

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THÔNG THÔNG TIN



BÁO CÁO ĐỒ ÁN

MÔN HỌC: KHAI THÁC DỮ LIỆU

Giảng viên hướng dẫn: **Ths. Mai Xuân Hùng**

GV. Phạm Nguyễn Thanh Bình

Tên đề tài: **DỰ ĐOÁN KHẢ NĂNG HỦY ĐẶT PHÒNG**

CỦA HAI KHÁCH SẠN TẠI BỒ ĐÀO NHA (2015 – 2017)

Lớp: **IS252.021**

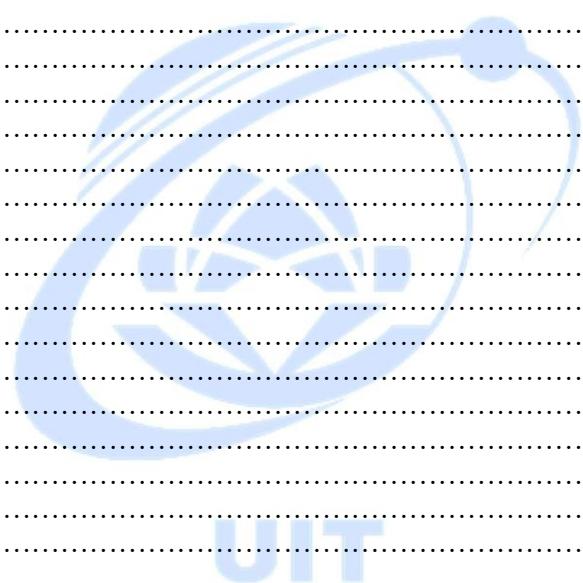
Phạm Quốc Hùng - 19521579

Lê Trần Anh Quý - 21520094

Đặng Lưu Hà - 21520798

Trần Minh Quang - 21522519

NHẬN XÉT CỦA GIẢNG VIÊN



MỤC LỤC

DANH MỤC HÌNH ẢNH.....	6
DANH MỤC BẢNG BIỂU.....	10
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI.....	11
1. Lý do chọn đề tài	11
2. Giới thiệu nguồn dữ liệu	11
3. Lọc dữ liệu	12
4. Mô tả thuộc tính	15
5. Giới thiệu bài toán.....	20
CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU.....	22
1. Trực quan hóa dữ liệu:	22
1.1. Thông kê số lượng nhận phòng và hủy phòng:	22
1.2. Thông kê loại hình khách sạn nào thu nhiều khách hàng nhất.....	23
1.3. Thông kê doanh thu mỗi tháng trong tập dữ liệu	23
1.4. Thông kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc	27
1.5. Thông kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng.	27
1.6. Tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công.....	29
1.7. Thông kê tỷ lệ phòng bị hủy	29
1.8. Kênh phân phối nào có tỷ lệ hủy cao nhất.....	30
1.9. Thể hiện hệ số tương quan giữa giá trị hủy đặt phòng và các thuộc tính còn lại.....	31
2. Mô tả cơ bản về tập dữ liệu:.....	33
3. Làm sạch dữ liệu:	36
3.1. Xử lý dữ liệu bị nhiễu (outlier):	36
3.2. Xử lý dữ liệu bị thiếu:	38
3.3. Kiểm tra ý nghĩa của dữ liệu:	41
3.4. Kiểm tra thuộc tính của dữ liệu:.....	43
3.5. Xử lý dữ liệu bị trùng lặp:	44
4. Thêm thuộc tính cho dữ liệu:	45

5. Giảm thuộc tính của dữ liệu:	47
6. Rời rạc hóa dữ liệu:	49
6.1. Thuộc tính LeadTime:	50
6.2. Thuộc tính Adults:.....	51
6.3. Thuộc tính ADR:	52
6.4. Thuộc tính StatusMinusArrivalDate:	52
6.5. Thuộc tính TotalNights:	53
6.6. Thuộc tính TotalPeople:.....	54
7. Dán nhãn số cho các thuộc tính định tính:	55
8. Lấy tập mẫu cho dữ liệu (Sampling data):	55
CHƯƠNG 3: CÁC THUẬT TOÁN KHAI THÁC DỮ LIỆU.....	57
1. Chuẩn bị dữ liệu để chạy các thuật toán:	57
2. Thuật toán Cây quyết định (Decision Tree):	59
2.1. Khái niệm:.....	59
2.2. Thực hiện:	60
2.2.1. Giải thuật Decision Tree ID3:	60
2.2.2. Giải thuật Decision Tree Cart:	62
3. Thuật toán Naïve Bayes:.....	65
3.1. Khái niệm:.....	65
3.2. Thực hiện:	66
4. Thuật toán K - Nearest Neighbors:	68
4.1. Khái niệm:.....	68
4.2. Thực hiện:	68
5. Thuật toán Neural Network:	70
5.1. Khái niệm:.....	70
5.2. Thực hiện:	70
6. Thuật toán Random Forest:	72
6.1. Khái niệm:.....	72
6.2. Thực hiện:	72
CHƯƠNG 4: ĐÁNH GIÁ CÁC THUẬT TOÁN	75

1. Đánh giá ma trận nhầm lẫn:	75
2. Đánh giá tỷ lệ TPR trên tỷ lệ FPR trong biểu đồ đường cong ROC:	76
3. Đánh giá về thời gian chạy từng thuật toán:	77
4. Đánh giá độ chính xác của thuật toán trên tập huấn luyện:	77
5. Đánh giá độ chính xác của thuật toán trên tập kiểm thử:	78
CHƯƠNG 5: KẾT LUẬN	80
 1. Ưu - Nhược điểm của từng thuật toán:	80
1.1. Decision Tree	80
1.2. Naïve Bayes	80
1.3. K – Nearest Neighbor	81
1.4. Neural Network	81
1.5. Random Forest	81
 2. Hướng phát triển:	82
BẢNG PHÂN CHIA CÔNG VIỆC	83
BẢNG ĐÁNH GIÁ CÁC THÀNH VIÊN TRONG NHÓM	84
TÀI LIỆU THAM KHẢO	84

DANH MỤC HÌNH ẢNH

Hình 1. 1 Bộ dữ liệu Hotel Booking Demands.....	12
Hình 1. 2 Thêm cột hotel vô 2 file csv.....	13
Hình 1. 3 Gộp 2 file csv thành 1 file csv mới	13
Hình 1. 4 Dữ liệu file hotel.csv	14
Hình 1. 5 Dữ liệu file hotel.csv	15
Hình 2. 1 Số lượng nhận và hủy phòng.....	22
Hình 2. 2 Biểu đồ thể hiện số lượng nhận và hủy đặt phòng	22
Hình 2. 3 Biểu đồ khách hàng từng khách sạn.....	23
Hình 2. 4 Tạo biến lưu trữ theo từng loại hình khách sạn và nhận phòng thành công	23
Hình 2. 5 Thống kê giá trị doanh thu trung bình theo từng tháng của khách sạn h1	24
Hình 2. 6 Thống kê giá trị doanh thu trung bình theo từng tháng của khách sạn h2	25
Hình 2. 7 Thống kê doanh thu trung bình của từng tháng của hai loại hình khách sạn	26
Hình 2. 8 Biểu đồ doanh thu theo từng tháng của từng khách sạn	26
Hình 2. 9 Biểu đồ thống kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc.....	27
Hình 2. 10 Code để vẽ biểu đồ thống kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng	28
Hình 2. 11 Biểu đồ thống kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng	28
Hình 2. 12 Biểu đồ tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công	29
Hình 2. 13 Biểu đồ thống kê tỷ lệ phòng bị hủy	30
Hình 2. 14 Biểu đồ thống kê kênh phân phối nào có tỷ lệ hủy cao nhất.....	31
Hình 2. 16 Biểu đồ hệ số tương quan giữa giá trị hủy đặt phòng và các thuộc tính còn lại	32
Hình 2. 17 Thông tin tập dữ liệu ban đầu.....	33
Hình 2. 18 Thông tin tập dữ liệu sau khi thay đổi kiểu dữ liệu 2 cột	34
Hình 2. 19 Thống kê thông tin thuộc tính định lượng	35

Hình 2. 20 Thống kê thông tin thuộc tính định tính	36
Hình 2. 21 Biểu đồ Boxplot cho cột ADR.....	37
Hình 2. 22 Loại bỏ cá ngoại lệ của cột ADR	37
Hình 2. 23 Liệt kê các giá trị null	38
Hình 2. 24 Tỉ lệ dữ liệu null.....	38
Hình 2. 25 Liệt kê số giá trị và tỉ lệ null của các cột	39
Hình 2. 26 Xóa các dòng dữ liệu Children bị null.....	39
Hình 2. 27 Xem thông tin của cột Country.....	40
Hình 2. 28 Thay thế giá trị rỗng của Country bằng PRT	40
Hình 2. 29 Biến đổi các giá trị null của Agent và Company thành giá trị mặc định .	40
Hình 2. 30 Liệt kê lại các giá trị null	41
Hình 2. 31 Kiểm tra adults, children và babies có đồng thời bằng không	42
Hình 2. 32 Xuất ra các dòng có adults, children và babies đồng thời bằng không....	42
Hình 2. 33 Xóa các dòng có 3 thuộc tính đó bằng 0	42
Hình 2. 34 Liệt kê kiểu dữ liệu của các cột	43
Hình 2. 35 Thay đổi dữ liệu từ float sang int.....	44
Hình 2. 36 Thay đổi dữ liệu từ object sang date	44
Hình 2. 37 Đếm số dòng trùng nhau.....	44
Hình 2. 38 Xóa các dòng trùng lặp, chỉ giữ lại dòng đầu	44
Hình 2. 39 Dữ liệu sau khi xóa các dòng trùng nhau.....	45
Hình 2. 40 Tạo thuộc tính ArrivalDate	45
Hình 2. 41 Tạo thuộc tính StatusMinusArrivalDate	46
Hình 2. 42	46
Hình 2. 43 Tạo thuộc tính RoomMatch	46
Hình 2. 44 Tạo thuộc tính TotalPeople.....	46
Hình 2. 45 Tạo thuộc tính HaveChild	47
Hình 2. 46 Tính hệ số tương quan	47
Hình 2. 47 Hiển thị các hệ số tương quan theo thứ tự giảm dần	48
Hình 2. 48 Xóa các thuộc tính không cần thiết	49
Hình 2. 49 Thống kê thuộc tính định lượng.....	50
Hình 2. 50 Chia giỏ thuộc tính LeadTime	51
Hình 2. 51 Chia dữ liệu theo giỏ thuộc tính LeadTime	51
Hình 2. 52 Chia giỏ thuộc tính Adults	51

Hình 2. 53 Chia dữ liệu theo giờ thuộc tính Adults.....	52
Hình 2. 54 Chia giờ thuộc tính ADR.....	52
Hình 2. 55 Chia dữ liệu theo giờ thuộc tính ADR	52
Hình 2. 56 Chia giờ thuộc tính StatusMinusArrivalDate	53
Hình 2. 57 Chia dữ liệu theo giờ thuộc tính StatusMinusArrivalDate	53
Hình 2. 58 Chia giờ thuộc tính TotalNights	53
Hình 2. 59 Chia dữ liệu theo giờ thuộc tính TotalNights.....	54
Hình 2. 60 Chia giờ thuộc tính TotalPeople	54
Hình 2. 61 Chia dữ liệu theo giờ thuộc tính TotalPeople.....	54
Hình 2. 62 Dán nhãn số cho các thuộc tính định tính.....	55
Hình 2. 63 Kiểm tra các giá trị trong IsCanceled	56
Hình 2. 64 Thực hiện Sampling dữ liệu	56
Hình 2. 65 Kết quả sau sampling dữ liệu	56
Hình 3. 1 Loại bỏ các thuộc tính có tương quan cao ra khỏi thuộc tính	57
Hình 3. 2 Chuẩn bị các mảng.....	58
Hình 3. 3 Chuẩn bị các mảng dữ liệu lưu lại thời gian chạy, độ chính xác thuật toán	59
Hình 3. 4 Hàm lưu ảnh cây quyết định ID3.....	60
Hình 3. 5 Hàm để vẽ biểu đồ đường cong ROC cho Decision Tree ID3	60
Hình 3. 6 Cài đặt thuật toán Decision Tree ID3	61
Hình 3. 7 Kết quả chạy ID3.....	62
Hình 3. 8 Mô hình cây quyết định ID3.....	62
Hình 3. 9 Hàm lưu ảnh cây quyết định CART	63
Hình 3. 10 Hàm để vẽ biểu đồ đường cong ROC cho Decision Tree Cart	63
Hình 3. 11 Cài đặt thuật toán Decision Tree CART.....	64
Hình 3. 12 Kết quả chạy CART	65
Hình 3. 13 Mô hình cây quyết định CART	65
Hình 3. 14 Hàm để vẽ biểu đồ đường cong ROC cho Naïve Bayes	66
Hình 3. 15 Cài đặt thuật toán Naïve Bayes.....	67
Hình 3. 16 Kết quả chạy Naïve Bayes	68
Hình 3. 17 Hàm để vẽ biểu đồ đường cong ROC cho K - Nearest Neighbors.....	68
Hình 3. 18 Cài đặt thuật toán K - Nearest Neighbors	69

Hình 3. 19 Kết quả chạy K - Nearest Neighbors.....	70
Hình 3. 20 Hàm để vẽ biểu đồ đường cong ROC cho Neural Network	70
Hình 3. 21 Cài đặt thuật toán Neural Network.....	71
Hình 3. 22 Kết quả chạy Neural Network	72
Hình 3. 23 Hàm lưu ảnh cây ngẫu nhiên Random Forest.....	72
Hình 3. 24 Hàm để vẽ biểu đồ đường cong ROC cho Random Forest	73
Hình 3. 25 Cài đặt thuật toán Random Forest.....	73
Hình 3. 26 Kết quả chạy Random Forest.....	74
Hình 3. 27 Mô hình cây ngẫu nhiên	74
Hình 4. 1 Ma trận nhầm lẫn của 6 thuật toán.....	75
Hình 4. 2 Biểu đồ đường cong ROC	76
Hình 4. 3 Thống kê thời gian chạy của từng thuật toán.....	77
Hình 4. 4 Biểu đồ thời gian chạy của từng thuật toán.....	77
Hình 4. 5 Thống kê độ chính xác trên tập train của từng thuật toán	78
Hình 4. 6 Biểu đồ độ chính xác trên tập train của từng thuật toán	78
Hình 4. 7 Thống kê độ chính xác trên tập test của từng thuật toán.....	79
Hình 4. 8 Biểu đồ độ chính xác trên tập test của từng thuật toán.....	79

DANH MỤC BẢNG BIỂU

Bảng 1. 1 Mô tả thuộc tính dữ liệu 20

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1. Lý do chọn đề tài

Ngành du lịch là một ngành quan trọng để phát triển kinh tế của một quốc gia, và việc đặt chỗ ở là một hoạt động không thể thiếu khi đi du lịch. Với một quốc gia có nhiều điểm đến thú vị như Bồ Đào Nha, dữ liệu về việc đặt phòng của các khách sạn sẽ là một tập dữ liệu lớn và đa dạng để phục vụ cho việc phân tích. Tìm hiểu về dữ liệu của các khách sạn sẽ đem lại những thông tin hữu ích để định hướng, phát triển khách sạn trong tương lai liên quan đến việc hủy phòng, đối tượng khách hàng, mức độ hài lòng của khách hàng, thời điểm thu hút khách, ...

Chính vì lẽ đó, nhóm chúng em đã chọn bộ dữ liệu về nhu cầu đặt phòng của 2 khách sạn điển hình cho hai khu vực du lịch nổi tiếng. Một là khách sạn dạng resort ở Algarve, khách sạn thứ hai nằm ở thành phố Lisbon - thủ đô Bồ Đào Nha với đa dạng các thuộc tính về đặt phòng như thông số phòng, thời gian lưu trú, thời gian từ khi đặt phòng đến khi lưu trú, liệu đặt phòng có bị hủy hay không và cách thức đặt phòng được thực hiện, ...

2. Giới thiệu nguồn dữ liệu

Bộ dữ liệu nhóm sử dụng có tên là Hotel booking demand datasets được đăng tải trên trang web ScienceDirect bởi ba tác giả là Nuno Antonio, Ana de Almeida và Luis Nunes.

 View PDF Download full issue

Outline Abstract 1. Data 2. Experimental design, materials and methods Acknowledgements Transparency document. Supplementary material Appendix A. Supplementary material References Cited by (29) Figures (3)  Tables (8)	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;">  </div> <div style="flex: 1; text-align: right;">  Volume 22, February 2019, Pages 41-49 </div> </div> <p style="font-size: 1em; margin-top: 10px;">Hotel booking demand datasets</p> <p style="margin-top: 10px;">Nuno Antonio^{a,b}  Ana de Almeida^{a,c,d}, Luis Nunes^{a,b,d}</p> <p style="margin-top: 10px;">Show more </p> <p style="margin-top: 10px;">+ Add to Mendeley  Share  Cite </p> <p style="margin-top: 10px;">https://doi.org/10.1016/j.dib.2018.11.126  Get rights and content  </p> <p style="margin-top: 10px;">Under a Creative Commons license </p> <p style="margin-top: 10px;">Abstract</p> <p>This data article describes two datasets with hotel demand data. One of the hotels (H1) is a resort hotel and the other is a city hotel (H2). Both datasets share the same structure, with 31 variables describing the 40,060 observations of H1 and 79,330 observations of H2. Each observation</p> <p style="margin-top: 10px;">Recommended articles</p> <p>Data on iron and turbidity in a drip irrigation system in New Jersey, USA Data in Brief, Volume 22, 2019, pp. 946-953 Steven Vergeau, Amy Raudenbush </p> <p>Chemical analysis on laser processed Ultrahydrophobic Ti-6Al-4V surface... Data in Brief, Volume 22, 2019, pp. 954-959 R. Jogdheesh, ..., J.L. Ocaña </p> <p>Draft genome sequence data of <i>Lactobacillus paracasei</i> strain DTA83... Data in Brief, Volume 22, 2019, pp. 1064-1067 Wilson José Fernandes Lemos Junior, ..., Viviana Corich </p> <p>Show 3 more articles  FEEDBACK </p>	
---	---	--

Hình 1. 1 Bộ dữ liệu Hotel Booking Demands

Bộ dữ liệu này chứa thông tin về nhu cầu đặt phòng khách sạn tại Bồ Đào Nha. Khách sạn H1 là một khu nghỉ dưỡng tại Algarve và H2 là một khách sạn trong thành phố Lisbon. Cả hai tập dữ liệu của hai khách sạn đều có cấu trúc giống nhau, bao gồm 31 biến mô tả với 40.060 lượt đặt phòng của khách sạn H1 và 79.330 lượt đặt phòng của khách sạn H2. Mỗi lượt đặt phòng được biểu diễn trên một dòng dữ liệu.

Các lượt đặt phòng được ghi nhận từ 01/07/2015 đến 31/08/2017, bao gồm cả đặt phòng thành công và đặt phòng đã hủy.

Nguồn dữ liệu:

<https://www.sciencedirect.com/science/article/pii/S2352340918315191#s0005>

3. Lọc dữ liệu

Do bộ dữ liệu bao gồm 2 file csv và có quá nhiều thuộc tính mà nhóm em không sử dụng hết nên nhóm em sử dụng Python để thực hiện lọc và tiền xử lý dữ liệu.

