nhãn là cột Revenue. Từ biểu đồ ở bảng Hình 8.3, ta thấy tỷ lệ khách hàng cũ không mang lại doanh thu chiếm 86.07%, điều này chỉ ra doanh nghiệp này không có khả năng giữ chân các khách hàng để mang lại lợi nhuận. Đồng thời, khách hàng mới cũng không mang lại doanh thu cho doanh nghiệp.

Kiểm tra tương quan với bảng Month, ta thu được thấy các tháng 3, 5, 11, 12 có tỷ lệ không mang lại doanh thu lớn hơn 85%.



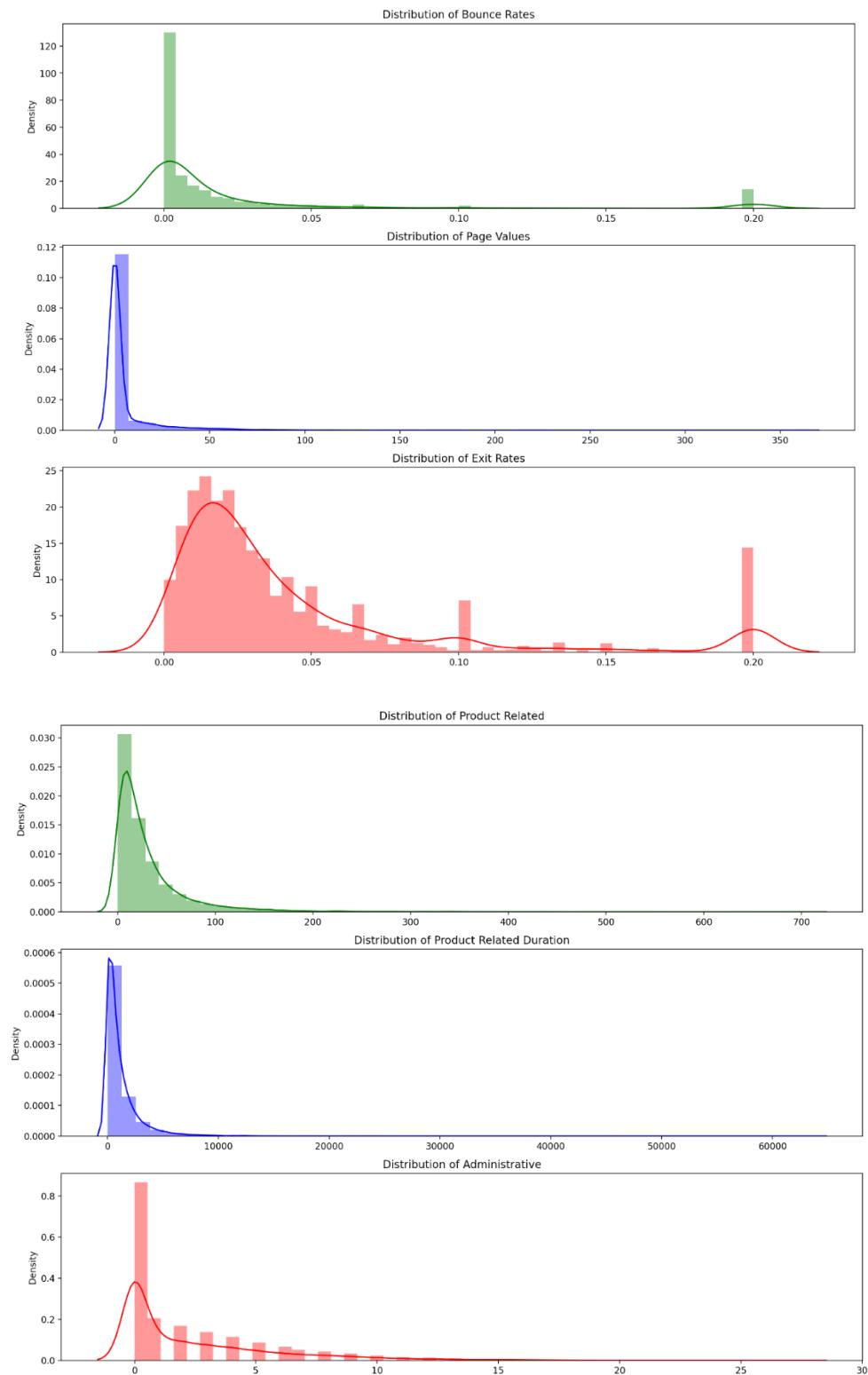
Hình 8.3. Biểu đồ tương quan giữa “Revenue - VisitType – Month”



Hình 8.4. Biểu đồ miêu tả vị trí phân bố dữ liệu của các biến độc lập

Tiếp theo ta sử dụng biểu đồ boxplot để sự phân bố của các giá trị trong bộ dữ liệu. Ta thấy thuộc tính “ProductRelated_Duration” có nhiều giá trị ngoại lai nhất so với các thuộc tính còn lại.

Cuối cùng, ta sử dụng hàm `displot()` để kiểm tra sự phân phối dữ liệu một vài biến trong bộ dữ liệu. Dựa vào Hình 8.5, ta thấy giá trị của các biến đều bị lệch về phía bên phải,



Hình 8.5. Biểu đồ thể hiện sự phân phối dữ liệu của các biến trong bộ dữ liệu

8.2.3. Tiền xử lý dữ liệu

Để thuận tiện cho việc phân lớp dữ liệu, chúng ta sẽ mã hóa các giá trị đặc trưng của hai thuộc tính “Month” và “VisitorType” từ dạng chuỗi thành hai giá trị 0 và 1 bằng phương pháp One Hot Encoding. Đây là quá trình biến đổi từng giá trị đặc trưng thành một biểu diễn vectơ nhị phân để sử dụng trong các thuật toán máy học.

Bên cạnh đó, ta sử dụng hàm map() để chuyển đổi 2 giá trị True, False của thuộc tính “Weekend” và “Revenue” về dạng nhị phân 0 và 1.

```
#Map the True/False values to 0 and 1
df['Weekend'] = df['Weekend'].map({False: 0, True: 1})
df['Revenue'] = df['Revenue'].map({False: 0, True: 1})
```

```
#One-hot encoding categorical data
df = pd.get_dummies(df)
df.head(10)
```

Month_May	Month_Nov	Month_Oct	Month_Sep	VisitorType_New_Visitor	VisitorType_Other	VisitorType_Returning_Visitor
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1

Hình 8.6. Giá trị của các biến phân loại sau khi mã hóa

Tiếp theo ta tiến hành tách cột “Revenue” đóng vai trò là biến phụ thuộc ra khỏi dataframe và giữ lại tất cả các biến còn lại để xây dựng mô hình và dự đoán.

```
#Remove the target column revenue from dataframe
X = df.drop(['Revenue'], axis = 1)
y = df['Revenue']

#Check the shapes
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

Shape of X: (12330, 28)
 Shape of y: (12330,)

Trước khi bắt đầu xây dựng các mô hình máy học ta sẽ chia tập dữ liệu thành 2 tập train, test với tỷ lệ 70: 30 và chuẩn hóa tất cả dữ liệu trên X_train, X_test.

```
#Split train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)

print("Shape of X_train :", X_train.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of X_test :", X_test.shape)
print("Shape of y_test :", y_test.shape)

Shape of X_train : (8631, 28)
Shape of y_train : (8631,)
Shape of X_test : (3699, 28)
Shape of y_test : (3699,)
```

```
#Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

8.2.4. Xây dựng mô hình

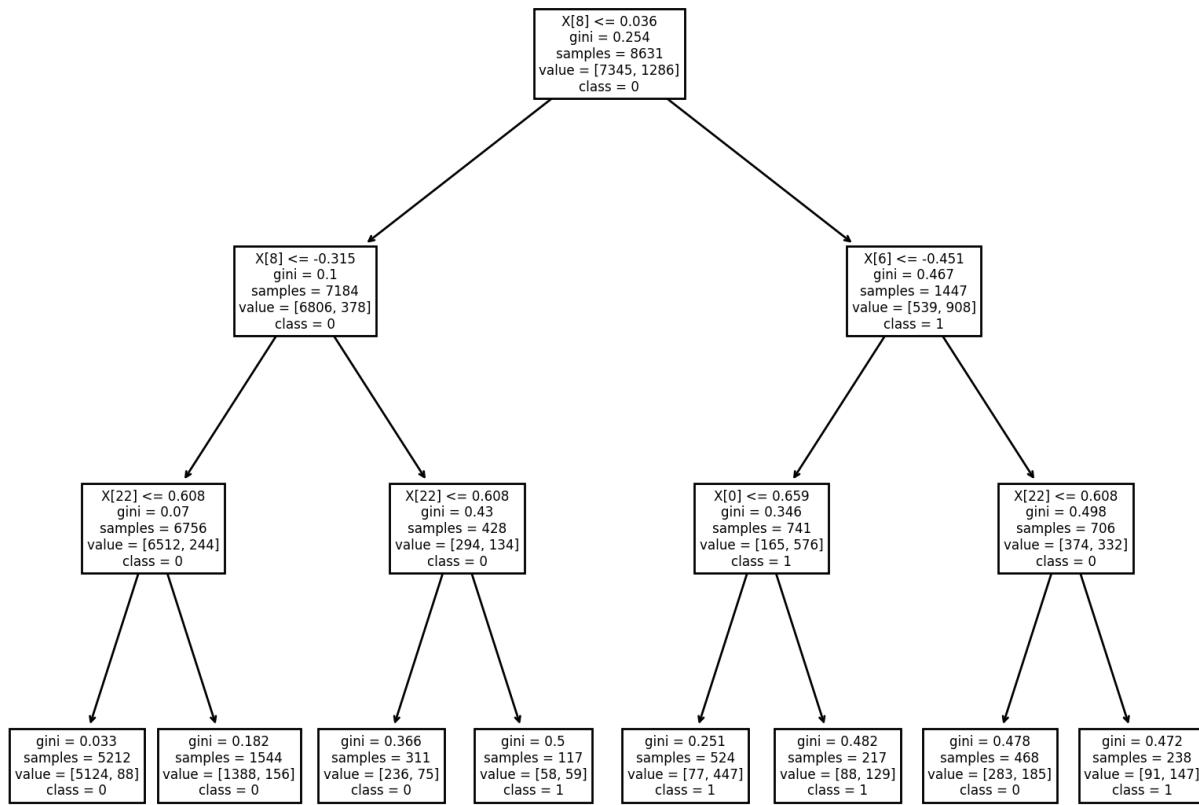
8.2.4.1. Mô hình Decision Tree

Để xây dựng một mô hình Decision Tree ta sử dụng hàm DecisionTreeClassifier() trong thư viện scikit-learn, đồng thời sử dụng phương thức fit() để huấn luyện mô hình trên tập train đã chia.

```
#Create model
model_DT = DecisionTreeClassifier(max_depth=3).fit(X_train,y_train)
```

Do bộ dữ liệu có 12 thuộc tính nên số lượng cây sẽ phát triển sẽ rất lớn và phức tạp. Nếu ở chế độ mặc định, các nút trên cây sẽ được mở rộng cho đến khi tất cả các lá đạt giá trị thuần túy. Điều này sẽ dẫn đến hiện tượng “overfitting”, khi mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán kém hiệu quả, và chất lượng mô hình không còn tốt trên dữ liệu test. Vì vậy, trong mô hình này ta sẽ giới hạn độ sâu của cây để tránh tình trạng trên.