Bước 1: Khai báo thư viện cần thiết, đọc file csv từ máy, thêm cột ‘hotel’ và xuất file csv.

```

# Khai báo thư viện
import pandas as pd

# Đọc dữ liệu từ file CSV
df1 = pd.read_csv('H1.csv')
df2 = pd.read_csv('H2.csv')

# Thêm cột 'hotel' với giá trị 'h1' cho df1, 'h2' cho df2
df1['hotel'] = 'h1'
df2['hotel'] = 'h2'

# Ghi dataframe đã chỉnh sửa vào file CSV mới
df1.to_csv('h1new.csv', index = False)
df2.to_csv('h2new.csv', index = False)

```

Hình 1. 2 Thêm cột hotel vô 2 file csv

Bước 2: Đọc 2 file vừa xuất từ bước 1, sau đó tiến hành gộp 2 bộ dữ liệu H1, H2 và xuất ra file csv mới.

```

# Đọc lại dữ liệu từ 2 file CSV vừa xuất trên
df1 = pd.read_csv('h1new.csv')
df2 = pd.read_csv('h2new.csv')

# Gộp 2 dataframe thành 1 dataframe mới
df_merged = pd.concat([df1, df2], ignore_index = True)

# Ghi dataframe mới vào file csv
df_merged.to_csv('hotel.csv', index = False)

# In thông tin của dataframe mới
print(df_merged)

```

Hình 1. 3 Gộp 2 file csv thành 1 file csv mới

Kết quả sau khi xử lý:

	IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	\		
0	0	342	2015	July			
1	0	737	2015	July			
2	0	7	2015	July			
3	0	13	2015	July			
4	0	14	2015	July			
...		
119385	0	23	2017	August			
119386	0	102	2017	August			
119387	0	34	2017	August			
119388	0	109	2017	August			
119389	0	205	2017	August			
	ArrivalDateWeekNumber	ArrivalDateDayOfMonth	StaysInWeekendNights	\			
0	27	1	0				
1	27	1	0				
2	27	1	0				
3	27	1	0				
4	27	1	0				
...		
119385	35	30	2				
119386	35	31	2				
119387	35	31	2				
119388	35	31	2				
119389	35	29	2				
	StaysInWeekNights	Adults	Children	...	Agent	Company	\
0	0	2	0.0	...	NULL	NULL	
1	0	2	0.0	...	NULL	NULL	
2	1	1	0.0	...	NULL	NULL	
3	1	1	0.0	...	304	NULL	
4	2	2	0.0	...	240	NULL	
...
119385	5	2	0.0	...	394	NULL	
119386	5	3	0.0	...	9	NULL	
119387	5	2	0.0	...	9	NULL	
119388	5	2	0.0	...	89	NULL	
119389	7	2	0.0	...	9	NULL	
	DaysInWaitingList	CustomerType	ADR	RequiredCarParkingSpaces	\		
0	0	Transient	0.00	0			
1	0	Transient	0.00	0			
2	0	Transient	75.00	0			
3	0	Transient	75.00	0			
4	0	Transient	98.00	0			
...

Hình 1.4 Dữ liệu file hotel.csv

```

DaysInWaitingList CustomerType      ADR  RequiredCarParkingSpaces \
0                  0    Transient   0.00           0
1                  0    Transient   0.00           0
2                  0    Transient  75.00           0
3                  0    Transient  75.00           0
4                  0    Transient  98.00           0
...
...          ...      ...
119385            0    Transient  96.14           0
119386            0    Transient 225.43           0
119387            0    Transient 157.71           0
119388            0    Transient 104.40           0
119389            0    Transient 151.20           0

TotalOfSpecialRequests ReservationStatus ReservationStatusDate hotel
0                      0        Check-Out       7/1/2015     h1
1                      0        Check-Out       7/1/2015     h1
2                      0        Check-Out       7/2/2015     h1
3                      0        Check-Out       7/2/2015     h1
4                      1        Check-Out       7/3/2015     h1
...
...          ...      ...
119385            0        Check-Out       9/6/2017     h2
119386            2        Check-Out       9/7/2017     h2
119387            4        Check-Out       9/7/2017     h2
119388            0        Check-Out       9/7/2017     h2
119389            2        Check-Out       9/7/2017     h2

[119390 rows x 32 columns]

```

Hình 1. 5 Dữ liệu file hotel.csv

4. Mô tả thuộc tính

Dữ liệu gồm 119390 dòng dữ liệu và 32 cột thuộc tính chứa quan sát các hoạt động đặt phòng từ các khách sạn H1 và khách sạn H2.

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	IsCanceled	boolean	Có hủy đặt phòng không. Nhận 2 giá trị: 0: không hủy 1: hủy

1	LeadTime	integer	Số ngày từ khi đặt phòng tới khi nhận phòng.
2	ArrivalDateYear	integer	Năm nhận phòng.
3	ArrivalDateMonth	string	Tháng nhận phòng trong năm.
4	ArrivalDateWeekNumber	integer	Tuần nhận phòng tính từ đầu năm.
5	ArrivalDateDayOfMonth	integer	Ngày nhận phòng trong tháng.
6	StayInWeekendNights	integer	Số đêm lưu trú cuối tuần.
7	StayIn WeekNights	integer	Số đêm lưu trú trong tuần.
8	Adults	integer	Số lượng người lớn.
9	Children	integer	Số lượng trẻ em.
10	Babies	integer	Số lượng em bé.
11	Meal	string	Loại bữa ăn được phục vụ. Có 3 giá trị: BB (Bed & Breakfast): Chỉ bao gồm bữa sáng. HB (Half Board): Bữa sáng và bữa tối được bao gồm trong giá phòng. Trong một số trường hợp, bạn có thể chọn nhận bữa trưa thay vì bữa sáng - khách sạn sẽ xác

			<p>nhận điều này khi khách hàng đến.</p> <p>FB (Full Board): Được bao gồm bữa sáng, bữa trưa và buổi tối.</p>
12	Country	string	Quốc tịch khách hàng.
	MarketSegment	string	<p>Thể hiện các phân khúc khách hàng của khách sạn:</p> <p>Offline TA/TO (Travel Agency/ Tour Operator): khách hàng từ các đại lý du lịch và công ty điều hành tour.</p> <p>Online TA: khách hàng từ các đại lý du lịch nhưng bằng hình thức online trực tuyến.</p> <p>Aviation: khách hàng đến từ các hãng hàng không.</p> <p>Complementary: khách hàng được hưởng ưu đãi miễn phí.</p> <p>Corporate: khách hàng đến từ các công ty, doanh nghiệp.</p> <p>Groups: khách hàng đến từ các hội nhóm, tổ chức.</p> <p>Direct: khách hàng đặt phòng trực tiếp tại khách sạn.</p> <p>Undefined: không xác định.</p>

	DistributionChannel	string	Kênh phân phối phòng khách sạn. Có ba loại: TA/TO (Travel Agency/Tour Operator): là kênh phân phối thông qua các đại lý du lịch và công ty điều hành tour. Direct: là kênh phân phối trực tiếp (khách hàng đặt phòng trực tiếp tại khách sạn) Corporate: là kênh phân phối thông qua các công ty, doanh nghiệp Undefined: không xác định.
13	IsRepeatedGuest	Boolean	Là khách quay lại. Có 2 giá trị: 0: khách tới lần đầu. 1: khách cũ.
14	PreviousCancellations	integer	Số lượng lần hủy đặt phòng trước đó.
15	PreviousBookingsNotCancelled	integer	Số lượng lần không hủy phòng trước đó.
16	ReservedRoomType	string	Loại phòng đã đặt.
17	AssignedRoomType	string	Loại phòng được nhận.
18	BookingChanges	integer	Số lần thay đổi thông tin đặt phòng.

19	DepositType	string	<p>Loại hình đặt cọc trước:</p> <p>No Deposit: không có tiền đặt cọc trước.</p> <p>Non-Refund: một khoản đặt cọc đã được thực hiện bằng giá trị của tổng chi phí lưu trú.</p> <p>Refundable: một khoản đặt cọc đã được thực hiện với giá trị dưới tổng chi phí lưu trú.</p>
20	Agent	integer	Mã của đại lý đặt phòng.
21	Company	interger	Mã của công ty đặt phòng.
22	DaysInWaitingList	integer	Số ngày đặt chỗ nằm trong danh sách chờ trước khi được xác nhận với khách hàng.
23	CustomerType	string	<p>Loại khách hàng:</p> <p>Group: Khi khách hàng đặt phòng theo một nhóm.</p> <p>Transient: Khách tạm trú là những khách lưu trú qua đêm không có ý định ở lại lâu dài.</p> <p>Transient party: Khách tạm trú theo nhóm lưu trú qua đêm không có ý định ở lại lâu dài.</p>

24	ADR	float	Doanh thu bình quân hằng ngày (Avarage Daily Rate) được tính bằng cách chia tổng của tất cả các giao dịch lưu trú cho tổng số đêm lưu trú.
25	RequiredCarParkingSpace s	integer	Số lượng chỗ đỗ xe ô tô được yêu cầu.
26	TotalOfSpecialRequests	integer	Tổng số lượng yêu cầu đặc biệt.
27	ReservationStatus	string	Trạng thái đặt phòng hiện tại: Check-Out: Khách hàng đã check-in nhưng đã rời đi. No-Show: Khách hàng không check-in và đã thông báo lý do cho khách sạn. Cancel: Khách hàng hủy lượt đặt phòng.
28	ReservationStatusDate	datetime	Ngày mà trạng thái cuối cùng của lượt đặt phòng được cập nhật.
29	hotel	string	Tên khách sạn: H1: Khách sạn ở resort. H2: Khách sạn trong thành phố.

Bảng 1. 1 Mô tả thuộc tính dữ liệu

5. Giới thiệu bài toán

Nhóm chúng em sẽ dự đoán mối quan hệ giữa khả năng hủy đặt phòng khách sạn (**IsCancelled**) với các thuộc tính khác để có thể khoanh vùng các đối tượng phục vụ thích hợp và đề ra các biện pháp phù hợp khi hủy đơn tăng cao.

CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU

1. Trực quan hóa dữ liệu:

1.1. Thống kê số lượng nhận phòng và hủy phòng:

- Số lượng trường hợp nhận và hủy phòng:

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

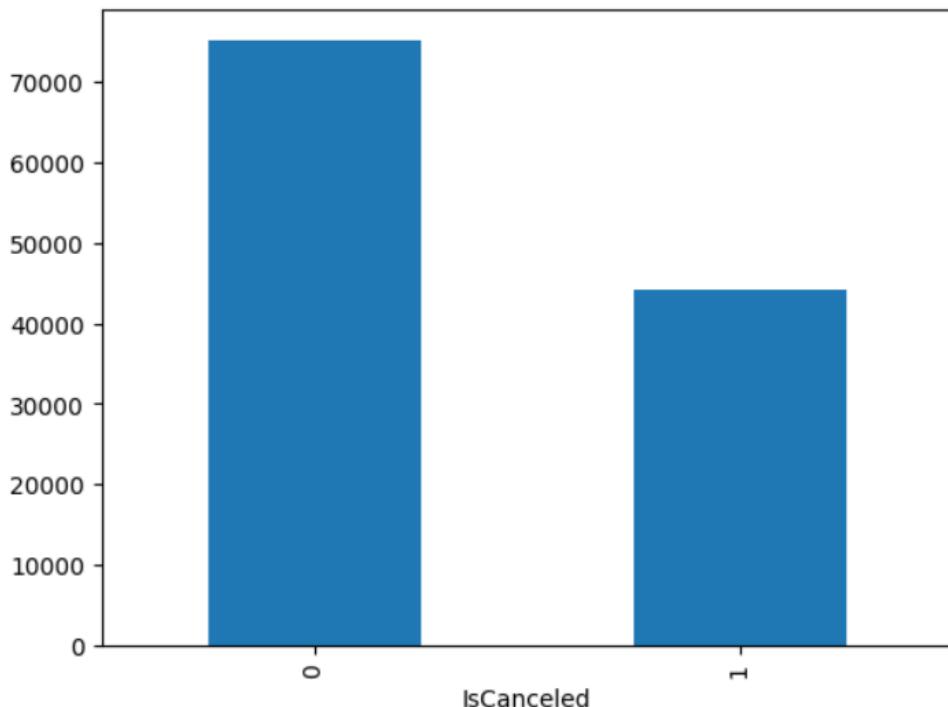
```
IsCanceled
0    75166
1    44224
Name: count, dtype: int64
```

Hình 2. 1 Số lượng nhận và hủy phòng

- Biểu đồ thể hiện:

```
df['IsCanceled'].value_counts().plot(kind = 'bar')
```

```
<Axes: xlabel='IsCanceled'>
```



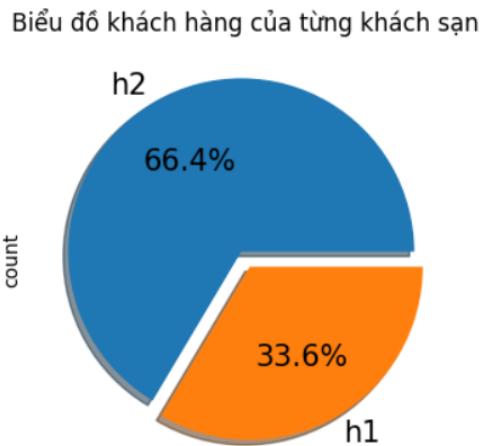
Hình 2. 2 Biểu đồ thể hiện số lượng nhận và hủy đặt phòng

- **Ta thấy:** Số lượng hủy nhận phòng chỉ bằng 2/3 số lượng nhận phòng.

1.2. Thống kê loại hình khách sạn nào thu nhiều khách hàng nhất

```
: df['hotel'].value_counts().plot.pie(explode=[0.05, 0.05], autopct='%1.1f%%', shadow=True, figsize=(6,4), fontsize=15)
```

```
: Text(0.5, 1.0, 'Biểu đồ khách hàng của từng khách sạn')
```



Hình 2. 3 Biểu đồ khách hàng từng khách sạn

Ta thấy: Khách sạn thành phố (h2) được đặt phòng nhiều nhất.

1.3. Thống kê doanh thu mỗi tháng trong tập dữ liệu

- Tạo biến để lưu trữ các giá trị trong tập dữ liệu theo từng loại hình khách sạn và nhận phòng thành công.

```
# Tạo biến để lưu trữ các giá trị trong tập dữ liệu theo từng loại hình khách sạn và nhận phòng thành công.
data_h1 = df[(df['hotel'] == 'h1') & (df['IsCanceled'] == 0)]
data_h2 = df[(df['hotel'] == 'h2') & (df['IsCanceled'] == 0)]
```

Hình 2. 4 Tạo biến lưu trữ theo từng loại hình khách sạn và nhận phòng thành công

- Tính giá trị trung bình từng tháng của từng loại khách sạn.

```
# Giá trị trung bình theo từng tháng của khách sạn resort
resort_hotel = data_h1.groupby(['ArrivalDateMonth'])['ADR'].mean().reset_index()
resort_hotel
```

	ArrivalDateMonth	ADR
0	April	75.867816
1	August	181.205892
2	December	68.322236
3	February	54.147478
4	January	48.708919
5	July	150.122528
6	June	107.921869
7	March	57.012487
8	May	76.657558
9	November	48.681640
10	October	61.727505
11	September	96.416860

Hình 2. 5 *Thống kê giá trị doanh thu trung bình theo từng tháng của khách sạn h1*

```
# Giá trị trung bình theo từng tháng của khách sạn city
city_hotel=data_h2.groupby(['ArrivalDateMonth'])['ADR'].mean().reset_index()
city_hotel
```

	ArrivalDateMonth	ADR
0	April	111.856824
1	August	118.412083
2	December	87.856764
3	February	86.183025
4	January	82.160634
5	July	115.563810
6	June	117.702075
7	March	90.170722
8	May	120.445842
9	November	86.500456
10	October	101.745956
11	September	112.598452

Hình 2. 6 Thống kê giá trị doanh thu trung bình theo từng tháng của khách sạn h2

- Ghép hai **Dataframe** vừa tạo lại thành một tập **Dataframe** mới chứa doanh thu trung bình của từng tháng của hai loại hình khách sạn.

```

final_hotel = city_hotel
final_hotel["Price_Resort"] = resort_hotel["ADR"]
final_hotel.rename(index=str,columns={'ADR':'Price_City'},inplace=True)
final_hotel

```

	ArrivalDateMonth	Price_City	Price_Resort
0	April	111.856824	75.867816
1	August	118.412083	181.205892
2	December	87.856764	68.322236
3	February	86.183025	54.147478
4	January	82.160634	48.708919
5	July	115.563810	150.122528
6	June	117.702075	107.921869
7	March	90.170722	57.012487
8	May	120.445842	76.657558
9	November	86.500456	48.681640
10	October	101.745956	61.727505
11	September	112.598452	96.416860

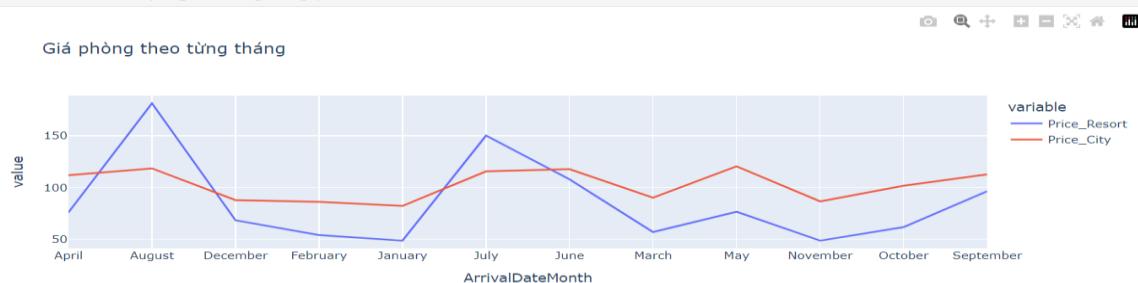
Hình 2. 7 Thông kê doanh thu trung bình của từng tháng của hai loại hình khách sạn

- Kết quả:

```

plt.figure(figsize = (25, 12))
px.line(final_hotel, x = 'ArrivalDateMonth', y = ['Price_Resort','Price_City'],
        title = 'Giá phòng theo từng tháng')

```

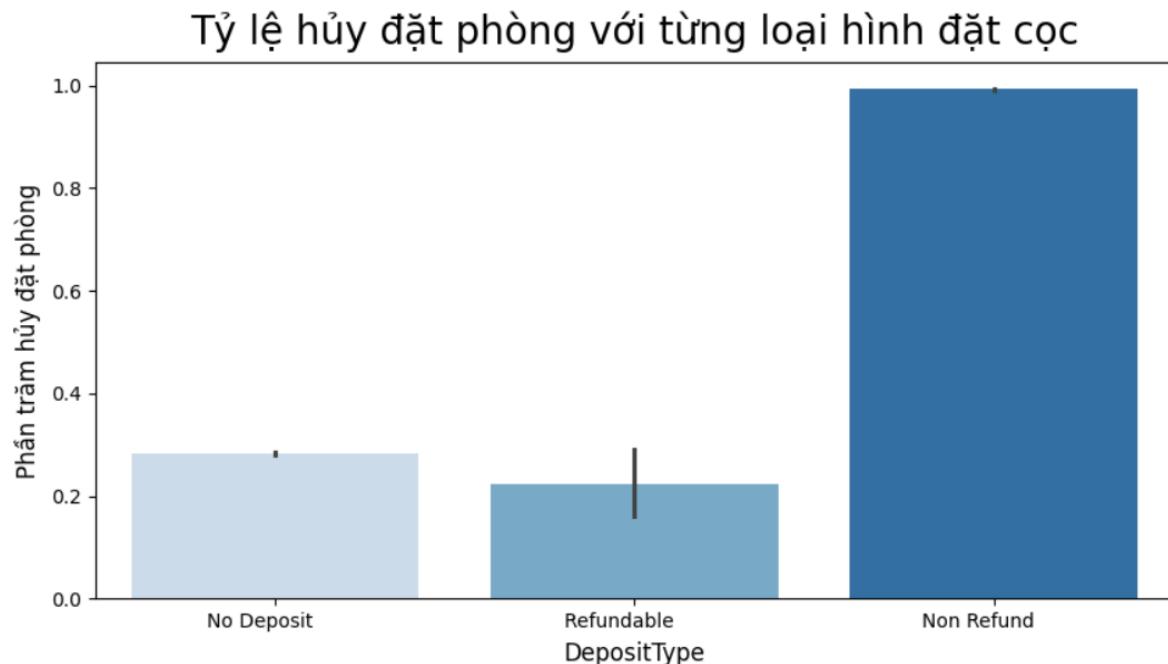


Hình 2. 8 Biểu đồ doanh thu theo từng tháng của từng khách sạn

Ta thấy: Doanh thu của khách sạn thành phố tương đối ổn định theo từng tháng trong từng năm. Doanh thu của khách sạn nghỉ dưỡng cao ở tháng 6, 7, 8 và thấp ở các tháng còn lại.

1.4. Thống kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc

```
plt.figure(figsize=(10,5))
plt.title("Tỷ lệ hủy đặt phòng với từng loại hình đặt cọc", fontsize = 20, pad = 10)
sns.barplot(x=df['DepositType'], y=df['IsCanceled'], palette='Blues')
plt.xlabel("DepositType", fontsize = 12, labelpad = 5)
plt.ylabel("Phần trăm hủy đặt phòng", fontsize = 12, labelpad = 5);
```



Hình 2. 9 Biểu đồ thống kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc

- **Ta thấy:** Lượng khách hàng hủy đặt cọc không nhận lại cọc có tỷ lệ không nhận phòng cao nhất.

1.5. Thống kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng.

```

# Khai báo dữ liệu
labels = ["Aviation", "Complementary", "Corporate", "Groups", "Direct", "Offline TA/TO", "Online TA"]

canceled = df[df['IsCanceled']==1][['MarketSegment']]
canceled = canceled.MarketSegment.value_counts().sort_values()

not_canceled = df[df['IsCanceled']==0][['MarketSegment']]
not_canceled = not_canceled.MarketSegment.value_counts().sort_values()

# Vẽ biểu đồ
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=labels, values=canceled, name="Canceled"),
              1, 1)
fig.add_trace(go.Pie(labels=labels, values=not_canceled, name="Check-in"),
              1, 2)

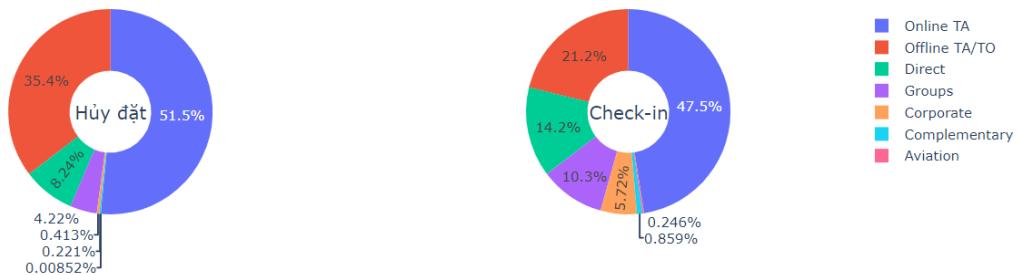
fig.update_traces(hole=.4, hoverinfo="label+percent+name")

# Cập nhật layout và thêm ghi chú
fig.update_layout(
    title_text="Phân phối đặt phòng và hủy đặt phòng theo phân khúc thị trường",
    annotations=[dict(text='Hủy đặt', x=0.185, y=0.5, font_size=16, showarrow=False),
                 dict(text='Check-in', x=0.82, y=0.5, font_size=16, showarrow=False)])
fig.show()

```

Hình 2. 10 Code để vẽ biểu đồ thống kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng

Phân phối đặt phòng và hủy đặt phòng theo phân khúc thị trường

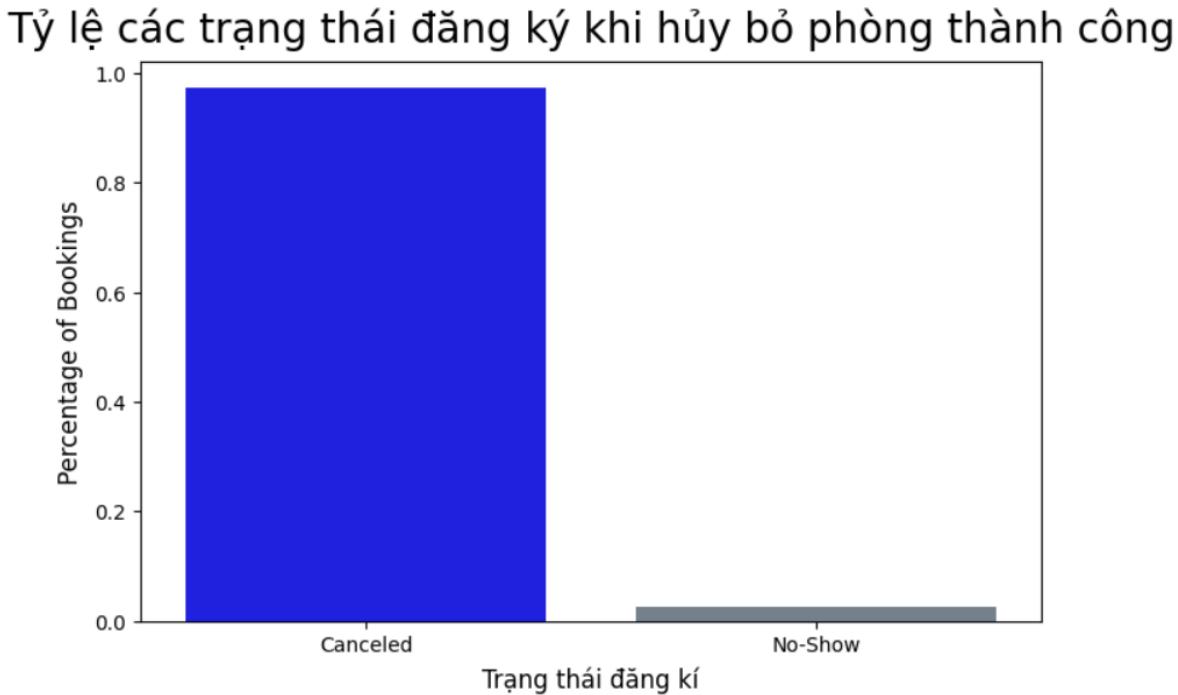


Hình 2. 11 Biểu đồ thống kê tỷ lệ phần trăm của các giá trị trong thuộc tính phân khúc thị trường với trạng thái hủy đặt phòng và nhận phòng

Ta thấy: Tỷ lệ phân khúc thị trường trong việc nhận phòng và hủy phòng ở từng thị trường là tỷ lệ thuận với nhau. Và tỉ lệ đặt phòng và hủy phòng online là lớn nhất.

1.6. Tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công

```
plt.figure(figsize=(8,5))
plt.title("Tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công", fontsize = 20, pad = 10)
sns.barplot(x=df[df['IsCanceled']==1]['ReservationStatus'].unique(),
             y=df[df['IsCanceled']==1]['ReservationStatus'].value_counts(normalize=True),
             palette=['blue', 'slategrey'])
plt.xlabel("Trạng thái đăng kí", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings", fontsize = 12, labelpad = 5);
```



Hình 2. 12 Biểu đồ tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công

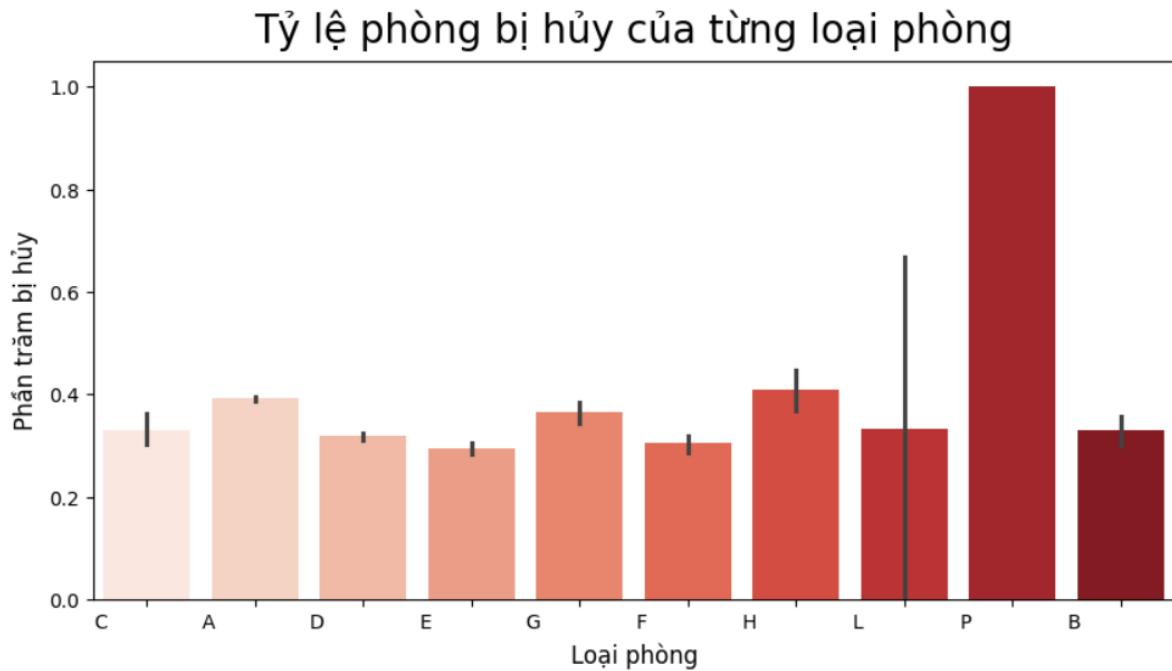
Ta thấy: Phần lớn các phòng bị hủy trước khi đến. Các dữ liệu chứa giá trị “**No – Show**” không đáng kể, có khả năng thuộc tính này sẽ không giúp ích được trong quá trình khai thác.

1.7. Thông kê tỷ lệ phòng bị hủy

```

plt.figure(figsize=(10,5))
plt.title("Tỷ lệ phòng bị hủy của từng loại phòng", fontsize = 20, pad = 10)
sns.barplot(x=df['ReservedRoomType'], y=df['IsCanceled'], palette='Reds')
plt.xlabel("Loại phòng", fontsize = 12, labelpad = 5)
plt.ylabel("Phần trăm bị hủy", fontsize = 12, labelpad = 5);

```



Hình 2. 13 Biểu đồ thống kê tỷ lệ phòng bị hủy

Ta thấy: Tỷ lệ khách hàng hủy đặt cao nhất đều với phòng P và thấp nhất đối với phòng E.

1.8. Kênh phân phối nào có tỷ lệ hủy cao nhất

```

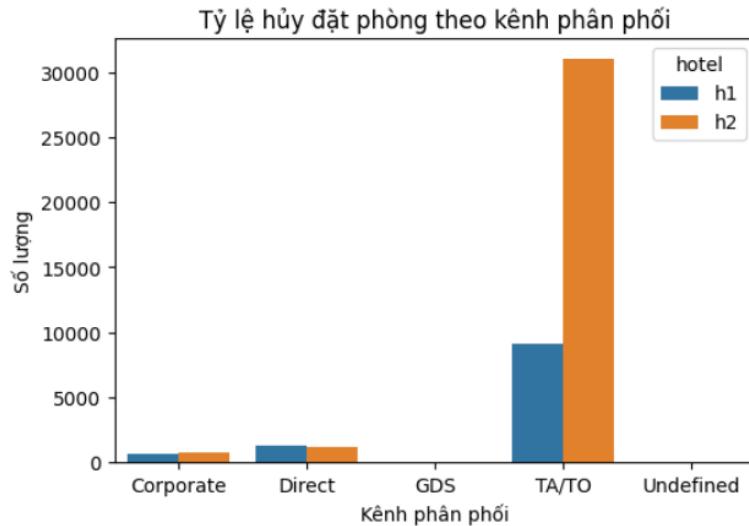
canceled_df = df[df['IsCanceled']==1]

# Nhóm theo kênh phân phối
canceled_df = canceled_df.groupby(['DistributionChannel', 'hotel']).size().reset_index().rename(columns={0:'Số lượng'})
canceled_df

# Vẽ biểu đồ
plt.figure(figsize=(6, 4))
sns.barplot(x='DistributionChannel', y='Số lượng', hue="hotel", data=canceled_df)
plt.xlabel('Kênh phân phối')
plt.ylabel('Số lượng')
plt.title('Tỷ lệ hủy đặt phòng theo kênh phân phối')

Text(0.5, 1.0, 'Tỷ lệ hủy đặt phòng theo kênh phân phối')

```



Hình 2. 14 Biểu đồ thống kê kênh phân phối nào có tỷ lệ hủy cao nhất

Ta thấy: Kênh phân phối thông qua các đại lý du lịch và công ty điều hành tour có tỷ lệ bị hủy đặt phòng cao nhất ở cả 2 khách sạn.

1.9. Thể hiện hệ số tương quan giữa giá trị hủy đặt phòng và các thuộc tính còn lại.

- Sử dụng hàm **heatmap** trong **Seaborn** để vẽ biểu đồ thể hiện sự tương quan giữa thuộc tính có hủy đặt phòng hay không và các thuộc tính khác trong tập dữ liệu. Biết rằng tương quan càng mạnh và đồng biến thì màu càng đậm và giảm dần về các tương quan mạnh nhưng nghịch biến với thuộc tính.

```

numeric_df = df.select_dtypes(include=['number'])
corr_df = numeric_df.corr() # Tính toán ma trận tương quan

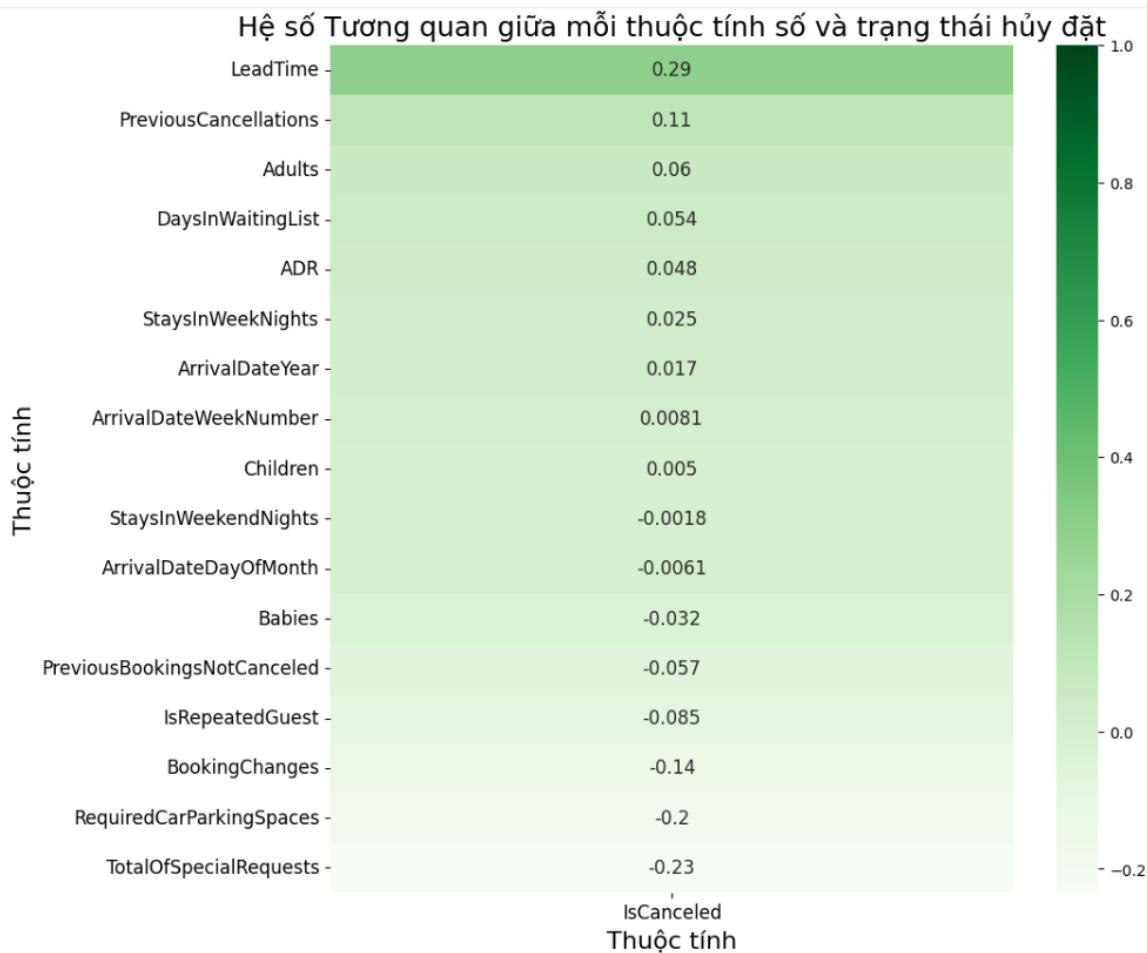
fig = plt.figure(figsize=(10, 10))
ax = sns.heatmap(corr_df[['IsCanceled']].sort_values('IsCanceled', ascending=False), annot=True, annot_kws={"size": 12}, cmap='Greens')

ax.set_title('Hệ số Tương quan giữa mỗi thuộc tính số và trạng thái hủy đặt', fontsize=18)
ax.set_xlabel('Thuộc tính', fontsize=16)
ax.set_ylabel('Thuộc tính', fontsize=16)
ax.tick_params(axis="both", labelsize=12)
y_min, y_max = ax.get_ylim()
ax.set_ylim(top=y_max + 1)

```

Hình 2. 15 Code để vẽ biểu đồ hệ số tương quan

- **Kết quả:**



Hình 2. 15 Biểu đồ hệ số tương quan giữa giá trị hủy đặt phòng và các thuộc tính còn lại

- **Ta thấy:** LeadTime có mối quan hệ tương quan cao nhất với việc có hủy đặt phòng. Có nghĩa là số ngày giữa thời điểm đặt phòng và ngày đến dự kiến tăng lên, khách

hàng càng có nhiều thời gian hơn để hủy đặt phòng. **TotalOfSpecialRequests** có mối tương quan yếu với mục đích hủy đặt phòng. Có nghĩa rằng khách hàng càng có nhiều yêu cầu, đòi hỏi, gắn bó với khách sạn thì khả năng hủy đặt phòng càng thấp.