Sau khi tạo mô hình ta sẽ trực tiếp vẽ được cây quyết định bằng lệnh `tree.plot_tree()`. Chỉ số Gini trên mỗi node hiện mức độ phân loại sai khi ta chọn ngẫu nhiên một phần tử từ tập train.



Hình 8.7. Mô hình cây quyết định

Tiếp theo ta tiến hành dự đoán trên dữ liệu test và kiểm tra kết quả dự đoán được so với kết quả thực tế.

```
#Predict on the test set  
y_pred_DT = model_DT.predict(X_test)  
  
#Actual value and predicted value  
new_DT = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred_DT})  
new_DT.head(10)
```

	Actual value	Predicted value
12245	0	0
9704	0	0
9177	0	0
8848	0	0
2768	0	0
11549	1	1
2193	0	0
7772	0	1
6338	1	0
4008	0	0

Sau đó ta sử dụng hàm confusion_matrix() để kiểm tra sự phân chia giữa các lớp trong kết quả dự đoán được.

```
#Confusion matrix  
cnf_matrix_DT = confusion_matrix(y_test, y_pred_DT)  
cnf_matrix_DT  
  
array([[2926,  151],  
       [ 265,  357]])
```

Từ ma trận hỗn loạn trên ta thấy mô hình đã dự đoán đúng được 2926 khách hàng không mua hàng và 357 người đã mua hàng trên trang web. Tuy nhiên do sự chênh lệch giữa khá lớn giữa hai lớp 0 và 1 nên mô hình có xu hướng thiên vị cho lớp 0.

Ta kiểm tra các độ đánh giá của mô hình trên tập test và so sánh trên tập train để xem mô hình có bị hiện tượng “overfitting” (quá khớp) hoặc “underfitting” (chưa khớp) hay không.

```
#Evaluation metrics  
print(classification_report(y_test, y_pred_DT))
```

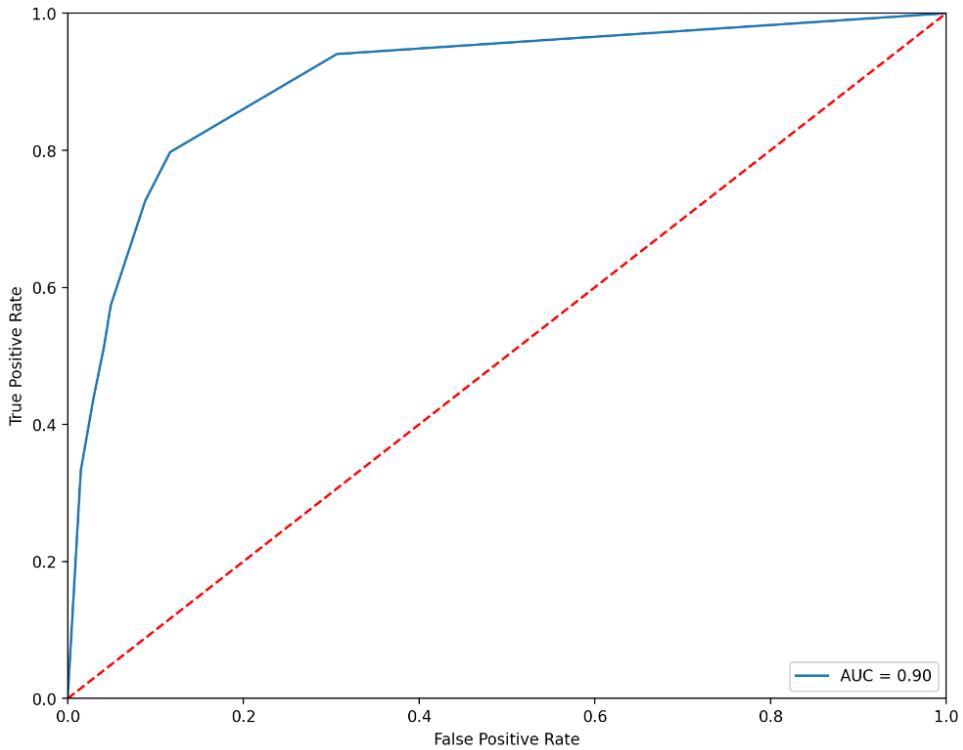
	precision	recall	f1-score	support
0	0.92	0.95	0.93	3077
1	0.70	0.57	0.63	622
accuracy			0.89	3699
macro avg	0.81	0.76	0.78	3699
weighted avg	0.88	0.89	0.88	3699

```
#Compare with train set  
from sklearn.metrics import accuracy_score  
y_pred_train = model_DT.predict(X_train)  
print('Training-set accuracy score: {:.4f}'.format(accuracy_score(y_train, y_pred_train)))
```

Training-set accuracy score: 0.9052

Từ hai kết quả trên ta thấy độ chính xác trên tập train và test gần như nhau.

Cuối cùng ta trực quan hóa biểu đồ đường cong ROC để đánh giá kết quả dự đoán. Từ Hình 8.7, chỉ số AUC = 0.90 và càng tiến gần về 1 nên có thể kết luận đây là mô hình có độ chính xác cao.



Hình 8.8. Biểu đồ ROC của mô hình Decision Tree

8.2.4.2. Mô hình Random Forest

Ta sử dụng hàm RandomForestClassifier trong thư viện Sklearn để tạo mô hình Random Forest.

```
#Create model
model_RF = RandomForestClassifier().fit(X_train,y_train)
```

Sau khi đã có mô hình ta tiến hành dự đoán bằng hàm predict() và in kết quả dự đoán so với kết quả thực tế ra màn hình.

```
#Predict on the test set
y_pred_RF = model_RF.predict(X_test)

#Actual value and predicted value
new_RF = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred_RF})
new_RF.head(10)
```

	Actual value	Predicted value
12245	0	0
9704	0	0
9177	0	0
8848	0	0
2768	0	0
11549	1	1
2193	0	0
7772	0	1
6338	1	0
4008	0	0

Tiếp theo ta dùng hàm confusion_matrix() để kiểm tra sự phân chia giữa các kết quả dự đoán được

```
#Confusion matrix
cnf_matrix_RF = metrics.confusion_matrix(y_test, y_pred_RF)
cnf_matrix_RF

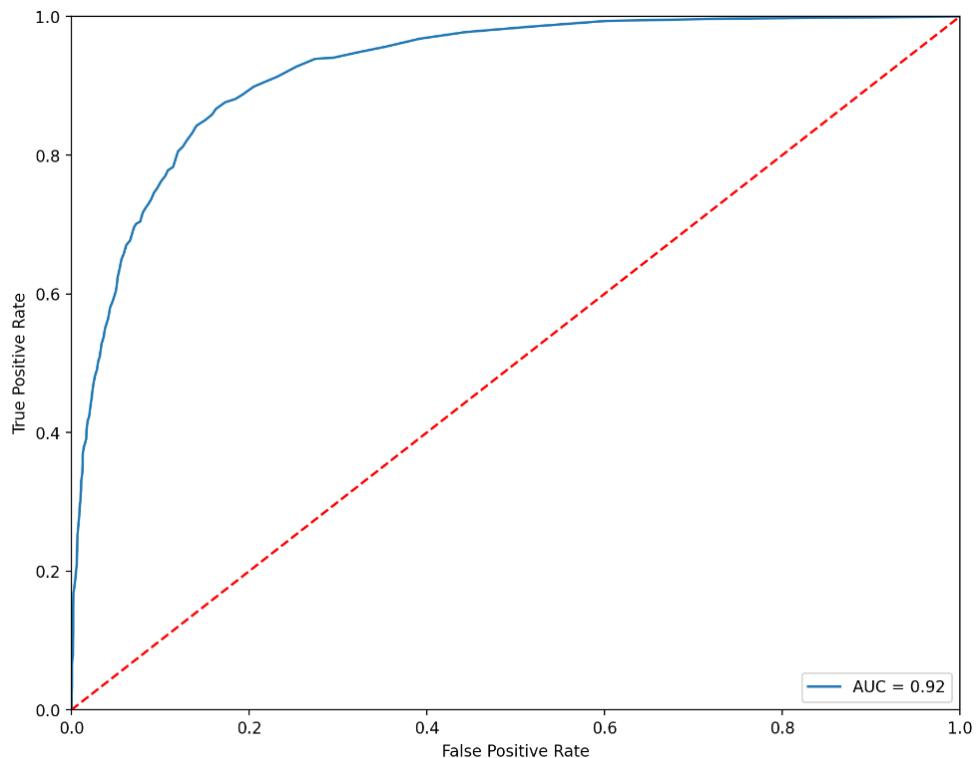
array([[2956, 121],
       [270, 352]])
```

Ta có bảng chi tiết các chỉ số đánh giá chất lượng mô hình sau khi được kiểm định dựa trên tập test với chỉ số Accuracy 0.89 Precision và Recall lần lượt là 0.74, 0.57 khá cao.

```
#Evaluation metrics
print(metrics.classification_report(y_test, y_pred_RF))

precision    recall   f1-score   support
0            0.92      0.96      0.94     3077
1            0.74      0.57      0.64      622
accuracy                           0.89     3699
macro avg       0.83      0.76      0.79     3699
weighted avg    0.89      0.89      0.89     3699
```

Bên cạnh đó đồ thi đường cong ROC với $AUC = 0.92$ là một con số cao đồng nghĩa với việc mô hình phân loại chính xác tốt.



Hình 8.9. Biểu đồ ROC của mô hình Random Forest

8.2.4.3. Mô hình Neural Network

Sử dụng mô hình MLPClassifier của scikit-learn, ta xây dựng mô hình mạng nơ ron đa lớp phân lớp (*Multilayer Perceptron Classifier - MLP Classifier*) với số lần lặp lại cho việc huấn luyện là 2000 vòng (càng nhiều vòng sẽ càng khiến các thông số mô hình hội tụ

hơn, nhưng sẽ tốn nhiều thời gian huấn luyện). Ta gọi hàm fit() thuộc mô hình để huấn luyện theo dữ liệu có được ở tập train.

```
#Create model  
model_NN = MLPClassifier(max_iter=2000).fit(X_train, y_train)
```

Tiếp theo ta sử dụng mô hình vừa được huấn luyện để dự đoán nhãn cho tập test bằng cách gọi hàm predict() với đầu vào là X_test chứa các đặc trưng của tập test.

```
#Predict on the test set  
y_pred_NN = model_NN.predict(X_test)
```

Tiến hành in ra các giá trị thuộc tập y_pred_NN (chứa nhãn dự đoán được của mô hình) và tập y_test (chứa nhãn thực tế của dữ liệu tập X_test) để kiểm tra xem các dự đoán của mô hình có chính xác hay không.

```
#Actual value and predicted value  
new_NN = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred_NN})  
new_NN.head(10)
```

	Actual value	Predicted value
12245	0	0
9704	0	0
9177	0	0
8848	0	0
2768	0	0
11549	1	0
2193	0	0
7772	0	0
6338	1	0
4008	0	0

Có thể nhận ra ở số thứ tự 11549 và 6338, các nhãn do mô hình đưa ra trái ngược với nhãn thực tế.

Sử dụng hàm confusion_matrix(), ta trích xuất được Confusion Matrix của kết quả dự đoán do mô hình đưa ra so với kết quả thực tế.

```
#Confusion matrix
cnf_matrix_NN = metrics.confusion_matrix(y_test, y_pred_NN)
cnf_matrix_NN

array([[2876,  201],
       [ 268,  354]])
```

Dựa vào Confusion Matrix ta thấy được mô hình dự đoán đúng khá nhiều (với 2876 điểm dữ liệu được phân loại đúng vào lớp 0 và 354 điểm dữ liệu được dự đoán đúng vào lớp 1 trên tổng số 3699 điểm dữ liệu). Số điểm dữ liệu dự đoán sai vào lớp 0 là 268, và số điểm dữ liệu dự đoán sai vào lớp 1 là 354 cho thấy mô hình có xu hướng thiên vị cho lớp 0 hơn lớp 1.

Tiếp theo, ta trích xuất các thông số đo chỉ số Accuracy, Precision và Recall của mô hình thông qua hàm classification_report() của thư viện scikit-learn.

```
#Evaluation metrics
print(metrics.classification_report(y_test, y_pred_NN))

precision    recall  f1-score   support

          0       0.91      0.93      0.92     3077
          1       0.64      0.57      0.60      622

   accuracy                           0.87     3699
  macro avg       0.78      0.75      0.76     3699
weighted avg       0.87      0.87      0.87     3699
```

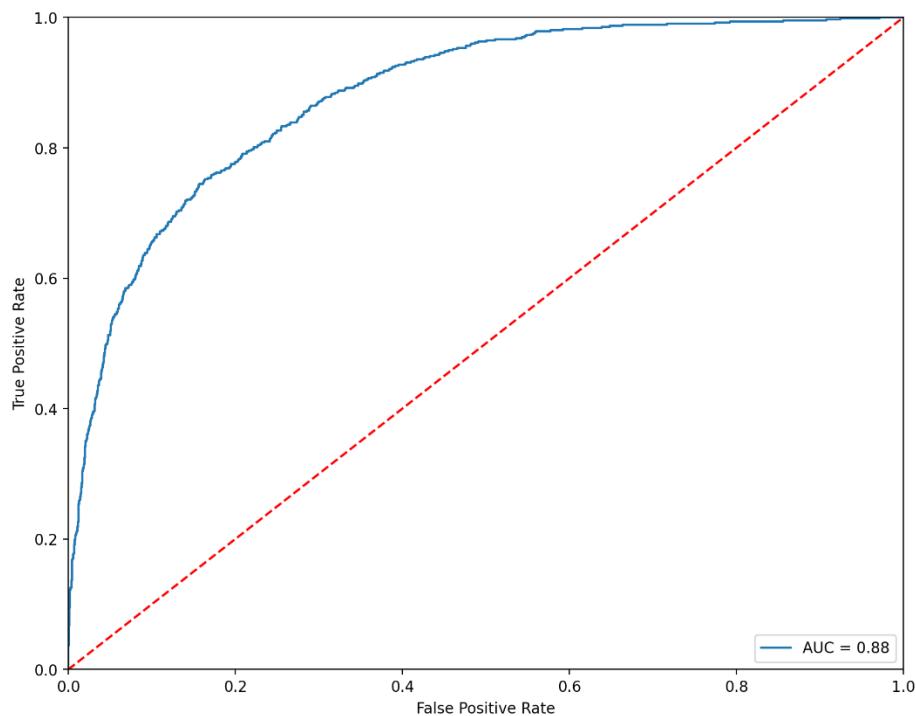
Các chỉ số Accuracy, Precision và Recall lần lượt là 0.87, 0.64 và 0.55. Điều này cho thấy mô hình có độ chính xác tương đối cao.

Ta sẽ dùng các giá trị xác suất đã được mô hình MLP Classifier đưa ra với dự đoán của tập test để trích xuất đường cong ROC và giá trị AUC của mô hình sử dụng hàm

`metrics.roc_curve()` và `metrics.auc()` của thư viện scikit-learn. Từ đây ta có luôn cả giá trị FPR và TPR.

```
#Compute ROC AUC
y_pred_proba = model_NN.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
auc = roc_auc_score(y_test, y_pred_proba)

#Visualize ROC curve
plt.plot(fpr,tpr,label="AUC = %0.2f" % auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```



Hình 8.10. Biểu đồ ROC của mô hình Neural Network

Giá trị AUC ở đây (tức diện tích của phần hình nằm dưới đường cong ROC) là 0.88, một giá trị khá cao, đồng nghĩa với việc mô hình cho kết quả dự đoán khá chính xác.

8.2.4.4. Mô hình Support Vector Machine

Sau khi tải bộ dữ liệu, làm sạch, phân chia các tập train và test, chuẩn hóa cũng như chuẩn bị đầy đủ các thư viện, đặc biệt không thể thiếu thư viện sklearn.svm.SVC, chúng ta tiến hành tạo mô hình theo đoạn lệnh dưới đây:

```
#Create model  
model_SVM = SVC(probability=True).fit(X_train,y_train)
```

Tiếp theo là sử dụng hàm predict() để dự đoán trên tập dữ liệu test đã phân chia trước đó. Dưới đây, nhóm thực hiện cũng đã lập bảng đối chiếu kết quả dự đoán và thực tế được ghi nhận trong bộ test.

```
#Predict on the test set  
y_pred_SVM = model_SVM.predict(X_test)  
  
#Actual value and predicted value  
new_SVM = pd.DataFrame({'Actual value': y_test, 'Predicted value':y_pred_SVM})  
new_SVM.head(10)
```

	Actual value	Predicted value
12245	0	0
9704	0	0
9177	0	0
8848	0	0
2768	0	0
11549	1	0
2193	0	0
7772	0	0
6338	1	0
4008	0	0

Ta tiếp tục tính toán ma trận hỗn loạn của kết quả dự đoán ở trên.

```
#Confusion matrix  
cnf_matrix_SVM = confusion_matrix(y_test, y_pred_SVM)  
cnf_matrix_SVM  
  
array([[2991,    86],  
       [ 346,   276]])
```

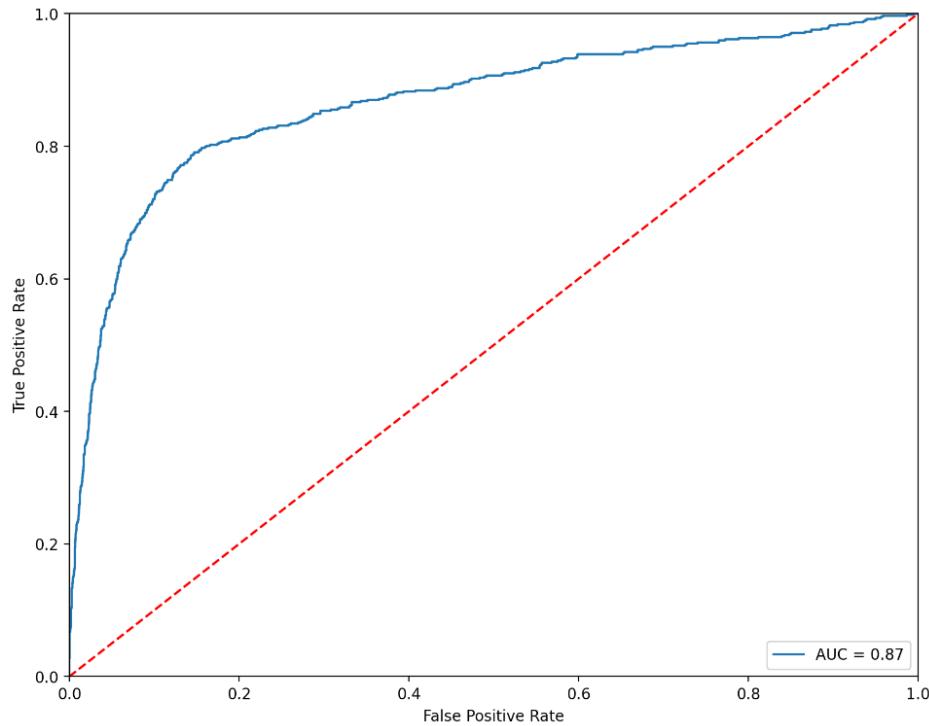
Như vậy, mô hình dự đoán đúng 2991 người không mua hàng và 276 người đã mua hàng.