2. Mô tả cơ bản về tập dữ liệu:

- Thông tin khái quát của dữ liệu:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   IsCanceled      119390 non-null   int64  
 1   LeadTime         119390 non-null   int64  
 2   ArrivalDateYear 119390 non-null   int64  
 3   ArrivalDateMonth 119390 non-null   object  
 4   ArrivalDateWeekNumber 119390 non-null   int64  
 5   ArrivalDateDayOfMonth 119390 non-null   int64  
 6   StaysInWeekendNights 119390 non-null   int64  
 7   StaysInWeekNights 119390 non-null   int64  
 8   Adults            119390 non-null   int64  
 9   Children          119386 non-null   float64 
 10  Babies             119390 non-null   int64  
 11  Meal               119390 non-null   object  
 12  Country            118902 non-null   object  
 13  MarketSegment      119390 non-null   object  
 14  DistributionChannel 119390 non-null   object  
 15  IsRepeatedGuest    119390 non-null   int64  
 16  PreviousCancellations 119390 non-null   int64  
 17  PreviousBookingsNotCanceled 119390 non-null   int64  
 18  ReservedRoomType   119390 non-null   object  
 19  AssignedRoomType   119390 non-null   object  
 20  BookingChanges     119390 non-null   int64  
 21  DepositType        119390 non-null   object  
 22  Agent               119390 non-null   object  
 23  Company             119390 non-null   object  
 24  DaysInWaitingList  119390 non-null   int64  
 25  CustomerType       119390 non-null   object  
 26  ADR                 119390 non-null   float64 
 27  RequiredCarParkingSpaces 119390 non-null   int64  
 28  TotalOfSpecialRequests 119390 non-null   int64  
 29  ReservationStatus   119390 non-null   object  
 30  ReservationStatusDate 119390 non-null   object  
 31  hotel               119390 non-null   object  
dtypes: float64(2), int64(16), object(14)
memory usage: 29.1+ MB
```

Hình 2. 16 Thông tin tập dữ liệu ban đầu

Theo trên, nhóm thấy có 2 thuộc tính **Agent** và **Company** nhận các giá trị số nhưng lại được định dạng thành kiểu dữ liệu **object**. Nhóm sẽ tiến hành đặt lại kiểu dữ liệu cho 2 thuộc tính đó như sau:

```

df['Agent'] = pd.to_numeric(df['Agent'], errors='coerce')
df['Company'] = pd.to_numeric(df['Company'], errors='coerce')
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   IsCanceled      119390 non-null   int64  
 1   LeadTime         119390 non-null   int64  
 2   ArrivalDateYear 119390 non-null   int64  
 3   ArrivalDateMonth 119390 non-null   object  
 4   ArrivalDateWeekNumber 119390 non-null   int64  
 5   ArrivalDateDayOfMonth 119390 non-null   int64  
 6   StaysInWeekendNights 119390 non-null   int64  
 7   StaysInWeekNights 119390 non-null   int64  
 8   Adults            119390 non-null   int64  
 9   Children          119386 non-null   float64 
 10  Babies            119390 non-null   int64  
 11  Meal               119390 non-null   object  
 12  Country            118902 non-null   object  
 13  MarketSegment      119390 non-null   object  
 14  DistributionChannel 119390 non-null   object  
 15  IsRepeatedGuest    119390 non-null   int64  
 16  PreviousCancellations 119390 non-null   int64  
 17  PreviousBookingsNotCanceled 119390 non-null   int64  
 18  ReservedRoomType    119390 non-null   object  
 19  AssignedRoomType    119390 non-null   object  
 20  BookingChanges      119390 non-null   int64  
 21  DepositType         119390 non-null   object  
 22  Agent              103050 non-null   float64 
 23  Company             6797 non-null    float64 
 24  DaysInWaitingList   119390 non-null   int64  
 25  CustomerType        119390 non-null   object  
 26  ADR                119390 non-null   float64 
 27  RequiredCarParkingSpaces 119390 non-null   int64  
 28  TotalOfSpecialRequests 119390 non-null   int64  
 29  ReservationStatus    119390 non-null   object  
 30  ReservationStatusDate 119390 non-null   object  
 31  hotel               119390 non-null   object  
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB

```

Hình 2. 17 Thông tin tập dữ liệu sau khi thay đổi kiểu dữ liệu 2 cột

- **Thuộc tính định lượng:** liệt kê các thông tin như: đếm số giá trị, giá trị lớn nhất, giá trị nhỏ nhất, trung bình, độ lệch chuẩn và tứ phân vị.

	count	mean	std	min	25%	50%	75%	max
IsCanceled	119390.0	0.370416	0.482918	0.00	0.00	0.000	1.0	1.0
LeadTime	119390.0	104.011416	106.863097	0.00	18.00	69.000	160.0	737.0
ArrivalDateYear	119390.0	2016.156554	0.707476	2015.00	2016.00	2016.000	2017.0	2017.0
ArrivalDateWeekNumber	119390.0	27.165173	13.605138	1.00	16.00	28.000	38.0	53.0
ArrivalDateDayOfMonth	119390.0	15.798241	8.780829	1.00	8.00	16.000	23.0	31.0
StaysInWeekendNights	119390.0	0.927599	0.998613	0.00	0.00	1.000	2.0	19.0
StaysInWeekNights	119390.0	2.500302	1.908286	0.00	1.00	2.000	3.0	50.0
Adults	119390.0	1.856403	0.579261	0.00	2.00	2.000	2.0	55.0
Children	119386.0	0.103890	0.398561	0.00	0.00	0.000	0.0	10.0
Babies	119390.0	0.007949	0.097436	0.00	0.00	0.000	0.0	10.0
IsRepeatedGuest	119390.0	0.031912	0.175767	0.00	0.00	0.000	0.0	1.0
PreviousCancellations	119390.0	0.087118	0.844336	0.00	0.00	0.000	0.0	26.0
PreviousBookingsNotCanceled	119390.0	0.137097	1.497437	0.00	0.00	0.000	0.0	72.0
BookingChanges	119390.0	0.221124	0.652306	0.00	0.00	0.000	0.0	21.0
Agent	103050.0	86.693382	110.774548	1.00	9.00	14.000	229.0	535.0
Company	6797.0	189.266735	131.655015	6.00	62.00	179.000	270.0	543.0
DaysInWaitingList	119390.0	2.321149	17.594721	0.00	0.00	0.000	0.0	391.0
ADR	119390.0	101.831122	50.535790	-6.38	69.29	94.575	126.0	5400.0
RequiredCarParkingSpaces	119390.0	0.062518	0.245291	0.00	0.00	0.000	0.0	8.0
TotalOfSpecialRequests	119390.0	0.571363	0.792798	0.00	0.00	0.000	1.0	5.0

Hình 2. 18 Thông kê thông tin thuộc tính định lượng

- **Thuộc tính định tính:** liệt kê số giá trị, số lượng các giá trị khác nhau trong từng thuộc tính, giá trị có tần số xuất hiện cao nhất của thuộc tính đó và tần số của giá trị đó.

		count	unique	top	freq
ArrivalDateMonth	119390	12	August	13877	
Meal	119390	5	BB	92310	
Country	118902	177	PRT	48590	
MarketSegment	119390	8	Online TA	56477	
DistributionChannel	119390	5	TA/TO	97870	
ReservedRoomType	119390	10	A	85994	
AssignedRoomType	119390	12	A	74053	
DepositType	119390	3	No Deposit	104641	
CustomerType	119390	4	Transient	89613	
ReservationStatus	119390	3	Check-Out	75166	
ReservationStatusDate	119390	926	10/21/2015	1461	
hotel	119390	2	h2	79330	

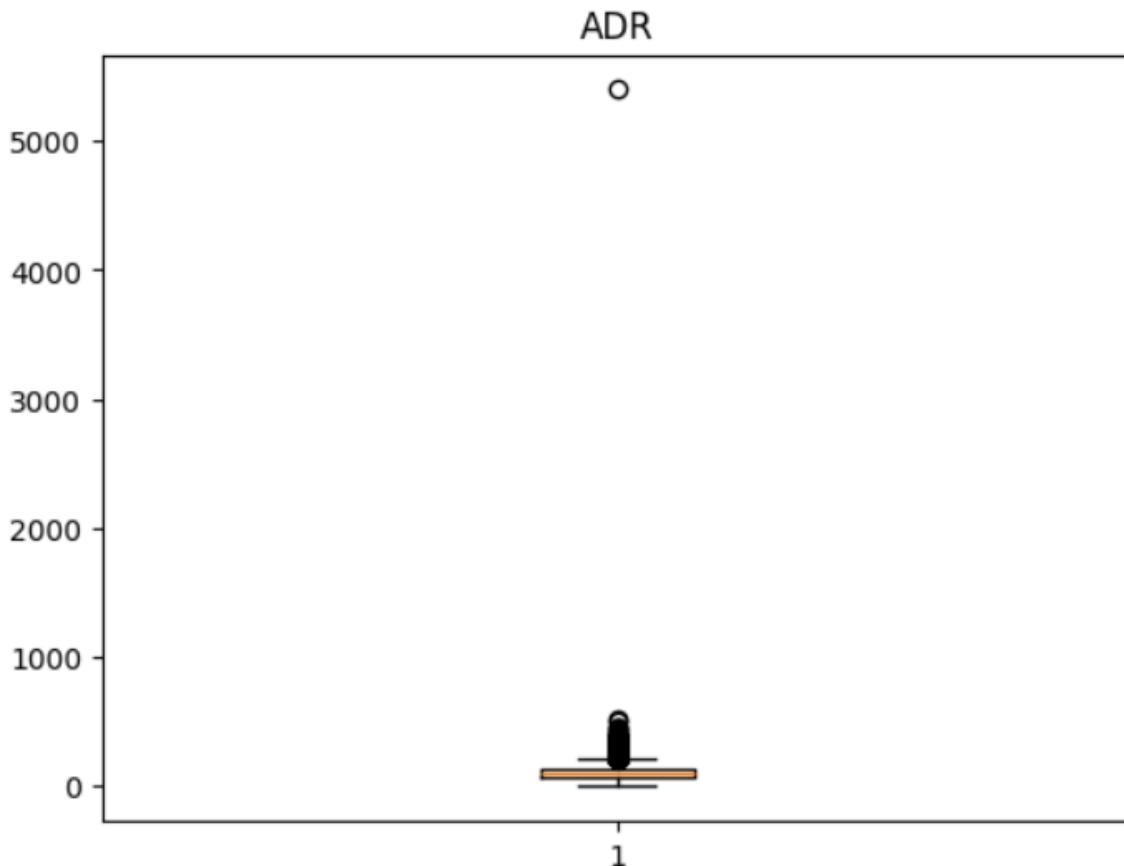
Hình 2. 19 Thông kê thông tin thuộc tính định tính

3. Làm sạch dữ liệu:

3.1. Xử lý dữ liệu bị nhiễu (outlier):

- Ngoại lệ (**Outlier**) chỉ một điểm dữ liệu có giá trị khác biệt đáng kể so với các điểm dữ liệu khác trong tập dữ liệu.
- Dựa vào bảng phân tích các thuộc tính định lượng trên, ta quan sát giá trị trung bình và giá trị nhỏ nhất hoặc lớn nhất của tất cả các cột, cột **ADR** có giá trị âm, và giá trị nhỏ nhất cực kỳ xa so với giá trị trung bình, điều này có nghĩa là giá trị nhỏ nhất là một ngoại lệ.
- Vẽ biểu đồ **Boxplot** để xem thông cột **ADR**:

```
#In ra biểu đồ Boxplot để xem giá trị ngoại lệ của cột ADR
plt.boxplot(df['ADR'])
plt.title('ADR')
plt.show()
```



Hình 2. 20 Biểu đồ Boxplot cho cột ADR

- **Nhận xét:** từ biểu đồ ta thấy, ngoại trừ giá trị âm thì cột **ADR** còn có một giá trị cực đại nằm xa so với các giá trị khác. Do đó giá trị cực đại đó cũng là 1 ngoại lệ.
- **Cách xử lý:** xóa đi những giá trị ngoại lệ đó của cột ADR.

```
# Loại bỏ ngoại lệ của cột ADR
df_no = df.copy()
df_no = df_no.drop(df_no[df_no.ADR < 0].index)
df_no = df_no.drop(df_no[df_no.ADR > 5000].index)
```

Hình 2. 21 Loại bỏ cá ngoại lệ của cột ADR

3.2. Xử lí dữ liệu bị thiếu:

- Liệt kê dữ liệu **NULL**:

```
# Liệt kê dữ liệu null  
df.isnull().sum()
```

IsCanceled	0
LeadTime	0
ArrivalDateYear	0
ArrivalDateMonth	0
ArrivalDateWeekNumber	0
ArrivalDateDayOfMonth	0
StaysInWeekendNights	0
StaysInWeekNights	0
Adults	0
Children	4
Babies	0
Meal	0
Country	488
MarketSegment	0
DistributionChannel	0
IsRepeatedGuest	0
PreviousCancellations	0
PreviousBookingsNotCanceled	0
ReservedRoomType	0
AssignedRoomType	0
BookingChanges	0
DepositType	0
Agent	16340
Company	112593
DaysInWaitingList	0
CustomerType	0
ADR	0
RequiredCarParkingSpaces	0
TotalOfSpecialRequests	0
ReservationStatus	0
ReservationStatusDate	0
hotel	0
dtvqe: int64	

Hình 2. 22 Liệt kê các giá trị null

- Với tỉ lệ dữ liệu **NULL** trên toàn dữ liệu là:

```
#Tỉ lệ dữ liệu null  
(df.isnull().sum().sum() / (119390 * 32)) * 100
```

3.387663330262166

Hình 2. 23 Tỉ lệ dữ liệu null

- Tìm và sắp xếp các dữ liệu **Null**:

```

# Tìm trong dữ liệu giá trị null, tính tổng chúng lại, sắp xếp giảm dần
total = df_no.isnull().sum().sort_values(ascending=False)

# Tổng giá trị null của một cột/ số dòng của cột null đó
percent_1=df_no.isnull().sum()/df_no.isnull().count()*100

# Làm tròn 1 chữ số và sắp giảm dần
percent_2 = (round(percent_1,2)).sort_values(ascending = False)

# Xuất 5 thuộc tính có số Lượng giá trị rỗng nhiều nhất
missing_data = pd.concat([total,percent_2],axis=1,keys=[ 'Total','%'])
missing_data.head(5)

```

	Total	%
Company	112593	94.31
Agent	16340	13.69
Country	488	0.41
Children	4	0.00
IsCanceled	0	0.00

Hình 2. 24 Liệt kê số giá trị và tỉ lệ null của các cột

- **Nhận xét:**

- Thuộc tính Children có 4 giá trị bị thiếu.
- Thuộc tính Country có 488 giá trị bị thiếu, chiếm 0,41% dữ liệu.
- Thuộc tính Agent có 16340 giá trị bị thiếu, chiếm 13,69% dữ liệu.
- Thuộc tính Company có 112593 giá trị bị thiếu, chiếm 94,31% dữ liệu.

- **Cách xử lý:**

- Đối với thuộc tính Children, do số lượng giá trị Null quá ít với tỷ lệ xấp xỉ 0%, nên ta loại bỏ các dòng dữ liệu đó sẽ không ảnh hưởng đến kết quả của dữ liệu.

```

# Xóa các dòng Children bị null
df_no.dropna(subset=[ 'Children' ], inplace=True)

```

Hình 2. 25 Xóa các dòng dữ liệu Children bị null

- Đối với thuộc tính Country, do có kiểu dữ liệu thuộc lớp object nên ta dùng giá trị xuất hiện nhiều nhất để điền vào những ô còn trống.

```
# Xem thông tin của Country
df_no['Country'].describe()
```

count	118898
unique	177
top	PRT
freq	48586
Name:	Country, dtype: object

Hình 2. 26 Xem thông tin của cột Country

```
# Nhận thấy rằng, giá trị "PRT" phổ biến nhất nên ta chọn giá trị đó để điền vào các vị trí còn trống
df_no['Country'].fillna(df_no['Country'].mode().to_string(), inplace=True)
```

Hình 2. 27 Thay thế giá trị rỗng của Country bằng PRT

- Đối với 2 thuộc tính Agent và Company do có số lượng giá trị null lớn chiếm lần lượt 13,69% và 94,31% trên tổng số dữ liệu nên ta sẽ cho các giá trị null thành lần lượt là ‘No Agent’ và ‘No Company’.

```
# Cho các giá trị null của Agent và Company thành Lần Lượt Là 'No Agent' và 'No Company'.
df_no['Agent'] = df_no['Agent'].fillna('No Agent')
df_no['Company'] = df_no['Company'].fillna('No Company')
```

Hình 2. 28 Biến đổi các giá trị null của Agent và Company thành giá trị mặc định

- *Kết quả:*

```
# Liệt kê dữ liệu null
df_no.isnull().sum()

IsCanceled          0
LeadTime             0
ArrivalDateYear      0
ArrivalDateMonth     0
ArrivalDateWeekNumber 0
ArrivalDateDayOfMonth 0
StaysInWeekendNights 0
StaysInWeekNights    0
Adults               0
Children              0
Babies                0
Meal                  0
Country               0
MarketSegment          0
DistributionChannel    0
IsRepeatedGuest        0
PreviousCancellations 0
PreviousBookingsNotCanceled 0
ReservedRoomType        0
AssignedRoomType        0
BookingChanges          0
DepositType             0
Agent                  0
Company                 0
DaysInWaitingList       0
CustomerType            0
ADR                     0
RequiredCarParkingSpaces 0
TotalOfSpecialRequests 0
ReservationStatus        0
ReservationStatusDate    0
hotel                   0
dtype: int64
```

Hình 2. 29 *Liệt kê lại các giá trị null*

3.3. Kiểm tra ý nghĩa của dữ liệu:

- Tại cùng một thời điểm thì các thuộc tính **adults**, **children** và **babies** có đồng thời bằng không hay không? Nếu có thì xóa các dòng dữ liệu đó.

```
# Cột filter dùng để kiểm tra có sự tồn tại của cùng lúc bằng 0 của các thuộc tính về số Lượng người
filter = (df_no.Children == 0) & (df_no.Adults == 0) & (df_no.Babies == 0)
filter

0      False
1      False
2      False
3      False
4      False
...
119385  False
119386  False
119387  False
119388  False
119389  False
Length: 119384, dtype: bool
```

Hình 2. 30 Kiểm tra adults, children và babies có đồng thời bằng không

	IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	ArrivalDateWeekNumber	ArrivalDateDayOfMonth	StaysInWeekendNights	StaysInWeekNights	Adults
2224	0	1	2015	October	41	6	0	3	0
2409	0	0	2015	October	42	12	0	0	0
3181	0	36	2015	November	47	20	1	2	0
3684	0	165	2015	December	53	30	1	4	0
3708	0	165	2015	December	53	30	2	4	0
...
115029	0	107	2017	June	26	27	0	3	0
115091	0	1	2017	June	26	30	0	1	0
116251	0	44	2017	July	28	15	1	1	0
116534	0	2	2017	July	28	15	2	5	0
117087	0	170	2017	July	30	27	0	2	0

180 rows × 32 columns

Hình 2. 31 Xuất ra các dòng có adults, children và babies đồng thời bằng không

- Nhận xét:** có 180 dòng dữ liệu nào mà cả 3 thuộc tính về người đều bằng 0.
- Cách xử lý:** xóa các dòng đó:

```
# Xóa các dòng cả 3 thuộc tính về người đều bằng 0
df_no = df_no.drop(df_no.index[filter])
```

Hình 2. 32 Xóa các dòng có 3 thuộc tính đó bằng 0

3.4. Kiểm tra thuộc tính của dữ liệu:

df_no.dtypes	
IsCanceled	int64
LeadTime	float64
ArrivalDateYear	int64
ArrivalDateMonth	object
ArrivalDateWeekNumber	int64
ArrivalDateDayOfMonth	int64
StaysInWeekendNights	float64
StaysInWeekNights	float64
Adults	float64
Children	float64
Babies	float64
Meal	object
Country	object
MarketSegment	object
DistributionChannel	object
IsRepeatedGuest	int64
PreviousCancellations	float64
PreviousBookingsNotCanceled	float64
ReservedRoomType	object
AssignedRoomType	object
BookingChanges	float64
DepositType	object
Agent	object
Company	object
DaysInWaitingList	float64
CustomerType	object
ADR	float64
RequiredCarParkingSpaces	float64
TotalOfSpecialRequests	float64
ReservationStatus	object
ReservationStatusDate	object
hotel	object
dtype: object	

Hình 2. 33 *Liệt kê kiểu dữ liệu của các cột*

- Ta nhận thấy có 1 số thuộc tính số lượng có kiểu dữ liệu là float trong khi các cột đó thể hiện giá trị thời gian hoặc là yes/no nên ta thay đổi kiểu dữ liệu cho các thuộc tính này.