```
#Evaluation metrics  
print(classification_report(y_test, y_pred_SVM))
```

	precision	recall	f1-score	support
0	0.90	0.97	0.93	3077
1	0.76	0.44	0.56	622
accuracy			0.88	3699
macro avg	0.83	0.71	0.75	3699
weighted avg	0.87	0.88	0.87	3699

Từ kết quả trên chúng ta nhận thấy độ chính xác khi dự đoán người không mua hàng là 90% và người mua hàng là 76%. Độ chính xác chung khá cao với 88%.

Dùng các giá trị xác suất đã được mô hình phân lớp SVM đưa với tập test để trích xuất giá trị của đường cong ROC và giá trị AUC của mô hình sử dụng hàm roc_curve() và roc_auc_score() của thư viện scikit-learn. Từ các giá trị trên, ta tiến hành trực quan hóa đường cong ROC của mô hình.



Hình 8.11. Biểu đồ ROC của mô hình SVM

Giá trị AUC ở đây (tức diện tích của phần hình nằm dưới đường cong ROC) là 0.87, một giá trị khá cao, đồng nghĩa với việc mô hình cho kết quả dự đoán khá chính xác.

8.2.5. Đánh giá

Sau khi chạy thực nghiệm từ cả 4 phương pháp phân lớp trên ta có bảng đánh giá dưới đây:

	Accuracy	Precision	Recall	F1-score
Decision Tree	0.89	0.70	0.57	0.63
Random Forest	0.89	0.74	0.57	0.64
Neural Network	0.87	0.64	0.57	0.60
Support Vector Machine	0.88	0.76	0.44	0.56

Bảng 8.2. Bảng đánh giá kết quả của 4 mô hình trên lớp 1

Dù lớp 0 là lớp chiếm ưu thế trong bộ dữ liệu nhưng trong thực tế việc xác định khách hàng nào đem lại doanh thu cho doanh thu là điều cần thiết. Vì vậy ta sẽ tập trung đánh giá kết quả của 4 mô hình dựa trên lớp 1.

Nhìn vào Bảng 8.2, ta thấy mô hình Random Forest cho kết quả tốt nhất với 57% dự đoán chính xác trên tổng số khách hàng được dự đoán là mua hàng. Ngoài ra, hệ số AUC của mô hình bằng 0.92 cho thấy mô hình có độ tin cậy cao. Vì vậy, ta kết luận rằng việc xây dựng mô hình phân lớp dựa trên thuật toán Random Forest đem lại hiệu quả cao trong việc dự đoán ý định mua hàng của người tiêu dùng trực tuyến. Thông qua đó, các nhà bán lẻ trực tuyến có thể cải thiện tỷ lệ khách hàng quay lại và làm tăng lợi nhuận cho hoạt động kinh doanh trong lĩnh vực thương mại điện tử.

8.2.6. Hướng phát triển

- Việc mất cân bằng giữa hai lớp 0 và 1 dẫn đến việc các mô hình có xu hướng thiên vị lớp chiếm ưu thế hơn mà bỏ qua lớp có giá trị thật sự. Vì vậy, ta có thể áp dụng các kỹ thuật cân bằng dữ liệu như SMOTE, Over-sampling, Under-sampling để cân bằng giá trị của các lớp nhằm cải thiện kết quả dự đoán của mô hình.
- Đối với những bài toán phân lớp, số lượng các thuộc tính thường rất lớn nhưng các thuộc tính không liên quan hoặc có độ ảnh hưởng rất thấp cũng là nguyên nhân dẫn đến việc học của mô hình không được chính xác và việc phân lớp trở nên phức tạp hơn. Ta có thể dùng biểu đồ Heatmap hoặc các giải thuật xếp hạng như RandomForestClassifier(), ExtraTreeClassifier() trước và sau khi xây dựng mô hình để lựa chọn các thuộc tính quan trọng nhằm thu gọn kích thước dữ liệu và tăng độ chính xác cho mô hình.
- Các giá trị ngoại lệ cũng ảnh hưởng đến chất lượng mô hình. Do đó ta cần loại bỏ những giá trị này để tăng cao độ chính xác cho mô hình dự đoán.
- Đối với các mô hình máy học, giá trị của các tham số trong mô hình ảnh hưởng đến việc huấn luyện và độ chính xác của mô hình đó. Việc điều chỉnh siêu tham số (*Hyperparameter tuning*) - quá trình chọn một tập hợp các siêu tham số tối ưu cho một thuật toán học, sẽ cải thiện độ chính xác tổng thể, hay nói cách khác, cung cấp dự đoán đầu ra tốt nhất có thể. Ta có thể áp dụng các kỹ thuật như GridSearchCV,

RandomizedSearchCV để tìm chỉ số learning rate khi training mô hình mạng nơ ron nhân tạo, tham số C và sigma khi training mô hình Support Vector Machine.

8.3. Phân tích dự báo với R

8.3.1. Mô hình Decision Tree

8.3.1.1. Cài đặt thư viện

- Thư viện caret: mã hóa các cột có giá trị phân loại
- Thư viện rpart: xây dựng mô hình cây quyết định.
- Thư viện rpart.plot: trực quan hóa cây quyết định.
- Thư viện pROC: trực quan hóa đường cong ROC.
- Thư viện e1071: in các giá trị thông kê đánh giá mô hình.

8.3.1.2. Tiền xử lý dữ liệu

Cũng tương tự như Python, đầu tiên ta sẽ chuyển hai cột “Revenue” và “Weekend” từ giá trị True, False sang nhị phân 0 và 1. Sau đó tiến hành thực hiện mã hóa các biến phân loại bằng hàm dummyVars().

```
#Convert 'Revenue' & 'Weekend' to binary number
df$Revenue <- ifelse(df$Revenue == "TRUE", 1, 0)
df$Weekend <- ifelse(df$Weekend == "TRUE", 1, 0)

#Encoding categorical variables
dummy <- dummyVars(~ ., data=df)
new_df <- data.frame(predict(dummy, newdata=df))
```

Bằng việc trực quan hóa dữ liệu, ta thấy các giá trị của từng biến trong bộ dữ liệu đều bị lệch về phía bên phải. Để chuẩn hóa các giá trị trên, ta sử dụng hàm scale() hay còn được gọi là Z chuẩn hóa, đưa tất cả các biến về cùng một dạng thang đo có giá trị trung bình bằng 0 và độ lệch chuẩn là 1.

Trước hết, tách biến mục tiêu thành 1 cột riêng và chỉ chuẩn hóa trên các biến độc lập. Sau khi đã chuẩn hóa xong ta đưa hai biến X (chứa các biến độc lập đã được chuẩn hóa) và biến y (chứa biến mục tiêu) vào dataframe trở lại.

```
#Remove the target column from data frame  
X <- new_df[, -which(names(new_df) %in% c("Revenue"))]  
y <- new_df[, "Revenue"]  
  
#Feature scaling  
X <- scale(X)  
  
#Add into dataframe again  
new_df <- as.data.frame(X)  
new_df$Revenue <- y
```

8.3.1.3. Xây dựng mô hình

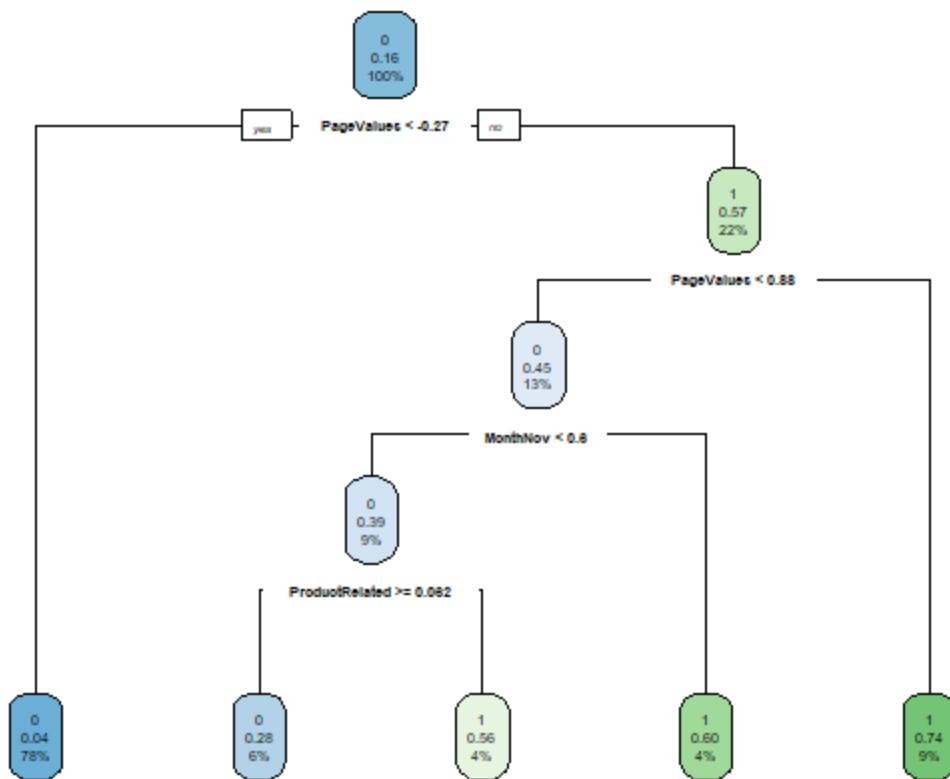
Ta chia bộ dữ liệu thành 2 tập train và test với tỷ lệ 7:3 với mục đích xây dựng mô hình trên tập train và kiểm tra lại kết quả trên tập test.

Trong R, để xây dựng mô hình cây quyết định ta sử dụng thư viện rpart, một thư viện máy học mạnh mẽ dùng để xây dựng cây phân loại và hồi quy. Rpart hoạt động bằng cách chia nhỏ dữ liệu theo đệ quy, nghĩa là các tập hợp con phát sinh trong quá trình phân chia sẽ được chia nhỏ thêm cho đến khi đạt được tiêu chí kết thúc đã xác định trước đó. Các quy tắc phân tách có thể được xây dựng theo nhiều cách khác nhau nhưng đều dựa trên khái niệm về tạp chất - một thước đo mức độ không đồng nhất của các nút lá. Nói cách khác, một nút lá chứa một lớp duy nhất là đồng nhất và có tạp chất bằng 0.

Ta sử dụng method = “class” để xây dựng mô hình cây quyết định dựa trên phương pháp phân loại vào các lớp 0 và 1 của biến mục tiêu. Ngoài ra, còn có khía khác như “anova”, “poison”, “exp”.

```
#Create model  
model_DT = rpart(Revenue ~ ., data = train, method='class')  
  
#Draw decision tree  
rpart.plot(model_DT)
```

Sau khi đã có mô hình ta sẽ trực quan hóa thành cây quyết định.



Hình 8.12. Mô hình cây quyết định bằng R

8.3.1.4. Đánh giá mô hình

Để kiểm tra lại mô hình ta sử dụng hàm predict() để dự đoán kết quả trên tập test.

```

> #Predict on test set
> pred_values = predict(model_DT, test, type='class')
> pred_values
   1   2   4   5   10   11   17   18   23   24   26   29   35   41   45   49   55   61 
   0   0   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0    0 
  63   65   67   70   71   72   74   75   76   79   83   86   94   102   109   113   118   126 
   0   0   0   0   0    0    0    0    0    0    0    0    0    0    1    0    0    0 
 127  129  133  135  136  137  141  145  147  157  158  163  168  169  171  178  180  187 
   0   0   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 189  192  197  203  204  209  212  213  217  221  223  227  232  235  238  244  248  255 
   0   0   1   0   0    0    0    0    0    1    0    0    0    0    0    0    0    0 
 256  262  264  265  266  269  279  282  284  288  291  294  295  297  298  300  301  303 
   0   0   0   0   1    0    0    1    0    0    0    0    1    0    0    0    0    0 
 307  309  312  315  318  320  322  339  341  342  344  345  347  350  354  357  362  364 
   0   0   0   0   0    0    0    0    0    0    0    0    0    0    0    0    0    0

```

Tiếp theo, ta sử dụng hàm confusionMatrix() để in ra các kết quả đánh giá độ chính xác của mô hình như Recall, Precision, F1-score.

```

> #Evaluation metrics
> cm <- confusionMatrix(pred_values, as.factor(test$Revenue))
> cm
Confusion Matrix and Statistics

Reference
Prediction   0      1
      0 2987  222
      1  132  358

Accuracy : 0.9043
95% CI  : (0.8944, 0.9136)
No Information Rate : 0.8432
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6137

McNemar's Test P-Value : 2.242e-06

Sensitivity : 0.9577
Specificity : 0.6172
Pos Pred Value : 0.9308
Neg Pred Value : 0.7306
Prevalence : 0.8432
Detection Rate : 0.8075
Detection Prevalence : 0.8675
Balanced Accuracy : 0.7875

'Positive' Class : 0

```

Ta có thể thấy mô hình đã dự đoán đúng 2987 người không mua hàng và 258 người đã mua hàng. Độ chính xác của mô hình 90%, một kết quả tương đối cao.

8.3.2. Mô hình Random Forest

8.3.2.1. Cài đặt thư viện

Để có thể xây dựng mô hình Random Forest ta cần phải cài đặt các thư viện “randomForest”, “ggplot2”, “caret” và “e1071”.

8.3.2.2. Tiền xử lý dữ liệu

Ta sẽ chia tập dữ liệu thành 2 tập train và test. Tập train sẽ dùng để xây dựng và huấn luyện mô hình, tập test sẽ dùng để kiểm định lại mô hình. Phân chia dữ liệu theo tỉ lệ 7:3, 7 cho train set và 3 cho test set.

```

17 #Split Data
18 data$Revenue = as.factor(data$Revenue)
19 data_set_size = floor(nrow(data)*.7)
20 index <- sample(1:nrow(data), size = data_set_size)
21 training <- data[index,]
22 testing <- data[-index,]

```

Tương tự như Python, ta thực chuyển 2 cột “Revenue” và “Weekend” về giá trị nhị phân 0 và 1, sau đó thực hiện mã hóa các cột biến phân loại.

```
#Convert 'Revenue' & 'Weekend' to binary number  
data$Revenue <- ifelse(data$Revenue == "TRUE", 1, 0)  
data$Weekend <- ifelse(data$Weekend == "TRUE", 1, 0)  
  
#Encoding categorical variables  
dummy <- dummyVars(~ ., data=data)  
df <- data.frame(predict(dummy, newdata=data))  
  
head(data)
```

Tiếp theo ta chuẩn hóa dữ liệu bằng hàm scale().

```
#Standardize Data  
dat <- data.frame(x = rnorm(10, 30, .2), y = runif(10, 3, 5))  
scaled.dat <- scale(dat)
```

8.3.2.3. Xây dựng mô hình

Ta cũng tiến hành chia dữ liệu thành hai tập train, test với tỷ lệ 7:3. Sau đó dùng hàm randomForest() để xây dựng mô hình trên tập train.

```
#Split Data  
data$Revenue = as.factor(data$Revenue)  
data_set_size = floor(nrow(data)*.7)  
index <- sample(1:nrow(data), size = data_set_size)  
train <- data[index,]  
test <- data[-index,]  
  
#Create the forest.  
model_RF <- randomForest(Revenue ~ ., data=train, importance = TRUE)
```

8.3.2.4. Đánh giá mô hình

Sau khi xây dựng mô hình ta dùng hàm predict() để so sánh kết quả dự đoán với kết quả thật trên tập test.

```

> #Predict on test set
> pred_test = predict(model_RF, test)
> pred_test
   6    7   12   19   20   26   29   31   36   38   41   46   49   51   52   53   54   58
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
  61   62   68   70   72   75   76   77   81   87   89   91   92   95   99   100   103   107
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
 110  112  115  117  120  129  132  133  136  139  141  144  147  152  153  155  158  159
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
 165  172  179  188  189  192  193  196  197  208  209  212  217  218  220  222  223  227
   0    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0
 230  231  232  234  237  240  242  250  251  260  263  267  268  274  275  276  277  283
   0    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0
 285  297  300  309  317  326  332  334  336  340  345  346  350  351  361  363  364  365
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

```

Dùng hàm confusionMatrix() để xem kết quả dự đoán và các phương pháp đánh giá mô hình.

```

#Confusion matrix
confusionMatrix(pred_test, test$Revenue, mode = 'everything')

Confusion Matrix and Statistics

             Reference
Prediction      0      1
      0 2988  240
      1 125   346

Accuracy : 0.9013
 95% CI : (0.8913, 0.9108)
No Information Rate : 0.8416
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.5979

McNemar's Test P-Value : 2.416e-09

Sensitivity : 0.9598
Specificity : 0.5904
Pos Pred Value : 0.9257
Neg Pred Value : 0.7346
Precision : 0.9257
Recall : 0.9598
F1 : 0.9424
Prevalence : 0.8416
Detection Rate : 0.8078
Detection Prevalence : 0.8727
Balanced Accuracy : 0.7751

'Positive' Class : 0

```

Ta thấy mô hình đã dự đoán 2988 người không mua hàng và 346 người mua hàng với độ chính xác của mô hình là 90%. Do sự mất cân bằng giữa các lớp trong bộ dữ liệu nên lớp 0 chiếm ưu thế hơn so với lớp 1.

8.3.2.5. Xếp hạng thuộc tính

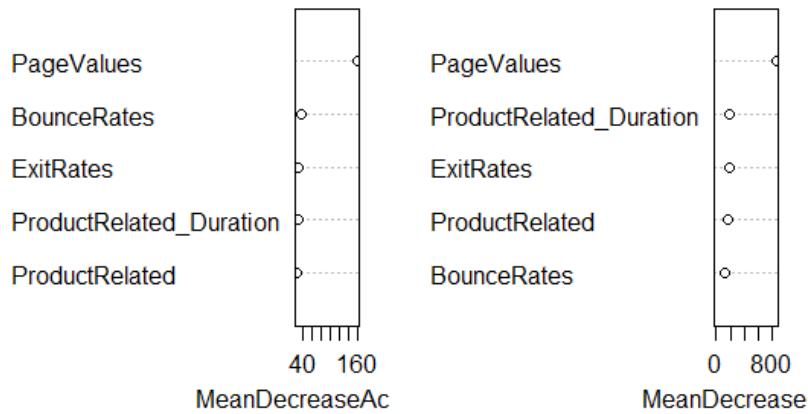
Việc lựa chọn ra các thuộc tính quan trọng là điều cần thiết để cải thiện độ chính xác của mô hình và kết quả dự đoán. Mức độ quan trọng của từng thuộc tính được đánh giá trên 2 tiêu chí:

- **MeanDecreaseAccuracy:** biểu thị mức độ chính xác mà mô hình bị mất đi khi loại trừ từng biến ra khỏi mô hình. Độ chính xác càng giảm thì mức độ quan trọng của biến đó với mô hình càng cao.
- **MeanDecreaseGini:** là thước đo cách mỗi biến đóng góp vào sự đồng nhất của các nút và lá trong kết quả của mô hình Random Forest. Giá trị của MeanDecreaseGini càng cao thì tầm quan trọng của biến trong mô hình càng lớn.

Trong R, ta sẽ sử dụng hàm varImpPlot() để trực quan hóa top 5 thuộc tính quan trọng theo mức độ giảm dần trong bộ dữ liệu để cải thiện mô hình.