```
#Thay đổi từ float sang int
df_no['StaysInWeekendNights'] = df_no['StaysInWeekendNights'].astype(int)
df_no['StaysInWeekNights'] = df_no['StaysInWeekNights'].astype(int)
df_no['Adults'] = df_no['Adults'].astype(int)
df_no['Children'] = df_no['Children'].astype(int)
df_no['Babies'] = df_no['Babies'].astype(int)
df_no['PreviousCancellations'] = df_no['PreviousCancellations'].astype(int)
df_no['PreviousBookingsNotCanceled'] = df_no['PreviousBookingsNotCanceled'].astype(int)
df_no['BookingChanges'] = df_no['BookingChanges'].astype(int)
df_no['DaysInWaitingList'] = df_no['DaysInWaitingList'].astype(int)
df_no['RequiredCarParkingSpaces'] = df_no['RequiredCarParkingSpaces'].astype(int)
df_no['TotalOfSpecialRequests'] = df_no['TotalOfSpecialRequests'].astype(int)
```

Hình 2. 34 Thay đổi dữ liệu từ float sang int

- Tương tự, ta thấy thuộc tính **ReservationStatusDate** là kiểu dữ liệu ngày nhưng lại có kiểu dữ liệu là **object**, có thể gây ra ảnh hưởng trong quá trình phân tích nên ta tiếp tục thay đổi kiểu dữ liệu.

```
# Đổi từ object sang date
df_no['ReservationStatusDate'] = pd.to_datetime(df_no['ReservationStatusDate'])
```

Hình 2. 35 Thay đổi dữ liệu từ object sang date

3.5. Xử lý dữ liệu bị trùng lặp:

- Kiểm tra dữ liệu có bị trùng lặp không.

```
# Đếm số dòng trùng nhau
num_duplicate_rows = df_no.duplicated().sum()
print(num_duplicate_rows)
```

31980

Hình 2. 36 Đếm số dòng trùng nhau

- Ta thấy có 31980 dòng dữ liệu bị trùng lặp. Nhóm sẽ xử lý bằng cách xóa đi các dòng trùng lặp, chỉ giữ lại 1 dòng đầu trong các cặp trùng lặp.

```
# Xóa các dòng trùng lặp, chỉ giữ lại dòng đầu tiên
df_cleaned = df_no.drop_duplicates(keep='first')
```

Hình 2. 37 Xóa các dòng trùng lặp, chỉ giữ lại dòng đầu

- **Kết quả:** không còn dòng trùng lặp trong dữ liệu. Dữ liệu thu gọn còn 87224 dòng.

	IsCanceled	LeadTime	ArrivalDateYear	ArrivalDateMonth	ArrivalDateWeekNumber	ArrivalDateDayOfMonth	StaysInWeekendNights	StaysInWeekNights	Adults
0	0	342	2015	July	27	1	0	0	2
1	0	737	2015	July	27	1	0	0	2
2	0	7	2015	July	27	1	0	1	1
3	0	13	2015	July	27	1	0	1	1
4	0	14	2015	July	27	1	0	2	2
...
119385	0	23	2017	August	35	30	2	5	2
119386	0	102	2017	August	35	31	2	5	3
119387	0	34	2017	August	35	31	2	5	2
119388	0	109	2017	August	35	31	2	5	2
119389	0	205	2017	August	35	29	2	7	2

87224 rows x 32 columns

Hình 2. 38 Dữ liệu sau khi xóa các dòng trùng nhau

4. Thêm thuộc tính cho dữ liệu:

- Tạo thuộc tính **ArrivalDate** là một ngày cụ thể theo format **%d/%m/%Y**.

```
# Chuyển đổi thuộc tính ArrivalDateMonth thành số
months = {
    "January": 1,
    "February": 2,
    "March": 3,
    "April": 4,
    "May": 5,
    "June": 6,
    "July": 7,
    "August": 8,
    "September": 9,
    "October": 10,
    "November": 11,
    "December": 12
}
for DF in [df_cleaned]:
    DF["ArrivalDateMonth"] = DF["ArrivalDateMonth"].map(months)

# Tạo thuộc tính ArrivalDate kiểu dữ liệu datetime thông qua ba thuộc tính: ArrivalDateYear, ArrivalDateMonth và ArrivalDateDayOfMonth
cols=["ArrivalDateYear","ArrivalDateMonth","ArrivalDateDayOfMonth"]
df_cleaned['ArrivalDate'] = df_cleaned[cols].apply(lambda x: '-'.join(x.values.astype(str)), axis="columns")
df_cleaned['ArrivalDate'] = pd.to_datetime(df_cleaned['ArrivalDate'])
```

Hình 2. 39 Tạo thuộc tính ArrivalDate

- Tạo thuộc tính **StatusMinusArrivalDate** là hiệu số giữa **ArrivalDate** và **ReservationStatusDate**.

```

# Tạo thuộc tính StatusMinusArrivalDate là hiệu số giữa ArrivalDate và ReservationStatusDate.
df_cleaned['StatusMinusArrivalDate'] = np.abs(df_cleaned['ArrivalDate'] - df_cleaned['ReservationStatusDate']).astype(str)
def format_lenght(date):
    return date[0]
df_cleaned['StatusMinusArrivalDate'] = df_cleaned['StatusMinusArrivalDate'].map(format_lenght).astype(int)

```

Hình 2. 40 Tạo thuộc tính StatusMinusArrivalDate

Nhận xét:

- Nếu khách hàng nhận phòng thì **status_minus_arrival_date** là số ngày khách hàng nghỉ lại tại khách sạn.
- Nếu khách hàng hủy đặt phòng từ **status_minus_arrival_date** là số ngày trước khi đến mà khách hàng đã hủy phòng.
- Tạo thuộc tính **TotalNights** là tổng số đêm ở vào ngày cuối tuần (StaysInWeekendNights) và ngày trong tuần (StaysInWeekNights).

```

# Tạo thuộc tính TotalNights là tổng của StaysInWeekendNights và StaysInWeekNights
df_cleaned['TotalNights'] = df_cleaned['StaysInWeekendNights'] + df_cleaned['StaysInWeekNights']

```

Hình 2. 41

- Tạo thuộc tính **RoomMatch**: là sự kết hợp giữa hai thuộc tính **ReservedRoomType** và **AssignedRoomType** chỉ ra liệu giữa phòng đã đặt trước và phòng đã chỉ định của khách có giống nhau hay không.

```

# Tạo thuộc tính RoomMatch: là sự kết hợp giữa hai thuộc tính ReservedRoomType và AssignedRoomType chỉ ra liệu giữa phòng đã đặt trước và phòng đã chỉ
df_cleaned['RoomMatch'] = (df_cleaned['ReservedRoomType'] == df_cleaned['AssignedRoomType']) * 1

```

Hình 2. 42 Tạo thuộc tính RoomMatch

Nhận xét:

- Nếu phòng đã đặt trước và phòng đã chỉ định của khách giống nhau thì trả về 1.
- Nếu phòng đã đặt trước và phòng đã chỉ định của khách khác nhau thì trả về 0.
- Tạo thuộc tính **TotalPeople** là tổng số lượng người lấy từ các thuộc tính **adults**, **children**, **babies**.

```

# Tạo thuộc tính TotalPeople là tổng số Lượng người lấy từ các thuộc tính Adults, Children, Babies.
df_cleaned['TotalPeople'] = df_cleaned['Adults'] + df_cleaned['Children'] + df_cleaned['Babies']

```

Hình 2. 43 Tạo thuộc tính TotalPeople

- Tạo thuộc tính **HaveChild** dựa trên các thuộc tính **Children**, **Babies** có số lượng nhiều hơn 0 trong tập dữ liệu.

```
# Tạo thuộc tính HaveChild dựa trên các thuộc tính Children, Babies có số lượng nhiều hơn 0 trong tập dữ liệu.
df_cleaned['HaveChild'] = ((df_cleaned['Children'] != 0) | (df_cleaned['Babies'] != 0)) * 1
```

Hình 2. 44 Tạo thuộc tính HaveChild

5. Giảm thuộc tính của dữ liệu:

- Thực hiện tính toán hệ số tương quan giữa cột **IsCanceled** với các cột khác:

```
# Chuyển đổi các cột phân loại thành dạng số
df_cor = df_cleaned.copy()
le = LabelEncoder()
col_ob = [
    'Meal',
    'Country',
    'MarketSegment',
    'DistributionChannel',
    'ReservedRoomType',
    'AssignedRoomType',
    'DepositType',
    'CustomerType',
    'ReservationStatus',
]
for col in col_ob:
    df_cor[col] = le.fit_transform(df_cor[col])

# Lọc ra các cột có kiểu dữ liệu là số
numerical_columns = df_cleaned.select_dtypes(include=[np.number])

# Tính hệ số tương quan tuyệt đối giữa các cột số và 'IsCanceled', sắp xếp theo độ lớn từ cao xuống
correlation_with_is_canceled = numerical_columns.corr()['IsCanceled'].abs().sort_values(ascending=False)[1:]

# Hiển thị kết quả
print(correlation_with_is_canceled)
```

Hình 2. 45 Tính hệ số tương quan

```

# Hiển thị kết quả
print(correlation_with_is_canceled)

RoomMatch          0.213332
LeadTime           0.184621
RequiredCarParkingSpaces 0.184445
ADR                0.132829
TotalOfSpecialRequests 0.120846
TotalPeople         0.098975
BookingChanges     0.093222
IsRepeatedGuest    0.088730
ArrivalDateYear    0.088179
TotalNights        0.085494
StaysInWeekNights 0.084208
Adults              0.080232
Children            0.067213
StaysInWeekendNights 0.061058
PreviousBookingsNotCanceled 0.052165
PreviousCancellations 0.051512
HaveChild           0.051458
Babies              0.020621
StatuMinusArrivalDate 0.016075
ArrivalDateDayOfMonth 0.005500
DaysInWaitingList   0.004717
ArrivalDateMonth    0.003931
ArrivalDateWeekNumber 0.001671
Name: IsCanceled, dtype: float64

```

Hình 2. 46 Hiển thị các hệ số tương quan theo thứ tự giảm dần

- Xóa thuộc tính **ArrivalDateYear**, **ArrivalDateMonth** và **ArrivalDateDayOfMonth** vì đã dùng các thuộc tính này để tạo **ArrivalDate**.
- Xóa thuộc tính **ArrivalDate**, **ReservationStatusDate** vì đã dùng các thuộc tính này để tạo **StatuMinusArrivalDate**.
- Xóa thuộc tính **ReservedRoomType** và **AssignedRoomType** vì đã tạo thuộc tính **RoomMatch**.
- Xóa thuộc tính **Children**, **Babies** vì đã dùng các thuộc tính này để tạo thuộc tính **TotalPeople** và **HaveChild**.
- Xóa thuộc tính **Agent** và **Company** vì chứa các dữ liệu không được định nghĩa cao.

- Xóa thuộc tính **StaysInWeekendNights** và **StaysInWeekNights** vì dùng các thuộc tính này để tạo **TotalNights**.
- Xóa thuộc tính **ArrivalDateWeekNumber** và **DaysInWaitingList** vì có mức tương quan rất thấp với thuộc tính **IsCanceled**.
- Xóa thuộc tính **ReservationStatus** vì có mức tương quan rất cao với thuộc tính **IsCanceled**.

```
# Xóa các thuộc tính không cần thiết
df_cleaned=df_cleaned.drop(['DaysInWaitingList', 'ArrivalDateYear', 'ArrivalDateMonth', 'ArrivalDateWeekNumber',
                           'ArrivalDateDayOfMonth', 'StaysInWeekendNights', 'StaysInWeekNights', 'Children', 'Babies',
                           'ReservedRoomType', 'AssignedRoomType', 'Agent', 'Company', 'ReservationStatusDate', 'ArrivalDate', 'ReservationStatus'],axis=1)
```

Hình 2. 47 Xóa các thuộc tính không cần thiết

6. Rời rạc hóa dữ liệu:

- Thống kê các thuộc tính định lượng:

```
df_cleaned.describe().T
```

	count	mean	std	min	25%	50%	75%	max
IsCanceled	87224.0	0.275199	0.446617	0.0	0.00	0.0	1.0	1.0
LeadTime	87224.0	79.973746	86.059046	0.0	11.00	49.0	125.0	737.0
Adults	87224.0	1.879345	0.621733	0.0	2.00	2.0	2.0	55.0
IsRepeatedGuest	87224.0	0.038556	0.192535	0.0	0.00	0.0	0.0	1.0
PreviousCancellations	87224.0	0.030404	0.369357	0.0	0.00	0.0	0.0	26.0
PreviousBookingsNotCanceled	87224.0	0.184043	1.733082	0.0	0.00	0.0	0.0	72.0
BookingChanges	87224.0	0.268481	0.710626	0.0	0.00	0.0	0.0	18.0
ADR	87224.0	106.462163	51.880086	0.0	72.25	98.2	134.1	510.0
RequiredCarParkingSpaces	87224.0	0.084312	0.281667	0.0	0.00	0.0	0.0	8.0
TotalOfSpecialRequests	87224.0	0.698925	0.832059	0.0	0.00	0.0	1.0	5.0
StatuMinusArrivalDate	87224.0	3.091970	2.142527	0.0	1.00	3.0	4.0	9.0
TotalNights	87224.0	3.628520	2.742894	0.0	2.00	3.0	5.0	69.0
RoomMatch	87224.0	0.851085	0.356007	0.0	1.00	1.0	1.0	1.0
TotalPeople	87224.0	2.029097	0.790162	1.0	2.00	2.0	2.0	55.0
HaveChild	87224.0	0.104363	0.305733	0.0	0.00	0.0	0.0	1.0

Hình 2. 48 Thông kê thuộc tính định lượng

6.1. Thuộc tính LeadTime:

Chuyển đổi thuộc tính **LeadTime** về kiểu số nguyên và tạo ra các nhóm thời gian chia thời gian làm 4 phần nhỏ để thể hiện hành vi đặt phòng sớm hay muộn, sử dụng qcut để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn **binning**).

```
# LeadTime có giá trị lớn nhất là 737 và nhỏ nhất là 0 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["LeadTime"], q=4)
# Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.
test.value_counts()
```

```
LeadTime
(-0.001, 11.0]    21880
(11.0, 49.0]      21836
(49.0, 125.0]     21796
(125.0, 737.0]    21712
Name: count, dtype: int64
```

Hình 2. 49 Chia giờ thuộc tính LeadTime

- Các giờ được đánh số theo thứ tự từ 1 đến 4 cho các khoảng tăng dần:

```
# Các giờ được đánh số theo thứ tự từ 1 đến 4 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['LeadTime']<=11, 'LeadTime']=1
    dataset.loc[(dataset['LeadTime']>11) & (dataset['LeadTime']<=49), 'LeadTime']=2
    dataset.loc[(dataset['LeadTime']>49) & (dataset['LeadTime']<=125), 'LeadTime']=3
    dataset.loc[dataset['LeadTime']>125, 'LeadTime']=4
    dataset['LeadTime']=dataset['LeadTime'].astype(int)
```

Hình 2. 50 Chia dữ liệu theo giờ thuộc tính LeadTime

6.2. Thuộc tính Adults:

Chuyển đổi thuộc tính **Adults** về kiểu số nguyên và tạo ra các nhóm người lớn chia làm 2 phần nhỏ để thể hiện quy mô đoàn khách, sử dụng qcut để phân chia đôi ra ở mỗi nhóm (phương pháp làm mịn **binning**).