```
#Variable importance  
varImpPlot(model_RF,  
            sort = T,  
            n.var = 5,  
            main = "Top 5 - Variable Importance")
```

Top 5 - Variable Importance



Hình 8.13. Top 5 thuộc tính quan trọng trong mô hình Random Forest

Ta dùng hàm importance() in ra giá trị MeanDecreaseAccuracy và MeanDecreaseGini của toàn bộ các biến để có cái nhìn tổng quan hơn.

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
Administrative	27.043107	-4.9348300	25.3792529	95.014987
Administrative_Duration	26.347527	-7.4316311	24.0915227	128.477574
Informational	11.660673	2.3189938	12.4976181	38.328809
Informational_Duration	15.190305	-0.7757112	15.4535095	58.918602
ProductRelated	21.827674	11.8400324	27.5764187	164.539239
ProductRelated_Duration	24.252506	7.2812022	29.5467662	198.358208
BounceRates	24.040595	24.0961428	37.6915439	128.547544
ExitRates	17.266222	34.2005096	30.6929038	197.164174
PageValues	95.673314	236.3639944	159.6705358	850.730646
SpecialDay	-4.167761	6.5051732	0.7667346	7.610919
Month	-6.392730	38.6826197	18.5111381	100.222151
OperatingSystems	3.546363	-0.4538799	2.8377077	37.030935
Browser	6.842165	-1.9413561	5.1081739	42.202802
Region	1.358417	2.5191367	2.5091661	66.544103
TrafficType	4.121967	6.4826292	7.1662100	67.593036
VisitorType	13.870713	3.7362030	14.0878079	23.631364
Weekend	5.375401	4.1820739	6.6864658	22.836807

Dựa vào Hình 8.13, ta có thể thấy biến “PageValues” có vai trò quan trọng lớn nhất đối với mô hình với MeanDecreaseGini bằng 850.73, nếu loại biến này ra khỏi mô hình thì độ chính xác giảm 159.67%. Các biến quan trọng khác lần lượt là “BounceRates”, “ExitRates”, “ProductRelated_Duration”, “ProductRelated”.

8.3.3. Mô hình Neural Network

8.3.3.1. Cài đặt thư viện

Sử dụng thư viện pacman để dễ dàng cài đặt và khai báo các thư viện khác, bao gồm: neuralnet (mô hình neural network), fastDummies (phục vụ one-hot encoding), corrplot, RColorBrewer và ggplot2 (cho trực quan hóa dữ liệu), rio (nạp dữ liệu), ROCR (đánh giá mô hình máy học) phục vụ mục đích khai phá dữ liệu và huấn luyện cũng như kiểm định mô hình.

```

# Import pacman (package manager)
if (!require("pacman")) install.packages("pacman")

# Use pacman to load add-on packages as desired
pacman::p_load(pacman, neuralnet, fastDummies,
                corrplot, RColorBrewer, rio, ROCR, ggplot2)

```

8.3.3.2. Tiền xử lý dữ liệu

Đầu tiên, ta chuyển hóa các cột chứa giá trị có kiểu dữ liệu logic (True/False) sang kiểu dữ liệu số tương ứng (1/0).

```

# Convert Logical Value into Integer
data$Weekend <- as.integer(as.logical(data$Weekend))
data$Revenue <- as.integer(as.logical(data$Revenue))

```

Tiếp theo, chúng ta chuyển hóa các cột chứa dữ liệu phân lớp sang các cột chứa dữ liệu số (1/0) bằng hàm dummy_cols() từ thư viện fastDummies. Sau đó ta bỏ các cột cũ đi.

```

# One-hot Encoding
data_d <- dummy_cols(data)
data_d <- data_d[, -which(names(data) %in% c("Month", "VisitorType"))]
head(data_d)

```

Tiếp theo ta tách cột Revenue (là cột chứa nhãn của Dataset) sang biến y, và để các cột còn lại (là các cột chứa các đặc trưng của Dataset) sang biến X.

```

# Remove the target column Revenue from data frame
X <- data_d[, -which(names(data_d) %in% c("Revenue"))]
y <- data_d[, "Revenue"]

```

Sau đó, ta tiếp tục chia tệp X và y thành các tệp X_train, y_train, X_test và y_test theo tỷ lệ train:test là 7:3, trong đó hai tệp X_train và y_train sẽ phục vụ mục đích huấn luyện mô hình và hai tệp X_test và y_test sẽ phục vụ mục đích kiểm định mô hình.

```

# Split train and test set
sample_size = 0.70 * nrow(data)
set.seed(1234)
index = sample(seq_len( nrow(data_d) ), size = sample_size)
X_train <- X[index, ]
y_train <- y[index]
X_test <- X[-index, ]
y_test <- y[-index]

```

Ta tiến hành chuẩn hóa các đặc trưng của dữ liệu với hàm scale() của R. Vì yêu cầu của thư viện neuralnet, ta phải dựng tập train hoàn chỉnh bằng cách kết hợp tập y_train với tập X_train sau khi đã scale xong.

```

# Data standardization
X_train <- scale(X_train)
X_test <- scale(X_test)
train <- as.data.frame(X_train)
train$Revenue <- y_train
head(train)

```

Do yêu cầu của thư viện neuralnet, ta dựng lại tập train hoàn chỉnh bằng cách chuyển hóa tập X_train lại thành một dataframe, tạo một cột Revenue mới, và thêm tập y_train vào cột này. Câu lệnh head() giúp in ra tập train mới được tạo để kiểm tra lại.

8.3.3.3. Xây dựng mô hình

Để có thể xây dựng mô hình mạng nơ ron bằng thư viện neuralnet, ta phải tạo ra một biến thuộc kiểu formula của R để xác định công thức tính toán cho mạng nơ ron. Ở đây, vì dùng toàn bộ đặc trưng có trong tập X_train, ta sẽ ghép toàn bộ tên cột của X_train vào chuỗi thành phần của công thức, và dùng hàm as.formula để chuyển hóa chuỗi này thành công thức trong R. Sau đó, ta sử dụng thư viện neuralnet để dựng mô hình mạng nơ ron, đưa vào đó tập train để mô hình được huấn luyện. Thông số linear.output = FALSE mang nghĩa mô hình là mô hình phân lớp (classifier), không phải mô hình hồi quy (regression).

```
# Create model
formula <- paste("Revenue ~ ", paste(colnames(X_train), collapse= " + "))
fmla <- as.formula(formula)
NN <- neuralnet(fmla, data=train, linear.output=FALSE) # For classification
```

8.3.3.4. Đánh giá mô hình

Sử dụng hàm compute(), ta sử dụng mô hình vừa huấn luyện dự đoán kết quả cho tập đặc trưng X_test.

```
# Predict on the test set
results = compute(NN, X_test)
```

Sử dụng kết quả dự đoán vừa thu được từ mô hình, ta tiến hành đổi chiều với nhãn thực tế được lưu trong tập y_test.

```
# Actual value and predicted value
prob_results <- results$net.result
predict_results <- data.frame(actual = y_test, prediction = prob_results)
head(predict_results)
rounded_results<-sapply(predict_results,round,digits=0)
rounded_results_df <- data.frame(rounded_results)
rounded_results_df

> head(rounded_results_df)
  actual prediction
1       0           0
2       0           0
3       0           0
4       0           0
5       0           0
6       0           0
```

Để kiểm định mô hình, ta trích xuất Confusion Matrix từ dữ liệu của bảng trên. Hàm attach và hàm table giúp gom bảng trên lại theo lớp 0 và 1 như confusion matrix.

```
# Confusion Matrix
attach(rounded_results_df)
table(actual,prediction)
```

```
> table(actual,prediction)
      prediction
actual      0     1
  0 2970 155
  1 210 364
```

Số lượng điểm dữ liệu được dự đoán đúng lần lượt vào lớp 0 và 1 là 2970 và 364. Số lượng điểm dữ liệu được dự đoán sai vào lớp 0 là 155 và số lượng điểm dữ liệu được dự đoán sai vào lớp 1 là 210.

Để kiểm định mô hình, ta sử dụng thư viện ROCR để trích xuất cái giá trị Accuracy, Precision, Recall, AUC và ROC của mô hình. Ta gọi hàm ROCR::prediction để thực hiện trích xuất đánh giá từ kết quả dự báo của mô hình và kết quả thực tế lưu trong tập y_test.

```
# Evaluation Metrics
nn_pred <- ROCR::prediction(prob_results, y_test)
```

Lần lượt sử dụng hàm performance() của thư viện ROCR để trích xuất các giá trị Accuracy, Precision, Recall, AUC và ROC của mô hình.

```
# Accuracy
acc.perf = performance(nn_pred, measure = "acc")
ind = which.max( slot(acc.perf, "y.values")[[1]] )
acc = slot(acc.perf, "y.values")[[1]][ind]
cutoff = slot(acc.perf, "x.values")[[1]][ind]
print(c(accuracy= acc, cutoff = cutoff))
```

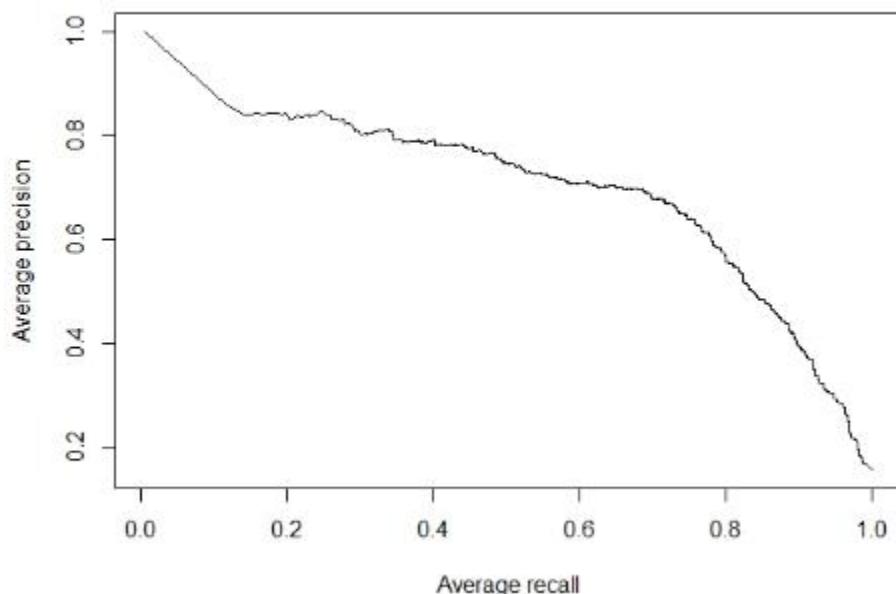
Chuỗi câu lệnh thực hiện việc trích xuất giá trị Accuray cao nhất mà mô hình đạt được và cutoff của giá trị xác suất trong tệp nhãn được dự đoán để mô hình đạt được chỉ số Accuracy đó.