```
# Adults có giá trị lớn nhất là 55 và nhỏ nhất là 0 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["Adults"], q=2)
# Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.
test.value_counts()
```

```
Adults
(-0.001, 2.0]    81214
(2.0, 55.0]       6010
Name: count, dtype: int64
```

Hình 2. 51 Chia giờ thuộc tính Adults

```
# Các giỏ được đánh số theo thứ tự từ 1 đến 2 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['Adults']<=2,'Adults']=1
    dataset.loc[dataset['Adults']>2,'Adults']=2
    dataset['Adults']=dataset['Adults'].astype(int)
```

Hình 2. 52 Chia dữ liệu theo giỏ thuộc tính Adults

6.3. Thuộc tính ADR:

Chuyển đổi thuộc tính **ADR** về kiểu số nguyên và tạo ra các nhóm doanh thu chia doanh thu làm 4 phần nhỏ, sử dụng **qcut** để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```
# ADR có giá trị Lớn nhất là 510 và nhỏ nhất là 0 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["ADR"], q=4)
# Đếm giá trị thuộc tính mỗi giỏ và hiển thị mỗi giỏ có bao nhiêu giá trị.
test.value_counts()
```

```
ADR
(-0.001, 72.25]    22042
(98.2, 134.1]     21879
(134.1, 510.0]    21732
(72.25, 98.2]     21571
Name: count, dtype: int64
```

Hình 2. 53 Chia giỏ thuộc tính ADR

```
# Các giỏ được đánh số theo thứ tự từ 1 đến 4 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['ADR']<=72.25,'ADR']=1
    dataset.loc[(dataset['ADR']>72.25) & (dataset['ADR']<=98.2),'ADR']=2
    dataset.loc[(dataset['ADR']>98.2) & (dataset['ADR']<=134.1),'ADR']=3
    dataset.loc[dataset['ADR']>134.1,'ADR']=4
    dataset['ADR']=dataset['ADR'].astype(int)
```

Hình 2. 54 Chia dữ liệu theo giỏ thuộc tính ADR

6.4. Thuộc tính StatusMinusArrivalDate:

Chuyển đổi thuộc tính **StatuMinusArrivalDate** về kiểu số nguyên và tạo ra các nhóm theo số ngày đặt trước chia làm 3 phần nhỏ, sử dụng **qcut** để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```
# StatuMinusArrivalDate có giá trị Lớn nhất là 9 và nhỏ nhất là 0 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["StatuMinusArrivalDate"], q=3)
# Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.
test.value_counts()
```

```
StatuMinusArrivalDate
(-0.001, 2.0]    41493
(2.0, 4.0]      26230
(4.0, 9.0]      19501
Name: count, dtype: int64
```

Hình 2. 55 Chia giờ thuộc tính StatuMinusArrivalDate

```
## Các giờ được đánh số theo số thứ tự từ 1 đến 3 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['StatuMinusArrivalDate']<=2, 'StatuMinusArrivalDate']=1
    dataset.loc[(dataset['StatuMinusArrivalDate']>2) & (dataset['StatuMinusArrivalDate']<=4), 'StatuMinusArrivalDate']=2
    dataset.loc[dataset['StatuMinusArrivalDate']>4, 'StatuMinusArrivalDate']=3
    dataset['StatuMinusArrivalDate']=dataset['StatuMinusArrivalDate'].astype(int)
```

Hình 2. 56 Chia dữ liệu theo giờ thuộc tính StatuMinusArrivalDate

6.5. Thuộc tính TotalNights:

Chuyển đổi thuộc tính **TotalNights** về kiểu số nguyên và tạo ra các nhóm tổng đêm ở chia làm 4 phần nhỏ đánh giá nhu cầu lưu trú của khách hàng, sử dụng **qcut** để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```
# TotalNights có giá trị Lớn nhất là 69 và nhỏ nhất là 0 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["TotalNights"], q=4)
# Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.
test.value_counts()
```

```
TotalNights
(-0.001, 2.0]    33490
(3.0, 5.0]      20294
(2.0, 3.0]      17827
(5.0, 69.0]     15613
Name: count, dtype: int64
```

Hình 2. 57 Chia giờ thuộc tính TotalNights

```

## Các giỏ được đánh số theo thứ tự từ 1 đến 4 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['TotalNights']<=2,'TotalNights']=1
    dataset.loc[(dataset['TotalNights']>2) & (dataset['TotalNights']<=3),'TotalNights']=2
    dataset.loc[(dataset['TotalNights']>3) & (dataset['TotalNights']<=5),'TotalNights']=3
    dataset.loc[dataset['TotalNights']>5,'TotalNights']=4
    dataset['TotalNights']=dataset['TotalNights'].astype(int)

```

Hình 2. 58 Chia dữ liệu theo giỏ thuộc tính TotalNights

6.6. Thuộc tính TotalPeople:

Chuyển đổi thuộc tính **TotalPeople** về kiểu số nguyên và tạo ra các nhóm tổng người ở chia làm 2 phần nhỏ, sử dụng qcut để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```

# TotalPeople có giá trị lớn nhất là 55 và nhỏ nhất là 1 nên ta sẽ dùng phương pháp làm mịn binning
test = pd.qcut(df_cleaned["TotalNights"], q=2)
# Đếm giá trị thuộc tính mỗi giỏ và hiển thị mỗi giỏ có bao nhiêu giá trị.
test.value_counts()

TotalNights
(0.999, 2.0]    51317
(2.0, 4.0]      35907
Name: count, dtype: int64

```

Hình 2. 59 Chia giỏ thuộc tính TotalPeople

```

## Các giỏ được đánh số theo thứ tự từ 1 đến 2 cho các khoảng tăng dần
for dataset in [df_cleaned]:
    dataset.loc[dataset['TotalPeople']<=2,'TotalPeople']=1
    dataset.loc[dataset['TotalPeople']>2,'TotalPeople']=2
    dataset['TotalPeople']=dataset['TotalPeople'].astype(int)

```

Hình 2. 60 Chia dữ liệu theo giỏ thuộc tính TotalPeople

- Các thuộc tính định lượng khác không rời rạc hóa dữ liệu vì:
 - Với thuộc tính **IsCanceled**, **IsRepeatedGuest**, **RoomMatch**, **HaveChild** chỉ có 2 giá trị 1 và 0 nên không cần rời rạc hóa.
 - Với các thuộc tính định lượng còn lại rời rạc hóa vì có quá nhiều giá trị trùng lặp (đặc biệt là 0) trong dữ liệu của thuộc tính.

7. Dán nhãn số cho các thuộc tính định tính:

- Sử dụng hàm **Label Encoding** để thực hiện:

```
# Sử dụng hàm Label Encoding
le = LabelEncoder()
col_об = [
    'Meal',
    'Country',
    'MarketSegment',
    'DistributionChannel',
    'DepositType',
    'CustomerType',
    'hotel',
]
for col in col_об:
    df_cleaned[col] = le.fit_transform(df_cleaned[col])

df_cleaned.head(10)
```

	IsCanceled	LeadTime	Adults	Meal	Country	MarketSegment	DistributionChannel	IsRepeatedGuest	PreviousCancellations	PreviousBookingsNotCanceled	...	Cust
0	0	4	1	0	136	3	1	0	0	0	0	...
1	0	4	1	0	136	3	1	0	0	0	0	...
2	0	1	1	0	60	3	1	0	0	0	0	...
3	0	2	1	0	60	2	0	0	0	0	0	...
4	0	2	1	0	60	6	3	0	0	0	0	...
6	0	1	1	0	136	3	1	0	0	0	0	...
7	0	1	1	1	136	3	1	0	0	0	0	...
8	1	3	1	0	136	6	3	0	0	0	0	...
9	1	3	1	2	136	5	3	0	0	0	0	...
10	1	2	1	0	136	6	3	0	0	0	0	...

Hình 2. 61 Dán nhãn số cho các thuộc tính định tính

8. Lấy tập mẫu cho dữ liệu (Sampling data):

Vì sự chênh lệch đáng kể giữa dữ liệu về việc nhận phòng và hủy phòng có thể dẫn đến bias (thiên vị) trong quá trình phát triển mô hình, chúng ta quyết định thực hiện một quy trình lọc ngẫu nhiên trên dữ liệu nhận phòng và chọn ra một số lượng mẫu tương đương với số lượng dữ liệu về hủy phòng. Điều này giúp đảm bảo rằng các mẫu dữ liệu sẽ đồng đều đại diện cho cả hai khía cạnh của quá trình đặt phòng và hủy phòng, tạo điều kiện thuận lợi cho quá trình xây dựng mô hình mà không gây ra sự thiên vị không mong muốn.

```
# Kiểm tra các giá trị trong thuộc tính IsCanceled  
value_counts = df_cleaned['IsCanceled'].value_counts()  
print(value_counts)
```

```
IsCanceled  
0      63220  
1      24004  
Name: count, dtype: int64
```

Hình 2. 62 Kiểm tra các giá trị trong IsCanceled

Phương pháp thực hiện:

```
# Trộn Lần dữ Liệu.  
shuffled_df = df_cleaned.sample(frac=1, random_state=4)  
  
# Đặt tất cả các trường hợp hủy phòng vào một bộ dữ liệu riêng.  
Canceled_df = shuffled_df.loc[shuffled_df['IsCanceled'] == 1]  
  
# Chọn ngẫu nhiên 492 quan sát từ nhóm không hủy phòng (Lớp đa số)  
NonCanceled_df = shuffled_df.loc[shuffled_df['IsCanceled'] == 0].sample(n=df_cleaned["IsCanceled"].value_counts()[1], random_state=42)  
  
# Ghép hai bảng dữ liệu lại với nhau  
sampling_df = pd.concat([Canceled_df, NonCanceled_df])
```

Hình 2. 63 Thực hiện Sampling dữ liệu

Kết quả:

```
# kiểm tra kết quả  
value_counts = sampling_df['IsCanceled'].value_counts()  
print(value_counts)
```

```
IsCanceled  
1      24004  
0      24004  
Name: count, dtype: int64
```

Hình 2. 64 Kết quả sau sampling dữ liệu

CHƯƠNG 3: CÁC THUẬT TOÁN KHAI THÁC DỮ LIỆU

1. Chuẩn bị dữ liệu để chạy các thuật toán:

- Loại bỏ các thuộc tính có tương quan cao ra khỏi thuộc tính.

```
corr = 0.7 # Nguồn tương quan để xác định xem hai biến có tương quan cao hay không.

def correlation(dataset, threshold):
    col_corr = set() # Tạo một tập hợp để lưu trữ tên của các cột có tương quan cao.
    corr_matrix = dataset.corr() # Tính ma trận tương quan giữa tất cả các cặp biến.

    # Duyệt qua tất cả các cặp biến trong ma trận tương quan.
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr # Trả về tập hợp các tên cột có tương quan cao.

# Tạo một bản sao của DataFrame gốc
df1 = df.copy()
corr_features = correlation(df1, corr)

# In ra thông tin về các thuộc tính có tương quan
print("-----With corr equal: {0}-----".format(corr))
print("Length of corr_feature equal: {0}".format(len(set(corr_features))))

# In ra danh sách các thuộc tính có tương quan bị loại bỏ
print("Correlated features to be removed:")
for feature in corr_features:
    print(feature)

# Loại bỏ các thuộc tính có tương quan cao từ DataFrame
df1 = df1.drop(corr_features, axis=1)

-----With corr equal: 0.7-----
Length of corr_feature equal: 2
Correlated features to be removed:
HaveChild
DistributionChannel
```

Hình 3. 1 Loại bỏ các thuộc tính có tương quan cao ra khỏi thuộc tính

- Chuẩn bị mảng **feature_np** chứa các thuộc tính của dữ liệu (không bao gồm nhãn) và một mảng **label_np** chứa nhãn tương ứng.

```
# Xóa cột 'IsCanceled' để lấy các thuộc tính khác
feature = df1.drop('IsCanceled',axis=1)
label = df1['IsCanceled']
# Chuyển đổi dữ liệu thành mảng NumPy
feature_np = np.array(feature)
label_np = np.array(label)
```

Hình 3. 2 Chuẩn bị các mảng

- Chuẩn bị các mảng dữ liệu lưu lại thời gian chạy, độ chính xác thuật toán qua các lần chạy:

```

# Thời gian và điểm số cho thuật toán Decision Tree ID3
TimeID3 = []
training_score_id3 = []
test_score_id3 = []
# Thời gian và điểm số cho thuật toán Decision Tree CART
TimeCart = []
training_score_cart = []
test_score_cart = []
# Thời gian và điểm số cho thuật toán Naïve Bayes
TimeNB = []
training_score_nb = []
test_score_nb = []
# Thời gian và điểm số cho thuật toán Random Forest
TimeRF = []
training_score_rf = []
test_score_rf = []
# Thời gian và điểm số cho thuật toán K - Nearest Neighbors
TimeKNN = []
training_score_knn = []
test_score_knn = []
# Thời gian và điểm số cho thuật toán Neural Network
TimeNN = []
training_score_nn = []
test_score_nn = []

```

Hình 3. 3 Chuẩn bị các mảng dữ liệu lưu lại thời gian chạy, độ chính xác thuật toán

2. Thuật toán Cây quyết định (Decision Tree):

2.1. Khái niệm:

- Thuật toán cây quyết định (**Decision Tree**) là một thuật toán học máy giám sát, không đổi số, được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Cây quyết định có cấu trúc cây phân cấp, bao gồm nút gốc, các nhánh, nút nội bộ và nút lá. Mục đích của thuật toán cây quyết định là tạo ra một mô hình huấn luyện có thể dự đoán lớp hoặc giá trị biến mục tiêu

bằng cách học các quy tắc quyết định đơn giản từ dữ liệu đã biết trước đó (dữ liệu huấn luyện).

- Hai thuật toán nổi bật, quan trọng để tạo ra cây quyết định:
 - + **ID3 (Iterative Dichotomiser 3)** là một thuật toán khá cổ điển trong việc xây dựng cây quyết định. Nó hoạt động bằng cách chọn thuộc tính tốt nhất để chia dữ liệu tại mỗi bước. ID3 sử dụng thuật toán **Entropy** và **Information Gain** để đánh giá thuộc tính tốt nhất cho việc chia dữ liệu. Tuy nhiên, ID3 có thể dễ dàng bị ảnh hưởng bởi nhiễu và không xử lý được dữ liệu liên tục.
 - + **CART (Classification and Regression Trees)** là một thuật toán khác để xây dựng cây quyết định, nó có thể xây dựng cả cây phân loại (**classification**) và cây hồi quy (**regression**). CART sử dụng thuật toán **Gini Impurity** để đánh giá thuộc tính tốt nhất cho việc chia dữ liệu hoặc giá trị trung bình của các điểm dữ liệu để thực hiện hồi quy.

2.2. Thực hiện:

2.2.1. Giải thuật Decision Tree ID3:

- Chuẩn bị hàm lưu ảnh cây quyết định ID3.

```
# Lưu cây quyết định dựa trên thuật toán ID3 vào file ảnh.  
def save_tree_id3(id3, i, corr):  
    fig, ax = plt.subplots(figsize=(50,24))  
    tree.plot_tree(id3,filled=True,fontsize=10)  
    plt.savefig('DecisionTreeID3_{0}_{1}.png'.format(corr, i))
```

Hình 3. 4 Hàm lưu ảnh cây quyết định ID3

- Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho Decision Tree ID3.

```
# Đổi với Decision Tree ID3  
id3_probs = id3.predict_proba(x_test) # Dự đoán xác suất cho x_test  
id3_probs = id3_probs[:, 1] # Chọn xác suất của lớp dương  
id3_auc = roc_auc_score(y_test, id3_probs) # Tính AUC cho Decision Tree ID3  
print('Decision Tree ID3: ROC AUC=% .3f' % (id3_auc)) # In AUC  
id3_fpr, id3_tpr, _ = roc_curve(y_test, id3_probs) # Tính FPR, TPR  
plt.plot(id3_fpr, id3_tpr, linestyle='--', label='Decision Tree ID3') # Vẽ đường ROC
```

Hình 3. 5 Hàm để vẽ biểu đồ đường cong ROC cho Decision Tree ID3

- Cài đặt thuật toán **Decision Tree ID3**.

```
# Decision Tree ID3
print("-----Decision Tree ID3-----")
start = time.time()
# Tạo một cây quyết định dựa trên giải thuật ID3
id3 = tree.DecisionTreeClassifier(criterion="entropy", random_state=0)
# Huấn Luyện mô hình
id3.fit(x_train,y_train)
end = time.time()
# Đo thời gian huấn luyện và lưu vào danh sách TimeID3
TimeID3.append(end - start)
# Đánh giá hiệu suất trên tập huấn luyện và lưu kết quả
training_score_id3.append(id3.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
id3_pred = id3.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_id3
id3_score = metrics.accuracy_score(y_test, id3_pred)
test_score_id3.append(id3_score)
# In kết quả accuracy và báo cáo phân loại
print("Decision Tree ID3")
print("Accuracy:", id3_score)
print("Report:", metrics.classification_report(y_test,id3_pred))
# Tạo ma trận nhầm lẫn
id3_cm = metrics.confusion_matrix(y_test,id3_pred)
# Lưu cây quyết định đã huấn luyện
save_tree_id3(id3, i, corr)
```

Hình 3. 6 Cài đặt thuật toán Decision Tree ID3

- Kết quả chạy:

-----Decision Tree ID3-----

Decision Tree ID3

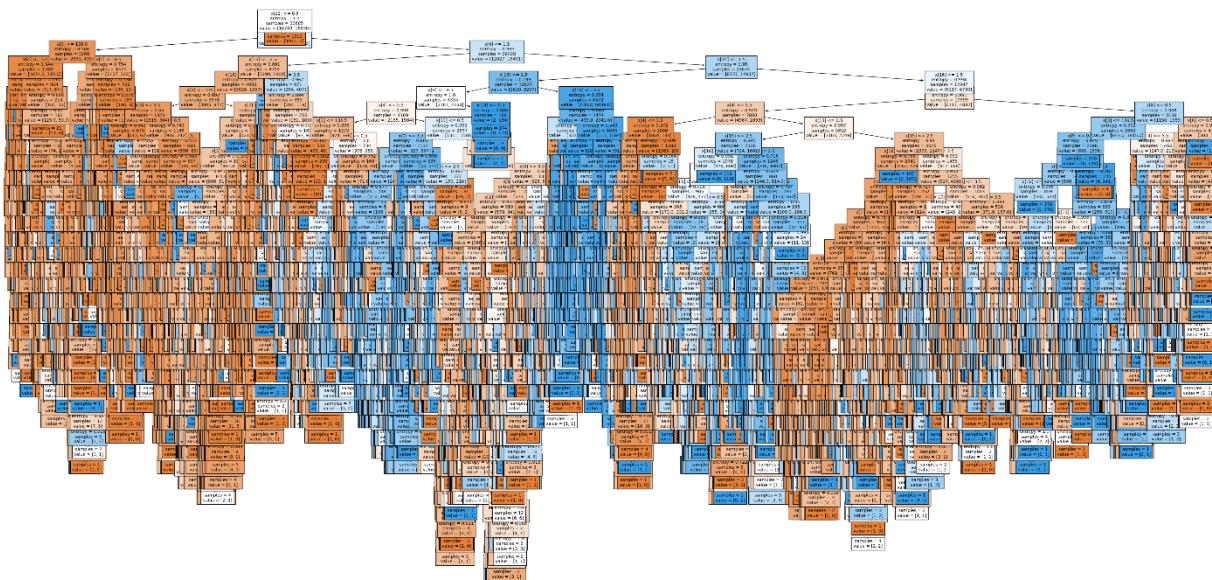
Accuracy: 0.8347566479205721

Report:

	precision	recall	f1-score	support
0	0.82	0.85	0.84	7235
1	0.85	0.82	0.83	7168
accuracy			0.83	14403
macro avg	0.84	0.83	0.83	14403
weighted avg	0.84	0.83	0.83	14403

Hình 3. 7 Kết quả chạy ID3

- Mô hình cây quyết định ID3:



Hình 3. 8 Mô hình cây quyết định ID3

2.2.2. Giải thuật Decision Tree Cart:

- Chuẩn bị hàm lưu ảnh cây quyết định CART:

```
# Lưu cây quyết định dựa trên thuật toán CART vào file ảnh.
def save_tree_cart(cart, i, corr):
    fig, ax = plt.subplots(figsize=(50,24))
    tree.plot_tree(cart,filled=True,fontsize=10)
    plt.savefig('DecisionTreeCart_{0}_{1}.png'.format(corr, i))
```

Hình 3. 9 Hàm lưu ảnh cây quyết định CART

Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho Decision Tree Cart.

```
# Đối với Decision Tree CART
cart_probs = cart.predict_proba(x_test) # Dự đoán xác suất cho x_test
cart_probs = cart_probs[:, 1] # Chọn xác suất của lớp dương
cart_auc = roc_auc_score(y_test, cart_probs) # Tính AUC cho Decision Tree CART
print('Decision Tree Cart: ROC AUC=% .3f' % (cart_auc)) # In AUC
cart_fpr, cart_tpr, _ = roc_curve(y_test, cart_probs) # Tính FPR, TPR
plt.plot(cart_fpr, cart_tpr, linestyle='--', label='Decision Tree Cart') # Vẽ đường ROC
```

Hình 3. 10 Hàm để vẽ biểu đồ đường cong ROC cho Decision Tree Cart

- Cài đặt thuật toán Decision Tree CART.

```

# Decision Tree CART
print("-----Decision Tree Cart-----")
start = time.time()
# Tạo một cây quyết định dựa trên giải thuật CART
cart = tree.DecisionTreeClassifier(criterion="gini", random_state=0)
# Huấn Luyện mô hình
cart.fit(x_train,y_train)
end = time.time()
# Đo thời gian huấn Luyện và lưu vào danh sách TimeCart
TimeCart.append(end - start)
# Đánh giá hiệu suất trên tập huấn Luyện và lưu kết quả
training_score_cart.append(cart.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
cart_pred = cart.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_cart
cart_score = metrics.accuracy_score(y_test, cart_pred)
test_score_cart.append(cart_score)
# In kết quả accuracy và báo cáo phân loại
print("Decision Tree Cart")
print("Accuracy:", cart_score)
print("Report:", metrics.classification_report(y_test,cart_pred))
# Tạo ma trận nhầm lẫn
cart_cm = metrics.confusion_matrix(y_test,cart_pred)
# Lưu cây quyết định đã huấn Luyện
save_tree_cart(cart, i, corr)

```

Hình 3. 11 Cài đặt thuật toán Decision Tree CART

- Kết quả chạy:

```

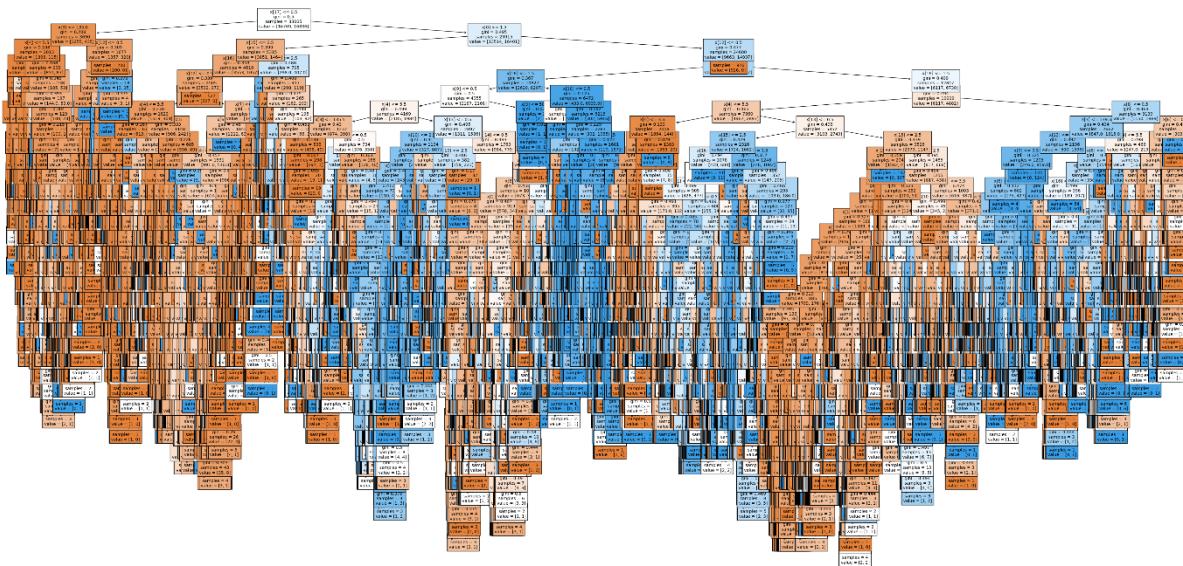
-----Decision Tree Cart-----
Decision Tree Cart
Accuracy: 0.8355203776990905
Report:
          precision    recall   f1-score   support
0           0.82      0.85      0.84      7235
1           0.85      0.82      0.83      7168

accuracy                           0.84      14403
macro avg                           0.84      0.84      14403
weighted avg                          0.84      0.84      14403

```

Hình 3. 12 Kết quả chạy CART

Mô hình cây quyết định CART:



Hình 3. 13 Mô hình cây quyết định CART

3. Thuật toán Naïve Bayes:

3.1. Khái niệm:

- **Naïve Bayes** là một thuật toán phân loại cơ bản dựa trên Định lý Bayes. Nó giả định rằng tất cả các thuộc tính là độc lập với nhau khi đã biết nhãn lớp, điều này làm cho việc tính toán xác suất trở nên đơn giản. Mặc dù có giả định đơn giản này, **Naïve Bayes** thường hoạt

động tốt trên các tập dữ liệu lớn với số lượng thuộc tính lớn. Thuật toán này đặc biệt hữu ích khi làm việc với các bài toán phân loại văn bản hoặc dữ liệu có cấu trúc đơn giản.

3.2. Thực hiện:

- Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho **Naïve Bayes**.

```
# Đối với Naive Bayes
nb_probs = nb.predict_proba(x_test) # Dự đoán xác suất cho x_test
nb_probs = nb_probs[:, 1] # Chọn xác suất của Lớp dương
nb_auc = roc_auc_score(y_test, nb_probs) # Tính AUC cho Naive Bayes
print('Naive Bayes: ROC AUC=% .3f' % (nb_auc)) # In AUC
nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs) # Tính FPR, TPR
plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes') # Vẽ đường ROC
```

Hình 3. 14 Hàm để vẽ biểu đồ đường cong ROC cho Naïve Bayes

- Cài đặt thuật toán **Naïve Bayes**.

```

# Naive Bayes
print("-----Naive Bayes-----")
start = time.time()
# Tạo một mô hình Naive Bayes
nb = GaussianNB()
# Huấn luyện mô hình
nb.fit(x_train,y_train)
end = time.time()
# Đo thời gian huấn luyện và Lưu vào danh sách TimeNB
TimeNB.append(end - start)
# Đánh giá hiệu suất trên tập huấn luyện và lưu kết quả
training_score_nb.append(nb.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
nb_pred = nb.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_nb
nb_score = metrics.accuracy_score(y_test, nb_pred)
test_score_nb.append(nb_score)
# In kết quả accuracy và báo cáo phân loại
print("Naive Bayes")
print("Accuracy:",nb_score)
print("Report:",metrics.classification_report(y_test,nb_pred))
#Tạo ma trận nhầm lẫn
nb_cm = metrics.confusion_matrix(y_test,nb_pred)

```

Hình 3. 15 Cài đặt thuật toán Naïve Bayes

- Kết quả chạy:

```

-----Naive Bayes-----
Naive Bayes
Accuracy: 0.6443796431299035
Report:
precision    recall   f1-score   support
0            0.89     0.33      0.48      7235
1            0.59     0.96      0.73      7168

accuracy          0.64      14403
macro avg       0.74     0.65      0.61      14403
weighted avg     0.74     0.64      0.61      14403

```

Hình 3. 16 Kết quả chạy Naïve Bayes

4. Thuật toán K - Nearest Neighbors:

4.1. Khái niệm:

- **K-Nearest Neighbors (KNN)** là một thuật toán phân loại không thông qua việc học mô hình mà dựa vào việc so sánh điểm dữ liệu mới với các điểm dữ liệu đã được gán nhãn trong tập huấn luyện. **KNN** sử dụng độ đo khoảng cách (thường là khoảng cách Euclidean) để tìm ra K điểm dữ liệu gần nhất và dùng đa số phiếu để quyết định nhãn của điểm dữ liệu mới. Thuật toán này đơn giản và dễ hiểu, nhưng có thể tốn nhiều thời gian khi phải tính toán khoảng cách với mỗi điểm dữ liệu trong tập huấn luyện.

4.2. Thực hiện:

- Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho K - Nearest Neighbors.

```

# Đối với K Nearest Neighbors
knn_probs = knn.predict_proba(x_test) # Dự đoán xác suất cho x_test
knn_probs = knn_probs[:, 1] # Chọn xác suất của lớp dương
knn_auc = roc_auc_score(y_test, knn_probs) # Tính AUC cho K Nearest Neighbors
print('K Nearest Neighbors ROC AUC=% .3f' % (knn_auc)) # In AUC
knn_fpr, knn_tpr, _ = roc_curve(y_test, knn_probs) # Tính FPR, TPR
plt.plot(knn_fpr, knn_tpr, linestyle='--', label='K Nearest Neighbors') # Vẽ đường ROC

```

Hình 3. 17 Hàm để vẽ biểu đồ đường cong ROC cho K - Nearest Neighbors

- Cài đặt thuật toán K - Nearest Neighbors.

```

# K-Nearest Neighbor
print("-----K-Nearest Neighbor-----")
start = time.time()
# Tạo một mô hình K-Nearest Neighbor
knn = KNeighborsClassifier()
# Huấn luyện mô hình
knn.fit(x_train,y_train)
end = time.time()
# Đo thời gian huấn luyện và lưu vào danh sách TimeKNN
TimeKNN.append(end - start)
# Đánh giá hiệu suất trên tập huấn luyện và lưu kết quả
training_score_knn.append(knn.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
knn_pred = knn.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_knn
knn_score = metrics.accuracy_score(y_test, knn_pred)
test_score_knn.append(knn_score)
# In kết quả accuracy và báo cáo phân loại
print("K-Nearest Neighbor")
print("Accuracy:",knn_score)
print("Report:",metrics.classification_report(y_test,knn_pred))
# Tạo ma trận nhầm lẫn
knn_cm = metrics.confusion_matrix(y_test,knn_pred)

```

Hình 3. 18 Cài đặt thuật toán K - Nearest Neighbors

- Kết quả chạy:

```

-----K-Nearest Neighbor-----
K-Nearest Neighbor
Accuracy: 0.8051794764979519
Report:
          precision    recall   f1-score   support
0           0.81      0.80      0.80      7235
1           0.80      0.81      0.81      7168

accuracy                      0.81      14403
macro avg                      0.81      0.81      14403
weighted avg                   0.81      0.81      14403

```

Hình 3. 19 Kết quả chạy K - Nearest Neighbors

5. Thuật toán Neural Network:

5.1. Khái niệm:

- Mạng nơ-ron (**Neural Network**) là một mô hình tính toán dựa trên cấu trúc của hệ thống nơ-ron trong não người. Mạng nơ-ron trong **Machine Learning** bao gồm các lớp nơ-ron liên kết với nhau qua các trọng số. Các thuật toán huấn luyện mạng nơ-ron như **Gradient Descent** được sử dụng để điều chỉnh các trọng số này sao cho mạng có thể học được cách biểu diễn dữ liệu và thực hiện các nhiệm vụ như phân loại hoặc dự đoán. **Neural Network** có thể mô hình hóa các mối quan hệ phức tạp trong dữ liệu, nhưng đòi hỏi nhiều dữ liệu huấn luyện và có thể cần nhiều thời gian và tài nguyên tính toán.

5.2. Thực hiện:

- Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho **Neural Network**.

```

# Đối với Neural Network
nn_probs = nn.predict_proba(x_test) # Dự đoán xác suất cho x_test
nn_probs = nn_probs[:, 1] # Chọn xác suất của Lớp dương
nn_auc = roc_auc_score(y_test, nn_probs) # Tính AUC cho Neural Network
print('Neural Network: ROC AUC=% .3f' % (nn_auc)) # In AUC
nn_fpr, nn_tpr, _ = roc_curve(y_test, nn_probs) # Tính FPR, TPR
plt.plot(nn_fpr, nn_tpr, linestyle='--', label='Neural Network') # Vẽ đường ROC

```

Hình 3. 20 Hàm để vẽ biểu đồ đường cong ROC cho Neural Network

- Cài đặt thuật toán **Neural Network**.

```
# Neural Network
print("-----Neural Network-----")
start = time.time()
# Tạo một mô hình Neural Network
nn = MLPClassifier(random_state=0, activation="logistic", solver="sgd")
# Huấn Luyện mô hình
nn.fit(x_train,y_train)
end = time.time()
# Đo thời gian huấn Luyện và Lưu vào danh sách TimeNN
TimeNN.append(end - start)
# Đánh giá hiệu suất trên tập huấn Luyện và Lưu kết quả
training_score_nn.append(nn.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
nn_pred = nn.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_nn
nn_score = metrics.accuracy_score(y_test, nn_pred)
test_score_nn.append(nn_score)
# In kết quả accuracy và báo cáo phân loại
print("Neural Network")
print("Accuracy:",nn_score)
print("Report:",metrics.classification_report(y_test,nn_pred))
# Tạo ma trận nhầm lẫn
nn_cm = metrics.confusion_matrix(y_test,nn_pred)
```

Hình 3. 21 Cài đặt thuật toán Neural Network

- Kết quả chạy:

```

-----Neural Network-----
Neural Network
Accuracy: 0.5378740540165243
Report:
precision    recall   f1-score   support
0            0.52     0.98      0.68     7235
1            0.84     0.09      0.16     7168

accuracy          0.54     14403
macro avg       0.68     0.54      0.42     14403
weighted avg    0.68     0.54      0.42     14403

```

Hình 3. 22 Kết quả chạy Neural Network

6. Thuật toán Random Forest:

6.1. Khái niệm:

- **Random Forest** là một phương pháp học máy dựa trên việc kết hợp nhiều cây quyết định (**decision trees**) thành một mô hình dự đoán. Mỗi cây trong **Random Forest** được huấn luyện trên một tập dữ liệu con được lấy mẫu ngẫu nhiên từ tập dữ liệu huấn luyện ban đầu và sử dụng một số thuộc tính ngẫu nhiên trong quá trình xây dựng cây. Kết quả dự đoán cuối cùng là kết quả trung bình hoặc đa số phiếu của các cây con. **RandomForest** thường cho hiệu suất tốt trên nhiều loại dữ liệu và ít dễ bị **overfitting** so với một cây quyết định đơn lẻ. Đặc biệt, **Random Forest** có thể xử lý dữ liệu lớn, có nhiều thuộc tính và có khả năng xử lý nhiễu tốt.

6.2. Thực hiện:

- Chuẩn bị hàm lưu ảnh cây ngẫu nhiên **Random Forest**:

```

# Lưu một cây ngẫu nhiên từ mô hình Random Forest vào file ảnh.
def save_randomForest(rf, i, corr, x_train):
    fig, ax = plt.subplots(figsize=(50,24))
    tree.plot_tree(rf.estimators_[0], filled=True)
    plt.savefig('RandomForest_{0}_{1}.png'.format(corr, i))

```

Hình 3. 23 Hàm lưu ảnh cây ngẫu nhiên Random Forest

- Chuẩn bị hàm để vẽ biểu đồ đường cong ROC cho Random Forest.

```
# Đổi với Random Forest
rf_probs = rf.predict_proba(x_test) # Dự đoán xác suất cho x_test
rf_probs = rf_probs[:, 1] # Chọn xác suất của Lớp dương
rf_auc = roc_auc_score(y_test, rf_probs) # Tính AUC cho Random Forest
print('Random Forest: ROC AUC=% .3f' % (rf_auc)) # In AUC
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs) # Tính FPR, TPR
plt.plot(rf_fpr, rf_tpr, linestyle='--', label='Random Forest') # Vẽ đường ROC
```

Hình 3. 24 Hàm để vẽ biểu đồ đường cong ROC cho Random Forest

- Cài đặt thuật toán Random Forest.

```
# Random Forest
print("-----Random Forest-----")
start = time.time()
# Tạo một mô hình Random Forest
rf = RandomForestClassifier()
# Huấn Luyện mô hình
rf.fit(x_train, y_train)
end = time.time()
# Đo thời gian huấn Luyện và lưu vào danh sách TimeRF
TimeRF.append(end - start)
# Đánh giá hiệu suất trên tập huấn Luyện và lưu kết quả
training_score_rf.append(rf.score(x_train, y_train))
# Dự đoán nhãn trên tập kiểm tra
rf_pred = rf.predict(x_test)
# Tính toán accuracy và lưu vào danh sách test_score_rf
rf_score = metrics.accuracy_score(y_test, rf_pred)
test_score_rf.append(rf_score)
# In kết quả accuracy và báo cáo phân loại
print("Random Forest")
print("Accuracy:", rf_score)
print("Report:", metrics.classification_report(y_test, rf_pred))
# Tạo ma trận nhầm lẫn
rf_cm = metrics.confusion_matrix(y_test, rf_pred)
# Lưu mô hình Random Forest
save_randomForest(rf, i, corr, x_train)
```

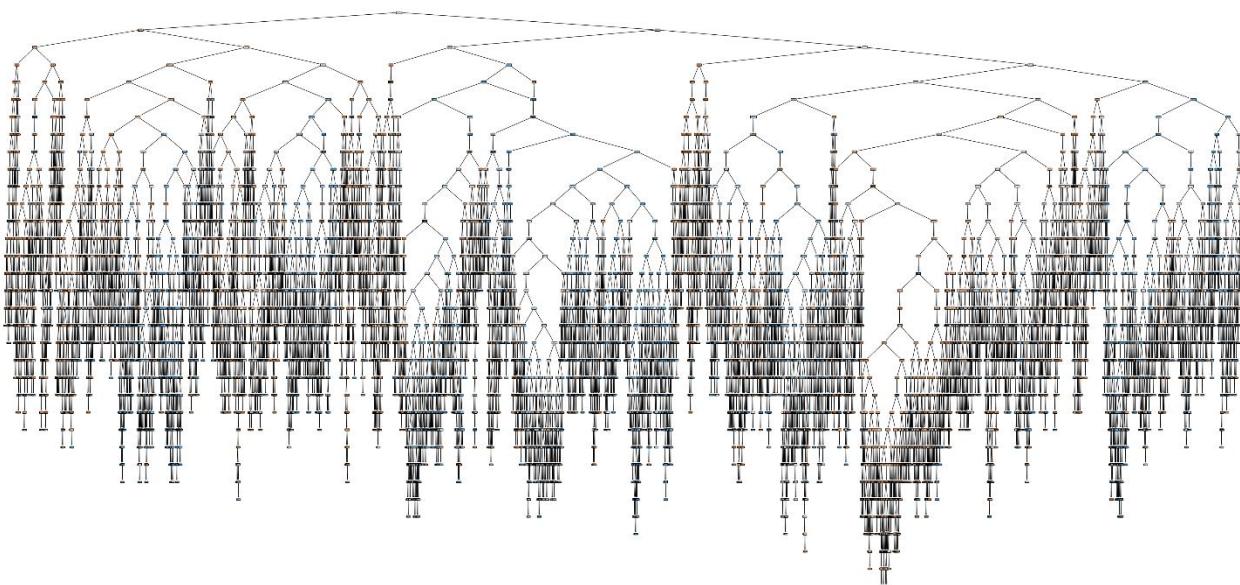
Hình 3. 25 Cài đặt thuật toán Random Forest

- Kết quả chạy:

```
-----Random Forest-----  
Random Forest  
Accuracy: 0.8564882316184128  
Report:  
precision recall f1-score support  
  