```
> print(c(accuracy= acc, cutoff = cutoff))
accuracy      cutoff
0.9051095 0.4256681
```

Chỉ số Accuracy của mô hình là xấp xỉ 0.91 với giá trị cutoff của xác suất là xấp xỉ 0.43, tức nghĩa với giá trị xác suất cao hơn 0.43 thì sẽ được coi là lớp 1 còn các giá trị nhỏ hơn sẽ được coi là lớp 0.

```
# Precision and Recall
```

```
RP.perf <- performance(nn_pred, "prec", "rec")
plot(RP.perf,
     avg="threshold")
```

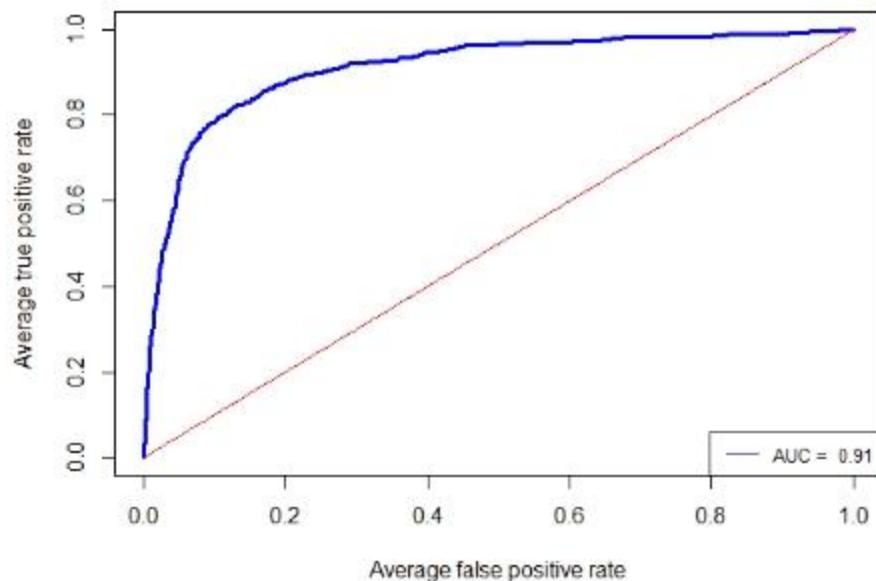


Hình 8.14. Biểu đồ thể hiện đường cong Precision - Recall của mô hình mạng nơ ron

Tiếp tục sử dụng hàm performance, ta trích xuất ra các giá trị TPR và FPR để hỗ trợ việc vẽ biểu đồ của đường cong ROC.

```
# ROC AUC
AUC_label = paste("AUC = ", round(as.numeric(auc.perf@y.values), digits=2))
perf <- ROCR::performance(nn_pred, "tpr", "fpr")

# Visualize ROC Curve
plot(perf, type="l", col = "blue", avg="threshold", lwd=3)
lines(y_test, y_test, type="l", lty=2, lwd=1, col = "red", pch=18)
legend("bottomright", legend=c(AUC_label),
       col=c("blue"), lty=1:2, cex=0.8)
```



Hình 8.15. Biểu đồ đường cong ROC của mô hình Neural Network bằng R

Chỉ số AUC lớn (0.91) chứng tỏ mô hình có độ chính xác cao và đáng tin cậy.

8.3.4. Mô hình Support Vector Machine

8.3.4.1. Xây dựng mô hình.

Tương tự như ở Python, sau khi đã hoàn thành hết các bước tiền xử lý dữ liệu như kiểm tra, chuẩn hóa và phân chia các tập train test, chúng ta lại xây dựng mô hình phân lớp SVM trên nền tảng Rstudio bằng ngôn ngữ R. Ở đây, chúng ta sử dụng tham số kernel dạng tuyến tính để xây dựng mô hình phân lớp SVM.

```
#Create SVM model
model_svm = svm(formula = Revenue ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = 'linear')

print(model_svm)
```

8.3.4.2. Đánh giá mô hình

Tiến hành kiểm định mô hình, ta sẽ kiểm tra trên tập test đã được phân tách ra trước đó bằng đoạn lệnh sau.

```
#Predicting the Test set results
predTest = predict(model_SVM,test,type="response")
predTest

> predTest = predict(model_SVM, test, type='response')
> predTest
   12   14   15   16   18   20   22   24   26   27   29   35   40   41   51   57   61   66   67   74   75   78   79
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
   81    90    94    97    99   100   104   107   108   112   113   118   121   123   124   132   136   141   148   151   154   164   168
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 173   186   187   188   189   192   194   195   197   202   204   207   215   223   226   234   235   237   238   239   240   242   243
   0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 252   253   256   263   265   266   270   274   277   281   282   283   285   287   288   289   291   293   295   298   299   302   307
   0     0     0     0     0     1     0     0     1     0     0     0     0     0     0     0     0     0     1     0     0     0     0
 314   315   321   328   342   346   347   353   357   361   372   374   375   381   382   384   385   391   395   396   398   399   401
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     1
 403   405   408   412   413   415   416   419   424   428   431   433   434   437   449   452   453   455   458   460   461   462   465
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
 467   472   479   489   493   497   500   501   502   504   505   506   508   518   519   520   521   522   523   528   529   533   534
   0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 540   541   543   545   547   550   551   561   562   564   568   570   575   576   582   583   586   594   595   596   600   602   604
   0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

Tiếp theo đến ma trận hỗn loạn của mô hình, ta lại tiến hành chạy đoạn lệnh và thu được kết quả.

```
#Confusion matrix
pred.train.dt <- predict(model_SVM,test,type = "class")
mean(pred.train.dt==test$Revenue)

cm <- table(pred.train.dt,test$Revenue)
cm

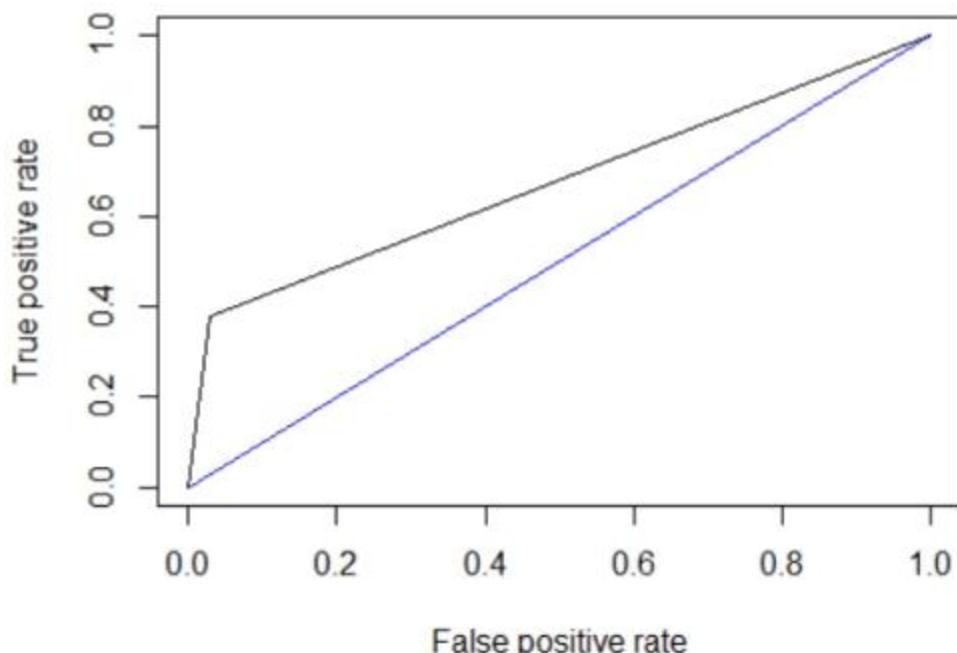
> pred.train.dt <- predict(model_SVM,test,type = "response")
> mean(pred.train.dt==test$Revenue)
[1] 0.87402
>
> cm <- table(pred.train.dt,test$Revenue)
> cm

pred.train.dt      0      1
          0 3004  377
          1   89  229
>
```

Như vậy, với 3381 người dự đoán không mua hàng, mô hình phân lớp SVM đã dự đoán chính xác 3004 trong số họ, chiếm 88%. Tương tự với số người được đoán mua hàng, ta lại có 229 người được dự đoán chính xác trên con số 318 người, chiếm 72%.

Sử dụng thư viện ROCR, chúng ta tiếp tục kiểm tra độ tin cậy của mô hình này thông qua trực quan đường cong ROC bằng đoạn lệnh dưới đây.

```
#ROC Curve
p <- predict(model_SVM ,test, type="decision")
p <- as.numeric(p)-1
pr <- prediction(p, test$Revenue)
pref <- performance(pr, "tpr", "fpr")
plot(pref, AUC = 'True')
lines(x = c(0,1), y = c(0,1), col="blue")
```



Hình 8.16. Biểu đồ đường cong ROC của mô hình SVM bằng R

Ta dễ dàng nhận thấy đường cong này còn cách điểm 1 trên trực true positive rate rất xa. Suy ra, chúng ta không có căn cứ nào để đánh giá cao độ tin cậy của nó được.

CHƯƠNG 9. TỔNG KẾT

Dù vẫn còn hạn chế về mặt kiến thức nhưng với sự nỗ lực và tìm hiểu kỹ lưỡng của các thành viên, nhóm đã hoàn thành các yêu cầu trong đồ án đề ra. Qua đồ án lần này, nhóm đã có thêm kiến thức trong việc sử dụng Python và R. Nhóm cũng đã vận dụng được những kiến thức đã học trên lớp vào việc phân tích dự báo với các mô hình hồi quy, time series và các mô hình máy học. Từ đó, nhóm đã được củng cố lại kiến thức, có cơ hội tiếp cận với các kỹ thuật phân tích và mô hình máy học, phát triển kỹ năng phân tích dữ liệu cũng như cách vận dụng các mô hình vào thực tế.

Trong quá trình phân tích dự báo, nhóm vẫn chưa thực sự đào sâu vào việc làm cho mô hình trở nên tối ưu nhất. Do đó, nhóm cần tập trung vào việc phân tích kỹ hơn ảnh hưởng của từng biến độc lập lên biến mục tiêu, cải thiện các tham số của mô hình để gia tăng độ chính xác kết quả dự đoán của mô hình.

Bên cạnh đó, các mô hình máy học trong đề tài này được huấn luyện thông qua dữ liệu từ cùng một nguồn (chỉ trong một năm). Điều này có nghĩa là dữ liệu được sử dụng nhất quán nhưng ảnh hưởng đến tính tổng quát hóa của các mô hình. Vì vậy, nhóm chúng sẽ xem xét việc thu thập thêm dữ liệu hành vi của khách hàng trong vài năm tiếp theo nhằm tạo ra sự vượt trội cho dữ liệu.

TÀI LIỆU THAM KHẢO

- [1] “Bài 1. Giới thiệu về hồi quy và tương quan,” [Trực tuyến]. Available: <https://nghiencuugiaoduc.com.vn/bai-1-gioi-thieu-ve-hoi-quy-va-tuong-quan/>. [Đã truy cập 09 August 2021].
- [2] “Phân tích hồi qui tuyến tính với SPSS,” [Trực tuyến]. Available: <http://bis.net.vn/forums/t/722.aspx>. [Đã truy cập 09 August 2021].
- [3] “Hồi quy tuyến tính (Triển khai trên Python),” [Trực tuyến]. Available: <https://cafedev.vn/tu-hoc-ml-hoi-quy-tuyen-tinh-trien-khai-tren-python/>. [Đã truy cập 09 August 2021].
- [4] “Correlation (tương quan) & Linear regression (hồi quy tuyến tính),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/tuong-quan-va-hoi-quy-tuyen-tinh-don-gian.html>. [Đã truy cập 10 August 2021].
- [5] “Tổng quan về Logistic regression (hồi quy Logistic) (Phần 1),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/tong-quan-ve-logistic-regression-hoi-quy-logistic-phan-1.html>. [Đã truy cập 12 August 2021].
- [6] “Tổng quan về Logistic regression (hồi quy Logistic) (Phần 2),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/tong-quan-ve-logistic-regression-hoi-quy-logistic-phan-2.html>. [Đã truy cập 12 August 2021].
- [7] “Tổng quan về Logistic regression (hồi quy Logistic) (Phần 3),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/tong-quan-ve-logistic-regression-hoi-quy-logistic-phan-3.html>. [Đã truy cập 12 August 2021].
- [8] “Dữ liệu chuỗi thời gian,” [Trực tuyến]. Available: https://machinelearningcoban.com/tabml_book/ch_data_processing/timeseries_data.html. [Đã truy cập 11 August 2021].

- [9] “Tìm hiểu về Time series (phân tích chuỗi thời gian) (P.1),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/tim-hieu-ve-time-series-phan-tich-chuoi-thoi-gian-p-1.html>. [Đã truy cập 11 August 2021].
- [10] “Recurrent Neural Network(Phần 1): Tổng quan và ứng dụng,” [Trực tuyến]. Available: <https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw>. [Đã truy cập 12 August 2021].
- [11] “[RNN] RNN là gì?,” [Trực tuyến]. Available: <https://dominhhai.github.io/vi/2017/10/what-is-rnn/>. [Đã truy cập 12 August 2021].
- [12] “Cây Quyết Định (Decision Tree),” [Trực tuyến]. Available: <https://trituenhantao.io/kien-thuc/decision-tree/>. [Đã truy cập 15 August 2021].
- [13] “Bài 34: Decision Trees (1): Iterative Dichotomiser 3,” [Trực tuyến]. Available: <https://machinelearningcoban.com/2018/01/14/id3/>. [Đã truy cập 14 August 2021].
- [14] “<https://bigdatauni.com/tin-tuc/thuat-toan-cay-quyet-dinh-classification-regression-tree-cart-p-1.html>,” [Trực tuyến]. Available: Thuật toán cây quyết định (P1) – Classification & Regression tree (CR&T). [Đã truy cập 14 August 2021].
- [15] “Thuật toán cây quyết định (P.2): CART (Gini index),” [Trực tuyến]. Available: <https://bigdatauni.com/tin-tuc/thuat-toan-cay-quyet-dinh-p-2-cart-gini-index.html>. [Đã truy cập 15 August 2021].
- [16] “Cây quyết định (Decision Tree) là gì? Tìm hiểu Thuật toán ID3,” [Trực tuyến]. Available: <https://1upnote.me/post/2018/10/ds-ml-decision-tree-id3/>. [Đã truy cập 15 August 2021].
- [17] “Thuật toán Random Forests,” [Trực tuyến]. Available: <https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz>. [Đã truy cập 16 August 2021].
- [18] “<https://ichi.pro/vi/thuat-toan-rung-ngau-nhien-58574608449016>,” [Trực tuyến]. Available: <https://ichi.pro/vi/thuat-toan-rung-ngau-nhien-58574608449016>. [Đã truy cập 16 August 2021].

- [19] “Bài 19: Support Vector Machine,” [Trực tuyến]. Available:
<https://machinelearningcoban.com/2017/04/09/sm/>. [Đã truy cập 19 August 2021].
- [20] “Giới thiệu về Support Vector Machine trong Machine Learning,” [Trực tuyến]. Available: <https://www.noron.vn/post/gioi-thieu-ve-support-vector-machine-trong-machine-learning-40dxtjcmrdye>. [Đã truy cập 19 August 2021].
- [21] “Support Vector Machine (SVM) là gì?,” [Trực tuyến]. Available:
<https://1upnote.me/post/2018/10/ds-ml-svm/>. [Đã truy cập 19 August 2021].
- [22] “Tổng quan về Neural Network(mạng Nơ Ron nhân tạo) là gì?,” [Trực tuyến]. Available:
<https://itnavi.com.vn/blog/neural-network-la-gi/>. [Đã truy cập 18 August 2021].
- [23] “Tổng quan về Neural Network(mạng Nơ Ron nhân tạo) là gì?,” [Trực tuyến]. Available:
<https://itnavi.com.vn/blog/neural-network-la-gi/>. [Đã truy cập 17 August 2021].
- [24] “Bài 33: Các phương pháp đánh giá một hệ thống phân lớp,” [Trực tuyến]. Available:
<https://machinelearningcoban.com/2017/08/31/evaluation/>. [Đã truy cập 22 August 2021].
- [25] “Precision, Recall và F1-score là gì?,” [Trực tuyến]. Available:
<https://caihuuthuc.wordpress.com/2020/02/23/precision-recall-va-f1-score-la-gi/>. [Đã truy cập 22 August 2021].
- [26] “Bài 46 - Đánh giá mô hình phân loại trong ML,” [Trực tuyến]. Available:
<https://phamdinhkhanh.github.io/2020/08/13/ModelMetric.html>. [Đã truy cập 21 August 2021].
- [27] “Linear Regression — Detailed View,” [Trực tuyến]. Available:
<https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86>. [Đã truy cập 11 August 2021].
- [28] “Logistic Regression in R Tutorial,” [Trực tuyến]. Available:
<https://www.datacamp.com/community/tutorials/logistic-regression-R>. [Đã truy cập 23 August 2021].

- [29] “Logistic Regression – A Complete Tutorial With Examples in R,” [Trực tuyến]. Available: <https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/>. [Đã truy cập 22 August 2021].
- [30] “Recurrent Neural Networks cheatsheet,” [Trực tuyến]. Available: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>. [Đã truy cập 20 August 2021].
- [31] “A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks,” [Trực tuyến]. Available: <https://builtin.com/data-science/recurrent-neural-networks-and-lstm>. [Đã truy cập 20 August 2021].
- [32] “Recursive Feature Elimination (RFE) for Feature Selection in Python,” [Trực tuyến]. Available: <https://machinelearningmastery.com/rfe-feature-selection-in-python/>. [Đã truy cập 17 August 2021].
- [33] “Variance Inflation Factor,” [Trực tuyến]. Available: <https://www.statisticshowto.com/variance-inflation-factor/>.
- [34] “Multiple Linear Regression,” [Trực tuyến]. Available: <https://corporatefinanceinstitute.com/resources/knowledge/other/multiple-linear-regression/>. [Đã truy cập 12 August 2021].
- [35] “An introduction to multiple linear regression,” [Trực tuyến]. Available: scribbr.com/statistics/multiple-linear-regression/. [Đã truy cập 13 August 2021].
- [36] “Introduction to Confusion Matrix in Python Sklearn,” [Trực tuyến]. Available: <https://intellipaat.com/blog/confusion-matrix-python/>. [Đã truy cập 15 August 2021].
- [37] “Decision Trees in R,” [Trực tuyến]. Available: <https://www.datacamp.com/community/tutorials/decision-trees-R>. [Đã truy cập 14 August 2021].
- [38] “Decision Tree in R Programming,” [Trực tuyến]. Available: <https://www.geeksforgeeks.org/decision-tree-in-r-programming/>. [Đã truy cập 14 August 2021].

- [39] “Decision Tree Classification in Python,” [Trực tuyến]. Available:
<https://www.datacamp.com/community/tutorials/decision-tree-classification-python>. [Đã truy cập 12 August 2021].
- [40] “Machine Learning - Decision Tree,” [Trực tuyến]. Available:
https://www.w3schools.com/python/python_ml_decision_tree.asp. [Đã truy cập 12 August 2021].
- [41] “Understanding Random Forests Classifiers in Python,” [Trực tuyến]. Available:
<https://www.datacamp.com/community/tutorials/random-forests-classifier-python>. [Đã truy cập 13 August 2021].
- [42] “Random Forest in R,” [Trực tuyến]. Available: <https://www.r-bloggers.com/2021/04/random-forest-in-r/>. [Đã truy cập 13 August 2021].
- [43] “Support Vector Machines in R,” [Trực tuyến]. Available:
<https://www.datacamp.com/community/tutorials/support-vector-machines-r>. [Đã truy cập 16 August 2021].
- [44] “Some R Packages for ROC Curves,” [Trực tuyến]. Available:
<https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/>. [Đã truy cập 18 August 2021].
- [45] “Linear Regression R,” [Trực tuyến]. Available:
<https://www.datacamp.com/community/tutorials/linear-regression-R>. [Đã truy cập 17 August 2021].