0 0.85 0.87 0.86 7235  
1 0.87 0.84 0.85 7168  
  
accuracy 0.86 0.86 0.86 14403  
macro avg 0.86 0.86 0.86 14403  
weighted avg 0.86 0.86 0.86 14403
```

Hình 3. 26 Kết quả chạy Random Forest

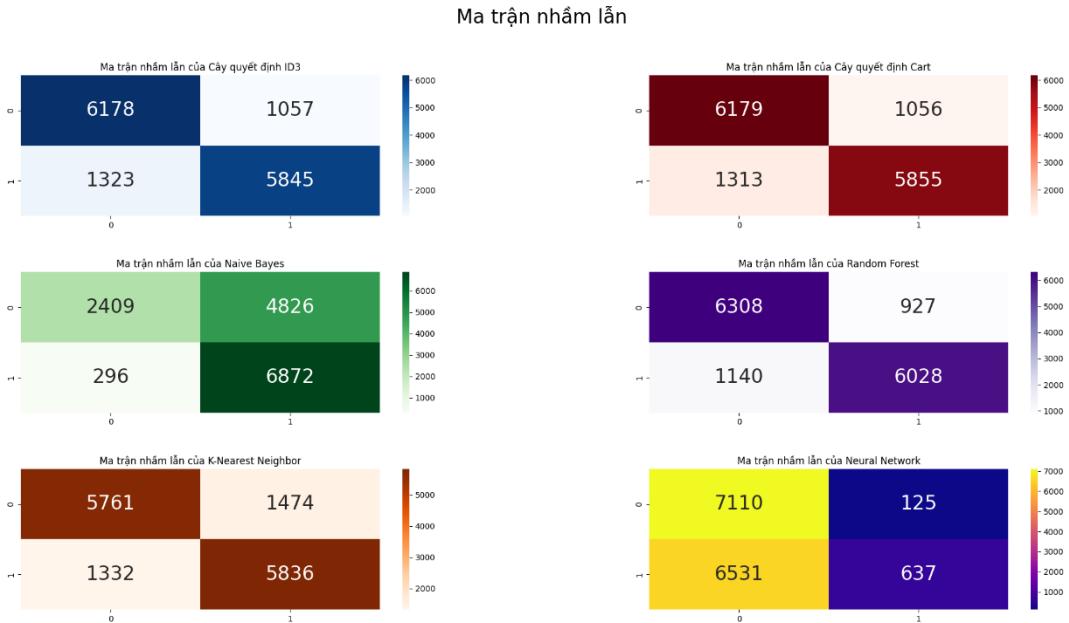
Mô hình cây ngẫu nhiên:



Hình 3. 27 Mô hình cây ngẫu nhiên

CHƯƠNG 4: ĐÁNH GIÁ CÁC THUẬT TOÁN

1. Đánh giá ma trận nhầm lẩn:



Hình 4. 1 Ma trận nhầm lẩn của 6 thuật toán

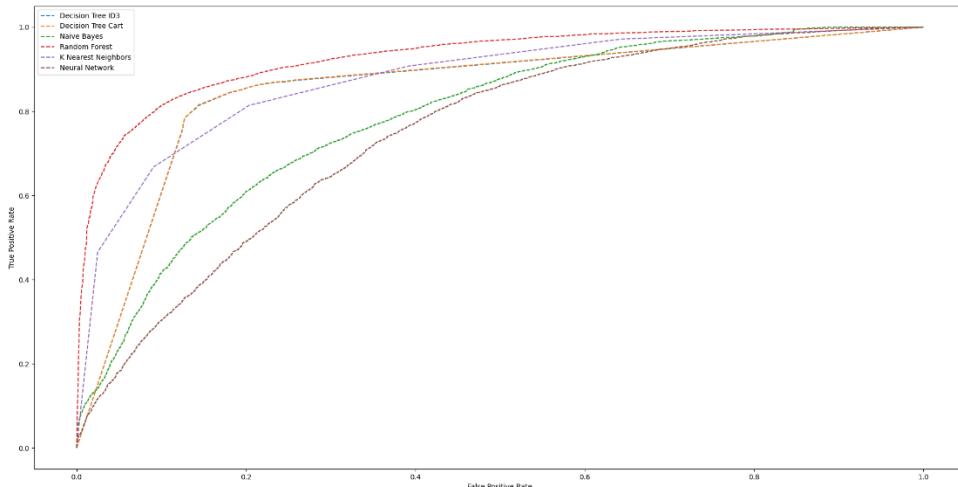
Nhận xét: Nhìn chung, ta thấy tỷ lệ **TPR** (True Positive Rate) và **FNR** (False Negative Rate) cao hơn **TNR** (True Negative Rate) và **FPR** (False Positive Rate) cho thấy mô hình phân loại có hiệu suất hợp lệ vì điều này phản ánh khả năng nhận biết chính xác của mô hình giữa các lớp phân loại, đặc biệt là khả năng nhận biết chính xác các trường hợp hủy đặt phòng.

- **TPR (Tỷ lệ đúng dương):** Tỷ lệ phần trăm của các dự đoán đúng là hủy đặt phòng (đã hủy) so với tổng số lượng thực sự là hủy đặt phòng. **TPR** cao cho thấy mô hình có khả năng nhận biết chính xác các trường hợp hủy đặt phòng, điều này rất quan trọng trong việc giảm thiểu rủi ro và đảm bảo lợi nhuận cho khách sạn.
- **FNR (Tỷ lệ sai âm):** Tỷ lệ phần trăm của các dự đoán sai là chưa hủy đặt phòng (đã hủy nhưng mô hình dự đoán chưa hủy) so với tổng số lượng thực sự là hủy đặt phòng. **FNR** thấp cho thấy mô hình ít bỏ sót các trường hợp hủy đặt phòng, điều này

giúp khách sạn nắm bắt kịp thời các trường hợp hủy để có thể điều chỉnh và tối ưu hóa nguồn lực.

- **TNR (Tỷ lệ đúng âm):** Tương tự như **TPR** nhưng dành cho các dự đoán thuộc lớp chưa hủy đặt phòng. **TNR** cao cho thấy mô hình có khả năng nhận biết chính xác các trường hợp chưa hủy đặt phòng, giúp giảm thiểu số lượng dự đoán sai là hủy đặt phòng.
- **FPR (Tỷ lệ sai dương):** Tương tự như FNR nhưng dành cho các dự đoán thuộc lớp chưa hủy đặt phòng. **FPR** thấp cho thấy mô hình ít báo nhầm các trường hợp chưa hủy đặt phòng là hủy đặt phòng, giúp giảm thiểu số lượng dự đoán sai và cải thiện chất lượng dịch vụ khách hàng.

2. Đánh giá tỷ lệ TPR trên tỷ lệ FPR trong biểu đồ đường cong ROC:



Hình 4. 2 Biểu đồ đường cong ROC

Nhận xét:

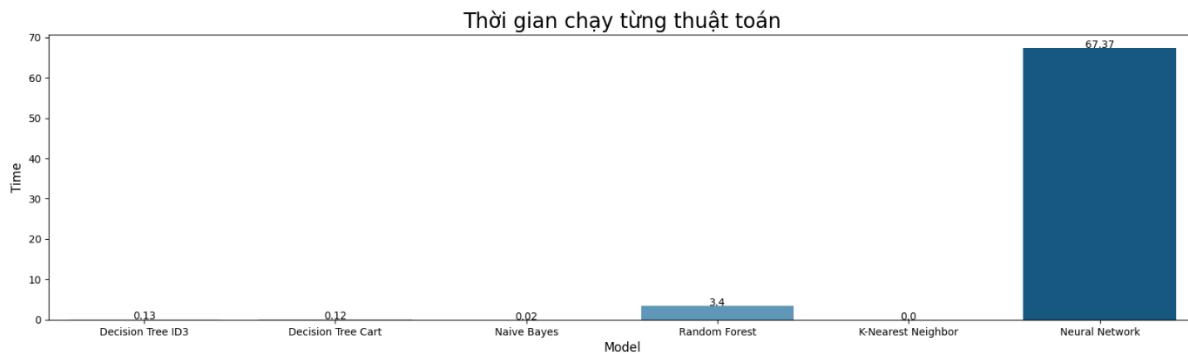
- Đường cong **ROC** của thuật toán **Random Forest** nằm gần góc trên bên trái của biểu đồ nhất cho thấy mô hình của thuật toán này là tốt nhất.
- Đường cong **ROC** của thuật toán **Neural Network** nằm xa góc trên bên trái của biểu đồ nhất cho thấy mô hình của thuật toán này là tệ nhất.

3. Đánh giá về thời gian chạy từng thuật toán:

```
df_time = pd.DataFrame(data=[('Decision Tree ID3',mean(TimeID3)), ('Decision Tree Cart',mean(TimeCart)),
                             ('Naive Bayes',mean(TimeNB)), ('Random Forest',mean(TimeRF)),
                             ('K-Nearest Neighbor',mean(TimeKNN)), ('Neural Network',mean(TimeNN))], columns=['Thuật toán', 'Thời gian chạy'])
df_time
```

	Thuật toán	Thời gian chạy
0	Decision Tree ID3	0.128959
1	Decision Tree Cart	0.123357
2	Naive Bayes	0.022168
3	Random Forest	3.403580
4	K-Nearest Neighbor	0.003069
5	Neural Network	67.365045

Hình 4. 3 Thông kê thời gian chạy của từng thuật toán



Hình 4. 4 Biểu đồ thời gian chạy của từng thuật toán

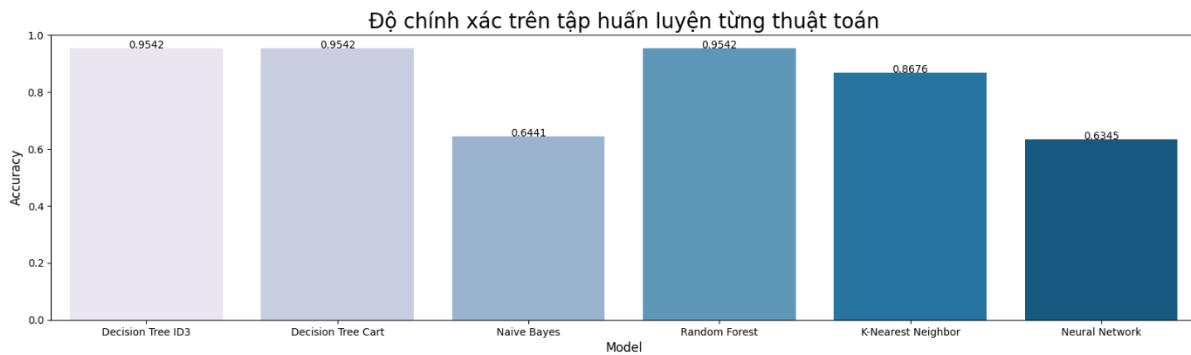
Nhận xét:

- **Neural Network** tốn nhiều thời gian nhất.
- Thời gian chạy của KNN gần như bằng không do đặc tính **Lazy Learner** của chính nó.

4. Đánh giá độ chính xác của thuật toán trên tập huấn luyện:

df_train = pd.DataFrame(data=[('Decision Tree ID3',mean(training_score_id3)), ('Decision Tree Cart',mean(training_score_cart)), ('Naive Bayes',mean(training_score_nb)), ('Random Forest',mean(training_score_rf)), ('K-Nearest Neighbor',mean(training_score_knn)), ('Neural Network',mean(training_score_nn))], columns=['Model', 'Accuracy'])
df_train
0 Decision Tree ID3 0.954233
1 Decision Tree Cart 0.954233
2 Naive Bayes 0.644071
3 Random Forest 0.954233
4 K-Nearest Neighbor 0.867579
5 Neural Network 0.634459

Hình 4. 5 Thông kê độ chính xác trên tập train của từng thuật toán



Hình 4. 6 Biểu đồ độ chính xác trên tập train của từng thuật toán

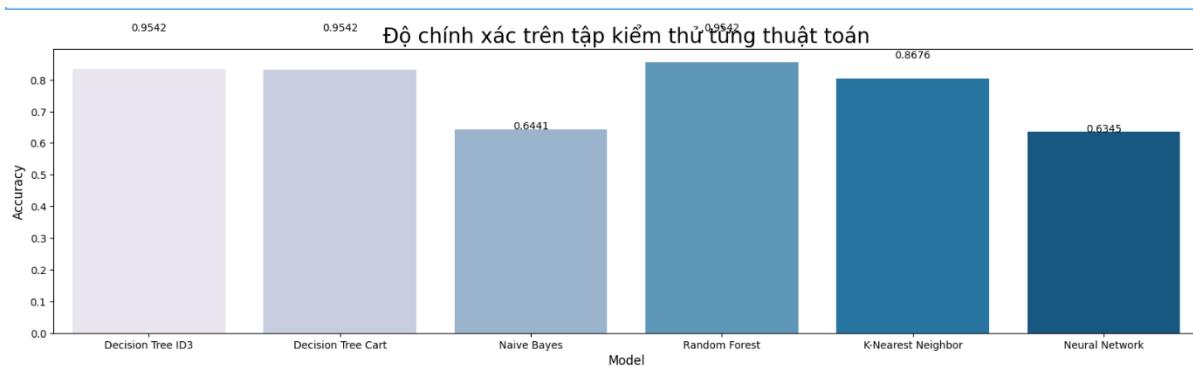
Nhận xét:

- Độ chính xác của thuật toán cây quyết định **ID3, CART** và **Random Forest** cao nhất (**95,42%**)
- Độ chính xác của thuật toán **Neural Network** thấp nhất (**63,45%**)

5. Đánh giá độ chính xác của thuật toán trên tập kiểm thử:

	Model	Accuracy
0	Decision Tree ID3	0.833275
1	Decision Tree Cart	0.831331
2	Naive Bayes	0.643431
3	Random Forest	0.855030
4	K-Nearest Neighbor	0.803860
5	Neural Network	0.636395

Hình 4. 7 Thông kê độ chính xác trên tập test của từng thuật toán



Hình 4. 8 Biểu đồ độ chính xác trên tập test của từng thuật toán

Nhận xét:

- Độ chính xác của thuật toán cây quyết định **ID3, CART** và **Random Forest** cao nhất (**95,42%**)
- Độ chính xác của thuật toán **Neural Network** thấp nhất (**63,45%**)

==> Độ chính xác của các mô hình thuật toán trong kiểm thử gần như giống nhau với mô hình khi huấn luyện.

CHƯƠNG 5: KẾT LUẬN

1. Ưu - Nhược điểm của từng thuật toán:

1.1. Decision Tree

Ưu điểm:

- Hiển thị rõ ràng: **Decision Tree** dễ hiểu và giải thích, giúp người dùng nắm bắt được quy luật mà thuật toán đã học từ dữ liệu.
- Không cần chuẩn hóa dữ liệu: Không yêu cầu tiền xử lý dữ liệu phức tạp như **Normalization** hoặc **Standardization**.
- Tính đơn giản: Thuật toán hoạt động dựa trên nguyên tắc cây quyết định, dễ dàng triển khai và duy trì.

Nhược điểm:

- Overfitting: Cây quyết định có thể quá phức tạp và dẫn đến **overfitting** nếu không được điều chỉnh đúng cách.
- Khả năng xử lý dữ liệu lớn hạn chế: Với lượng dữ liệu lớn, việc xây dựng và duy trì cây quyết định có thể trở nên khó khăn.

1.2. Naïve Bayes

Ưu điểm:

- Hiệu quả với dữ liệu rời rạc: Đạt hiệu quả cao với dữ liệu dạng rời rạc, thường thấy trong dữ liệu khách sạn như loại phòng, ngày **check-in/check-out**.
- Khả năng xử lý nhanh: Thuật toán nhanh chóng và phù hợp cho việc phân loại lớn lượng dữ liệu.
- Không cần nhiều tính huống: Dữ liệu đầu vào không cần phải được chuẩn hóa trước khi sử dụng.

Nhược điểm:

- Giả định độc lập: Giả định rằng các đặc trưng là hoàn toàn độc lập với nhau, điều này không thực tế trong nhiều trường hợp.

- Độ chính xác thấp với dữ liệu liên quan: Khi các đặc trưng có mối liên hệ mạnh mẽ, **Naïve Bayes** có thể không hiệu quả.

1.3. K – Nearest Neighbor

Ưu điểm:

- Thời gian chạy nhanh nhờ cơ chế **Lazy Learner**.
- Độ chính xác sau khi được chuẩn hóa đạt giá trị cao nhất.

Khuyết điểm:

- Độ chính xác giữa tập kiểm định và tập kiểm thử khi chưa chuẩn hóa dữ liệu có độ chênh lệch rất cao nên mô hình không phù hợp.
- KNN không hoạt động tốt trên tập dữ liệu lớn và kích thước cao. Hơn nữa, KNN còn nhạy cảm với nhiễu trong tập dữ liệu nên cần loại bỏ.

1.4. Neural Network

Ưu điểm:

- Biểu đồ độ chính xác trên tập train của từng thuật toán **Neural Network** có thể học từ dữ liệu phức tạp và mô hình hóa các mối quan hệ gián tiếp giữa các biến.
- Cải thiện độ chính xác: Có thể đạt được độ chính xác cao hơn so với các thuật toán truyền thống.

Nhược điểm:

- Tốn nhiều thời gian chạy đối với cả ba trường hợp.
- Độ chính xác khi huấn luyện dữ liệu khá cao khi dữ liệu chưa được chuẩn hóa tuy nhiên độ chênh lệch với độ chính xác của tập thử nghiệm khá cao.
- Phụ thuộc vào phần cứng do cần bộ xử lý hoạt động cao khi xử lý song song trong mạng neural.

1.5. Random Forest

Ưu điểm:

- **Khả năng giảm thiểu overfitting:** Bằng cách tạo ra nhiều cây quyết định và lấy trung bình, **Random Forest** giảm thiểu nguy cơ **overfitting**.
- **Hiệu suất cao:** Thường xuyên đạt được hiệu suất phân loại cao hơn so với các thuật toán đơn lẻ.
- **Khả năng xử lý dữ liệu lớn:** Tốt hơn so với **Decision Trees** và **KNN**.

Nhược điểm:

- Thời gian tính toán: Thời gian tính toán có thể dài hơn so với các thuật toán đơn giản hơn như **Naïve Bayes**.
- Khối lượng dữ liệu: Cần đủ lượng dữ liệu để đảm bảo hiệu suất tốt của mô hình.

==> Các thuật toán học máy như **Decision Tree**, **Naïve Bayes**, **K-Nearest Neighbor**, **Neural Network** và **Random Forest** đều mang lại những ưu và nhược điểm riêng biệt khi được áp dụng vào dự đoán khả năng hủy phòng của khách sạn. Việc lựa chọn thuật toán phù hợp đòi hỏi việc cân nhắc kỹ lưỡng dựa trên đặc điểm của dữ liệu và mục tiêu của dự án.

2. Hướng phát triển:

- Cập nhật và thử nghiệm với các mô hình học máy mới hơn, ví dụ như deep learning models, attention mechanisms, transfer learning.
- Phát triển ứng dụng hoặc trang web cho phép khách hàng đặt phòng trực tuyến, nhận thông báo và cập nhật về tình trạng đặt phòng của họ.

BẢNG PHÂN CHIA CÔNG VIỆC

Công việc	Phạm Quốc Hùng - 19521579	Lê Trần Anh Quý - 21520094	Trần Minh Quang - 21522519	Đặng Lưu Hà - 21520798
Chọn dataset, xác định đề tài	X	X	X	X
Thực hiện tổng quan đề tài				X
Trục quan hóa dữ liệu, Mô tả cơ bản về dữ liệu, làm sạch dữ liệu		X		
Thêm, giảm, rời rạc hóa dữ liệu			X	
Dán nhãn dữ liệu và lấy tập mẫu	X			
Thực hiện thuật toán Decison Tree, Random Forest		X		
Thực hiện thuật toán Naive Bayes và KNN			X	X
Thực hiện thuật toán Neural Network	X			
Đánh giá các thuật toán		X	X	
Ưu, nhược điểm và hướng phát triển	X			
Viết báo cáo			X	X

BẢNG ĐÁNH GIÁ CÁC THÀNH VIÊN TRONG NHÓM

Thành viên	Phạm Quốc Hùng - 19521579	Lê Trần Anh Quí - 21520094	Trần Minh Quang - 21522519	Đặng Lưu Hà - 21520798
Điểm	9	10	9	9

TÀI LIỆU THAM KHẢO

- [1] Tài liệu lý thuyết của GV. Mai Xuân Hùng.
- [2] Tài liệu thực hành của GV. Phạm Nguyễn Thanh Bình
- [3] Random Forest Classification with Scikit-Learn: <https://www.datacamp.com/tutorial/random-forests-classifier-python>
- [4] Binning Data in Pandas with cut and qcut: <https://dataqy.io/pandas-cut-qcut/#:~:text=Pandas%20qcut%3A%20Binning%20Data%20into%20Equal-Sized%20Bins%20The,quantiles.%20This%20process%20is%20known%20as%20quantile-based%20discretization>